package com.twitter.ann.annoy

import com.twitter.ann.common.\_

import com.twitter.bijection.Injection

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

import com.twitter.util.FuturePool

// Class to provide Annoy based ann index.

object TypedAnnoyIndex {

/\*\*

\* Create Annoy based typed index builder that serializes index to a directory (HDFS/Local file system).

\* It cannot be used in scalding as it leverage C/C++ jni bindings, whose build conflicts with version of some libs installed on hadoop.

\* You can use it on aurora or with IndexBuilding job which triggers scalding job but then streams data to aurora machine for building index.

\* @param dimension dimension of embedding

\* @param numOfTrees builds a forest of numOfTrees trees.

\* More trees gives higher precision when querying at the cost of increased memory and disk storage requirement at the build time.

\* At runtime the index will be memory mapped, so memory wont be an issue but disk storage would be needed.

\* @param metric distance metric for nearest neighbour search

\* @param injection Injection to convert bytes to Id.

\* @tparam T Type of Id for embedding

\* @tparam D Typed Distance

\* @return Serializable AnnoyIndex

\*/

def indexBuilder[T, D <: Distance[D]](

dimension: Int,

numOfTrees: Int,

metric: Metric[D],

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

futurePool: FuturePool

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

TypedAnnoyIndexBuilderWithFile(dimension, numOfTrees, metric, injection, futurePool)

}

/\*\*

\* Load Annoy based queryable index from a directory

\* @param dimension dimension of embedding

\* @param metric distance metric for nearest neighbour search

\* @param injection Injection to convert bytes to Id.

\* @param futurePool FuturePool

\* @param directory Directory (HDFS/Local file system) where serialized index is stored.

\* @tparam T Type of Id for embedding

\* @tparam D Typed Distance

\* @return Typed Queryable AnnoyIndex

\*/

def loadQueryableIndex[T, D <: Distance[D]](

dimension: Int,

metric: Metric[D],

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

futurePool: FuturePool,

directory: AbstractFile

): Queryable[T, AnnoyRuntimeParams, D] = {

TypedAnnoyQueryIndexWithFile(dimension, metric, injection, futurePool, directory)

}

}