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

import com.twitter.bijection.scrooge.BinaryScalaCodec

import com.twitter.bijection.scrooge.CompactScalaCodec

import com.twitter.bijection.Bufferable

import com.twitter.bijection.Injection

import com.twitter.hermit.candidate.thriftscala.Candidates

import com.twitter.scalding.DateRange

import com.twitter.scalding.commons.source.VersionedKeyValSource

import com.twitter.scalding\_internal.source.lzo\_scrooge.DailySuffixMostRecentLzoScrooge

import com.twitter.scalding\_internal.source.lzo\_scrooge.FixedPathLzoScrooge

import com.twitter.scalding\_internal.source.lzo\_scrooge.HourlySuffixMostRecentLzoScrooge

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

case class EdgeWithDecayedWtsFixedPathSource(path: String)

extends FixedPathLzoScrooge[EdgeWithDecayedWeights](path, EdgeWithDecayedWeights)

case class UserAndNeighborsFixedPathSource(path: String)

extends FixedPathLzoScrooge[UserAndNeighbors](path, UserAndNeighbors)

case class NormsAndCountsFixedPathSource(path: String)

extends FixedPathLzoScrooge[NormsAndCounts](path, NormsAndCounts)

case class UserToInterestedInClustersFixedPathSource(path: String)

extends FixedPathLzoScrooge[UserToInterestedInClusters](path, UserToInterestedInClusters)

case class TimelineDataExtractorFixedPathSource(path: String)

extends FixedPathLzoScrooge[ReferenceTweets](path, ReferenceTweets)

case class TweetClusterScoresHourlySuffixSource(path: String, override val dateRange: DateRange)

extends HourlySuffixMostRecentLzoScrooge[TweetAndClusterScores](path, dateRange)

case class TweetTopKClustersHourlySuffixSource(path: String, override val dateRange: DateRange)

extends HourlySuffixMostRecentLzoScrooge[TweetTopKClustersWithScores](

path,

dateRange

)

case class ClusterTopKTweetsHourlySuffixSource(path: String, override val dateRange: DateRange)

extends HourlySuffixMostRecentLzoScrooge[ClusterTopKTweetsWithScores](

path,

dateRange

)

case class TweetSimilarityUnhydratedPairsSource(path: String, override val dateRange: DateRange)

extends DailySuffixMostRecentLzoScrooge[LabelledTweetPairs](

path,

dateRange

)

case class WTFCandidatesSource(path: String)

extends FixedPathLzoScrooge[Candidates](path, Candidates)

case class EmbeddingsLiteSource(path: String)

extends FixedPathLzoScrooge[EmbeddingsLite](path, EmbeddingsLite)

object AdhocKeyValSources {

def interestedInSource(path: String): VersionedKeyValSource[Long, ClustersUserIsInterestedIn] = {

implicit val keyInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val valInject: Injection[ClustersUserIsInterestedIn, Array[Byte]] =

CompactScalaCodec(ClustersUserIsInterestedIn)

VersionedKeyValSource[Long, ClustersUserIsInterestedIn](path)

}

def clusterDetailsSource(path: String): VersionedKeyValSource[(String, Int), ClusterDetails] = {

implicit val keyInject: Injection[(String, Int), Array[Byte]] =

Bufferable.injectionOf[(String, Int)]

implicit val valInject: Injection[ClusterDetails, Array[Byte]] =

CompactScalaCodec(ClusterDetails)

VersionedKeyValSource[(String, Int), ClusterDetails](path)

}

def bipartiteQualitySource(

path: String

): VersionedKeyValSource[(String, Int), BipartiteClusterQuality] = {

implicit val keyInject: Injection[(String, Int), Array[Byte]] =

Bufferable.injectionOf[(String, Int)]

implicit val valInject: Injection[BipartiteClusterQuality, Array[Byte]] =

CompactScalaCodec(BipartiteClusterQuality)

VersionedKeyValSource[(String, Int), BipartiteClusterQuality](path)

