package com.twitter.ann.faiss

import com.twitter.conversions.DurationOps.richDurationFromInt

import com.twitter.search.common.file.AbstractFile

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

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Time

import com.twitter.util.Try

import com.twitter.util.logging.Logging

import java.util.Locale

object HourlyDirectoryWithSuccessFileListing extends Logging {

private val SUCCESS\_FILE\_NAME = "\_SUCCESS"

def listHourlyIndexDirectories(

root: AbstractFile,

startingFrom: Time,

count: Int,

lookbackInterval: Int

): Seq[AbstractFile] = listingStep(root, startingFrom, count, lookbackInterval)

private def listingStep(

root: AbstractFile,

startingFrom: Time,

remainingDirectoriesToFind: Int,

remainingAttempts: Int

): List[AbstractFile] = {

if (remainingDirectoriesToFind == 0 || remainingAttempts == 0) {

return List.empty

}

val head = getSuccessfulDirectoryForDate(root, startingFrom)

val previousHour = startingFrom - 1.hour

head match {

case Throw(e) =>

listingStep(root, previousHour, remainingDirectoriesToFind, remainingAttempts - 1)

case Return(directory) =>

directory ::

listingStep(root, previousHour, remainingDirectoriesToFind - 1, remainingAttempts - 1)

}

}

private def getSuccessfulDirectoryForDate(

root: AbstractFile,

date: Time

): Try[AbstractFile] = {

val folder = root.getPath + "/" + date.format("yyyy/MM/dd/HH", Locale.ROOT)

val successPath =

folder + "/" + SUCCESS\_FILE\_NAME

debug(s"Checking ${successPath}")

Try(FileUtils.getFileHandle(successPath)).flatMap { file =>

if (file.canRead) {

Try(FileUtils.getFileHandle(folder))

} else {

Throw(new IllegalArgumentException(s"Found ${file.toString} but can't read it"))

}

}

}

}