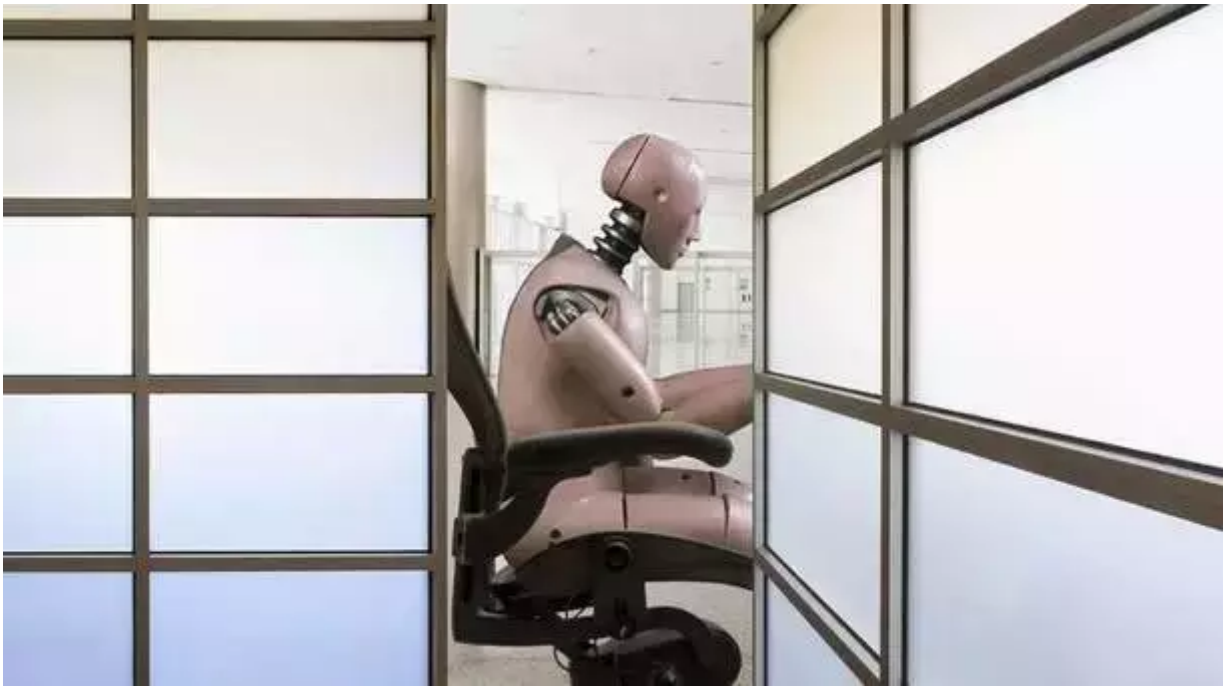
重磅|Facebook发布AI搜索引擎Faiss：比最先进搜索算法快8.5倍

人工智能学家 2017-04-01



概要：Facebook对相似的多媒体文档进行快速搜索，这是以前传统的搜索引擎无法做到的挑战。

来源：全球人工智能

译者：马卓奇博士

近日，Facebook发布了 人工智能相似性搜索（Faiss）库，可以让Facebook对相似的多媒体文档进行快速搜索，这是以前传统的搜索引擎无法做到的挑战。Facebook实现的最近邻搜索算法，在十亿级的数据库上，比当前已知的最好方法，以及目前文献中已知的GPU上最快的k近邻搜索算法的速度快大约8.5倍。Faiss打破了一些记录，比如Facebook是第一个在十亿高维向量上建立k近邻图的。

相似性学习

传统的数据库是由包含符号信息的结构性表格组成的。例如，一个图像集可以表示成一个表格，表格的每一行代表一副索引图像。每一行都包含类似图像编号和描述文本的信息。并且这些行可以链接到其他表格的项，例如一张含有人的图片可以和一个姓名表链接起来。

AI工具，例如文本嵌入模型（word2vec），或卷积神经网络（CNN）描述子使用深度学习训练，会产生高维度向量。与固定的符号表示相比，这些表示更有表现力并且更灵活，Facebook将在文中进一步解释。然而用SQL索引的传统数据库不适合这些新的表示方法。首先，新媒体项目的巨大数据流会创造几十

亿个失量。其次，并且更重要的是，找到相似的项，意味着找到相似的高维向量，即使用标准的索引语言能做到，也是很无效的。

向量表示的用途是什么?

