package com.twitter.ann.scalding.offline.faissindexbuilder

import com.twitter.ann.common.Distance

import com.twitter.ann.common.Metric

import com.twitter.cortex.ml.embeddings.common.\_

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

import com.twitter.scalding.Args

import com.twitter.scalding.DateOps

import com.twitter.scalding.DateParser

import com.twitter.scalding.DateRange

import com.twitter.scalding.Execution

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

import com.twitter.search.common.file.FileUtils

import com.twitter.util.logging.Logging

import java.util.Calendar

import java.util.TimeZone

trait IndexBuilderExecutable extends Logging {

// This method is used to cast the entityKind and the metric to have parameters.

def indexBuilderExecution[T <: UserId, D <: Distance[D]](

args: Args

): Execution[Unit] = {

// parse the arguments for this job

val uncastEntityKind = EntityKind.getEntityKind(args("entity\_kind"))

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

val entityKind = uncastEntityKind.asInstanceOf[EntityKind[T]]

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

val uncastDateRange = args.list("embedding\_date\_range")

val embeddingDateRange = if (uncastDateRange.nonEmpty) {

Some(DateRange.parse(uncastDateRange)(DateOps.UTC, DateParser.default))

} else {

None

}

val embeddingFormat =

entityKind.parser.getEmbeddingFormat(args, "input", providedDateRange = embeddingDateRange)

val numDimensions = args.int("num\_dimensions")

val embeddingLimit = args.optional("embedding\_limit").map(\_.toInt)

val outputDirectory = FileUtils.getFileHandle(args("output\_dir"))

val factoryString = args.optional("factory\_string").get

val sampleRate = args.float("training\_sample\_rate", 0.05f)

logger.debug(s"Job args: ${args.toString}")

val finalOutputDirectory = embeddingDateRange

.map { range =>

val cal = Calendar.getInstance(TimeZone.getTimeZone("UTC"))

cal.setTime(range.end)

outputDirectory

.getChild(s"${cal.get(Calendar.YEAR)}")

.getChild(f"${cal.get(Calendar.MONTH) + 1}%02d")

.getChild(f"${cal.get(Calendar.DAY\_OF\_MONTH)}%02d")

}.getOrElse(outputDirectory)

logger.info(s"Final output directory is ${finalOutputDirectory.getPath}")

IndexBuilder

.run(

embeddingFormat,

embeddingLimit,

sampleRate,

factoryString,

metric,

finalOutputDirectory,

numDimensions

).onComplete { \_ =>

Unit

}

}

}

object IndexBuilderApp extends TwitterExecutionApp with IndexBuilderExecutable {

override def job: Execution[Unit] = Execution.getArgs.flatMap { args: Args =>

indexBuilderExecution(args)

}

}