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

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.UniqueID

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

import com.twitter.simclusters\_v2.common.clustering.ConnectedComponentsClusteringMethod

import com.twitter.simclusters\_v2.common.clustering.LargestDimensionClusteringMethod

import com.twitter.simclusters\_v2.common.clustering.LouvainClusteringMethod

import com.twitter.simclusters\_v2.common.clustering.MedoidRepresentativeSelectionMethod

import com.twitter.simclusters\_v2.common.clustering.MaxFavScoreRepresentativeSelectionMethod

import com.twitter.simclusters\_v2.common.clustering.SimilarityFunctions

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

import com.twitter.simclusters\_v2.hdfs\_sources.ClustersMembersLargestDimApeSimilarity2DayUpdateScalaDataset

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

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

import com.twitter.simclusters\_v2.hdfs\_sources.InterestedInTwiceByLargestDim2DayUpdateScalaDataset

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

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

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

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

import com.twitter.simclusters\_v2.scalding.embedding.twice.InterestedInTwiceBaseApp.ProducerEmbeddingSource

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

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

import java.util.TimeZone

/\*\*

To build & deploy the TWICE scheduled jobs via workflows:

scalding workflow upload \

--workflow interested\_in\_twice-batch \

--jobs src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_largest\_dim-batch,src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_louvain-batch,src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_connected\_components-batch \

--scm-paths "src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice/\*" \

--autoplay \

-> See workflow here: https://workflows.twitter.biz/workflow/cassowary/interested\_in\_twice-batch

(Use `scalding workflow upload --help` for a breakdown of the different flags)

\*/\*/

object InterestedInTwiceLargestDimScheduledApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2021-09-02")

override def batchIncrement: Duration = Days(7)

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersMatchingLargestDimension

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runScheduledApp(

new LargestDimensionClusteringMethod(),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_by\_largest\_dim",

"clusters\_members\_largest\_dim\_ape\_similarity",

InterestedInTwiceByLargestDimScalaDataset,

ClustersMembersLargestDimApeSimilarityScalaDataset,

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceLargestDimMaxFavScoreScheduledApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2022-06-30")

override def batchIncrement: Duration = Days(7)

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersMatchingLargestDimension

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runScheduledApp(

new LargestDimensionClusteringMethod(),

new MaxFavScoreRepresentativeSelectionMethod[SimClustersEmbedding](),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_by\_largest\_dim\_fav\_score",

"clusters\_members\_largest\_dim\_ape\_similarity",

InterestedInTwiceByLargestDimFavScoreScalaDataset,

ClustersMembersLargestDimApeSimilarityScalaDataset,

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceLouvainScheduledApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2021-09-02")

override def batchIncrement: Duration = Days(7)

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runScheduledApp(

new LouvainClusteringMethod(

args.required("cosine\_similarity\_threshold").toDouble,

args.optional("resolution\_factor").map(\_.toDouble)),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_louvain",

"clusters\_members\_louvain\_ape\_similarity",

InterestedInTwiceLouvainScalaDataset,

ClustersMembersLouvainApeSimilarityScalaDataset,

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceConnectedComponentsScheduledApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2021-09-02")

override def batchIncrement: Duration = Days(7)

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runScheduledApp(

new ConnectedComponentsClusteringMethod(

args.required("cosine\_similarity\_threshold").toDouble),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_connected\_components",

"clusters\_members\_connected\_components\_ape\_similarity",

InterestedInTwiceConnectedComponentsScalaDataset,

ClustersMembersConnectedComponentsApeSimilarityScalaDataset,

args.getOrElse("num-reducers", "4000").toInt

)

}

}

/\*\* Production Scalding job that calculates TWICE embeddings in a shorter period (every two days).

\*

\* Given that the input sources of TWICE are updated more frequently (e.g., user\_user\_graph is

\* updated every 2 day), updating TWICE embedding every 2 day will better capture interests of new

\* users and the interest shift of existing users.

