package com.twitter.ann.util

import com.twitter.ann.common.{Appendable, EntityEmbedding}

import com.twitter.concurrent.AsyncStream

import com.twitter.logging.Logger

import com.twitter.util.Future

import java.util.concurrent.atomic.AtomicInteger

object IndexBuilderUtils {

val Log = Logger.apply()

def addToIndex[T](

appendable: Appendable[T, \_, \_],

embeddings: Seq[EntityEmbedding[T]],

concurrencyLevel: Int

): Future[Int] = {

val count = new AtomicInteger()

// Async stream allows us to procss at most concurrentLevel futures at a time.

Future.Unit.before {

val stream = AsyncStream.fromSeq(embeddings)

val appendStream = stream.mapConcurrent(concurrencyLevel) { annEmbedding =>

val processed = count.incrementAndGet()

if (processed % 10000 == 0) {

Log.info(s"Performed $processed updates")

}

appendable.append(annEmbedding)

}

appendStream.size

}

}

}