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

import com.twitter.dal.client.dataset.KeyValDALDataset

import com.twitter.scalding.DateRange

import com.twitter.scalding.typed.TypedPipe

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

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

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

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

import com.twitter.wtf.entity\_real\_graph.thriftscala.EntityType

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

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

object EntityEmbeddingsSources {

final val SemanticCoreSimClustersEmbeddingsDec11Dataset =

SemanticCoreSimclustersEmbeddingsScalaDataset

final val SemanticCoreSimClustersEmbeddingsUpdatedDataset =

SemanticCoreSimclustersEmbeddingsUpdatedScalaDataset

final val SemanticCoreSimClustersEmbeddings2020Dataset =

SemanticCoreSimclustersEmbeddings2020ScalaDataset

final val SemanticCorePerLanguageSimClustersEmbeddingsDataset =

SemanticCorePerLanguageSimclustersEmbeddingsScalaDataset

final val LogFavSemanticCorePerLanguageSimClustersEmbeddingsDataset =

LogFavSemanticCorePerLanguageSimclustersEmbeddingsScalaDataset

final val HashtagSimClustersEmbeddingsUpdatedDataset =

HashtagSimclustersEmbeddingsUpdatedScalaDataset

final val ReverseIndexSemanticCoreSimClustersEmbeddingsDec11Dataset =

ReverseIndexSemanticCoreSimclustersEmbeddingsScalaDataset

final val ReverseIndexSemanticCoreSimClustersEmbeddingsUpdatedDataset =

ReverseIndexSemanticCoreSimclustersEmbeddingsUpdatedScalaDataset

final val ReverseIndexSemanticCoreSimClustersEmbeddings2020Dataset =

ReverseIndexSemanticCoreSimclustersEmbeddings2020ScalaDataset

final val ReverseIndexSemanticCorePerLanguageSimClustersEmbeddingsDataset =

ReverseIndexSemanticCorePerLanguageSimclustersEmbeddingsScalaDataset

final val LogFavReverseIndexSemanticCorePerLanguageSimClustersEmbeddingsDataset =

LogFavReverseIndexSemanticCorePerLanguageSimclustersEmbeddingsScalaDataset

final val ReverseIndexHashtagSimClustersEmbeddingsUpdatedDataset =

ReverseIndexHashtagSimclustersEmbeddingsUpdatedScalaDataset

// Fav-based TFG topic embeddings built from user device languages

// Keyed by SimClustersEmbeddingId with InternalId.TopicId ((topic, language) pair, with country = None)

final val FavTfgTopicEmbeddingsDataset = FavTfgTopicEmbeddingsScalaDataset

final val FavTfgTopicEmbeddingsParquetDataset = FavTfgTopicEmbeddingsParquetScalaDataset

final val FavTfgTopicEmbeddings2020Dataset = FavTfgTopicEmbeddings2020ScalaDataset

final val FavTfgTopicEmbeddings2020ParquetDataset = FavTfgTopicEmbeddings2020ParquetScalaDataset

// Logfav-based TFG topic embeddings built from user device languages

// Keyed by SimClustersEmbeddingId with InternalId.LocaleEntityId ((topic, language) pair)

final val LogFavTfgTopicEmbeddingsDataset = LogFavTfgTopicEmbeddingsScalaDataset

final val LogFavTfgTopicEmbeddingsParquetDataset = LogFavTfgTopicEmbeddingsParquetScalaDataset

// Fav-based TFG topic embeddings built from inferred user consumed languages

// Keyed by SimClustersEmbeddingId with InternalId.TopicId ((topic, country, language) tuple)

final val FavInferredLanguageTfgTopicEmbeddingsDataset =

FavInferredLanguageTfgTopicEmbeddingsScalaDataset

private val validSemanticCoreEmbeddingTypes = Seq(

EmbeddingType.FavBasedSematicCoreEntity,

EmbeddingType.FollowBasedSematicCoreEntity

)

/\*\*

\* Given a fav/follow/etc embedding type and a ModelVersion, retrieve the corresponding dataset to

\* (SemanticCore entityId -> List(clusterId)) from a certain dateRange.

\*/

def getSemanticCoreEntityEmbeddingsSource(

embeddingType: EmbeddingType,

modelVersion: String,

dateRange: DateRange

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

val dataSet = modelVersion match {

case ModelVersions.Model20M145KDec11 => SemanticCoreSimClustersEmbeddingsDec11Dataset

case ModelVersions.Model20M145KUpdated => SemanticCoreSimClustersEmbeddingsUpdatedDataset

case \_ => throw new IllegalArgumentException(s"ModelVersion $modelVersion is not supported")

}

assert(validSemanticCoreEmbeddingTypes.contains(embeddingType))

entityEmbeddingsSource(dataSet, embeddingType, dateRange)

}

/\*\*

\* Given a fav/follow/etc embedding type and a ModelVersion, retrieve the corresponding dataset to

\* (clusterId -> List(SemanticCore entityId)) from a certain dateRange.

