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

import com.twitter.ann.common.Distance

import com.twitter.ann.common.EntityEmbedding

import com.twitter.ann.common.Metric

import com.twitter.ann.faiss.FaissIndexer

import com.twitter.cortex.ml.embeddings.common.EmbeddingFormat

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

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

import com.twitter.scalding.Execution

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

import com.twitter.util.logging.Logging

object IndexBuilder extends FaissIndexer with Logging {

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

embeddingFormat: EmbeddingFormat[T],

embeddingLimit: Option[Int],

sampleRate: Float,

factoryString: String,

metric: Metric[D],

outputDirectory: AbstractFile,

numDimensions: Int

): Execution[Unit] = {

val embeddingsPipe = embeddingFormat.getEmbeddings

val limitedEmbeddingsPipe = embeddingLimit

.map { limit =>

embeddingsPipe.limit(limit)

}.getOrElse(embeddingsPipe)

val annEmbeddingPipe = limitedEmbeddingsPipe.map { embedding =>

val embeddingSize = embedding.embedding.length

assert(

embeddingSize == numDimensions,

s"Specified number of dimensions $numDimensions does not match the dimensions of the " +

s"embedding $embeddingSize"

)

EntityEmbedding[Long](embedding.entityId.userId, Embedding(embedding.embedding.toArray))

}

build(annEmbeddingPipe, sampleRate, factoryString, metric, outputDirectory)

}

}