package com.twitter.simclusters\_v2.scio.bq\_generation

package simclusters\_index\_generation

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

import com.spotify.scio.ScioContext

import com.spotify.scio.coders.Coder

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

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

import com.twitter.beam.job.DateRangeOptions

import com.twitter.conversions.DurationOps.richDurationFromInt

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.hdfs\_sources.AdsFavBasedSimclustersClusterToTweetIndexScalaDataset

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

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

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

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

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

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

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

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

import com.twitter.simclusters\_v2.scio.bq\_generation.common.BQGenerationUtil.buildActionTypesEngagementIndicatorString

import com.twitter.simclusters\_v2.scio.bq\_generation.common.BQGenerationUtil.getInterestedIn2020SQL

import com.twitter.simclusters\_v2.scio.bq\_generation.common.BQTableDetails

import com.twitter.simclusters\_v2.scio.bq\_generation.simclusters\_index\_generation.Config.AdsClickEngagementTypeIds

import com.twitter.simclusters\_v2.scio.bq\_generation.simclusters\_index\_generation.Config.AdsFavEngagementTypeIds

import com.twitter.simclusters\_v2.scio.bq\_generation.simclusters\_index\_generation.EngagementEventBasedClusterToTweetIndexFromBQ.getTopKTweetsForClusterKeyBQ

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.unified\_user\_actions.thriftscala.ActionType

import java.time.Instant

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

import org.joda.time.DateTime

trait EngagementEventBasedClusterToTweetIndexGenerationJob extends ScioBeamJob[DateRangeOptions] {

// Configs to set for different type of embeddings and jobs

val isAdhoc: Boolean

val getConsumerEmbeddingsSQLFunc: (DateTime, Int) => String

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

// Point to different user tweet interaction table generation sql

// UUA-supported events: Config.unifiedUserTweetActionPairGenerationSQLPath

val userTweetEngagementEventPairSqlPath: String

lazy val userTweetEngagementEventPairTemplateVariable: Map[String, String] = Map.empty

// Enable Video-only filters and health filters (for VideoViewBased embeddings)

val enableHealthAndVideoFilters: Boolean = Config.enableHealthAndVideoFilters

val enableFavClusterTopKTweetsIntersection: Boolean =

Config.enableIntersectionWithFavBasedClusterTopKTweetsIndex

// Min fav/interaction threshold

val minInteractionCount: Int = Config.minInteractionCount

val minFavCount: Int = Config.minFavCount

// Tweet embeddings parameters

val tweetEmbeddingsLength: Int = Config.tweetEmbeddingsLength

val tweetEmbeddingsHalfLife: Int = Config.tweetEmbeddingsHalfLife

// Clusters-to-tweet index parameters

val clusterTopKTweets: Int = Config.clusterTopKTweets

val maxTweetAgeHours: Int = Config.maxTweetAgeHours

val minEngagementPerCluster: Int = Config.minEngagementPerCluster

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

// Read consumer embeddings SQL

val consumerEmbeddingsSQL = getConsumerEmbeddingsSQLFunc(queryTimestamp, 21)

// Generate SimClusters cluster-to-tweet index via BQ

val topKtweetsForClusterKey =

getTopKTweetsForClusterKeyBQ(

sc,

queryTimestamp,

maxTweetAgeHours,

consumerEmbeddingsSQL,

userTweetEngagementEventPairSqlPath,

userTweetEngagementEventPairTemplateVariable,

enableHealthAndVideoFilters,

enableFavClusterTopKTweetsIntersection,

minInteractionCount,

minFavCount,

tweetEmbeddingsLength,

tweetEmbeddingsHalfLife,

minEngagementPerCluster,

clusterTopKTweets

)

// 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.RootMHPath

else

Config.AdhocRootPath)

+ keyValDatasetOutputPath)),

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

environmentOverride = environment,

)

)

}

}

// This abstract class is used to define parameters specific to UUA events.

abstract class UUABasedClusterToTweetIndexGenerationJob

extends EngagementEventBasedClusterToTweetIndexGenerationJob {

// UUA Action types and column names

val contributingActionTypes: Seq[String]

val contributingActionReferenceTweetIdColumn: String = Config.actionTweetIdColumn

val undoActionTypes: Seq[String]

// Default undo tweet id is same as the actionTweetId (e.g. for favs these are the same tweet id)

val undoActionReferenceTweetIdColumn: String = Config.actionTweetIdColumn

// Get the string that represents the list of undo event ids

lazy val undoActionTypesStr: String = {

// Populate the action type list with a placeholder action if its empty

val actionTypes =

if (undoActionTypes.nonEmpty) undoActionTypes

else Seq(Config.PlaceholderActionType)

convertActionTypesSeqToString(actionTypes)

