MIT导师计算机大数据-刘铭超-总结报告

刘铭超

2020.8.3

MIT导师计算机大数据-刘铭超-总结报告

一、研究背景

在大数据时代，能够开发基于大数据的人工智能微服务并提供给他人使用是数据分析从业者所必须掌握的技能，也是让一般人难以接触的人工智能算法落地实践的主要方式之一。而大多数人工智能算法的落地都会面临以下的问题：操作系统不同带来的指令差别，本地环境配置不成功。就连专业人员往往也需要花费半天甚至一天的时间才能将环境配置好，让一个程序正常执行，而这显然对于广大非专业群众用户而言是及其不友好的，诗人望而却步。就算对于专业人士来讲，团队开发时各个成员的环境同步和版本的迭代更新问题上也会因此产生许多的困难，这些问题阻碍了开发者的合作和所开发的软件走向商业化部署的路线。

docker便是为了解决这一问题而产生的。它是一个开源的应用容器引擎。使用虚拟化技术将一个软件所需要的一切文件和环境都封装到一个称为“容器”container的内部，从而使代码和代码所依赖的环境合为一体。docker容器还可以以“镜像”--image--的方式更加紧凑地存储并在云端共享（通常是dockerhub）。任何人只要成功安装了docker就会有docker daemon这个守护进程，这个进程保证了任何docker image都能被成功运行，并且使用相同image的人的运行效果都是相同的。

使用虚拟机也可以完成相同的任务，但是虚拟机占用的存储空间极大，究其原因是虚拟机作为一个完整的计算机，它的运行需要一个庞大的操作系统文件支撑。而这个庞大的操作系统对于我们的软件开发并不重要。docker优势就在于他除了软件所必须依赖的文件系统和环境以外不需要将庞大的操作系统OS也存入image中或是运行在container中。这使得docker container非常的轻量级，一台可能只能同时运行四五个虚拟机的电脑完全可以支持运行成百上千个docker container。

而谈到一个可以正常工作，服务于广大用户的服务时，我们无法避免数据库的使用。如今内存变得越来越廉价，而与此同时时间却变得比原来更加宝贵。数据库所面临的难题从过去的：“如何用更小的空间整齐地存下所有数据”，慢慢变化为了：“如何让数据库可以对用户的请求有更为迅速的反馈”。NoSQL类型数据库从此产生，其中cassandra是一个非常优秀的产品，其使用的CQL语言与业内长期使用的SQL语言比较接近，易于学习，而且它为python语言提供了非常友好的接口，是目前python大数据项目的首选。

二、项目实践

项目目的：训练一个图片分类神经网络，并将图片分类功能实现为一个网络应用。将网络应用容器化，并与数据库容器互联通信。

训练神经网络

采用tensorflow深度学习框架，使用mnist-fashion数据集进行训练，将训练得到的参数以checkpoint的形式存储在项目目录下的.checkpoint/文件夹中。得到checkpoint之后可以删去训练使用的train.py文件以减小最终成品的大小。

在predict.py中复现网络结构定义，并加载checkpoint的参数。定义predict_imgpath函数，该函数接收图片路径位置，返回图片经过神经网络判断后得到的类别名称的字符串。

网络应用设计

采用轻量级网络应用框架flask，网络结构如下

route	功能	函数名
/	重定向到/upload	index
/upload(GET方法)	渲染网络界面upload.html，提供上传图像的按钮和相应文字提示。	upload
/upload(POST方法)	获得上传图片，进行文件名合法性检查。保存到指定路径下，并调用prediction.predict_imgpath方法得到预测类别。渲染网络页面upload_ok.html，展示图片，图片类别。将数据按照规范存入cassandra数据库	upload

网页HTML书写和渲染

由于需要和用户交互操作，故书写网页蓝图upload.html和upload_ok.html分别用来展现用户上传图片前和上传图片后的页面。其中需要动态更新的部分定义合适的变量名，此后通过flask.render_template来进行更新渲染。上传文件通过html里面如下两句`<input type="file" name="file" style="margin-top:20px;"/>`和`<input type="submit" value="上传" class="button-new" style="margin-top:15px;"/>`所渲染出来的按钮即可实现。

cassandra链接

直接使用docker pull下来的cassandra-latest镜像

通信关系分析如下：

- 容器A中的flask应用需要访问容器B中的cassandra数据库
- 容器A中的flask应用的接口需要映射到本机的某个端口上，使得本机用户可以访问容器中的服务

故为该项目单独创建一个本地网络deewhy-network，网络服务主程序run_service.py所在的容器A和运行cassandra数据库的容器B均接入到deewhy-network之中，容器A通过ip寻找容器B并获取链接。容器A运行时进行端口映射，将我设置的容器内的端口8987映射到本机的8987号端口。从而实现在本机的8987号端口上访问到容器内的服务。

实践上使用的是在cassandra-driver这一提供给python的第三方库作为cassandra数据库接口。

容器化

书写requirements.txt文件，指明项目必须的环境配置

在项目文件夹下书写Dockerfile，指明所使用的基镜像，运行一些更新调整基镜像的命令，之后递归下载requirements.txt中的环境依赖，并设置启动容器时默认运行run_service.py文件。

最终呈现

使用docker network创建deewhy-network之后先后在deewhy-network运行B和容器A即可（network名称可以自由选择，这里仅仅是我的个人喜好）

（因为容器A的初始化命令需要容器B提前存在）

之后在本机上的浏览器访问localhost:8987即可通过简单的鼠标点选上传图片并运行图片识别功能，并返回一个易于阅读的反馈页面。通过命令行查询cassandra所在容器也可以看到历史上所有的上传记录。

三、学习感受与收获

容器化的过程就像是软件的标准化。这种通过docker基镜像生成的容器于所有docker用户而言都是下载即可使用的，并且他人更是可以进一步把任何镜像作为基镜像进行进一步的开发。

在亲身经历过各种人工智能项目繁琐的环境配置以及因为机器硬件不同所带来的问题之后，docker容器和镜像这种软件储存和传播的途径让我体会到了软件开发和算法落地的大趋势。虽然自己在docker build的过程经历了很长时间的找错和调试，但是最终展现上前所未有的简洁性证明了搭建容器所花费的精力是值得的。而且在这个过程中并没有为了使用docker而大量返工源代码，整体体验非常好。

在本次项目中综合了大量此前在课内外学习过的知识，让我对自己的知识体系进行了一次更为系统化的重构。从机器学习的理论和框架使用开始，到书写了交互式的网页前端内容的书写，进而实践了容器间通信和非关系型数据库Cassandra的数据存取，让我亲身体会到了一个更为完整的计算机技术栈大致的样貌。与此同时也熟练使用起了版本控制工具和容器管理工具，使得自己管理工程进度的手段更为丰富和优雅，在开发过程中遇到问题时可以井井有条地进行不同的尝试，这从根本上增强了自己动手解决问题的效率。

同时，这也是我第一次进行微服务app的开发，这次四个星期的经验和教训为我将来进行更加深入的开发和研究打开了一条通路，同时作为一名计算机大数据专业的学生，此后我也会从一名开发者的视角来寻找当下的技术痛点，以此作为理论和科研方向的指导，让自己的科研更具有现世价值。