package com.twitter.simclusters\_v2.scalding.optout

import com.twitter.dal.client.dataset.{KeyValDALDataset, SnapshotDALDataset}

import com.twitter.scalding.{

Args,

DateRange,

Days,

Duration,

Execution,

RichDate,

TypedPipe,

TypedTsv,

UniqueID

}

import com.twitter.scalding\_internal.dalv2.DALWrite.D

import com.twitter.scalding\_internal.dalv2.DALWrite.\_

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

import com.twitter.simclusters\_v2.common.{ClusterId, ModelVersions, SemanticCoreEntityId, UserId}

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

import com.twitter.simclusters\_v2.scalding.inferred\_entities.InferredEntities

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

ClusterType,

ClustersUserIsInterestedIn,

SemanticCoreEntityWithScore,

UserToInterestedInClusters

}

import com.twitter.wtf.scalding.jobs.common.{AdhocExecutionApp, ScheduledExecutionApp}

import com.twitter.simclusters\_v2.scalding.common.TypedRichPipe.\_

import com.twitter.simclusters\_v2.scalding.common.Util

import java.util.TimeZone

object InterestedInOptOut {

def filterOptedOutInterestedIn(

interestedInPipe: TypedPipe[(UserId, ClustersUserIsInterestedIn)],

optedOutEntities: TypedPipe[(UserId, Set[SemanticCoreEntityId])],

clusterToEntities: TypedPipe[(ClusterId, Seq[SemanticCoreEntityWithScore])]

): TypedPipe[(UserId, ClustersUserIsInterestedIn)] = {

val validInterestedIn = SimClustersOptOutUtil.filterOptedOutClusters(

userToClusters = interestedInPipe.mapValues(\_.clusterIdToScores.keySet.toSeq),

optedOutEntities = optedOutEntities,

legibleClusters = clusterToEntities

)

interestedInPipe

.leftJoin(validInterestedIn)

.mapValues {

case (originalInterestedIn, validInterestedInOpt) =>

val validInterestedIn = validInterestedInOpt.getOrElse(Seq()).toSet

originalInterestedIn.copy(

clusterIdToScores = originalInterestedIn.clusterIdToScores.filterKeys(validInterestedIn)

)

}

.filter(\_.\_2.clusterIdToScores.nonEmpty)

}

/\*\*

\* Writes InterestedIn data to HDFS

\*/

def writeInterestedInOutputExecution(

interestedIn: TypedPipe[(UserId, ClustersUserIsInterestedIn)],

interestedInDataset: KeyValDALDataset[KeyVal[Long, ClustersUserIsInterestedIn]],

outputPath: String

): Execution[Unit] = {

interestedIn

.map { case (k, v) => KeyVal(k, v) }

.writeDALVersionedKeyValExecution(

interestedInDataset,

D.Suffix(outputPath)

)

}

/\*\*

\* Convert InterestedIn to thrift structs, then write to HDFS

\*/

def writeInterestedInThriftOutputExecution(

interestedIn: TypedPipe[(UserId, ClustersUserIsInterestedIn)],

modelVersion: String,

interestedInThriftDatset: SnapshotDALDataset[UserToInterestedInClusters],

thriftOutputPath: String,

dateRange: DateRange

): Execution[Unit] = {

interestedIn

.map {

case (userId, clusters) =>

UserToInterestedInClusters(userId, modelVersion, clusters.clusterIdToScores)

}

.writeDALSnapshotExecution(

interestedInThriftDatset,

D.Daily,

D.Suffix(thriftOutputPath),

D.EBLzo(),

dateRange.end

)

}

}

/\*\*

capesospy-v2 update --build\_locally --start\_cron \

--start\_cron interested\_in\_optout\_daily \

src/scala/com/twitter/simclusters\_v2/capesos\_config/atla\_proc.yaml

\*/

object InterestedInOptOutDailyBatchJob extends ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2019-11-24")

override def batchIncrement: Duration = Days(1)

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val userOptoutEntities =

SimClustersOptOutUtil

.getP13nOptOutSources(dateRange.embiggen(Days(4)), ClusterType.InterestedIn)

.count("num\_users\_with\_optouts")

.forceToDisk

val interestedIn2020Pipe = InterestedInSources

.simClustersRawInterestedIn2020Source(dateRange, timeZone)

.count("num\_users\_with\_2020\_interestedin")

val interestedInLite2020Pipe = InterestedInSources

.simClustersRawInterestedInLite2020Source(dateRange, timeZone)

.count("num\_users\_with\_2020\_interestedin\_lite")

val clusterToEntities = InferredEntities

.getLegibleEntityEmbeddings(dateRange.prepend(Days(21)), timeZone)

.count("num\_cluster\_to\_entities")

val filtered2020InterestedIn = InterestedInOptOut

.filterOptedOutInterestedIn(interestedIn2020Pipe, userOptoutEntities, clusterToEntities)