假设你有一张建筑物的图片，比如说某个中等城市的市政厅，但你不记得城市的名字了，然后你想在图集中找到这个建筑的其他照片。SQL中传统的关键字搜索就没有办法用了，因为你已经忘记了这个城市的名字。

这里，相似性搜索就派上用场了。图像的矢量表示可以对相似的图像产生相似的矢量，相似矢量的定义是欧几里德空间距离相近的矢量。

矢量表示的另一个应用在分类中。比如说你需要一个分类器来决定图像集中的哪副图像包含雏菊。训练分类器是一个众所周知的过程：算法将输入图像分为雏菊的图像和不是雏菊的图像（猫咪，山羊，玫瑰，矢车菊）。如果分类器是线性的，他会输出一个分类向量，这个向量和图像矢量的点积反映了这个图像包含雏菊的可能性。然后可以计算该向量和图集中所有的项的点积，然后返回具有最高值的图像。这一类搜索算法是“内积最大”搜索。

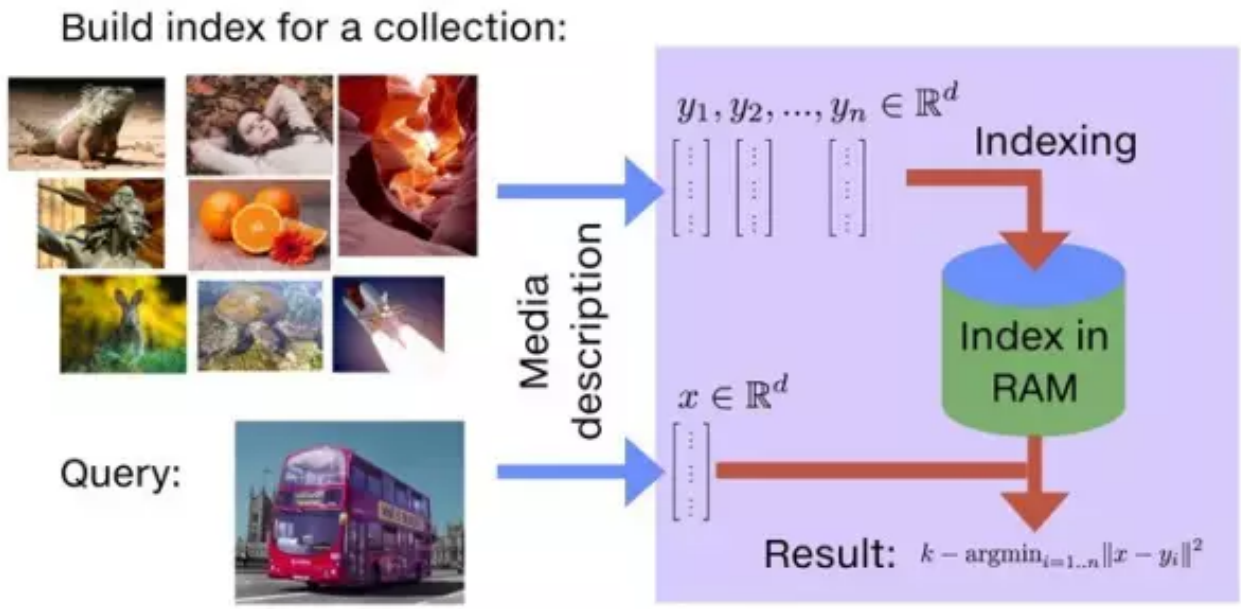
所以，对于相似性搜索和分类，我们需要进行如下的操作

•给出一个搜索矢量，返回与该矢量欧几里德距离最近的数据库目标的列表。
•给出一个搜索矢量，返回与该矢量点积值最高的数据库目标的列表

一个附加的挑战是我们想在很巨大的数据级，在几十亿的向量上进行这些操作。

软件工具

目前可用的软件工具对于上述的数据库搜索操作是不够的。传统的SQL数据库系统很不实际，因为他们已经针对哈希搜索或一维区间搜索做了优化。像OpenCV等库中可用的相似性搜索函数严重受到了扩展性的限制，其他的相似性搜索库只考虑"小"数据集（例如只有一百万个矢量）。还有一些其他的安装包都是发表论文的研究成果，只能在某些特殊设定下显示他们的性能。



Facebook开源的Faiss是一个重点解决上述局限性的库。有如下优点：

- Faiss提供了几个相似性搜索方法，跨越了用法折衷的范围
- Faiss针对内存使用和速度进行了优化
- Faiss针对最相关的索引方法提供了最先进的GPU实现

