package com.twitter.ann.common

import com.twitter.ann.common.EmbeddingType.\_

import com.twitter.ann.common.thriftscala.{

NearestNeighborQuery,

NearestNeighborResult,

Distance => ServiceDistance,

RuntimeParams => ServiceRuntimeParams

}

import com.twitter.bijection.Injection

import com.twitter.finagle.Service

import com.twitter.mediaservices.commons.codec.ArrayByteBufferCodec

import com.twitter.util.Future

class ServiceClientQueryable[T, P <: RuntimeParams, D <: Distance[D]](

service: Service[NearestNeighborQuery, NearestNeighborResult],

runtimeParamInjection: Injection[P, ServiceRuntimeParams],

distanceInjection: Injection[D, ServiceDistance],

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

extends Queryable[T, P, D] {

override def query(

embedding: EmbeddingVector,

numOfNeighbors: Int,

runtimeParams: P

): Future[List[T]] = {

service

.apply(

NearestNeighborQuery(

embeddingSerDe.toThrift(embedding),

withDistance = false,

runtimeParamInjection(runtimeParams),

numOfNeighbors

)

)

.map { result =>

result.nearestNeighbors.map { nearestNeighbor =>

idInjection.invert(ArrayByteBufferCodec.decode(nearestNeighbor.id)).get

}.toList

}

}

override def queryWithDistance(

embedding: EmbeddingVector,

numOfNeighbors: Int,

runtimeParams: P

): Future[List[NeighborWithDistance[T, D]]] =

service

.apply(

NearestNeighborQuery(

embeddingSerDe.toThrift(embedding),

withDistance = true,

runtimeParamInjection(runtimeParams),

numOfNeighbors

)

)

.map { result =>

result.nearestNeighbors.map { nearestNeighbor =>

NeighborWithDistance(

idInjection.invert(ArrayByteBufferCodec.decode(nearestNeighbor.id)).get,

distanceInjection.invert(nearestNeighbor.distance.get).get

)

}.toList

}

}