package com.twitter.ann.manhattan

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

import com.twitter.ann.common.{EmbeddingProducer, EmbeddingType}

import com.twitter.bijection.Injection

import com.twitter.ml.api.embedding.{EmbeddingBijection, EmbeddingSerDe}

import com.twitter.ml.api.{thriftscala => thrift}

import com.twitter.stitch.Stitch

import com.twitter.storage.client.manhattan.bijections.Bijections

import com.twitter.storage.client.manhattan.bijections.Bijections.BinaryScalaInjection

import com.twitter.storage.client.manhattan.kv.ManhattanKVEndpoint

import com.twitter.storage.client.manhattan.kv.impl.{

DescriptorP1L0,

ReadOnlyKeyDescriptor,

ValueDescriptor

}

private[manhattan] class ManhattanEmbeddingProducer[T](

keyDescriptor: DescriptorP1L0.DKey[T],

valueDescriptor: ValueDescriptor.EmptyValue[EmbeddingVector],

manhattanEndpoint: ManhattanKVEndpoint)

extends EmbeddingProducer[T] {

/\*\*

\* Lookup an embedding from manhattan given a key of type T.

\*

\* @return An embedding stitch.

\* An easy way to get a Future from a Stitch is to run Stitch.run(stitch)

\*/

override def produceEmbedding(input: T): Stitch[Option[EmbeddingVector]] = {

val fullKey = keyDescriptor.withPkey(input)

val stitchResult = manhattanEndpoint.get(fullKey, valueDescriptor)

stitchResult.map { resultOption =>

resultOption.map(\_.contents)

}

}

}

object ManhattanEmbeddingProducer {

private[manhattan] def keyDescriptor[T](

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

dataset: String

): DescriptorP1L0.DKey[T] =

ReadOnlyKeyDescriptor(injection.andThen(Bijections.BytesBijection))

.withDataset(dataset)

private[manhattan] val EmbeddingDescriptor: ValueDescriptor.EmptyValue[

EmbeddingType.EmbeddingVector

] = {

val embeddingBijection = new EmbeddingBijection(EmbeddingSerDe.floatEmbeddingSerDe)

val thriftInjection = BinaryScalaInjection[thrift.Embedding](thrift.Embedding)

ValueDescriptor(embeddingBijection.andThen(thriftInjection))

}

def apply[T](

dataset: String,

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

manhattanEndpoint: ManhattanKVEndpoint

): EmbeddingProducer[T] = {

val descriptor = keyDescriptor(injection, dataset)

new ManhattanEmbeddingProducer(descriptor, EmbeddingDescriptor, manhattanEndpoint)

}

}