|  |
| --- |
| **评阅成绩** |
|  |



**Northwestern Polytechnical University**

**计算机学院生产实习报告**

|  |  |
| --- | --- |
| 专业名称 | 计算机科学与技术 |
| 实习题目 | chatbot聊天机器人 |
| 姓 名 | 袁粼 |
| 班 级 | 10011804 |
| 学 号 | 2018302344 |
| 实习时间 | 2021/07/05 - 2021/07/16 |
| 指导教师 | 邱晨 |

2021 年 7月 15日

1. 实习名称

构建自己的聊天机器人

1. 实习目的

通过构建出具有特色的聊天机器人，经历和实践一个项目的完整开发过程，培养团队合作能力和协作精神。并初步具备应用软件或系统软件的设计与开发能力、计算机算法的设计与优化能力以及运用机器学习、自然语言处理等主流计算机科学技术分析解决实际问题的能力。

1. 实习内容
2. 分组、项目简介、安装并在本地运行chatterbot web app
3. 学习和使用Python爬虫、正则匹配
4. 学习、安装并配置Rasa，得到能进行对话的Rasa聊天机器人
5. 安装、学习使用Docker
6. 在Docker内实现第一天的聊天机器人，从宿主机上访问
7. 各个小组制作NLP相关内容的ppt并进行演示
8. 学习意图识别相关内容，完善开发中的项目
9. 学习命名实体抽取，进一步完善开发中的项目
10. 完成项目的总装与测试，编撰文档
11. 项目演示与比赛
12. 实习过程

第一步是配置环境。需要配置的包括python、flask、chatterbot、rasa以及docker。主要在实习的第一周完成。接着是项目各个模块开发过程，将占用三至四天。然后是项目各个部分的总装与测试环节，耗时一至二天。最后是最后一天的项目验收。

环境配置：

python环境：在项目开始前本地机器已经安装并配置了python3.8，但该版本的python与第一天要演示的chatterbot不兼容。故从Microsoft Store中下载了python3.7，由于Windows Appx的特性，两个版本的python可以共存，使用python或python3.8调用python3.8，python3.7调用python3.7.9，但同样由于Appx的特性，后面做了大量无用功在尝试修改Python安装文件夹及其内容的访问和写属性上。项目的后续开发都将在python3.7.9上进行。双python3导致现有的图形调试界面不能正常工作。

chatterbot可以通过pip来安装。但chatterbot所需的spacy不会自动安装，因此需要手动配置spacy。安装好spacy后提示需要安装spacy的英文语言包en，从spacy官网上下载兼容版本的en\_web\_core\_sm.tar.gz,第一次安装时尝试将安装包解压再运行setup.py，但是总是报错。检查命令行backtrace，发现错误出在目标文件不能写入上。在3个小时尝试对python文件夹属性的修改无用后，偶然发现指令pip install xxx.gz会自动解压调取setup.py进行安装配置。由于pip3.7属于python3.7安装文件夹的完全控制权限用户‘Trusted Installer’，终于顺利安装上了en\_core\_web\_sm。在此之后打开带有管理员权限的命令行（由于接下来的指令需要调用系统的mklink /d）运行指令python3.7 -m spacy link en\_core\_web\_sm en --force，将spacy的默认en链接到en\_core\_web\_sm。最后安装剩余的依赖包，成功在本地机上运行了chatterbot demo。

实际上pip下载到缓存里以备安装的whl (wheel) 文件本质上都是gzip压缩包。这点可以从部分提供whl下载的官网上与其相邻等价的gz下载项和whl大小相同上发现。因此whl的安装过程是使用gunzip指令解压，再执行setup.py。推测pip安装依赖包经过这几个过程：分析出要安装的库和版本后，根据python版本，首先从可更新的资料库中查找其依赖库及其依赖库，其次从指定或默认的数据库中下载需要安装的包的安装文件，最后按照依赖关系从最不需要依赖的末端开始安装，并将安装记录添加到pip的本地数据库内。

coronabot的对话采取一问一答的形式。其网页代码的构造不算复杂，但有许多不方便修改和可以造成格式不统一或需要来回反复修改之处。第二天对chatterbot的训练文本进行了一定程度的修改。并通过提问聊天机器人的回答得到的是训练文本中的下一句发现chatterbot的本质是通过对输入语句进行解析，匹配训练数据中的下一文本。为避免重复计算，将一部分固定不变的中间结果写入sqlite database中，chatterbot的模型完全存储在这个数据库内。对chatterbot demo和database.sqlite3的进一步解析显示，chatterbot库使用了基于spacy的模糊查找，同时能识别空格划分的数字表达式然后输出答案（避免输入'(\_)'）。它读取训练数据组成一个长文本(str类型)file，然后训练产生tag、relation两张表保存到sql数据库里。应答的原理是遍历整个tag表找到输入对应的最大似然估计，提取tag号，再以此为依据在relation表中查找应答的tag号，返回tag表找出应答字符串返回。因此如果输入被对应到了某个‘答案’，那chatbot会回应这个答案在训练文本中的下一句；如果答案有多个下一句，则会返回训练数据文本文件中最前面的下一句。通过少量的训练，chatterbot出现‘具备少量记忆’的特征。同时，chatbot的name属性将影响其所能触及的域：名为huming与Coronabot的实例相互间似乎是透明的。

scrapy的配置、运行很顺利。爬虫利用python的网络通信，并行地从网站上抓取页面，提取特征后抓取它指向的页面或根据一定规则生成的url的页面，重复上述功能。简单来说是在以url为节点，超链接为边的有向图上的广度优先搜索，但指向外部域名的分支通常会被剪去。

