package com.twitter.simclusters\_v2

package scio.bq\_generation.ftr\_tweet

import com.google.api.services.bigquery.model.TimePartitioning

import com.twitter.conversions.DurationOps.richDurationFromInt

import com.spotify.scio.ScioContext

import com.spotify.scio.coders.Coder

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

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

import com.twitter.simclusters\_v2.scio.bq\_generation.common.IndexGenerationUtil.parseClusterTopKTweetsFn

import java.time.Instant

import com.twitter.beam.job.DateRangeOptions

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

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.scio.bq\_generation.common.BQTableDetails

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

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

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

import com.twitter.tcdc.bqblaster.beam.syntax.\_

import com.twitter.tcdc.bqblaster.core.avro.TypedProjection

import com.twitter.tcdc.bqblaster.core.transform.RootTransform

import com.twitter.wtf.beam.bq\_embedding\_export.BQQueryUtils

import org.apache.beam.sdk.io.gcp.bigquery.BigQueryIO

trait FTRClusterToTweetIndexGenerationJob extends ScioBeamJob[DateRangeOptions] {

val isAdhoc: Boolean

val outputTable: BQTableDetails

val keyValDatasetOutputPath: String

val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

]

// Base configs

val projectId = "twttr-recos-ml-prod"

val environment: DAL.Env = if (isAdhoc) DAL.Environment.Dev else DAL.Environment.Prod

// Variables for Tweet Embedding SQL

val scoreKey: String

val scoreColumn: String

// Variables for spam treatment

val maxTweetFTR: Double

val maxUserFTR: Double

// Tweet embeddings parameters

val tweetEmbeddingsLength: Int = Config.SimClustersTweetEmbeddingsGenerationEmbeddingLength

// Clusters-to-tweet index parameters

val clusterTopKTweets: Int = Config.clusterTopKTweets

val maxTweetAgeHours: Int = Config.maxTweetAgeHours

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

ThriftStructLazyBinaryScroogeCoder.scroogeCoder

override def configurePipeline(sc: ScioContext, opts: DateRangeOptions): Unit = {

// The time when the job is scheduled

val queryTimestamp = opts.interval.getEnd

val tweetEmbeddingTemplateVariables =

Map(

"START\_TIME" -> queryTimestamp.minusDays(1).toString(),

"END\_TIME" -> queryTimestamp.toString(),

"TWEET\_SAMPLE\_RATE" -> Config.TweetSampleRate.toString,

"ENG\_SAMPLE\_RATE" -> Config.EngSampleRate.toString,

"MIN\_TWEET\_FAVS" -> Config.MinTweetFavs.toString,

"MIN\_TWEET\_IMPS" -> Config.MinTweetImps.toString,

"MAX\_TWEET\_FTR" -> maxTweetFTR.toString,

"MAX\_USER\_LOG\_N\_IMPS" -> Config.MaxUserLogNImps.toString,

"MAX\_USER\_LOG\_N\_FAVS" -> Config.MaxUserLogNFavs.toString,

"MAX\_USER\_FTR" -> maxUserFTR.toString,

"TWEET\_EMBEDDING\_LENGTH" -> Config.SimClustersTweetEmbeddingsGenerationEmbeddingLength.toString,

"HALFLIFE" -> Config.SimClustersTweetEmbeddingsGenerationHalfLife.toString,

"SCORE\_COLUMN" -> scoreColumn,

"SCORE\_KEY" -> scoreKey,

)

val tweetEmbeddingSql = BQQueryUtils.getBQQueryFromSqlFile(

"/com/twitter/simclusters\_v2/scio/bq\_generation/ftr\_tweet/sql/ftr\_tweet\_embeddings.sql",

tweetEmbeddingTemplateVariables)

val clusterTopTweetsTemplateVariables =

Map(

"CLUSTER\_TOP\_K\_TWEETS" -> Config.clusterTopKTweets.toString,

"TWEET\_EMBEDDING\_SQL" -> tweetEmbeddingSql

)

val clusterTopTweetsSql = BQQueryUtils.getBQQueryFromSqlFile(

"/com/twitter/simclusters\_v2/scio/bq\_generation/sql/cluster\_top\_tweets.sql",

clusterTopTweetsTemplateVariables

)

// Generate SimClusters cluster-to-tweet index

val topKtweetsForClusterKey = sc.customInput(

s"SimClusters cluster-to-tweet index generation BQ job",

BigQueryIO

.read(parseClusterTopKTweetsFn(Config.TweetEmbeddingHalfLife))

.fromQuery(clusterTopTweetsSql)

.usingStandardSql()

)

// Setup BQ writer

val ingestionTime = opts.getDate().value.getEnd.toDate

val bqFieldsTransform = RootTransform

.Builder()

.withPrependedFields("dateHour" -> TypedProjection.fromConstant(ingestionTime))

val timePartitioning = new TimePartitioning()

.setType("HOUR").setField("dateHour").setExpirationMs(3.days.inMilliseconds)

val bqWriter = BigQueryIO

.write[ClusterIdToTopKTweetsWithScores]

