package com.twitter.ann.scalding.offline

import com.twitter.ann.common.Distance

import com.twitter.ann.common.Metric

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

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

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

import com.twitter.scalding.source.TypedText

import com.twitter.scalding.Args

import com.twitter.scalding.Execution

import com.twitter.scalding.UniqueID

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

/\*\*

\* This job reads index embedding data, query embeddings data, and split into index set, query set and true nearest neigbor set

\* from query to index.

\*/

object KnnTruthSetGenerator extends TwitterExecutionApp {

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

Execution.getArgs.flatMap { args: Args =>

val queryEntityKind = EntityKind.getEntityKind(args("query\_entity\_kind"))

val indexEntityKind = EntityKind.getEntityKind(args("index\_entity\_kind"))

val metric = Metric.fromString(args("metric"))

run(queryEntityKind, indexEntityKind, metric, args)

}

}

private[this] def run[A <: EntityId, B <: EntityId, D <: Distance[D]](

uncastQueryEntityKind: EntityKind[\_],

uncastIndexSpaceEntityKind: EntityKind[\_],

uncastMetric: Metric[\_],

args: Args

)(

implicit uniqueID: UniqueID

): Execution[Unit] = {

val queryEntityKind = uncastQueryEntityKind.asInstanceOf[EntityKind[A]]

val indexEntityKind = uncastIndexSpaceEntityKind.asInstanceOf[EntityKind[B]]

val metric = uncastMetric.asInstanceOf[Metric[D]]

val reducers = args.int("reducers")

val mappers = args.int("mappers")

val numNeighbors = args.int("neighbors")

val knnOutputPath = args("truth\_set\_output\_path")

val querySamplePercent = args.double("query\_sample\_percent", 100) / 100

val indexSamplePercent = args.double("index\_sample\_percent", 100) / 100

val queryEmbeddings = queryEntityKind.parser

.getEmbeddingFormat(args, "query")

.getEmbeddings

.sample(querySamplePercent)

val indexEmbeddings = indexEntityKind.parser

.getEmbeddingFormat(args, "index")

.getEmbeddings

.sample(indexSamplePercent)

// calculate and write knn

val knnExecution = KnnHelper

.findNearestNeighbours(

queryEmbeddings,

indexEmbeddings,

metric,

numNeighbors,

reducers = reducers,

mappers = mappers

)(queryEntityKind.ordering, uniqueID).map(

nearestNeighborsToString(\_, queryEntityKind, indexEntityKind)

)

.shard(1)

.writeExecution(TypedText.tsv(knnOutputPath))

// write query set embeddings

val querySetExecution = queryEntityKind.parser

.getEmbeddingFormat(args, "query\_set\_output")

.writeEmbeddings(queryEmbeddings)

// write index set embeddings

val indexSetExecution = indexEntityKind.parser

.getEmbeddingFormat(args, "index\_set\_output")

.writeEmbeddings(indexEmbeddings)

Execution.zip(knnExecution, querySetExecution, indexSetExecution).unit

}

}