\*/

def getReverseIndexedSemanticCoreEntityEmbeddingsSource(

embeddingType: EmbeddingType,

modelVersion: String,

dateRange: DateRange

): TypedPipe[(ClusterId, Seq[SemanticCoreEntityWithScore])] = {

val dataSet = modelVersion match {

case ModelVersions.Model20M145KDec11 =>

ReverseIndexSemanticCoreSimClustersEmbeddingsDec11Dataset

case ModelVersions.Model20M145KUpdated =>

ReverseIndexSemanticCoreSimClustersEmbeddingsUpdatedDataset

case ModelVersions.Model20M145K2020 =>

ReverseIndexSemanticCoreSimClustersEmbeddings2020Dataset

case \_ => throw new IllegalArgumentException(s"ModelVersion $modelVersion is not supported")

}

assert(validSemanticCoreEmbeddingTypes.contains(embeddingType))

reverseIndexedEntityEmbeddingsSource(dataSet, embeddingType, dateRange)

}

// Return the raw DAL dataset reference. Use this if you're writing to DAL.

def getEntityEmbeddingsDataset(

entityType: EntityType,

modelVersion: String,

isEmbeddingsPerLocale: Boolean = false

): KeyValDALDataset[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]] = {

(entityType, modelVersion) match {

case (EntityType.SemanticCore, ModelVersions.Model20M145KDec11) =>

SemanticCoreSimClustersEmbeddingsDec11Dataset

case (EntityType.SemanticCore, ModelVersions.Model20M145KUpdated) =>

if (isEmbeddingsPerLocale) {

SemanticCorePerLanguageSimClustersEmbeddingsDataset

} else {

SemanticCoreSimClustersEmbeddingsUpdatedDataset

}

case (EntityType.SemanticCore, ModelVersions.Model20M145K2020) =>

SemanticCoreSimClustersEmbeddings2020Dataset

case (EntityType.Hashtag, ModelVersions.Model20M145KUpdated) =>

HashtagSimClustersEmbeddingsUpdatedDataset

case (entityType, modelVersion) =>

throw new IllegalArgumentException(

s"(Entity Type, ModelVersion) ($entityType, $modelVersion) not supported.")

}

}

// Return the raw DAL dataset reference. Use this if you're writing to DAL.

def getReverseIndexedEntityEmbeddingsDataset(

entityType: EntityType,

modelVersion: String,

isEmbeddingsPerLocale: Boolean = false

): KeyValDALDataset[KeyVal[SimClustersEmbeddingId, InternalIdEmbedding]] = {

(entityType, modelVersion) match {

case (EntityType.SemanticCore, ModelVersions.Model20M145KDec11) =>

ReverseIndexSemanticCoreSimClustersEmbeddingsDec11Dataset

case (EntityType.SemanticCore, ModelVersions.Model20M145KUpdated) =>

if (isEmbeddingsPerLocale) {

ReverseIndexSemanticCorePerLanguageSimClustersEmbeddingsDataset

} else {

ReverseIndexSemanticCoreSimClustersEmbeddingsUpdatedDataset

}

case (EntityType.SemanticCore, ModelVersions.Model20M145K2020) =>

ReverseIndexSemanticCoreSimClustersEmbeddings2020Dataset

case (EntityType.Hashtag, ModelVersions.Model20M145KUpdated) =>

ReverseIndexHashtagSimClustersEmbeddingsUpdatedDataset

case (entityType, modelVersion) =>

throw new IllegalArgumentException(

s"(Entity Type, ModelVersion) ($entityType, $modelVersion) not supported.")

}

}

private def entityEmbeddingsSource(

dataset: KeyValDALDataset[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]],

embeddingType: EmbeddingType,

dateRange: DateRange

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

val pipe = DAL

.readMostRecentSnapshot(dataset, dateRange)

.withRemoteReadPolicy(AllowCrossDC)

.toTypedPipe

filterEntityEmbeddingsByType(pipe, embeddingType)

}

private def reverseIndexedEntityEmbeddingsSource(

dataset: KeyValDALDataset[KeyVal[SimClustersEmbeddingId, InternalIdEmbedding]],

embeddingType: EmbeddingType,

dateRange: DateRange

): TypedPipe[(ClusterId, Seq[SemanticCoreEntityWithScore])] = {

val pipe = DAL

.readMostRecentSnapshot(dataset, dateRange)

.withRemoteReadPolicy(AllowCrossDC)

.toTypedPipe

filterReverseIndexedEntityEmbeddingsByType(pipe, embeddingType)

}

private[hdfs\_sources] def filterEntityEmbeddingsByType(

pipe: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]],

embeddingType: EmbeddingType

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

pipe.collect {

case KeyVal(

SimClustersEmbeddingId(\_embeddingType, \_, InternalId.EntityId(entityId)),

embedding

) if \_embeddingType == embeddingType =>

(entityId, embedding)

}

}

private[hdfs\_sources] def filterReverseIndexedEntityEmbeddingsByType(

pipe: TypedPipe[KeyVal[SimClustersEmbeddingId, InternalIdEmbedding]],

embeddingType: EmbeddingType

): TypedPipe[(ClusterId, Seq[SemanticCoreEntityWithScore])] = {

pipe.collect {

case KeyVal(

SimClustersEmbeddingId(\_embeddingType, \_, InternalId.ClusterId(clusterId)),

embedding

) if \_embeddingType == embeddingType =>

val entitiesWithScores = embedding.embedding.collect {

case InternalIdWithScore(InternalId.EntityId(entityId), score) =>

SemanticCoreEntityWithScore(entityId, score)

}

(clusterId, entitiesWithScores)

}

}

}