链滴

Jina AI x 矩池云 Matpool ｜神经搜索引擎，一键构建

作者：matpool

原文链接：https://ld246.com/article/1652345535476

来源网站：链滴

图片、视频、语音等非结构化数据在快速增长，随着深度学习技术的不断升级，非结构化数据的搜索逐渐形成可能。在这样的背景下，专注于神经搜索技术的商业开源软件公司——Jina AI，提出了神经索 (Neural Search)，借助深度学习技术搜索非结构化数据。

现在，矩池云 Matpool 已经支持 Jina 框架，学习者和研究者只需在租用机器时选择 Jina 镜像，就以体会高效的、大规模非结构化数据搜索，同时矩池云团队版用户还可实现基于镜像、资料、算力、储等共享，完成高效协作。

神经搜索 (Neural Search)

首先，我们来了解一下什么是神经搜索 (Neural Search) 。

神经搜索是指用深度学习技术对海量信息进行搜索。与传统搜索不同的点在于，传统搜索主要基于文标签，而神经搜索则可以处理文本、图像、视频、音频甚至 3D Mesh 之间的多模态和跨模态搜索问。神经搜索系统的应用，包括以图搜图、以文字搜图、Question-Answering（问答机器人）、照片重、海量标签分类等。

神经搜索借助深度学习模型，把非结构化数据表示为向量。向量空间中，相似数据会聚合在一起，不数据则会分散在空间的不同位置。根据用户查询的数据，在向量空间中寻找最近邻，就可以实现非结化数据的搜索。

虽然多模态和跨模态搜索是神经搜索的重要应用场景，但是搭建神经搜索解决方案却非常复杂，往往及到工程化、AI 建模和以及 DevOps。简化这种复杂性，正是 Jina AI 在做的事情。

Jina AI 神经搜索生态

Jina AI 提供了一个涵盖整个开发过程的端到端开源技术栈，即神经搜索生态。

DocArray: The data structure for unstructured data

在神经搜索生态中的第一个产品是 DocArray，这也是创建神经搜索项目的第一步。

DocArray 将非结构化数据，统一成同一种数据结构，利用 Python API，研发人员可以高效地处理、量化、搜索、推荐、存储及传输数据。它适用于大型项目的构建，并且针对数据科学家和 AI 工程师行了大量优化。

Jina: Cloud-native neural search framework for any kind of data

Jina 是生态中最早的产品，诞生于 2020 年。Jina 是一个云原生神经搜索框架，简单来讲，它可以把地 DocArray 程序升级为一个高度可扩展的云服务。

开发者如果自己设计神经搜索系统，往往需要自行维护一套工具链，包括构建模型预测服务、维护向索引等，Jina 通过将整个流程封装成一个完整系统，提供统一的接口，极大降低了神经搜索系统的开成本。

Finetuner: Fintune any DNN for better embedding on neural search tasks

优秀的神经搜索解决方案，往往需要一个绝佳的向量模型，公开的预训练模型一般就能解决这个问题 Finetuner 可以借助特定领域的数据，进一步微调模型，以获得更高的准确率，更好地应用于搜索任 。为此，Finetuner 可以被视作神经搜索的最后一步。

CLIP-as-service: Embed images and sentences into fixed-length vectors with CLIP

CLIP-as-service 利用 CLIP 模型，可以将图像和句子嵌入固定长度的向量中，开发者可以在构建新的索解决方案时，将其作为向量化服务，或者在用 Jina 在生产中构建服务时，简单地将其作为最佳实 。

Hub: Share and discover building blocks for neural search applications

实际应用中，一个搜索解决方案通常包括许多组件，其中一些组件可以在不同任务中重复使用，利用用组件，可以极大简化开发过程。为此 Hub 应运而生。通过 Hub，开发者可以分享和发现来自官方社区的组件，只需几分钟就可以从零开始，快速搭建全新的搜索解决方案。

JCloud: Simplify deploying and managing Jina project on Jina Cloud

将 Jina 项目部署到云端，你可以使用 JCloud。它是一个命令行界面，用于管理 Jina Cloud 上 Jina 目的生命周期，Jina Cloud 是一个云主机平台，它承载着 Jina 项目，并提供免费的计算和存储资源。

NOW: One line to host them all. Bootstrap your image search case in minutes

Jina NOW 通过一行代码解决文本到图像的搜索问题。对于首次使用 Jina 的用户，它提供了更精简用户体验。

使用 Jina “全家桶”的方法众多，通常研发人员会从 DocArray 开始设计原型，然后用 Jina 把它变服务，再通过参考 Hub 上的组件加速开发进度，最后通过 JCloud 进行部署。部署后，如果对准确率 Accuracy)、精确率 (Precision) 和召回率 (Recall) 不满意，这时候就可以使用 Finetuner 进行调优。果此刻任务处理的是文本和图像，我们也可以直接用 CLIP-as-service 作为向量服务。

