package com.twitter.ann.service.query\_server.common.warmup

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

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

import com.twitter.util.Await

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

import com.twitter.util.logging.Logging

import scala.annotation.tailrec

import scala.util.Random

trait Warmup extends Logging {

protected def minSuccessfulTries: Int

protected def maxTries: Int

protected def randomQueryDimension: Int

protected def timeout: Duration

@tailrec

final protected def run(

iteration: Int = 0,

successes: Int = 0,

name: String,

f: => Future[\_]

): Unit = {

if (successes == minSuccessfulTries || iteration == maxTries) {

info(s"Warmup finished after ${iteration} iterations with ${successes} successes")

} else {

Try(Await.result(f.liftToTry, timeout)) match {

case Return(Return(\_)) =>

debug(s"[$name] Iteration $iteration Success")

run(iteration + 1, successes + 1, name, f)

case Return(Throw(e)) =>

warn(s"[$name] Iteration $iteration has failed: ${e.getMessage}. ", e)

run(iteration + 1, successes, name, f)

case Throw(e) =>

info(s"[$name] Iteration $iteration was too slow: ${e.getMessage}. ", e)

run(iteration + 1, successes, name, f)

}

}

}

private val rng = new Random()

protected def randomQuery(): EmbeddingVector =

Embedding(Array.fill(randomQueryDimension)(-1 + 2 \* rng.nextFloat()))

def warmup(): Unit

}