Faiss: 入门导读

原创 果冻虾仁 编程往事 7月10日

收录于话题 #后台公论 6个

引言

Faiss是Facebook于2017年开源的一个相似度检索工具。

相似度检索是啥？搜索、广告、推荐都需要用到相似度的检索。因为无论是网页、广告抑或推荐博文一定要符合你的查询意图才能带来更好的用户体验。

Faiss支持的不止是文本的相似检索，它支持多媒体文档。图片，视频都可以，只要把它们向量化就行。

本文主要是解读一下Faiss的官方Get Started文档中的Demo代码：

https://github.com/facebookresearch/faiss/wiki/Getting-started

虽然代码只有潦潦数行，但对于初学者也值得玩味。另外注意本文对demo代码有微调。

正文

```
import numpy as np
d = 64                           # dimension
nb = 100000                      # database size
nq = 10000                       # nb of queries
np.random.seed(1234)             # make reproducible
xb = np.random.random((nb, d)).astype('float32')
xb[:, 0] += np.arange(nb) / 1000.
```

numpy 随机数

np.random.random((nb, d)) 使用numpy随机数生成二维数组（矩阵）。其中nb表示矩阵的行数，d表示矩阵的列数。

随机数的数值范围在 [0.0, 1.0)区间，本来就是浮点型，貌似可以不用再astype了。

 numpy.array

np.random.random((nb, d)) 生成的数据类型是numpy.array。

python3虽然也有array类型，但是只支持一维。普通的list虽然可以二维，但是性能太差。所以numpy有自己的array类型，并且有更丰富的api。

 numpy.array 切片

xb 就是一个numpy.array了。然后 xb[:, 0] 表示的是对二维数组切片。

这个方括号里冒号逗号分隔，可以视作三个参数：

- 参数1和参数2表示的选择的行范围。用法类型list的切片，只是这里选择的是行。
- 参数3表示在选择完行之后，要选择的列的下标。

所以xb[:, 0]表示的是选择所有行的第一列。
悄悄告诉你：别试了，即使是二维的list不支持这个写法哦。

```
import faiss                   # make faiss available
index = faiss.IndexFlatL2(d)   # build the index
print(index.is_trained)
print(index.ntotal)
index.add(xb)                  # add vectors to the index
print(index.ntotal)
```

这个算是进入正题了，导入faiss包（需要事先安装哦，建议使用conda安装）

 faiss.IndexFlatL2(d)

faiss.IndexFlatL2(d)创建了一个IndexFlatL2类型的索引。faiss支持丰富的索引类型，这里创建的只是最简单的索引，它进行暴力的L2距离搜索。

基于向量空间计算相似度，主要有两种方法，一种就是L2（即欧几里得距离），另外一种是计算夹角cosin（即余弦相似度），本文这里不做展开，后续会有文章单独介绍。

另外创建索引一定要指定维度，也就是参数d。

 index.is_trained

index.is_trained 表示是否训练完成。大部分索引需要训练，而IndexFlatL2不需要，所以这里会直接返回True。

 index.add(xb)

xb是前面用numpy生成的随机二维数组（一组向量），将其添加到索引中。
或者可以说成是给xb构建了一个索引。

 index.ntotal

这个表示被索引数据的数目，在执行index.add之前ntotal是0，在index.add之后ntotal为100000，也就是nb的值。

```
k = 4                          # we want to see 4 nearest neighbors
D, I = index.search(xb[:5], k) # sanity check
print(I)
print(D)
```

 index.search

index.search就是在进行相似性检索了。参数1是输入数据，参数2是个数。

k = 4，表示要搜索4个近邻（NN）。也就是通常说的KNN，K-means的K。

xb[:5]是xb的0 - 4行共5组向量，在xb中找到与输入的5个向量最相似的4个向量。

 返回值：I

I表示的是id。输出如下：

```
[[  0 393 363  78]
 [  1 555 277 364]
```

```
[  2 304 101  13]
[  3 173  18 182]
[  4 288 370 531]]
```

因为输入数据xb[:5]含有5个向量，所以返回的结果也是5个（5行）。
每一行有4个元素（因为k=4）。从左到右表示距离从近到远。元素的值是xb中的向量的id。


返回值：D

D表示的就是计算出来的距离。输出如下：

```
[[0.        7.1751738 7.20763   7.2511625]
 [0.        6.3235645 6.684581  6.799946 ]
 [0.        5.7964087 6.391736  7.2815123]
 [0.        7.2779055 7.527987  7.6628466]
 [0.        6.7638035 7.2951202 7.3688145]]
```

也就是I的矩阵中，返回的向量id和输入向量之间的距离。
从结果可以验证，确实从左到右其距离越来越远。

```
xq = np.random.random((nq, d)).astype('float32')
xq[:, 0] += np.arange(nq) / 1000.
D, I = index.search(xq, k)     # actual search
print(I[:5])                   # neighbors of the 5 first queries
print(I[-5:])                  # neighbors of the 5 last queries
print(D)
```

这个代码片段其实和上一个类似的，只是这个是在模拟真实的检索。

因为真实的相似检索过程，输入数据可不是文档集合的xb[:5]，而是另外一组向量。

比如用户看完一篇文章，要推荐其他文章给用户。那么xq就是将看完的这篇文章的特征向量化，然后去所有候选的文章集合中去找最与之相似的几篇。

这个过程也就是『召回』。在信息检索和推荐领域都有召回的概念。