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

import com.twitter.scalding.\_

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

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

import com.twitter.scalding\_internal.source.lzo\_scrooge.FixedPathLzoScrooge

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

AggregatableProducerSimclustersEmbeddingsByLogFavScoreScalaDataset,

AggregatableProducerSimclustersEmbeddingsByLogFavScoreThriftScalaDataset,

AggregatableProducerSimclustersEmbeddingsByLogFavScore2020ScalaDataset,

AggregatableProducerSimclustersEmbeddingsByLogFavScore2020ThriftScalaDataset,

AggregatableProducerSimclustersEmbeddingsByLogFavScoreRelaxedFavEngagementThreshold2020ScalaDataset,

AggregatableProducerSimclustersEmbeddingsByLogFavScoreRelaxedFavEngagementThreshold2020ThriftScalaDataset

}

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

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

EmbeddingType,

ModelVersion,

NeighborWithWeights,

SimClustersEmbedding,

SimClustersEmbeddingId,

SimClustersEmbeddingWithId,

UserToInterestedInClusterScores

}

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

import java.util.TimeZone

/\*\*

\* This file implements a new Producer SimClusters Embeddings.

\* The differences with existing producer embeddings are:

\*

\* 1) the embedding scores are not normalized, so that one can aggregate multiple producer embeddings by adding them.

\* 2) we use log-fav scores in the user-producer graph and user-simclusters graph.

\* LogFav scores are smoother than fav scores we previously used and they are less sensitive to outliers

\*

\*

\*

\* The main difference with other normalized embeddings is the `convertEmbeddingToAggregatableEmbeddings` function

\* where we multiply the normalized embedding with producer's norms. The resulted embeddings are then

\* unnormalized and aggregatable.

\*

\*/

/\*\*

\* Production job:

capesospy-v2 update aggregatable\_producer\_embeddings\_by\_logfav\_score src/scala/com/twitter/simclusters\_v2/capesos\_config/atla\_proc3.yaml

\*/

object AggregatableLogFavBasedProducerEmbeddingsScheduledApp

extends AggregatableLogFavBasedProducerEmbeddingsBaseApp

with ScheduledExecutionApp {

override val modelVersion: ModelVersion = ModelVersion.Model20m145kUpdated

// Not using the EmbeddingUtil.getHdfsPath to preserve the previous functionality.

private val outputPath: String =

"/user/cassowary/manhattan\_sequence\_files/producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score"

private val outputPathThrift: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = false,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score\_thrift"

)

override def batchIncrement: Duration = Days(7)

override def firstTime: RichDate = RichDate("2020-04-05")

override def writeToManhattan(

output: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALVersionedKeyValExecution(

AggregatableProducerSimclustersEmbeddingsByLogFavScoreScalaDataset,

D.Suffix(outputPath),

version = ExplicitEndTime(dateRange.end)

)

}

override def writeToThrift(

output: TypedPipe[SimClustersEmbeddingWithId]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALSnapshotExecution(

dataset = AggregatableProducerSimclustersEmbeddingsByLogFavScoreThriftScalaDataset,

updateStep = D.Daily,

pathLayout = D.Suffix(outputPathThrift),

fmt = D.Parquet,

endDate = dateRange.end

)

}

}

/\*\*

\* Production job:

capesospy-v2 update --build\_locally --start\_cron aggregatable\_producer\_embeddings\_by\_logfav\_score\_2020 src/scala/com/twitter/simclusters\_v2/capesos\_config/atla\_proc3.yaml

\*/

object AggregatableLogFavBasedProducerEmbeddings2020ScheduledApp

extends AggregatableLogFavBasedProducerEmbeddingsBaseApp

with ScheduledExecutionApp {

override val modelVersion: ModelVersion = ModelVersion.Model20m145k2020

// Not using the EmbeddingUtil.getHdfsPath to preserve the previous functionality.

private val outputPath: String =

"/user/cassowary/manhattan\_sequence\_files/producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score\_20m145k2020"

private val outputPathThrift: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = false,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score\_thrift"

)

override def batchIncrement: Duration = Days(7)

override def firstTime: RichDate = RichDate("2021-03-05")

override def writeToManhattan(

output: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALVersionedKeyValExecution(

AggregatableProducerSimclustersEmbeddingsByLogFavScore2020ScalaDataset,

D.Suffix(outputPath),

version = ExplicitEndTime(dateRange.end)

)

}

override def writeToThrift(

output: TypedPipe[SimClustersEmbeddingWithId]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALSnapshotExecution(

dataset = AggregatableProducerSimclustersEmbeddingsByLogFavScore2020ThriftScalaDataset,

updateStep = D.Daily,

pathLayout = D.Suffix(outputPathThrift),

fmt = D.Parquet,

endDate = dateRange.end

)

}

}

/\*\*

\* Production job:

capesospy-v2 update --build\_locally --start\_cron aggregatable\_producer\_embeddings\_by\_logfav\_score\_relaxed\_fav\_engagement\_threshold\_2020 src/scala/com/twitter/simclusters\_v2/capesos\_config/atla\_proc3.yaml

\*/

object AggregatableLogFavBasedProducerEmbeddingsRelaxedFavEngagementThreshold2020ScheduledApp

extends AggregatableLogFavBasedProducerEmbeddingsBaseApp

with ScheduledExecutionApp {

override val modelVersion: ModelVersion = ModelVersion.Model20m145k2020

override val embeddingType: EmbeddingType = EmbeddingType.RelaxedAggregatableLogFavBasedProducer