相似性搜索的评价

一旦学习机器提取了矢量（从图像，视频，文档或其他内容），矢量就可以输入到相似性搜索库中。

Faiss和暴力算法进行对照，准确详尽的计算所有的相似性，然后返回最相似元素的列表值。这可以说是“金牌标准”的参考结果表了。高效地实现暴力算法并不明显，并且它经常影响了其他组件的性能。

如果愿意牺牲准确性，相似性搜索可以快几个数量级，也就是会比对照结果偏离一点点。例如说，如果图像相似性搜索的第一个和第二个结果进行了交换，并不是很严重的事情，因为他们可能都是给出的查询要求的正确结果。加速这一搜索过程需要对数据集进行一些预处理，将其称作索引。

这也就引出了Facebook感兴趣的三个参数

- 速度。要根据查询找到10个（或其他数量）最相似的矢量需要花费多长时间？我们希望比暴力算法的时间要短，不然索引的意义在哪里？
- 内存使用。该方法需要使用多少RAM？比最初的向量多还是少？Faiss支持只用RAM进行搜索，因为硬盘数据库要慢得多，没错，即使是固态硬盘也是这样的。
- 准确率。返回的结果列表和暴力搜索算法的结果对比如何？可以通过统计所有查询中，最近邻真实值是结果列表的第一个返回值的数量来评价准确率，（1召回率法），或者通过计算前10个返回的最近邻的平均分数（10重叠度测量）

通常固定内存使用，来评估速度和准确率之间的权衡。Faiss重点在于压缩初始向量的方法，因为他们是唯一能缩放到数十亿向量的数据集的工具：当有十亿个矢量需要被索引时，即使每个向量只有32位也会占据大量的存储空间。

许多索引库只针对1百万级的矢量，称其为小规模。例如nmslib就含有针对这种库的很有效的算法。它比Faiss快但是需要更多的存储。

在十亿矢量进行评估

由于工程界对于这么大的数据集没有一个完善的基准，所以和研究结果进行比较

。

准确率在Deep1B上进行评估，Deep1B是一个具有十亿张图像的图像集。每张图像都被卷积神经网络（CNN）处理过，并且CNN中的一个激活图作为图像的描述子。这些矢量可以用欧几里德距离进行比较