.to(outputTable.toString)

.withExtendedErrorInfo()

.withTimePartitioning(timePartitioning)

.withLoadJobProjectId(projectId)

.withThriftSupport(bqFieldsTransform.build(), AvroConverter.Legacy)

.withCreateDisposition(BigQueryIO.Write.CreateDisposition.CREATE\_IF\_NEEDED)

.withWriteDisposition(BigQueryIO.Write.WriteDisposition.WRITE\_APPEND)

// Save SimClusters index to a BQ table

topKtweetsForClusterKey

.map { clusterIdToTopKTweets =>

{

ClusterIdToTopKTweetsWithScores(

clusterId = clusterIdToTopKTweets.clusterId,

topKTweetsWithScores = clusterIdToTopKTweets.topKTweetsWithScores

)

}

}

.saveAsCustomOutput(s"WriteToBQTable - $outputTable", bqWriter)

// Save SimClusters index as a KeyValSnapshotDataset

topKtweetsForClusterKey

.map { clusterIdToTopKTweets =>

KeyVal(clusterIdToTopKTweets.clusterId, clusterIdToTopKTweets.topKTweetsWithScores)

}.saveAsCustomOutput(

name = s"WriteClusterToKeyIndexToKeyValDataset at $keyValDatasetOutputPath",

DAL.writeVersionedKeyVal(

clusterToTweetIndexSnapshotDataset,

PathLayout.VersionedPath(prefix =

((if (!isAdhoc)

Config.FTRRootMHPath

else

Config.FTRAdhocpath)

+ keyValDatasetOutputPath)),

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

environmentOverride = environment,

)

)

}

}

object FTRClusterToTweetIndexGenerationAdhoc extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = true

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simcluster\_adhoc\_test\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

"ftr\_tweets\_adhoc/ftr\_cluster\_to\_tweet\_adhoc"

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersFtrAdhocClusterToTweetIndexScalaDataset

override val scoreColumn = "ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1\_embedding"

override val scoreKey = "ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1"

override val maxUserFTR: Double = Config.MaxUserFTR

override val maxTweetFTR: Double = Config.MaxTweetFTR

}

object OONFTRClusterToTweetIndexGenerationAdhoc extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = true

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simcluster\_adhoc\_test\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

"oon\_ftr\_tweets\_adhoc/oon\_ftr\_cluster\_to\_tweet\_adhoc"

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersOonFtrAdhocClusterToTweetIndexScalaDataset

override val scoreColumn = "oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_embedding"

override val scoreKey = "oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay"

override val maxUserFTR: Double = Config.MaxUserFTR

override val maxTweetFTR: Double = Config.MaxTweetFTR

}

object FTRPop1000RankDecay11ClusterToTweetIndexGenerationBatch

extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = false

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_ftr\_pop1000\_rnkdecay11\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

Config.FTRPop1000RankDecay11ClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersFtrPop1000Rnkdecay11ClusterToTweetIndexScalaDataset

override val scoreColumn = "ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1\_embedding"

override val scoreKey = "ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_1\_1"

override val maxUserFTR: Double = Config.MaxUserFTR

override val maxTweetFTR: Double = Config.MaxTweetFTR

}

object FTRPop10000RankDecay11ClusterToTweetIndexGenerationBatch

extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = false

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_ftr\_pop10000\_rnkdecay11\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

Config.FTRPop10000RankDecay11ClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersFtrPop10000Rnkdecay11ClusterToTweetIndexScalaDataset

override val scoreColumn = "ftrat5\_decayed\_pop\_bias\_10000\_rank\_decay\_1\_1\_embedding"

override val scoreKey = "ftrat5\_decayed\_pop\_bias\_10000\_rank\_decay\_1\_1"

override val maxUserFTR: Double = Config.MaxUserFTR

override val maxTweetFTR: Double = Config.MaxTweetFTR

}

object OONFTRPop1000RankDecayClusterToTweetIndexGenerationBatch

extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = false

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_oon\_ftr\_pop1000\_rnkdecay\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

Config.OONFTRPop1000RankDecayClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersOonFtrPop1000RnkdecayClusterToTweetIndexScalaDataset

override val scoreColumn = "oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay\_embedding"

override val scoreKey = "oon\_ftrat5\_decayed\_pop\_bias\_1000\_rank\_decay"

override val maxUserFTR: Double = Config.MaxUserFTR

override val maxTweetFTR: Double = Config.MaxTweetFTR

}

object DecayedSumClusterToTweetIndexGenerationBatch extends FTRClusterToTweetIndexGenerationJob {

override val isAdhoc: Boolean = false

override val outputTable: BQTableDetails =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_decayed\_sum\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath: String =

Config.DecayedSumClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = SimclustersDecayedSumClusterToTweetIndexScalaDataset

override val scoreColumn = "dec\_sum\_logfavScoreClusterNormalizedOnly\_embedding"

override val scoreKey = "dec\_sum\_logfavScoreClusterNormalizedOnly"

override val maxUserFTR = 1.0

override val maxTweetFTR = 1.0

}