}

def entityToClustersSource(

path: String

): VersionedKeyValSource[SimClustersEmbeddingId, SimClustersEmbedding] = {

implicit val keyInject: Injection[SimClustersEmbeddingId, Array[Byte]] =

BinaryScalaCodec(SimClustersEmbeddingId)

implicit val valInject: Injection[SimClustersEmbedding, Array[Byte]] =

BinaryScalaCodec(SimClustersEmbedding)

VersionedKeyValSource[SimClustersEmbeddingId, SimClustersEmbedding](path)

}

def clusterToEntitiesSource(

path: String

): VersionedKeyValSource[SimClustersEmbeddingId, InternalIdEmbedding] = {

implicit val keyInject: Injection[SimClustersEmbeddingId, Array[Byte]] = BinaryScalaCodec(

SimClustersEmbeddingId)

implicit val valInject: Injection[InternalIdEmbedding, Array[Byte]] =

BinaryScalaCodec(InternalIdEmbedding)

VersionedKeyValSource[SimClustersEmbeddingId, InternalIdEmbedding](path)

}

// For storing producer-simclusters embeddings

def topProducerToClusterEmbeddingsSource(

path: String

): VersionedKeyValSource[Long, TopSimClustersWithScore] = {

implicit val keyInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val valInject: Injection[TopSimClustersWithScore, Array[Byte]] =

CompactScalaCodec(TopSimClustersWithScore)

VersionedKeyValSource[Long, TopSimClustersWithScore](path)

}

// For storing producer-simclusters embeddings

def topClusterEmbeddingsToProducerSource(

path: String

): VersionedKeyValSource[PersistedFullClusterId, TopProducersWithScore] = {

implicit val keyInject: Injection[PersistedFullClusterId, Array[Byte]] =

CompactScalaCodec(PersistedFullClusterId)

implicit val valInject: Injection[TopProducersWithScore, Array[Byte]] =

CompactScalaCodec(TopProducersWithScore)

VersionedKeyValSource[PersistedFullClusterId, TopProducersWithScore](path)

}

def userToInferredEntitiesSource(

path: String

): VersionedKeyValSource[Long, SimClustersInferredEntities] = {

implicit val keyInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val valInject: Injection[SimClustersInferredEntities, Array[Byte]] =

CompactScalaCodec(SimClustersInferredEntities)

VersionedKeyValSource[Long, SimClustersInferredEntities](path)

}

def knownForAdhocSource(path: String): VersionedKeyValSource[Long, ClustersUserIsKnownFor] = {

implicit val keyInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val valInject: Injection[ClustersUserIsKnownFor, Array[Byte]] =

CompactScalaCodec(ClustersUserIsKnownFor)

VersionedKeyValSource[Long, ClustersUserIsKnownFor](path)

}

def knownForSBFResultsDevelSource(

path: String

): VersionedKeyValSource[Long, Array[(Int, Float)]] = {

implicit val keyInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

implicit val valInject: Injection[Array[(Int, Float)], Array[Byte]] =

Bufferable.injectionOf[Array[(Int, Float)]]

VersionedKeyValSource[Long, Array[(Int, Float)]](path)

}

// injection to store adjlist in the mapped indices space for users

def intermediateSBFResultsDevelSource(

path: String

): VersionedKeyValSource[Int, List[(Int, Float)]] = {

implicit val keyInject: Injection[Int, Array[Byte]] = Injection.int2BigEndian

implicit val valInject: Injection[List[(Int, Float)], Array[Byte]] =

Bufferable.injectionOf[List[(Int, Float)]]

VersionedKeyValSource[Int, List[(Int, Float)]](path)

}

def mappedIndicesDevelSource(path: String): VersionedKeyValSource[Int, Long] = {

implicit val keyInject: Injection[Int, Array[Byte]] = Injection.int2BigEndian

implicit val valInject: Injection[Long, Array[Byte]] = Injection.long2BigEndian

VersionedKeyValSource[Int, Long](path)

}

}