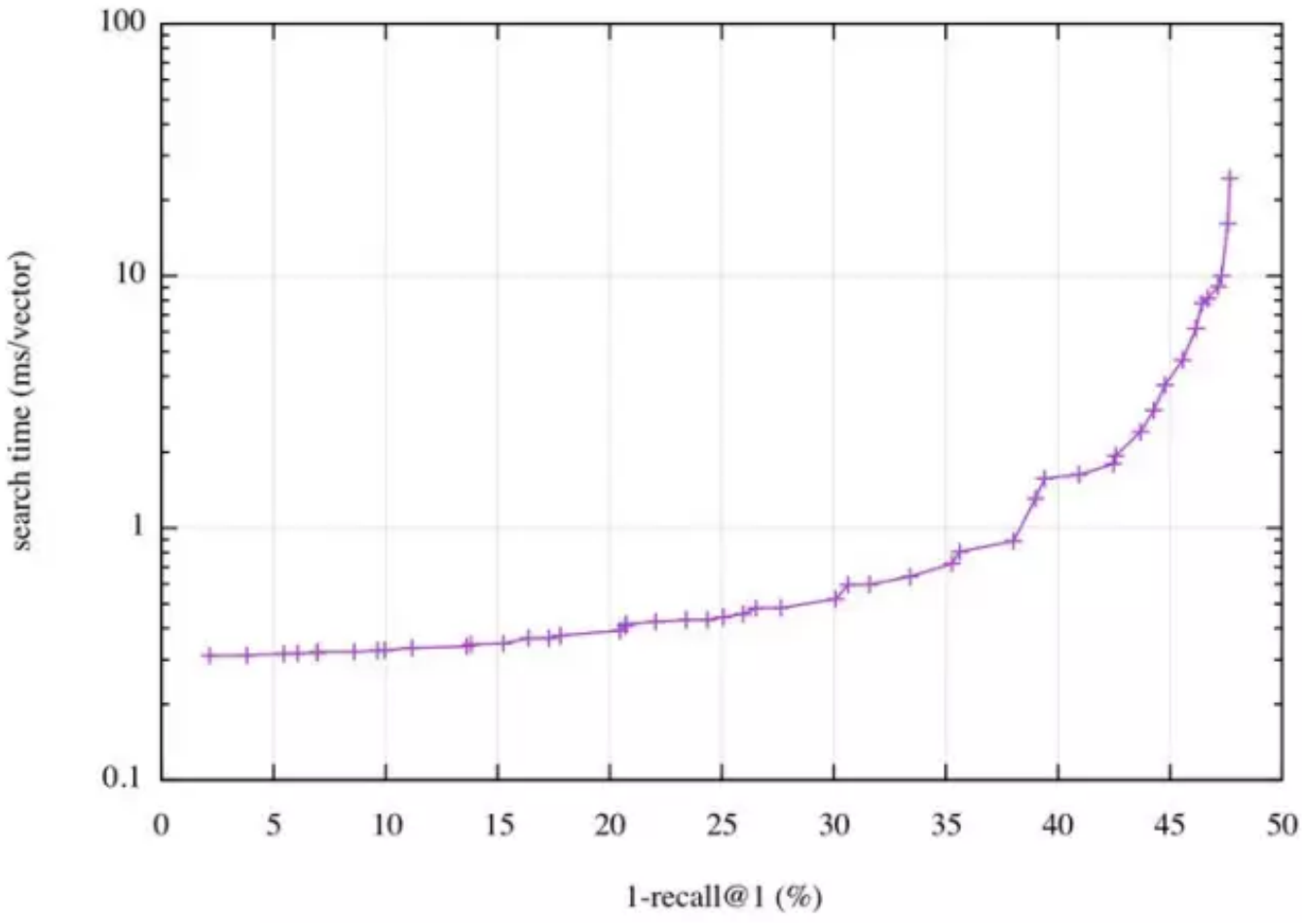
来量化图像之间的相似性。

Deep1B有一个小的查询图像集，并且提供这些图片用暴力算法得到的相似性搜索结果的真实值。所以，如果运行搜索算法，可以估计结果的1召回率。

选择索引

为了进行评估，把RAM内存使用限定在30GB。这个内存的限制条件决定了对于索引方法和参数的选择。Faiss中，索引方法用字符串表示；在这个示例中，是OPQ20_80,IMI2x14,PQ20。

这个字符串说明了一个应用到向量上的预处理步骤（OPQ20_80），一个表示数据库划分方法的选择机制（IMI2x14），一个编码部件（PQ20），说明向量是用乘积量化（PQ）方法产生的20位编码。这样一来，内存使用，包括系统占用的，都低于RAM的30GB。



这听起来有些工程技术性，也就是为什么Faiss的文档提供了指导，教你如何根据你的需求选择最合适的索引方法。

在索引中搜索

当索引准备好之后，可以设置一系列搜索时间参数来调整方法。为了进行评估，Facebook采用单线搜索。由于内存使用已经固定了，需要优化准确率和搜索时间之间的权衡。这也就意味着，可以设置参数，让算法在最短的搜索时间内给出百分之四十的1召回率。

幸运的是，Faiss自带一个自动的调节机制，可以扫描参数空间然后收集能提供最好的操作分的参数。也就是说，牺牲一定的准确性来达到最好的搜索时间，反之亦然。在Deep1B上，操作分可以用曲线显示出来：

在这个曲线图上，可以发现，40%的1召回率的每个矢量的搜索时间少于2毫秒，或者时间达到0.5毫秒，召回率达到30%。2毫秒的查询时间可以换算成在单核处理器上一秒进行500次查询（QPS）。

该结果可以与《使用深度描述子在十亿级数据库上的高效索引》，Babenko and Lempitsky, CVPR 2016这一最新研究结果相比，Deep1B数据库就来自该论文。他们需要20毫秒来达到45%的召回率。

用GPU处理十亿级数据集

许多努力都花费到了GPU实现上，用普通的多GPU支持产生了惊人的单机器性能。GPU实现也是很多CPU的直接替代，并且你不需要知道CUDA API才能使用GPU。GPU Faiss支持所有2012年以后的Nvidia GPU（Kepler，计算能力3.5+）。

用roofline模型作为指导，也就是会尽力占用内存带宽或者浮点单元。 Faiss GPU在单块GPU上的速度比对应的Faiss CPU实现大约快5-10倍。新的Pascal级的硬件，例如P100，把这一数字提高到了20倍。