// Relax fav engagement threshold

override val minNumFavers = 15

// Not using the EmbeddingUtil.getHdfsPath to preserve the previous functionality.

private val outputPath: String =

"/user/cassowary/manhattan\_sequence\_files/producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score\_relaxed\_fav\_engagement\_threshold\_20m145k2020"

private val outputPathThrift: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = false,

modelVersion = modelVersion,

pathSuffix =

"producer\_simclusters\_aggregatable\_embeddings\_by\_logfav\_score\_relaxed\_fav\_score\_threshold\_thrift"

)

override def batchIncrement: Duration = Days(7)

override def firstTime: RichDate = RichDate("2021-07-26")

override def writeToManhattan(

output: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALVersionedKeyValExecution(

AggregatableProducerSimclustersEmbeddingsByLogFavScoreRelaxedFavEngagementThreshold2020ScalaDataset,

D.Suffix(outputPath),

version = ExplicitEndTime(dateRange.end)

)

}

override def writeToThrift(

output: TypedPipe[SimClustersEmbeddingWithId]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeDALSnapshotExecution(

dataset =

AggregatableProducerSimclustersEmbeddingsByLogFavScoreRelaxedFavEngagementThreshold2020ThriftScalaDataset,

updateStep = D.Daily,

pathLayout = D.Suffix(outputPathThrift),

fmt = D.Parquet,

endDate = dateRange.end

)

}

}

/\*\*\*

\* Adhoc job:

scalding remote run --user recos-platform \

--main-class com.twitter.simclusters\_v2.scalding.embedding.producer.AggregatableLogFavBasedProducerEmbeddingsAdhocApp \

--target src/scala/com/twitter/simclusters\_v2/scalding/embedding/producer:aggregatable\_logfav\_based\_producer\_embeddings\_job-adhoc \

-- --date 2020-04-08

\*/

object AggregatableLogFavBasedProducerEmbeddingsAdhocApp

extends AggregatableLogFavBasedProducerEmbeddingsBaseApp

with AdhocExecutionApp {

override val modelVersion: ModelVersion = ModelVersion.Model20m145kUpdated

private val outputPath: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = true,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_log\_fav\_score"

)

private val outputPathThrift: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = false,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_log\_fav\_score\_thrift"

)

override def writeToManhattan(

output: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.flatMap { keyVal =>

keyVal.value.embedding.map { simClusterWithScore =>

(

keyVal.key.embeddingType,

keyVal.key.modelVersion,

keyVal.key.internalId,

simClusterWithScore.clusterId,

simClusterWithScore.score

)

}

}

.writeExecution(

// Write to TSV for easier debugging of the adhoc job.

TypedTsv(outputPath)

)

}

override def writeToThrift(

output: TypedPipe[SimClustersEmbeddingWithId]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeExecution(

new FixedPathLzoScrooge(outputPathThrift, SimClustersEmbeddingWithId)

)

}

}

/\*\*

./bazel bundle src/scala/com/twitter/simclusters\_v2/scalding/embedding/producer:aggregatable\_logfav\_based\_producer\_embeddings\_job\_2020-adhoc

scalding remote run \

--user cassowary \

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

--principal service\_acoount@TWITTER.BIZ \

--cluster bluebird-qus1 \

--main-class com.twitter.simclusters\_v2.scalding.embedding.producer.AggregatableLogFavBasedProducerEmbeddings2020AdhocApp \

--target src/scala/com/twitter/simclusters\_v2/scalding/embedding/producer:aggregatable\_logfav\_based\_producer\_embeddings\_job\_2020-adhoc \

--hadoop-properties "scalding.with.reducers.set.explicitly=true mapreduce.job.reduces=4000" \

-- --date 2020-06-28

\*/

object AggregatableLogFavBasedProducerEmbeddings2020AdhocApp

extends AggregatableLogFavBasedProducerEmbeddingsBaseApp

with AdhocExecutionApp {

override val modelVersion: ModelVersion = ModelVersion.Model20m145k2020

private val outputPath: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = true,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_log\_fav\_score"

)

private val outputPathThrift: String = EmbeddingUtil.getHdfsPath(

isAdhoc = false,

isManhattanKeyVal = false,

modelVersion = modelVersion,

pathSuffix = "producer\_simclusters\_aggregatable\_embeddings\_by\_log\_fav\_score\_thrift"

)

override def writeToManhattan(

output: TypedPipe[KeyVal[SimClustersEmbeddingId, SimClustersEmbedding]]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.flatMap { keyVal =>

keyVal.value.embedding.map { simClusterWithScore =>

(

keyVal.key.embeddingType,

keyVal.key.modelVersion,

keyVal.key.internalId,

simClusterWithScore.clusterId,

simClusterWithScore.score

)

}

}

.writeExecution(

// Write to TSV for easier debugging of the adhoc job.

TypedTsv(outputPath)

)

}

override def writeToThrift(

output: TypedPipe[SimClustersEmbeddingWithId]

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueID: UniqueID

): Execution[Unit] = {

output

.writeExecution(

new FixedPathLzoScrooge(outputPathThrift, SimClustersEmbeddingWithId)

)

}

}

trait AggregatableLogFavBasedProducerEmbeddingsBaseApp

extends AggregatableProducerEmbeddingsBaseApp {

override val userToProducerScoringFn: NeighborWithWeights => Double = \_.logFavScore.getOrElse(0.0)

override val userToClusterScoringFn: UserToInterestedInClusterScores => Double =

\_.logFavScore.getOrElse(0.0)

override val embeddingType: EmbeddingType = EmbeddingType.AggregatableLogFavBasedProducer

}