**DLP Lab01 实验报告**

计算机科学与技术学院 PB20050978 关浩祥

一、任务说明

本次实验我完成了参考题目2 —— 输入一个计算机类文章的PDF文件（实际实现中需要规定输入的是不含附录的CVPR文章），解析出所有的参考文献并显示给用户，用户可以选择任意一篇文章下载其对应的BIB文件（从DBLP下载）。这些功能都封装在一个Python包中，利用Flask部署在后端，前端使用Vue与后端通信并将结果渲染在网页上构成与用户交互的图形化界面。

二、实验细节

（一）文章PDF解析类

用户上传文章的pdf文件后，首先需要解析出其中包含的参考文献。经过调研后，决定使用PyMuPDF包对pdf文件进行解析，并将解析功能封装为名为PaperPdfExtract的类。

首先利用\_get\_ref\_pages函数获取含有参考文件的pdf页面，思路是搜索关键词“References”、“REFERENCES”、“referenCes”。其中使用到了PyMuPDF包的fitz.open()函数，函数接受pdf文件路径，返回一个可迭代遍历的包含pdf每一页的对象。

然后利用\_get\_ref\_text函数将References关键词后的参考文献提取出来，返回含有参考文献字符串的列表。

这时遇到一个难点，PyMuPDF提取出的每一个参考文献可能只是一个完整的参考文献的一部分字符串。所以，要解决该问题，就需要考察提取出的每一个参考文献字符串，判断其是一个完整的参考文献字符串，还是其中的一部分。若是其中的一部分，则需要与其后的剩余部分进行拼接，组合成完整的参考文献字符串。将这一部分功能封装在\_get\_split\_ref\_text函数中。



最后，需要提取出参考文献中的作者，以供爬取参考文献的BIB文件使用。注意到参考文献大部分的作者都以“，”进行分隔，则我们可以从左至右扫描参考文献字符串，以逗号分隔将作者提取出来。注意到参考文献中论文名称等部分也会有逗号，容易“误提取”，所以我们设定作者名称的最大长度为20，基本上可以避免“误提取”的发生。将上述功能封装为一个方法。



以上便是PaperPdfExtract类的所有功能。

（二）参考文献爬取类

此处一开始遇到的困难是难以从参考文献字符串中提取出文献名称这一部分，原因是参考文献字符串难以通过一些的特定的分隔符将其分割为不同部分，也难以编写对应的正则表达式。因此，代码实现思路并不是直接在dblp中搜索论文名称，而是搜索PaperPdfExtract类提取出的参考文献作者，在作者的所有论文著作中寻找目标。这些功能封装在了RefPaperCrawler类中。



该类最主要的方法是get\_bib(self, no)，传入参考文献的编号，返回对应的BIB。实现的思路是，考察每一个作者名字在dblp上的检索结果，获取包含“Exact matches”的检索结果。然后获取该精确匹配作者的论文页面，获取其中的每一篇论文的名称。

如何知道获取的某一篇论文名称正是需要寻找的目标？由于此时获取的论文名称与对应参考文献字符串中包含的论文名称并不一定完全一样（即便正是需要的目标，也可能会有一两个字符的差别），所以我们采取子串相似度匹配的思路。顺序截取参考文献字符串的每一个长度与待匹配论文名相同的子串，将二者进行相似度匹配，若某一子串的匹配相似度大于阈值（0.95），则认为匹配成功，直接下载这篇论文的BIB文件。上述相似度匹配使用到了difflib包，difflib.SequenceMatcher(None, s1, s2).quick\_ratio()返回s1与s2两个字符串的相似度。

上述名称检索、页面获取、BIB获取都使用到了requests与BeautifulSoup包，便于html的解析与元素的获取。目标的class、id等属性名的获取使用了浏览器的检查网页源文件功能，便于定位目标元素。

（三）Flask后端设计

Flask进行后端设计，只需要设定对应的路由与对应的响应函数即可。一共提供了两个函数，upload\_and\_analyse获取上传的pdf文件，调用PaperPdfExtract类解析pdf，返回json格式的参考文献列表；get\_bib函数根据POST请求包含的论文编号调用RefPaperCrawler类的get\_bib方法返回json格式的bib字符串。