一些令人印象深刻的数字

- 用近似索引，暴力算法k近邻图（k=10）在128维CNN描述子，YFCC100M数据集的95百万张图像，0.8的10重叠度测量可以在四块Maxwell Titan X GPU上用35分钟建立，包括索引的建立时间。

- 十亿向量k近邻图现在可以很容易做到。可以建立Deep1B数据集的暴力算法k近邻图（k=10），0.65的10重叠度测量在四块Maxwell Titan X GPU上用不到12小时建立，或0.8的10重叠度测量在八块Pascal P100-PCIe GPU上用不到12小时建立。低质量的图可以在Titan X配置上用不到5小时产生。

- 其他的组件也能达到不错的性能。例如，建立上述的Deep1B索引需要用k-means将67.1百万个120维的向量聚成262，144个中心，需要用25E-M迭代，在四块Titan X GPU上花费139分钟(12.6 tflop/s of compute)，或者八块P100 GPU上花费43.8分钟(40 tflop/s of compute)。注意到聚类的训练集不需要用到GPU内存，因为数据是按需要流向GPU的，不会对性能产生影响。

底层设计

Facebook人工智能研究组从2015年开始开发Faiss，主要基于研究结果和大量的工程努力。对于这个库，选择重点关注对几个基础技术的合适优化版本。尤其是重点使用的CPU方面：

- 多线程开发多核并且在多GPU上进行并行搜索
- BLAS库，通过矩阵乘法进行有效准确的距离计算。如果不使用BLAS，高效的暴力算法实现无法达到最优。BLAS/LAPACK是Faiss唯一的强制软件依赖项。
- 机器SIMD向量化和计数用来对独立向量进行加速计算。

GPU方面

对于之前相似性搜索在GPU上的实现，k选择（找到k个最小或最大的元素）一直是个性能问题，因为一些典型的CPU的算法（例如堆选择）与GPU不兼容。对于Faiss GPU，设计了目前文献资料已知的最快的小k选择算法（k<=1024）。所有的中间状态都完整的保存在缓存器中，有助于提高速度。可以对输入数据在单通道进行k选择，操作可以达到可能峰值性能的55%，这一数字由GPU内存带宽的峰值决定。

大部分注意力都放在了有效的平铺方法和用于相似搜索的核实现。多GPU支持由数据分表或复制来提供，该方法不受单GPU的可用内存的限制。在支持GPU上的全float16计算提供了半精度浮点支持(float16)，并且更早的结构提供了中间的float16存储支持。发现用float16进行矢量编码，在加速的同时几乎不会产生准确性损失。

简而言之，不变的系统开销因素在实现中是有影响的。Faiss花费了大量的工作来关注工程细节。

Gaiss用C++实现，并且和Python有捆绑。开始，首先从GitHub上下载Faiss，进行编译，并且把Faiss模块导入Python。Faiss可以和numpy完全结合，并且所有的函数都采用numpy阵列（32浮点）。

索引目标

Faiss（C++ 和 Python）提供了索引实例。

每个索引的子类别都是一个索引结构，在其中可以增加和搜索向量。例如，IndexFlatL2就是一个暴力算法索引，用L2距离进行查找。

```
import faiss                   # make faiss available
index = faiss.IndexFlatL2(d)   # build the index, d=size of vectors

# here we assume xb contains a n-by-d numpy matrix of type float32

index.add(xb)                  # add vectors to the index
print index.ntotal
```

这个可以显示索引矢量的数目。添加到IndexFlat意味着复制矢量到索引的内部存储，由于对矢量没有处理。

进行搜索：

```
# xq is a n2-by-d matrix with query vectors
k = 4                          # we want 4 similar vectors
D, I = index.search(xq, k)     # actual search
print I
```

I是一个整数矩阵，输出如下：

```
[[  0 393 363  78]
 [  1 555 277 364]
 [  2 304 101  13]]
```

对于xq的第一个矢量，xb中最相似矢量的索引是0（基于0的），第二相似的是#393，第三是#363，以此类推。对于xq的第二个矢量，相似矢量列表是#1, #555,等等。xq的前三个矢量和xb的前三个矢量看起来是相同的。

矩阵D是平方距离矩阵。和I形状相同，表示的是用欧几里德平方距离查询的每个结果矢量。

Faiss实现了12中索引类别，一般是其他索引的组合。可选的GPU版本有完全一样的界面，并且有CPU和GPU索引之间转换的桥。Python界面基本是由C++产生的，以显示C++索引，所以很容易将Python验证编码转换到整合的C++。

来源:全球人工智能