}

override lazy val userTweetEngagementEventPairTemplateVariable: Map[String, String] = {

Map(

"CONTRIBUTING\_ACTION\_TYPES\_STR" -> convertActionTypesSeqToString(contributingActionTypes),

"CONTRIBUTING\_ACTION\_TWEET\_ID\_COLUMN" -> contributingActionReferenceTweetIdColumn,

"UNDO\_ACTION\_TYPES\_STR" -> undoActionTypesStr,

"UNDO\_ACTION\_TWEET\_ID\_COLUMN" -> undoActionReferenceTweetIdColumn

)

}

/\*\*\*

\* Convert a list of actions to a string that could be easily used in SQLs

\* Example input: Seq("ServerTweetFav", "ClientTweetFav")

\* output: "ServerTweetFav","ClientTweetFav"

\* SQL use case: SELECT \* FROM table WHERE actionType IN ("ServerTweetFav","ClientTweetFav")

\*/

private def convertActionTypesSeqToString(actionTypes: Seq[String]): String = {

actionTypes.map(action => f"""\"${action}\"""").mkString(",")

}

}

abstract class AdsClusterToTweetIndexGenerationJob

extends EngagementEventBasedClusterToTweetIndexGenerationJob {

// Ads contributing action types - fav, click, etc

val contributingActionTypes: Seq[Int]

override lazy val userTweetEngagementEventPairTemplateVariable: Map[String, String] = {

Map(

"CONTRIBUTING\_ACTION\_TYPES\_STR" -> convertActionTypesSeqToString(contributingActionTypes)

)

}

private def convertActionTypesSeqToString(actionTypes: Seq[Int]): String = {

actionTypes.map(action => f"""${action}""").mkString(",")

}

}

object FavBasedClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val minInteractionCount: Int = 8

override val minFavCount: Int = 8

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_fav\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.FavBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedSimclustersClusterToTweetIndexScalaDataset

}

object FavBasedClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val minInteractionCount: Int = 8

override val minFavCount: Int = 8

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_fav\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.FavBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedSimclustersClusterToTweetIndexScalaDataset

}

object VideoViewBasedClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(

ActionType.ClientTweetVideoPlayback50.name)

override val undoActionTypes: Seq[String] = Seq.empty

override val enableHealthAndVideoFilters: Boolean = true

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_video\_view\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.VideoViewBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

VideoViewBasedSimclustersClusterToTweetIndexScalaDataset

}

object VideoViewBasedClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(

ActionType.ClientTweetVideoPlayback50.name)

override val undoActionTypes: Seq[String] = Seq.empty

override val enableHealthAndVideoFilters: Boolean = true

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_video\_view\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.VideoViewBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

VideoViewBasedSimclustersClusterToTweetIndexScalaDataset

}

object RetweetBasedClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetRetweet.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnretweet.name)

override val undoActionReferenceTweetIdColumn: String = Config.retweetTweetIdColumn

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_retweet\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.RetweetBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

RetweetBasedSimclustersClusterToTweetIndexScalaDataset

}

object RetweetBasedClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetRetweet.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnretweet.name)

override val undoActionReferenceTweetIdColumn: String = Config.retweetTweetIdColumn

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_retweet\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.RetweetBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

RetweetBasedSimclustersClusterToTweetIndexScalaDataset

}

object ReplyBasedClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.combinedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetReply.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetDelete.name)

override val undoActionReferenceTweetIdColumn: String = Config.replyTweetIdColumn

override val minInteractionCount: Int = 8

override val minFavCount: Int = 8

override val minEngagementPerCluster: Int = 3

// Add supplemental positive signals to the user tweet engagement event template

// We bundle each reply signal with a positive signal (fav or retweet)

val supplementalPositiveSignals: Seq[String] =

Seq(ActionType.ServerTweetFav.name, ActionType.ServerTweetRetweet.name)

override lazy val userTweetEngagementEventPairTemplateVariable: Map[String, String] = {

Map(

"CONTRIBUTING\_ACTION\_TYPE\_STR" -> contributingActionTypes.head,

"UNDO\_ACTION\_TYPES\_STR" -> undoActionTypesStr,

"UNDO\_ACTION\_TWEET\_ID\_COLUMN" -> undoActionReferenceTweetIdColumn,

"SUPPLEMENTAL\_ACTION\_TYPES\_ENGAGEMENT\_STR" -> buildActionTypesEngagementIndicatorString(

supplementalPositiveSignals)

