package com.twitter.simclusters\_v2.common

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

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

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

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

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

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

SimClustersEmbeddingId => ThriftSimClustersEmbeddingId

}

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

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

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

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

import com.twitter.simclusters\_v2.thriftscala.{EmbeddingType => SimClustersEmbeddingType}

object SimClustersEmbeddingId {

val DefaultModelVersion: ModelVersion = ModelVersion.Model20m145k2020

// Embeddings which is available in Content-Recommender

val TweetEmbeddingTypes: Set[EmbeddingType] =

Set(

FavBasedTweet,

FollowBasedTweet,

LogFavBasedTweet,

LogFavLongestL2EmbeddingTweet

)

val DefaultTweetEmbeddingType: EmbeddingType = LogFavLongestL2EmbeddingTweet

val UserInterestedInEmbeddingTypes: Set[EmbeddingType] =

Set(

FavBasedUserInterestedIn,

FollowBasedUserInterestedIn,

LogFavBasedUserInterestedIn,

RecentFollowBasedUserInterestedIn,

FilteredUserInterestedIn,

FavBasedUserInterestedInFromPE,

FollowBasedUserInterestedInFromPE,

LogFavBasedUserInterestedInFromPE,

FilteredUserInterestedInFromPE,

LogFavBasedUserInterestedInFromAPE,

FollowBasedUserInterestedInFromAPE,

UnfilteredUserInterestedIn

)

val DefaultUserInterestInEmbeddingType: EmbeddingType = FavBasedUserInterestedIn

val ProducerEmbeddingTypes: Set[EmbeddingType] =

Set(

FavBasedProducer,

FollowBasedProducer,

AggregatableFavBasedProducer,

AggregatableLogFavBasedProducer,

RelaxedAggregatableLogFavBasedProducer,

KnownFor

)

val DefaultProducerEmbeddingType: EmbeddingType = FavBasedProducer

val LocaleEntityEmbeddingTypes: Set[EmbeddingType] =

Set(

FavTfgTopic,

LogFavTfgTopic

)

val DefaultLocaleEntityEmbeddingType: EmbeddingType = FavTfgTopic

val TopicEmbeddingTypes: Set[EmbeddingType] =

Set(

LogFavBasedKgoApeTopic

)

val DefaultTopicEmbeddingType: EmbeddingType = LogFavBasedKgoApeTopic

val AllEmbeddingTypes: Set[EmbeddingType] =

TweetEmbeddingTypes ++

UserInterestedInEmbeddingTypes ++

ProducerEmbeddingTypes ++

LocaleEntityEmbeddingTypes ++

TopicEmbeddingTypes

def buildTweetId(

tweetId: TweetId,

embeddingType: EmbeddingType = DefaultTweetEmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

assert(TweetEmbeddingTypes.contains(embeddingType))

ThriftSimClustersEmbeddingId(

embeddingType,

modelVersion,

InternalId.TweetId(tweetId)

)

}

def buildUserInterestedInId(

userId: UserId,

embeddingType: EmbeddingType = DefaultUserInterestInEmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

assert(UserInterestedInEmbeddingTypes.contains(embeddingType))

ThriftSimClustersEmbeddingId(

embeddingType,

modelVersion,

InternalId.UserId(userId)

)

}

def buildProducerId(

userId: UserId,

embeddingType: EmbeddingType = DefaultProducerEmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

assert(ProducerEmbeddingTypes.contains(embeddingType))

ThriftSimClustersEmbeddingId(

embeddingType,

modelVersion,

InternalId.UserId(userId)

)

}

def buildLocaleEntityId(

entityId: SemanticCoreEntityId,

language: String,

embeddingType: EmbeddingType = DefaultLocaleEntityEmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

ThriftSimClustersEmbeddingId(

embeddingType,

modelVersion,

InternalId.LocaleEntityId(

LocaleEntityId(entityId, language)

)

)

}

def buildTopicId(

topicId: TopicId,

language: Option[String] = None,

country: Option[String] = None,

embeddingType: EmbeddingType = DefaultTopicEmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

ThriftSimClustersEmbeddingId(

embeddingType,

modelVersion,

InternalId.TopicId(

TopicId(topicId, language, country)

)

)

}

// Extractor object for InternalIds that wrap Long

object LongInternalId {

def unapply(iid: InternalId): Option[Long] = iid match {

case InternalId.TweetId(id) => Some(id)

case InternalId.UserId(id) => Some(id)

case InternalId.EntityId(id) => Some(id)

case \_ => None

}

}

// Extractor object for SimClusterEmbeddingIds with InternalIds that wrap Long

object LongSimClustersEmbeddingId {

def unapply(id: ThriftSimClustersEmbeddingId): Option[Long] =

LongInternalId.unapply(id.internalId)

}

// Only for debuggers.

def buildEmbeddingId(

entityId: String,

embeddingType: EmbeddingType,

modelVersion: ModelVersion = DefaultModelVersion

): ThriftSimClustersEmbeddingId = {

if (TweetEmbeddingTypes.contains(embeddingType)) {

buildTweetId(entityId.toLong, embeddingType, modelVersion)

} else if (UserInterestedInEmbeddingTypes.contains(embeddingType)) {

buildUserInterestedInId(entityId.toLong, embeddingType, modelVersion)

} else if (ProducerEmbeddingTypes.contains(embeddingType)) {

buildProducerId(entityId.toLong, embeddingType, modelVersion)

} else if (LocaleEntityEmbeddingTypes.contains(embeddingType)) {

buildLocaleEntityId(entityId.toLong, "en", embeddingType, modelVersion)

} else if (TopicEmbeddingTypes.contains(embeddingType)) {

buildTopicId(

entityId.toLong,

Some("en"),

embeddingType = embeddingType,

modelVersion = modelVersion)

} else {

throw new IllegalArgumentException(s"Invalid embedding type: $embeddingType")

}

}

implicit val internalIdOrdering: Ordering[InternalId] =

Ordering.by(internalId => internalId.hashCode())

implicit val simClustersEmbeddingIdOrdering: Ordering[ThriftSimClustersEmbeddingId] =

Ordering.by(embeddingId =>

(embeddingId.embeddingType.value, embeddingId.modelVersion.value, embeddingId.internalId))

// Use Enum for feature switch

object TopicEnum extends Enumeration {

protected case class EmbeddingType(embeddingType: SimClustersEmbeddingType) extends super.Val

import scala.language.implicitConversions

implicit def valueToEmbeddingType(value: Value): EmbeddingType =

value.asInstanceOf[EmbeddingType]

val FavTfgTopic: Value = EmbeddingType(SimClustersEmbeddingType.FavTfgTopic)

val LogFavBasedKgoApeTopic: Value = EmbeddingType(

SimClustersEmbeddingType.LogFavBasedKgoApeTopic)

}

}