package com.twitter.ann.annoy

import com.twitter.ann.annoy.AnnoyCommon.IndexIdMappingFileName

import com.twitter.ann.common.\_

import com.twitter.ann.file\_store.WritableIndexIdFileStore

import com.twitter.bijection.Injection

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

import com.twitter.util.Future

import com.twitter.util.FuturePool

import org.apache.beam.sdk.io.fs.ResourceId

private[annoy] object TypedAnnoyIndexBuilderWithFile {

private[annoy] def apply[T, D <: Distance[D]](

dimension: Int,

numOfTrees: Int,

metric: Metric[D],

injection: Injection[T, Array[Byte]],

futurePool: FuturePool

): Appendable[T, AnnoyRuntimeParams, D] with Serialization = {

val index = RawAnnoyIndexBuilder(dimension, numOfTrees, metric, futurePool)

val writableFileStore = WritableIndexIdFileStore(injection)

new TypedAnnoyIndexBuilderWithFile[T, D](index, writableFileStore)

}

}

private[this] class TypedAnnoyIndexBuilderWithFile[T, D <: Distance[D]](

indexBuilder: RawAppendable[AnnoyRuntimeParams, D] with Serialization,

store: WritableIndexIdFileStore[T])

extends Appendable[T, AnnoyRuntimeParams, D]

with Serialization {

private[this] val transformedIndex = IndexTransformer.transformAppendable(indexBuilder, store)

override def append(entity: EntityEmbedding[T]): Future[Unit] = {

transformedIndex.append(entity)

}

override def toDirectory(directory: ResourceId): Unit = {

indexBuilder.toDirectory(directory)

toDirectory(new IndexOutputFile(directory))

}

override def toDirectory(directory: AbstractFile): Unit = {

indexBuilder.toDirectory(directory)

toDirectory(new IndexOutputFile(directory))

}

private def toDirectory(directory: IndexOutputFile): Unit = {

val indexIdFile = directory.createFile(IndexIdMappingFileName)

store.save(indexIdFile)

}

override def toQueryable: Queryable[T, AnnoyRuntimeParams, D] = {

transformedIndex.toQueryable

}

}