)

}

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_reply\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.ReplyBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

ReplyBasedSimclustersClusterToTweetIndexScalaDataset

}

object ReplyBasedClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.combinedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetReply.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetDelete.name)

override val undoActionReferenceTweetIdColumn: String = Config.replyTweetIdColumn

override val minInteractionCount: Int = 8

override val minFavCount: Int = 8

override val minEngagementPerCluster: Int = 3

// Add supplemental positive signals to the user tweet engagement event template

// We bundle each reply signal with a positive signal (fav or retweet)

val supplementalPositiveSignals: Seq[String] =

Seq(ActionType.ServerTweetFav.name, ActionType.ServerTweetRetweet.name)

override lazy val userTweetEngagementEventPairTemplateVariable: Map[String, String] = {

Map(

"CONTRIBUTING\_ACTION\_TYPE\_STR" -> contributingActionTypes.head,

"UNDO\_ACTION\_TYPES\_STR" -> undoActionTypesStr,

"UNDO\_ACTION\_TWEET\_ID\_COLUMN" -> undoActionReferenceTweetIdColumn,

"SUPPLEMENTAL\_ACTION\_TYPES\_ENGAGEMENT\_STR" -> buildActionTypesEngagementIndicatorString(

supplementalPositiveSignals)

)

}

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_reply\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.ReplyBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

ReplyBasedSimclustersClusterToTweetIndexScalaDataset

}

object PushOpenBasedClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ClientNotificationOpen.name)

override val contributingActionReferenceTweetIdColumn: String = Config.pushTweetIdColumn

override val undoActionTypes: Seq[String] = Seq.empty

override val minInteractionCount = 1

override val minFavCount = 0

override val enableFavClusterTopKTweetsIntersection = true

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_push\_open\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.PushOpenBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

PushOpenBasedSimclustersClusterToTweetIndexScalaDataset

}

object PushOpenBasedClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.unifiedUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ClientNotificationOpen.name)

override val contributingActionReferenceTweetIdColumn: String = Config.pushTweetIdColumn

override val undoActionTypes: Seq[String] = Seq.empty

override val minInteractionCount = 1

override val minFavCount = 0

override val enableFavClusterTopKTweetsIntersection = true

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_push\_open\_based\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath = Config.PushOpenBasedClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

PushOpenBasedSimclustersClusterToTweetIndexScalaDataset

}

object AdsFavBasedClusterToTweetIndexGenerationAdhocJob

extends AdsClusterToTweetIndexGenerationJob {

val isAdhoc: Boolean = true

val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val contributingActionTypes: Seq[Int] = AdsFavEngagementTypeIds // fav

override val tweetEmbeddingsHalfLife: Int = 345600000 // 4 days

// The earliest user tweet engagement event we consider is 7 days ago

// The tweet could be older than 7 days

override val maxTweetAgeHours: Int = 168 // 7 days

override val minInteractionCount: Int = 3

override val minFavCount: Int = 3

override val minEngagementPerCluster: Int = 2

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_ads\_fav\_based\_cluster\_to\_tweet\_index")

val keyValDatasetOutputPath: String = Config.AdsFavBasedClusterToTweetIndexOutputPath

val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = AdsFavBasedSimclustersClusterToTweetIndexScalaDataset

val userTweetEngagementEventPairSqlPath: String =

Config.adsUserTweetActionPairGenerationSQLPath

}

object AdsFavBasedClusterToTweetIndexGenerationBatchJob

extends AdsClusterToTweetIndexGenerationJob {

val isAdhoc: Boolean = false

val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val contributingActionTypes: Seq[Int] = AdsFavEngagementTypeIds // fav

override val tweetEmbeddingsHalfLife: Int = 345600000 // 4 days

// The earliest user tweet engagement event we consider is 7 days ago

// The tweet could be older than 7 days

override val maxTweetAgeHours: Int = 168 // 7 days

override val minInteractionCount: Int = 3

override val minFavCount: Int = 3

override val minEngagementPerCluster: Int = 2

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_ads\_fav\_based\_cluster\_to\_tweet\_index")

val keyValDatasetOutputPath: String = Config.AdsFavBasedClusterToTweetIndexOutputPath

val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = AdsFavBasedSimclustersClusterToTweetIndexScalaDataset

val userTweetEngagementEventPairSqlPath: String =

Config.adsUserTweetActionPairGenerationSQLPath

}

object AdsFavClickBasedClusterToTweetIndexGenerationAdhocJob

extends AdsClusterToTweetIndexGenerationJob {

val isAdhoc: Boolean = true

val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val contributingActionTypes: Seq[Int] =