（四）Vue前端设计

Vue的MVVM特性便于数据的动态渲染，从而可以动态生成表格，这极大方便了前端根据后端返回的参考文献列表动态渲染html展示在网页上。在实现中，使用了基于Vue的Element UI框架进行表格的设计。



与后端的通信使用了axios网络请求库，上传时将pdf文件放在post请求的body中，后端以json返回数据。



为区别不同用户上传的不同pdf，利用js的Math.random().toString(36)方法为每一个用户的每一次上传生成一个随机码，以此区别不同的上传，便于后端知道该返回哪一篇论文的参考文献的BIB文件。

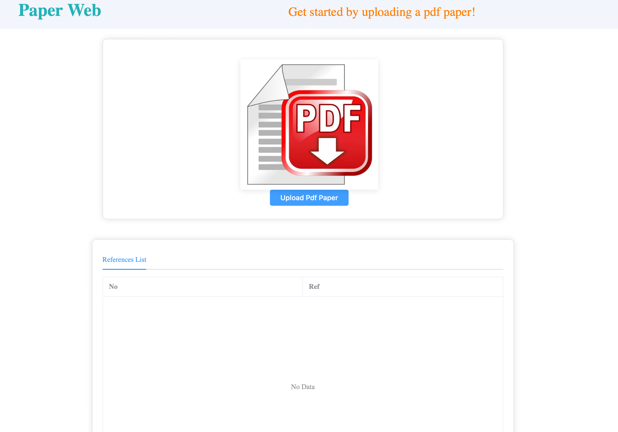
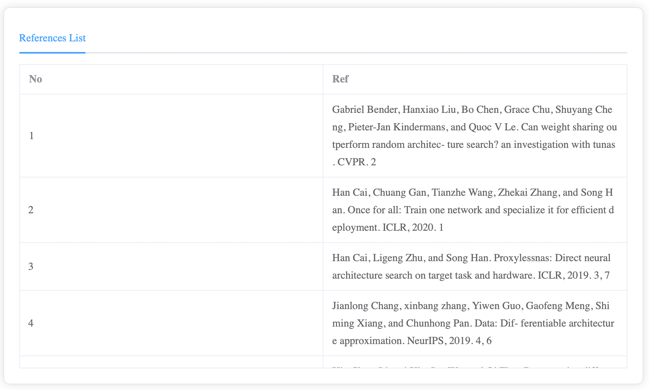
三、实验总结

将前后端部署后便可进行测试。测试后发现，所有参考文献都被正确地解析并显示出来，绝大部分的参考文献都可以获取对应的BIB文件，一小部分文献因为使用上述方法无法将作者名解析出来（不符合以逗号分隔的特性），所以无法获取其BIB文件，该漏洞有待日后完善。前后端通信功能测试良好。解析pdf并返回参考文献列表可在一秒内完成，检索并获取一个参考文献对应的BIB文件的平均可在20秒内完成（会受网络状态的影响）。

本次实验给了我一个锻炼自己前后端分离设计并合并的机会，让我对前后端设计的理解加深了。同时，BIB文件的爬取也锻炼了我分析页面DOM结构的能力，增长了我关于各类型http请求的知识。实验过程中遇到了不少困难，但通过查阅各类资料最后也都解决了，这也提高了我自主学习、查阅资料的能力。虽然过程非常的苦，但最后能做出一个比较完整的demo还是很有成就感的。

四、界面布局

前后端部署后，打开<http://localhost:5173/>，可以看到主界面（左）：

上面提供按钮让用户上传文件，底下是展示参考文献的表格。

上传文件后如右图所示，表格被更新为参考文献数据，由于文献很多，所以表格采用固定大小的显示方式，用户可通过滑动方式查看后面的参考文献。

当用户想要下载某一参考文献的BIB文件，只需点击表格对应行，便会出现等待约20秒提示，然后等待过后便会将文件下载到本地。

