package com.twitter.simclusters\_v2.scalding.embedding.abuse

import com.twitter.scalding.\_

import com.twitter.scalding.source.TypedText

import com.twitter.scalding\_internal.dalv2.DALWrite.{D, \_}

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

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

import com.twitter.simclusters\_v2.scalding.common.matrix.SparseMatrix

import com.twitter.simclusters\_v2.scalding.embedding.abuse.AbuseSimclusterFeaturesScaldingJob.buildKeyValDataSet

import com.twitter.simclusters\_v2.scalding.embedding.abuse.AdhocAbuseSimClusterFeaturesScaldingJob.{

abuseInteractionSearchGraph,

buildSearchAbuseScores,

impressionInteractionSearchGraph

}

import com.twitter.simclusters\_v2.scalding.embedding.abuse.DataSources.getUserInterestedInSparseMatrix

import com.twitter.simclusters\_v2.scalding.embedding.common.EmbeddingUtil

import com.twitter.simclusters\_v2.scalding.embedding.common.EmbeddingUtil.{ClusterId, UserId}

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

ModelVersion,

SimClustersEmbedding,

SingleSideUserScores

}

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

import java.util.TimeZone

object AbuseSimclusterFeaturesScaldingJob {

val HealthyConsumerKey = "healthyConsumer"

val UnhealthyConsumerKey = "unhealthyConsumer"

val HealthyAuthorKey = "healthyAuthor"

val UnhealthyAuthorKey = "unhealthyAuthor"

private[this] val EmptySimCluster = SimClustersEmbedding(List())

def buildKeyValDataSet(

normalizedSimClusterMatrix: SparseMatrix[UserId, ClusterId, Double],

unhealthyGraph: SparseMatrix[UserId, UserId, Double],

healthyGraph: SparseMatrix[UserId, UserId, Double]

): TypedPipe[KeyVal[Long, SingleSideUserScores]] = {

val searchAbuseScores =

buildSearchAbuseScores(

normalizedSimClusterMatrix,

unhealthyGraph = unhealthyGraph,

healthyGraph = healthyGraph

)

val pairedScores = SingleSideInteractionTransformation.pairScores(

Map(

HealthyConsumerKey -> searchAbuseScores.healthyConsumerClusterScores,

UnhealthyConsumerKey -> searchAbuseScores.unhealthyConsumerClusterScores,

HealthyAuthorKey -> searchAbuseScores.healthyAuthorClusterScores,

UnhealthyAuthorKey -> searchAbuseScores.unhealthyAuthorClusterScores

)

)

pairedScores

.map { pairedScore =>

val userPairInteractionFeatures = PairedInteractionFeatures(

healthyInteractionSimClusterEmbedding =

pairedScore.interactionScores.getOrElse(HealthyConsumerKey, EmptySimCluster),

unhealthyInteractionSimClusterEmbedding =

pairedScore.interactionScores.getOrElse(UnhealthyConsumerKey, EmptySimCluster)

)

val authorPairInteractionFeatures = PairedInteractionFeatures(

healthyInteractionSimClusterEmbedding =

pairedScore.interactionScores.getOrElse(HealthyAuthorKey, EmptySimCluster),

unhealthyInteractionSimClusterEmbedding =

pairedScore.interactionScores.getOrElse(UnhealthyAuthorKey, EmptySimCluster)

)

val value = SingleSideUserScores(

pairedScore.userId,

consumerHealthyScore = userPairInteractionFeatures.healthySum,

consumerUnhealthyScore = userPairInteractionFeatures.unhealthySum,

authorUnhealthyScore = authorPairInteractionFeatures.unhealthySum,

authorHealthyScore = authorPairInteractionFeatures.healthySum

)

KeyVal(pairedScore.userId, value)

}

}

}

/\*\*

\* This job creates single-side features used to predict the abuse reports in search. The features

\* are put into manhattan and availabe in feature store. We expect that search will be able to use

\* these features directly. They may be useful for other models as well.

\*/

object SearchAbuseSimclusterFeaturesScaldingJob extends ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2021-02-01")

override def batchIncrement: Duration =

Days(7)

private val OutputPath: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = true,

modelVersion = ModelVersion.Model20m145kUpdated,

pathSuffix = "search\_abuse\_simcluster\_features"

)

def buildDataset(

)(

implicit dateRange: DateRange,

): Execution[TypedPipe[KeyVal[Long, SingleSideUserScores]]] = {

Execution.getMode.map { implicit mode =>

val normalizedSimClusterMatrix = getUserInterestedInSparseMatrix.rowL2Normalize

val abuseSearchGraph = abuseInteractionSearchGraph()(dateRange, mode)

val impressionSearchGraph = impressionInteractionSearchGraph()(dateRange, mode)

buildKeyValDataSet(normalizedSimClusterMatrix, abuseSearchGraph, impressionSearchGraph)

}

}

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

// Extend the date range to a total of 19 days. Search keeps 21 days of data.

val dateRangeSearchData = dateRange.prepend(Days(12))

buildDataset()(dateRangeSearchData).flatMap { dataset =>

dataset.writeDALVersionedKeyValExecution(

dataset = SearchAbuseSimclusterFeaturesManhattanScalaDataset,

pathLayout = D.Suffix(OutputPath)

)

}

}

}

/\*\*

\* You can check the logic of this job by running this query.

\*

\* scalding remote run \

\* --target src/scala/com/twitter/simclusters\_v2/scalding/embedding/abuse:abuse-prod \

\* --main-class com.twitter.simclusters\_v2.scalding.embedding.abuse.AdhocSearchAbuseSimclusterFeaturesScaldingJob \

\* --hadoop-properties "mapreduce.job.split.metainfo.maxsize=-1" \

\* --cluster bluebird-qus1 --submitter hadoopnest-bluebird-1.qus1.twitter.com \

\* -- --date 2021-02-01 2021-02-02 \

\* --outputPath AdhocSearchAbuseSimclusterFeaturesScaldingJob-test1

\*/

object AdhocSearchAbuseSimclusterFeaturesScaldingJob extends AdhocExecutionApp {

def toTsv(

datasetExecution: Execution[TypedPipe[KeyVal[Long, SingleSideUserScores]]],

outputPath: String

): Execution[Unit] = {

datasetExecution.flatMap { dataset =>

dataset

.map { keyVal =>

(

keyVal.key,

keyVal.value.consumerHealthyScore,

keyVal.value.consumerUnhealthyScore,

keyVal.value.authorHealthyScore,

keyVal.value.authorUnhealthyScore

)

}

.writeExecution(TypedText.tsv(outputPath))

}

}

override def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

toTsv(

SearchAbuseSimclusterFeaturesScaldingJob.buildDataset()(dateRange),

args("outputPath")

)

}

}