package com.twitter.ann.scalding.offline.com.twitter.ann.scalding.benchmark

/\*

This job will generate KNN ground truth based user and item embeddings.

\*/

import com.twitter.scalding.typed.TypedPipe

import com.twitter.scalding.\_

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

import com.twitter.ann.knn.thriftscala.Knn

import com.twitter.ann.knn.thriftscala.Neighbor

import com.twitter.ann.scalding.offline.IndexingStrategy

import com.twitter.ann.scalding.offline.KnnHelper

import com.twitter.ann.common.Distance

import com.twitter.ml.featurestore.lib.embedding.EmbeddingWithEntity

import com.twitter.cortex.ml.embeddings.common.EmbeddingFormatArgsParser

import com.twitter.cortex.ml.embeddings.common.EntityKind

import java.util.TimeZone

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

import com.twitter.ann.scalding.benchmark.UserItemKnnScalaDataset

import com.twitter.scalding\_internal.job.TwitterExecutionApp

import com.twitter.ml.featurestore.lib.EntityId

import com.twitter.ml.featurestore.lib.UserId

/\*\*

\* This job will take consumer and item embeddings(either url or tweet) and output Knn entities (user id, (distance, item id)).

\*

\* Example command to run this adhoc job:

\*

\* scalding remote run \

\* --target ann/src/main/scala/com/twitter/ann/scalding/benchmark:benchmark-adhoc \

\* --hadoop-properties "mapreduce.map.memory.mb=8192 mapreduce.map.java.opts='-Xmx7618M' mapreduce.reduce.memory.mb=8192 mapreduce.reduce.java.opts='-Xmx7618M' mapred.task.timeout=0" \

\* --submitter hadoopnest3.smf1.twitter.com \

\* --user cortex-mlx \

\* --submitter-memory 8000.megabyte \

\* --main-class com.twitter.ann.scalding.offline.com.twitter.ann.scalding.benchmark.KnnJob -- \

\* --dalEnvironment Prod \

\* --search\_space\_entity\_type user \

\* --user.feature\_store\_embedding ConsumerFollowEmbedding300Dataset \

\* --user.feature\_store\_major\_version 1569196895 \

\* --user.date\_range 2019-10-23 \

\* --search\_space.feature\_store\_embedding ConsumerFollowEmbedding300Dataset \

\* --search\_space.feature\_store\_major\_version 1569196895 \

\* --search\_space.date\_range 2019-10-23 \

\* --date 2019-10-25 \

\* --version "consumer\_follower\_test" \

\* --reducers 10000 \

\* --num\_of\_random\_groups 20 \

\* --num\_replicas 1000 \

\* --indexing\_strategy.metric InnerProduct \

\* --indexing\_strategy.type hnsw \

\* --indexing\_strategy.dimension 300 \

\* --indexing\_strategy.ef\_construction 30 \

\* --indexing\_strategy.max\_m 10 \

\* --indexing\_strategy.ef\_query 50 \

\* --search\_space\_shards 3000 \

\* --query\_shards 3000 \

\* --search\_space.read\_sample\_ratio 0.038

\*/

trait KnnJobBase {

val seed: Long = 123

def getKnnDataset[B <: EntityId, D <: Distance[D]](

args: Args

)(

implicit uniqueID: UniqueID

): TypedPipe[Knn] = {

val consumerPipe: TypedPipe[EmbeddingWithEntity[UserId]] = EmbeddingFormatArgsParser.User

.getEmbeddingFormat(args, "user")

.getEmbeddings

val itemPipe = EntityKind

.getEntityKind(args("search\_space\_entity\_type"))

.parser

.getEmbeddingFormat(args, "search\_space")

.getEmbeddings

KnnHelper

// Refer to the documentation of findNearestNeighboursWithIndexingStrategy for more

// information about how to set these settings.

.findNearestNeighboursWithIndexingStrategy[UserId, B, D](

queryEmbeddings = consumerPipe,

searchSpaceEmbeddings = itemPipe.asInstanceOf[TypedPipe[EmbeddingWithEntity[B]]],

numNeighbors = args.int("candidate\_per\_user", 20),

reducersOption = args.optional("reducers").map(\_.toInt),

numOfSearchGroups = args.int("num\_of\_random\_groups"),

numReplicas = args.int("num\_replicas"),

indexingStrategy = IndexingStrategy.parse(args).asInstanceOf[IndexingStrategy[D]],

queryShards = args.optional("query\_shards").map(\_.toInt),

searchSpaceShards = args.optional("search\_space\_shards").map(\_.toInt)

)

.map {

case (user, items) =>

val neighbors = items.map {

case (item, distance) =>

Neighbor(

distance.distance,

item.toThrift

)

}

Knn(user.toThrift, neighbors)

}

}

}

object KnnJob extends TwitterExecutionApp with KnnJobBase {

val KnnPathSuffix: String = "/user/cortex-mlx/qualatative\_analysis/knn\_ground\_truth/"

val partitionKey: String = "version"

override def job: Execution[Unit] = Execution.withId { implicit uniqueId =>

Execution.getArgs.flatMap { args: Args =>

implicit val timeZone: TimeZone = TimeZone.getDefault

implicit val dateParser: DateParser = DateParser.default

implicit val dateRange: DateRange = DateRange.parse(args.list("date"))(timeZone, dateParser)

getKnnDataset(args).writeDALExecution(

UserItemKnnScalaDataset,

D.Daily,

D.Suffix(KnnPathSuffix),

D.Parquet,

Set(D.Partition(partitionKey, args("version"), D.PartitionType.String))

)

}

}

}