package com.twitter.simclusters\_v2.scio

package multi\_type\_graph.multi\_type\_graph\_sims

import com.spotify.scio.ScioContext

import com.spotify.scio.coders.Coder

import com.spotify.scio.values.SCollection

import com.twitter.beam.io.dal.DAL

import com.twitter.beam.io.fs.multiformat.PathLayout

import com.twitter.beam.job.DateRangeOptions

import com.twitter.common.util.Clock

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

import com.twitter.dal.client.dataset.SnapshotDALDataset

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

import com.twitter.scio\_internal.coders.ThriftStructLazyBinaryScroogeCoder

import com.twitter.scio\_internal.job.ScioBeamJob

import com.twitter.scrooge.ThriftStruct

import com.twitter.simclusters\_v2.hdfs\_sources.RightNodeSimHashScioScalaDataset

import com.twitter.simclusters\_v2.scio.multi\_type\_graph.common.MultiTypeGraphUtil

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

import com.twitter.util.Duration

import com.twitter.wtf.dataflow.cosine\_similarity.ApproximateMatrixSelfTransposeMultiplicationJob

import java.time.Instant

trait RightNodeCosineSimilarityScioBaseApp

extends ScioBeamJob[DateRangeOptions]

with ApproximateMatrixSelfTransposeMultiplicationJob[RightNode] {

override implicit def scroogeCoder[T <: ThriftStruct: Manifest]: Coder[T] =

ThriftStructLazyBinaryScroogeCoder.scroogeCoder

override val ordering: Ordering[RightNode] = MultiTypeGraphUtil.rightNodeOrdering

val isAdhoc: Boolean

val cosineSimKeyValSnapshotDataset: KeyValDALDataset[KeyVal[RightNode, SimilarRightNodes]]

val rightNodeSimHashSnapshotDataset: SnapshotDALDataset[RightNodeSimHashSketch] =

RightNodeSimHashScioScalaDataset

val cosineSimJobOutputDirectory: String = Config.cosineSimJobOutputDirectory

override def graph(

implicit sc: ScioContext,

coder: Coder[RightNode]

): SCollection[(Long, RightNode, Double)] =

MultiTypeGraphUtil.getTruncatedMultiTypeGraph(noOlderThan = Duration.fromDays(14))

override def simHashSketches(

implicit sc: ScioContext,

coder: Coder[RightNode]

): SCollection[(RightNode, Array[Byte])] = {

sc.customInput(

"ReadSimHashSketches",

DAL

.readMostRecentSnapshotNoOlderThan(

rightNodeSimHashSnapshotDataset,

Duration.fromDays(14),

Clock.SYSTEM\_CLOCK,

DAL.Environment.Prod

)

).map { sketch =>

sketch.rightNode -> sketch.simHashOfEngagers.toArray

}

}

override def configurePipeline(

sc: ScioContext,

opts: DateRangeOptions

): Unit = {

implicit def scioContext: ScioContext = sc

// DAL.Environment variable for WriteExecs

val dalEnv = if (isAdhoc) DAL.Environment.Dev else DAL.Environment.Prod

val topKRightNodes: SCollection[(RightNode, SimilarRightNodes)] = topK

.collect {

case (rightNode, simRightNodes) =>

val sims = simRightNodes.collect {

case (simRightNode, score) => SimilarRightNode(simRightNode, score)

}

(rightNode, SimilarRightNodes(sims))

}

topKRightNodes

.map {

case (rightNode, sims) => KeyVal(rightNode, sims)

}.saveAsCustomOutput(

name = "WriteRightNodeCosineSimilarityDataset",

DAL.writeVersionedKeyVal(

cosineSimKeyValSnapshotDataset,

PathLayout.VersionedPath(prefix =

((if (!isAdhoc)

MultiTypeGraphUtil.RootMHPath

else

MultiTypeGraphUtil.AdhocRootPath)

+ Config.cosineSimJobOutputDirectory)),

instant = Instant.ofEpochMilli(opts.interval.getEndMillis - 1L),

environmentOverride = dalEnv,

)

)

}

}