接下来是rasa的安装配置。使用pip安装的rasa无法运行，并导致之前的coronabot报同样的错误："OSError: [WinError 126] 找不到指定的模块"。分析了一系列可能的原因、经历了numpy、spacy重装、更换版本、使用numpy+mkl后，终于发现是当前版本的spacy与安装的python不兼容的问题。使用rasa允许的最低版本spacy==1.4.1后成功解决。问题的解决过程比较曲折，并且使用到了相当多的命令行调试，加之对python语言的不熟悉以及Appx版本的python3.7只能用pip进行修改等诸多弊端，更加剧其曲折程度。最后的问题发生在scipy的linalg\\_\_init\_\_.py上。根据github上对numpy与Store python不兼容的讨论，确定了问题是\_distributor\_init.py没能在这个版本的python上正确运行导致的。对其适当修改后rasa就能正常工作了。



后面两天配置了docker，并成功运行起了第一天的演示样例，从浏览器访问127.0.0.1:5000访问到了docker内运行的flask服务器。由于docker环境采用了标准的Linux系统，安装配置都很顺利。在这个过程中发现chatterbot实际上可以兼容python3.8及以上版本：chatterbot不兼容python3.8的原因是其依赖包spacy随着en链接的取消只有较低版本支持chatterbot，而低版本的spacy没有python3.8的安装包；问题的解决方案是修改chatterbot库里的源代码文件：/usr/local/lib/python3.7/site-packages/chatterbot/tagging.py，错误是其使用了nlq=spacy.load("en")而非spacy.load("en\_web\_core\_sm")，但仅有后者被spacy3.1所支持。将第13行的"self.nlq = spacy.loac(self.language.ISO\_639\_1.lower());"修改为"self.nlq = spacy.loac(self.language.ISO\_639\_1.lower()+'\_core\_web\_sm');"即可。由于中文的chatterbot的CPU计算版本为zh\_core\_web\_sm，这个修改使其同时也支持了中文的chatterbot与python3.8的兼容。

第六天进行了词向量化相关内容的演示。除此之外，第六到八天对第一天的demo的网页部分进行了详细解析，并学习了很多网站开发相关的内容以及HTTP相关的理论知识，且对python语法有更进一步了解。

python早期没有装饰器语法，但是有相似的概念。即定义一个函数后，使用参数为函数，返回函数的函数，将前者赋值为以前者为参数的后者的调用的返回值。由于python内一切对象，后者函数可以对前者在调用前后作修饰而不影响后者的功能本身，而这个变化只体现在后者的声明和被调用上，适合被理解为后者对前者做了一些装饰，故将def func1 ...; func1=func2(fun1);固定为语法@func2; def func1...;取名装饰器。

查看templates文件夹中的网页代码，发现向屏幕上添加气泡的过程体：

$.get("/get", { msg: rawText }).done(function (data) {...});

内存在以下对应关系：

格式：$.get(route, {vararg1: vararg2}).done(function (args) {...});

route: 用双引号标注的url，如"/get"相当于函数运行于"127.0.0.1:5000/get"；

vararg1: vararg2: 第一个参数代表其在python代码中的名称，第二个参数属于一个枚举，表征第一个参数的类型；

args: flask中该路由的函数的返回值

可认为"$.get(...)"可存在多个实现。经实践得出结论：多个"$.get(...)"若采取不同的url则可以并存且点击Send将使其全部被触发。这意味着在这种规模的网页应用中，一次请求可以触发多次响应。顺藤摸瓜，找到了JavaScript中的eval方法，作用是将json格式字符串转换为对应的数据结构。这成为了后来输出多个回复框，在此之上进一步对index.html、app.py进行改造的基础，为style.css的解析留下了时间。

第八天成功让chatbot输出的图片在网页中显示了出来，预定的所有功能达成。这意味着源代码的基本结构、通信机理基本上固定了下来，之后只有小幅的修改完善和界面美化工作了。

所有源代码编写完成后，绘制了一些符合项目主题的svg图片，并修改了style.css进行页面美化。替换了rasa模型。

至此，项目已经可以演示了。

1. 心得体会

为期十二天的实习终于引来了尾声。这段时间给我最大的感悟是千万不要用Windows做开发，否则不仅会遇到各种莫名其妙又极具个性的问题，还有可能损坏已有的开发环境，为遇到更多问题埋下种子。例如最初配置rasa时最后遇到的一个问题中，通过命令行pdb找到报错的地方位于\_\_init\_\_.py，根据错误函数内的各种参数，再结合为解决scipy不能运行而浏览的GitHub上numpy和Appx版本python3.6不兼容问题的讨论帖，最终确定问题原因和解决方法和\_distributor\_init.py相关。

与简单明了的C/C++不同，python用添加大量的特性和隐含的条件的方式换取了日常意义下的便利，导致python更简单便利的入门和更复杂繁琐的精通。

因此用python进行团队项目开发，必须注意规范。开发团队项目需要注意的另一要点是和其他成员的配合与交流。除非是完全不相关的领域，不经历互动和协作，项目有好成果的概率接近于零。

1. 附上实习单位鉴定意见（需要指导老师签字）

**要求：正文4号宋体，1.5倍行距。提交报告以班号-学号-姓名命名。**