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

import com.twitter.scalding.Args

import com.twitter.scalding.DateRange

import com.twitter.scalding.Days

import com.twitter.scalding.Duration

import com.twitter.scalding.Execution

import com.twitter.scalding.RichDate

import com.twitter.scalding.TypedPipe

import com.twitter.scalding.TypedTsv

import com.twitter.scalding.UniqueID

import com.twitter.simclusters\_v2.common.ClusterId

import com.twitter.simclusters\_v2.common.SemanticCoreEntityId

import com.twitter.simclusters\_v2.common.UserId

import com.twitter.scalding\_internal.dalv2.DAL

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

import com.twitter.scalding\_internal.dalv2.remote\_access.AllowCrossClusterSameDC

import com.twitter.scalding\_internal.dalv2.remote\_access.ExplicitLocation

import com.twitter.scalding\_internal.dalv2.remote\_access.ProcAtla

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

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

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

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

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

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

import com.twitter.wtf.scalding.jobs.common.AdhocExecutionApp

import com.twitter.wtf.scalding.jobs.common.ScheduledExecutionApp

import java.util.TimeZone

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

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

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

/\*\*

\* Creates opt-out compliant KnownFor datasets based on plain user -> KnownFor data and users'

\* opt-out selections from YourTwitterData. In essence, we remove any cluster whose inferred

\* entities were opted out by the user.

\* The opted out KnownFor dataset should be the default dataset to be consumed, instead of the

\* plain KnownFor, which is not opt-out compliant.

\*/

object KnownForOptOut {

def filterOptedOutKnownFor(

knownForPipe: TypedPipe[(UserId, ClustersUserIsKnownFor)],

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

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

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

val validKnownFor = SimClustersOptOutUtil.filterOptedOutClusters(

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

optedOutEntities = optedOutEntities,

legibleClusters = clusterToEntities

)

knownForPipe

.leftJoin(validKnownFor)

.mapValues {

case (originalKnownFors, validKnownForOpt) =>

val validKnownFor = validKnownForOpt.getOrElse(Seq()).toSet

originalKnownFors.copy(

clusterIdToScores = originalKnownFors.clusterIdToScores.filterKeys(validKnownFor)

)

}

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

}

}

/\*\*

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

--start\_cron known\_for\_optout\_daily \

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

\*/

object KnownForOptOutDailyBatchJob extends ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2021-03-29")

override def batchIncrement: Duration = Days(1)

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val optedOutEntitiesPipe = SimClustersOptOutUtil

.getP13nOptOutSources(dateRange.embiggen(Days(2)), ClusterType.KnownFor)

.forceToDisk

val clusterToEntitiesPipe = InferredEntities.getLegibleEntityEmbeddings(dateRange, timeZone)

val knownFor2020 = DAL

.readMostRecentSnapshot(

SimclustersV2RawKnownFor20M145K2020ScalaDataset,

dateRange.embiggen(Days(10)))

.withRemoteReadPolicy(AllowCrossClusterSameDC)

.toTypedPipe

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

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

val filtered2020KnownForExec = {

val filtered2020KnownForData = KnownForOptOut

.filterOptedOutKnownFor(

knownForPipe = knownFor2020,

optedOutEntities = optedOutEntitiesPipe,

clusterToEntities = clusterToEntitiesPipe

)

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

.forceToDisk

Execution

.zip(

filtered2020KnownForData

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

.writeDALVersionedKeyValExecution(

SimclustersV2KnownFor20M145K2020ScalaDataset,

D.Suffix(DataPaths.KnownFor2020Path)

),

filtered2020KnownForData

.map {

case (userId, ClustersUserIsKnownFor(modelVersion, clusters)) =>

UserToKnownForClusters(userId, modelVersion, clusters)

}

.writeDALSnapshotExecution(

dataset = SimclustersV2KnownFor20M145K2020ThriftScalaDataset,

updateStep = D.Daily,

pathLayout = D.Suffix(DataPaths.KnownFor2020ThriftDatasetPath),

fmt = D.Parquet,

endDate = dateRange.end

)

).unit

}

Util.printCounters(filtered2020KnownForExec)

}

}

/\*\*

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

./bazel bundle src/scala/com/twitter/simclusters\_v2/scalding/optout:knownfor\_optout-adhoc && \

oscar hdfs --user recos-platform --screen --tee your\_ldap \

--bundle knownfor\_optout-adhoc \

--tool com.twitter.simclusters\_v2.scalding.optout.KnownForOptOutAdhocJob \

-- --date 2019-10-12

\*/

object KnownForOptOutAdhocJob extends AdhocExecutionApp {

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

val knownForPipe = DAL

.readMostRecentSnapshotNoOlderThan(SimclustersV2RawKnownFor20M145KDec11ScalaDataset, Days(30))

.withRemoteReadPolicy(ExplicitLocation(ProcAtla))

.toTypedPipe

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

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

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

SimClustersOptOutUtil

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

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

val clusterToEntities = InferredEntities

.getLegibleEntityEmbeddings(dateRange, timeZone)

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

val filteredKnownForPipe = KnownForOptOut.filterOptedOutKnownFor(

knownForPipe,

userOptoutEntities,

clusterToEntities

)

val output = knownForPipe

.join(filteredKnownForPipe)

.collect {

case (userId, (originalKnownFor, filtered))

if originalKnownFor.clusterIdToScores != filtered.clusterIdToScores =>

(userId, (originalKnownFor, filtered))

}

.join(userOptoutEntities)

.map {

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

Seq(

"userId=" + userId,

"originalKnownFor=" + originalKnownFor,

"filteredKnownFor=" + filtered,

"optoutEntities=" + optoutEntities

).mkString("\t")

}

val outputPath = "/user/recos-platform/adhoc/knownfor\_optout"

output.writeExecution(TypedTsv(outputPath))

}

}