AdsFavEngagementTypeIds ++ AdsClickEngagementTypeIds // fav + click

override val tweetEmbeddingsHalfLife: Int = 604800000 // 7 days

// The earliest user tweet engagement event we consider is 21 days ago

// The tweet could be older than 21 days

override val maxTweetAgeHours: Int = 504 // 21 days

override val minInteractionCount: Int = 3

override val minFavCount: Int = 3

override val minEngagementPerCluster: Int = 2

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_ads\_fav\_click\_ sbased\_cluster\_to\_tweet\_index")

val keyValDatasetOutputPath: String = Config.AdsFavClickBasedClusterToTweetIndexOutputPath

val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = AdsFavClickBasedSimclustersClusterToTweetIndexScalaDataset

val userTweetEngagementEventPairSqlPath: String =

Config.adsUserTweetActionPairGenerationSQLPath

}

object AdsFavClickBasedClusterToTweetIndexGenerationBatchJob

extends AdsClusterToTweetIndexGenerationJob {

val isAdhoc: Boolean = false

val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val contributingActionTypes: Seq[Int] =

AdsFavEngagementTypeIds ++ AdsClickEngagementTypeIds // fav + click

override val tweetEmbeddingsHalfLife: Int = 604800000 // 7 days

// The earliest user tweet engagement event we consider is 21 days ago

// The tweet could be older than 21 days

override val maxTweetAgeHours: Int = 504 // 21 days

override val minInteractionCount: Int = 3

override val minFavCount: Int = 3

override val minEngagementPerCluster: Int = 2

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_ads\_fav\_click\_based\_cluster\_to\_tweet\_index")

val keyValDatasetOutputPath: String = Config.AdsFavClickBasedClusterToTweetIndexOutputPath

val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] = AdsFavClickBasedSimclustersClusterToTweetIndexScalaDataset

val userTweetEngagementEventPairSqlPath: String =

Config.adsUserTweetActionPairGenerationSQLPath

}

object FavBasedEvergreenContentClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.evergreenContentUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val tweetEmbeddingsHalfLife: Int = 57600000 // 16 hours

override val maxTweetAgeHours: Int = 48 // 2 days

override val minInteractionCount: Int = 8

override val minFavCount: Int = 0

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_fav\_based\_evergreen\_content\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath =

Config.FavBasedEvergreenContentClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedEvergreenContentSimclustersClusterToTweetIndexScalaDataset

}

object FavBasedEvergreenContentClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.evergreenContentUserTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val tweetEmbeddingsHalfLife: Int = 57600000 // 16 hours

override val maxTweetAgeHours: Int = 48 // 2 days

override val minInteractionCount: Int = 8

override val minFavCount: Int = 0

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_fav\_based\_evergreen\_content\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath =

Config.FavBasedEvergreenContentClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedEvergreenContentSimclustersClusterToTweetIndexScalaDataset

}

object FavBasedVideoClusterToTweetIndexGenerationAdhocJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = true

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.favBasedVideoTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val minInteractionCount: Int = 8

override val minFavCount: Int = 0

override val outputTable =

BQTableDetails(

"twttr-recos-ml-prod",

"simclusters",

"simclusters\_fav\_based\_video\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath =

Config.FavBasedVideoClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedVideoSimclustersClusterToTweetIndexScalaDataset

}

object FavBasedVideoClusterToTweetIndexGenerationBatchJob

extends UUABasedClusterToTweetIndexGenerationJob {

override val isAdhoc = false

override val getConsumerEmbeddingsSQLFunc = getInterestedIn2020SQL

override val userTweetEngagementEventPairSqlPath: String =

Config.favBasedVideoTweetActionPairGenerationSQLPath

override val contributingActionTypes: Seq[String] = Seq(ActionType.ServerTweetFav.name)

override val undoActionTypes: Seq[String] = Seq(ActionType.ServerTweetUnfav.name)

override val minInteractionCount: Int = 8

override val minFavCount: Int = 0

override val outputTable =

BQTableDetails(

"twttr-bq-cassowary-prod",

"user",

"simclusters\_fav\_based\_video\_cluster\_to\_tweet\_index")

override val keyValDatasetOutputPath =

Config.FavBasedVideoClusterToTweetIndexOutputPath

override val clusterToTweetIndexSnapshotDataset: KeyValDALDataset[

KeyVal[FullClusterId, TopKTweetsWithScores]

] =

FavBasedVideoSimclustersClusterToTweetIndexScalaDataset

}