代码实例：搭建图片搜索引擎

矩池云现已支持 Jina 镜像，开发者进入 matpool.com，**通过「主机市场-租用-输入Jina」即可直接行。**

在这个实例中，我们使用 mini ImageNet 数据进行展示。

矩池云的公共数据集已经为大家提供了相应的数据，我们将数据复制到当前文件夹下。

```
!cp /public/data/image/mini-imagenet/train.tar .
!tar -xf train.tar
```

我们使用 docarray 库提供的 Document 对图片进行封装。多个 Document 构成一个 DocumentArray 。

```
from docarray import Document, DocumentArray
```

DocumentArray提供的from_files函数可以帮助我们快速加载文件夹下的所有图片。图片文件的位置息保存在uri属性中。plot_image_sprites函数可以预览图片。

```
data_path = 'train/*/*.jpg'
docs = DocumentArray.from_files(data_path)
print(f"{len(docs)} Documents in DocumentArray")
docs[:16].plot_image_sprites() # Preview the images
```

```
38400 Documents in DocumentArray
```

定义preproc函数用于对图片进行预处理，使用apply函数对 DocumentArray 中的每个 Document 行preproc函数。

```
def preproc(d: Document):
    return (d.load_uri_to_image_tensor()  # load
            .set_image_tensor_shape((80, 60))  # ensure all images right size (dataset image size _
hould_ be (80, 60))
            .set_image_tensor_normalization()  # normalize color
            .set_image_tensor_channel_axis(-1, 0))  # switch color axis for the PyTorch model later

docs.apply(preproc)
```

通过 Document 的 uri 信息，我们加载图片内容并以 ndarray 的格式保存在 tensor 属性中。

```
╭──────────────────────────────────── Documents Summary ─────────────────────────────────────╮
│                                                                                            │
│   Length                 38400                                                             │
│   Homogenous Documents   True                                                              │
│   Common Attributes      ('id', 'tensor', 'mime_type', 'uri')                              │
│                                                                                            │
╰────────────────────────────────────────────────────────────────────────────────────────────╯
╭──────────────────────────────────── Attributes Summary ────────────────────────────────────╮
│                                                                                            │
│   Attribute   Data type      #Unique values   Has empty value                              │
│  ──────────────────────────────────────────────────────────────────────────────────────    │
│   id          ('str',)       38400            False                                      │
│   mime_type   ('str',)       1                False                                      │
```

```
| tensor    ('ndarray',)  38400          False             |
| uri       ('str',)      38400          False             |
|                                                        |
╰──────────────────────────────────────────────────────────
───╯
```

加载 ResNet50 模型，调用embed函数计算 DocumentArray 中的每个 Document 的向量表示。这我们使用矩池云的预训练模型仓库中的 ResNet50 模型。

```
# Load ResNet50 from pytorch
import torch
if torch.cuda.is_available():
    device = "cuda"
else:
    device = "cpu"

from torchvision.models.resnet import resnet50

model_path = '/public/pytorch_models/resnet/resnet50-19c8e357.pth'

model = resnet50()
state_dict = torch.load(model_path)
model.load_state_dict(state_dict)

# Embed images
docs.embed(model, device=device)
```

每张图片的向量表示以 Tensor 格式保存在 embedding 属性中。

```
╭──────────────────────────── Documents Summary ────────────────────────────
──────────────╮
|                                                        |
|  Length                 38400                                  |
|  Homogenous Documents   True                                   |
|  Common Attributes      ('id', 'tensor', 'mime_type', 'uri', 'embedding')  |
|                                                        |
╰──────────────────────────────────────────────────────────
───────────────────╯
╭───────────────────────── Attributes Summary ─────────────────────────╮
|                                                        |
|  Attribute   Data type     #Unique values   Has empty value  |
|  ──────────────────────────────────────────────────────────
|
|  embedding   ('Tensor',)   38400          False             |
|  id          ('str',)      38400          False             |
|  mime_type   ('str',)      1              False             |
|  tensor      ('ndarray',)  38400          False             |
|  uri         ('str',)      38400          False             |
|                                                        |
╰──────────────────────────────────────────────────────────
───╯
```

构建查询用的 DocumentArray ，调用match函数在先前步骤中创建的 DocumentArray 中搜索最相的图片。搜索结果保存在 matches 变量中，使用plot_matches_sprites函数进行可视化。

```
# Match nearest neighbours
query_docs = (DocumentArray.from_files(index_data_path, size=10).apply(preproc).embed(m
del, device=device))

query_docs.match(docs,limit=10)

# Visualize the matches
query_docs[6].plot_matches_sprites(channel_axis=0, inv_normalize=True)
```