package com.twitter.ann.hnsw

import com.twitter.ann.common.EmbeddingType.EmbeddingVector

import com.twitter.bijection.Injection

import com.twitter.ml.api.embedding.Embedding

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

import java.io.OutputStream

import org.mapdb.DBMaker

import org.mapdb.HTreeMap

import org.mapdb.Serializer

import scala.collection.JavaConverters.\_

/\*\*

\* This class currently only support querying and creates map db on fly from thrift serialized embedding mapping

\* Implement index creation with this or altogether replace mapdb with some better performing solution as it takes a lot of time to create/query or precreate while serializing thrift embeddings

\*/

private[hnsw] object MapDbBasedIdEmbeddingMap {

/\*\*

\* Loads id embedding mapping in mapDB based container leveraging memory mapped files.

\* @param embeddingFile: Local/Hdfs file path for embeddings

\* @param injection : Injection for typed Id T to Array[Byte]

\*/

def loadAsReadonly[T](

embeddingFile: AbstractFile,

injection: Injection[T, Array[Byte]]

): IdEmbeddingMap[T] = {

val diskDb = DBMaker

.tempFileDB()

.concurrencyScale(32)

.fileMmapEnable()

.fileMmapEnableIfSupported()

.fileMmapPreclearDisable()

.cleanerHackEnable()

.closeOnJvmShutdown()

.make()

val mapDb = diskDb

.hashMap("mapdb", Serializer.BYTE\_ARRAY, Serializer.FLOAT\_ARRAY)

.createOrOpen()

HnswIOUtil.loadEmbeddings(

embeddingFile,

injection,

new MapDbBasedIdEmbeddingMap(mapDb, injection)

)

}

}

private[this] class MapDbBasedIdEmbeddingMap[T](

mapDb: HTreeMap[Array[Byte], Array[Float]],

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

extends IdEmbeddingMap[T] {

override def putIfAbsent(id: T, embedding: EmbeddingVector): EmbeddingVector = {

val value = mapDb.putIfAbsent(injection.apply(id), embedding.toArray)

if (value == null) null else Embedding(value)

}

override def put(id: T, embedding: EmbeddingVector): EmbeddingVector = {

val value = mapDb.put(injection.apply(id), embedding.toArray)

if (value == null) null else Embedding(value)

}

override def get(id: T): EmbeddingVector = {

Embedding(mapDb.get(injection.apply(id)))

}

override def iter(): Iterator[(T, EmbeddingVector)] = {

mapDb

.entrySet()

.iterator()

.asScala

.map(entry => (injection.invert(entry.getKey).get, Embedding(entry.getValue)))

}

override def size(): Int = mapDb.size()

override def toDirectory(embeddingFile: OutputStream): Unit = {

HnswIOUtil.saveEmbeddings(embeddingFile, injection, iter())

}

}