\*

\* To build & deploy the scheduled job via workflows:

\* {{{

\* scalding workflow upload \

\* --workflow interested\_in\_twice\_2\_day\_update-batch \

\* --jobs src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_largest\_dim\_2\_day\_update-batch \

\* --scm-paths "src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice/\*" \

\* --autoplay

\* }}}

\*

\*/\*/

object InterestedInTwiceLargestDim2DayUpdateScheduledApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with ScheduledExecutionApp {

override def firstTime: RichDate = RichDate("2022-04-06")

override def batchIncrement: Duration = Days(2)

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersMatchingLargestDimension

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runScheduledApp(

new LargestDimensionClusteringMethod(),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_by\_largest\_dim\_2\_day\_update",

"clusters\_members\_largest\_dim\_ape\_similarity\_2\_day\_update",

InterestedInTwiceByLargestDim2DayUpdateScalaDataset,

ClustersMembersLargestDimApeSimilarity2DayUpdateScalaDataset,

args.getOrElse("num-reducers", "4000").toInt

)

}

}

/\*\*

[Preferred way] To run a locally built adhoc job:

./bazel bundle src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_<CLUSTERING\_METHOD>-adhoc

scalding remote run --target src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_<CLUSTERING\_METHOD>-adhoc

To build and run a adhoc job with workflows:

scalding workflow upload \

--workflow interested\_in\_twice-adhoc \

--jobs src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_largest\_dim-adhoc,src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_louvain-adhoc,src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice:interested\_in\_twice\_connected\_components-adhoc \

--scm-paths "src/scala/com/twitter/simclusters\_v2/scalding/embedding/twice/\*" \

--autoplay \

\*/\*/

object InterestedInTwiceLargestDimAdhocApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with AdhocExecutionApp {

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersMatchingLargestDimension

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runAdhocApp(

new LargestDimensionClusteringMethod(),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_by\_largest\_dim",

"clusters\_members\_largest\_dim\_ape\_similarity",

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceLargestDimMaxFavScoreAdhocApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with AdhocExecutionApp {

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersMatchingLargestDimension

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runAdhocApp(

new LargestDimensionClusteringMethod(),

new MaxFavScoreRepresentativeSelectionMethod[SimClustersEmbedding](),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_by\_largest\_dim\_fav\_score",

"clusters\_members\_largest\_dim\_ape\_similarity",

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceLouvainAdhocApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with AdhocExecutionApp {

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runAdhocApp(

new LouvainClusteringMethod(

args.required("cosine\_similarity\_threshold").toDouble,

args.optional("resolution\_factor").map(\_.toDouble)),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_louvain",

"clusters\_members\_louvain\_ape\_similarity",

args.getOrElse("num-reducers", "4000").toInt

)

}

}

object InterestedInTwiceConnectedComponentsAdhocApp

extends InterestedInTwiceBaseApp[SimClustersEmbedding]

with AdhocExecutionApp {

override def producerProducerSimilarityFnForClustering: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

override def producerProducerSimilarityFnForClusterRepresentative: (

SimClustersEmbedding,

SimClustersEmbedding

) => Double =

SimilarityFunctions.simClustersCosineSimilarity

/\*\*

\* Top-level method of this application.

\*/

def runOnDateRange(

args: Args

)(

implicit dateRange: DateRange,

timeZone: TimeZone,

uniqueId: UniqueID

): Execution[Unit] = {

runAdhocApp(

new ConnectedComponentsClusteringMethod(

args.required("cosine\_similarity\_threshold").toDouble),

new MedoidRepresentativeSelectionMethod[SimClustersEmbedding](

producerProducerSimilarityFnForClusterRepresentative),

ProducerEmbeddingSource.getAggregatableProducerEmbeddings,

"interested\_in\_twice\_connected\_components",

"clusters\_members\_connected\_components\_ape\_similarity",

args.getOrElse("num-reducers", "4000").toInt

)

}

}