.count("num\_users\_with\_compliant\_2020\_interestedin")

val write2020Exec = InterestedInOptOut.writeInterestedInOutputExecution(

filtered2020InterestedIn,

SimclustersV2InterestedIn20M145K2020ScalaDataset,

DataPaths.InterestedIn2020Path

)

val write2020ThriftExec = InterestedInOptOut.writeInterestedInThriftOutputExecution(

filtered2020InterestedIn,

ModelVersions.Model20M145K2020,

SimclustersV2UserToInterestedIn20M145K2020ScalaDataset,

DataPaths.InterestedIn2020ThriftPath,

dateRange

)

val sanityCheck2020Exec = SimClustersOptOutUtil.sanityCheckAndSendEmail(

oldNumClustersPerUser = interestedIn2020Pipe.map(\_.\_2.clusterIdToScores.size),

newNumClustersPerUser = filtered2020InterestedIn.map(\_.\_2.clusterIdToScores.size),

modelVersion = ModelVersions.Model20M145K2020,

alertEmail = SimClustersOptOutUtil.AlertEmail

)

val filtered2020InterestedInLite = InterestedInOptOut

.filterOptedOutInterestedIn(interestedInLite2020Pipe, userOptoutEntities, clusterToEntities)

.count("num\_users\_with\_compliant\_2020\_interestedin\_lite")

val write2020LiteExec = InterestedInOptOut.writeInterestedInOutputExecution(

filtered2020InterestedInLite,

SimclustersV2InterestedInLite20M145K2020ScalaDataset,

DataPaths.InterestedInLite2020Path

)

val write2020LiteThriftExec = InterestedInOptOut.writeInterestedInThriftOutputExecution(

filtered2020InterestedInLite,

ModelVersions.Model20M145K2020,

SimclustersV2UserToInterestedInLite20M145K2020ScalaDataset,

DataPaths.InterestedInLite2020ThriftPath,

dateRange

)

val sanityCheck2020LiteExec = SimClustersOptOutUtil.sanityCheckAndSendEmail(

oldNumClustersPerUser = interestedInLite2020Pipe.map(\_.\_2.clusterIdToScores.size),

newNumClustersPerUser = filtered2020InterestedInLite.map(\_.\_2.clusterIdToScores.size),

modelVersion = ModelVersions.Model20M145K2020,

alertEmail = SimClustersOptOutUtil.AlertEmail

)

Util.printCounters(

Execution.zip(

Execution.zip(

write2020Exec,

write2020ThriftExec,

sanityCheck2020Exec),

Execution.zip(

write2020LiteExec,

write2020LiteThriftExec,

sanityCheck2020LiteExec

)

)

)

}

}

/\*\*

\* For debugging only. Does a filtering run and prints the differences before/after the opt out

scalding remote run --target src/scala/com/twitter/simclusters\_v2/scalding/optout:interested\_in\_optout-adhoc \

--user cassowary --cluster bluebird-qus1 \

--main-class com.twitter.simclusters\_v2.scalding.optout.InterestedInOptOutAdhocJob -- \

--keytab /var/lib/tss/keys/fluffy/keytabs/client/cassowary.keytab \

--principal service\_acoount@TWITTER.BIZ \

-- \

--outputDir /user/cassowary/adhoc/interestedin\_optout \

--date 2020-09-03

\*/

object InterestedInOptOutAdhocJob extends AdhocExecutionApp {

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val outputDir = args("outputDir")

val interestedInPipe = InterestedInSources

.simClustersInterestedInUpdatedSource(dateRange, timeZone)

.count("num\_users\_with\_interestedin")

val userOptoutEntities: TypedPipe[(UserId, Set[SemanticCoreEntityId])] =

SimClustersOptOutUtil

.getP13nOptOutSources(dateRange.embiggen(Days(4)), ClusterType.InterestedIn)

.count("num\_users\_with\_optouts")

val clusterToEntities = InferredEntities

.getLegibleEntityEmbeddings(dateRange, timeZone)

.count("num\_cluster\_to\_entities")

val filteredInterestedInPipe = InterestedInOptOut

.filterOptedOutInterestedIn(

interestedInPipe,

userOptoutEntities,

clusterToEntities

)

.count("num\_users\_with\_interestedin\_after\_optout")

val output = interestedInPipe

.join(filteredInterestedInPipe)

.filter {

case (userId, (originalInterestedIn, filtered)) =>

originalInterestedIn.clusterIdToScores != filtered.clusterIdToScores

}

.join(userOptoutEntities)

.map {

case (userId, ((originalInterestedIn, filtered), optoutEntities)) =>

Seq(

"userId=" + userId,

"originalInterestedInVersion=" + originalInterestedIn.knownForModelVersion,

"originalInterestedIn=" + originalInterestedIn.clusterIdToScores.keySet,

"filteredInterestedIn=" + filtered.knownForModelVersion,

"filteredInterestedIn=" + filtered.clusterIdToScores.keySet,

"optoutEntities=" + optoutEntities

).mkString("\t")

}

Util.printCounters(

output.writeExecution(TypedTsv(outputDir))

)

}

}