

# Python程序设计报告

设计题目： 小说云图

学生姓名： 徐辙

专 业：计算机科学与技术

班 级： 19-3班

学 号： 2019218042

指导教师： 马学森

完成日期： 2021.1.3

# （一）需求和规格说明

**问题描述：**这是一款了解小说主要任务或事件的软件。网上有对应小说网站，可以爬取到相关数据。用户输入小说的名字，软件会自动找到指定的小说，然后爬取相应数据，再进行解析，最后生成一张云图。

**编程任务：**

1. 用户输入小说，软件进行爬取。
2. 爬取网站小说章节链接。
3. 根据小说章节链接获取相应内容。
4. 对所有章节的内容进行整合。
5. 将小说内容的高频词汇生成云图。

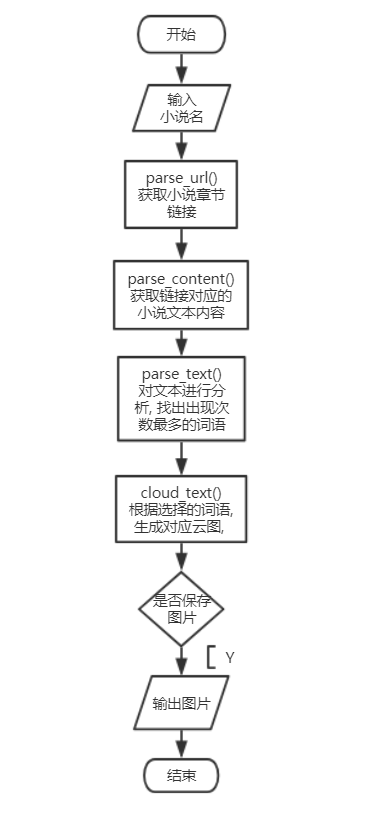
注：由于爬取需要耗费时间，故只选取了其中的部分章节进行爬取。由于部分原因（网络），首次爬取可能失败，需要进行第二次爬取。

# （二）设计

## 1．设计思想

本软件主要考察爬虫，api的调用，列表和字符串的相关操作等。因为需要获取输入的小说名，然后进行选择性爬取，故和一般的爬虫不太一样。这里选用Web自动化工具selenium进行爬取，其能够模拟用户的输入和点击等功能。在用户输入小说名后，会跳转到另一个页面，这就需要切换页面，后面，就能够找到小说章节页。在该页面上，爬取小说每一章节的链接。获取到链接后，用requests库中的requests.get(url)方法，能够获取到链接网页的内容。对内容进行解析，可以使用bs4，lxml……在这里，我选用lxml进行文本解析。我对小说页面中的主题内容进行保存，因为链接有很多，而etree.HTML()返回的是数组，故用””.join()将数组连接为一个字符串，并且注意处理无关字符和编码格式，最后将文本数据以字符串形式返回。获取字符串后，调用jieba库中的lcut()函数，返回一个字符串列表。然后用字典保存每个词语出现的次数，且按字典值进行降序排列，然后以列表的形式返回出现次数最多的20个词语。在生成词云时，首先选择屏蔽词，然后用wordcloud库中的方法生成词云对象，再结合matplotlib库中的方法将词云进行展示，用户可以选择保留图片。整个程序大致是一个全自动的过程，符合当下的软件发展形势，软件只需要用户输入想要查看的小说名，即可获取到对应小说的词云图。

## 2. 设计表示

1. 流程图：
2. 函数：

|  |  |  |
| --- | --- | --- |
| **函数名称** | **输入参数** | **描述** |
| parse\_url() | text\_name: 小说名 | 爬起指定小说的章节链接 |
| parse\_content() | url\_list: 章节链接列表 | 根据章节链接, 获取该链接包含的文本信息 |
| parse\_text() | texts: 文本内容 | 根据文本内容, 进行分词, 然后统计出现次数最多的词语 |
| cloud\_text() | text: 词语列表  text\_name: 小说名 | 根据词语列表输出词云图  根据小说名确定图片名 |

## 3. 核心算法

程序的核心部分是几个函数，即parse\_url()，parse\_content()，parse\_text()，cloud\_text()。

第一步：用户输入想要查看的小说名。

第二步：调用parse\_url()函数。本函数主要是通过Web自动化工具selenium进行爬虫。首先对Chrome进行参数配置，使用无头模式。然后driver.get(url)访问网址。在页面中，根据find\_element\_by\_id()找到输入框，调用send\_keys()方法将小说名送入输入框，再根据find\_elements\_by\_xpath()找到搜索按钮，此处会返回列表，选择第三个，调用click()方法，然后会跳转到下一个页面。跳转到下一个页面，即switch\_to.window(driver.window\_headless[1])跳转到新打开的页面。在第二个网页中找到小说名，然后进行点击，跳转到第三个网页。在第三个网页上，定位每一个章节的位置，能获取到跳转链接href属性，因为链接较多，所以用一个列表进行保存，即url\_list.append()方法，然后返回url\_list。

第三步：调用parse\_content()函数。本函数主要是通过requests库发送request请求。首先要设置headers，配置请求头，否则会被当做爬虫而被反爬。然后遍历url\_list，对每个请求进行requests.get(url)。然后通过etree.HTML(r.text)获取网页源码，此处需要处理中文乱码问题。然后通过tree.xpath()找到文章主题内容，此方法返回的是一个列表数据，需要通过””.join()对其进行拼接转换为一个字符串，此处需要处理字符乱码，即一些空格和回车之类的，需要用到replace()函数，最后返回texts。

第四步：调用parse\_text()方法。本函数主要是通过jieba库对文本进行分词。首先通过jieba.lcut()方法，即在精确模式下，返回一个分词列表。定义一个字典，然后遍历列表，去除长度为1的词语，通过counts[word]=counts.get(word, 0)+1方法对词语出现的次数进行统计。再将字典转化为列表，字典的键值对转变为元组的形式。通过列表的sorted内置方法，加上lambda表达式取元组的第二位排序，加上设置reverse=True进行降序排列。最后返回items。

第五步：调用cloud\_text()函数。本函数主要是通过wordcloud库和matplotlib库生成词云。首先设置需要屏蔽的词语，然后定义字典wc，定义词云字符串，然后通过generate()方法生成词云对象，再通过figure()、imshow()和show()等方法对词云进行绘制。

# （三）用户手册

因为本程序是一个自动化的程序，符合时代趋势，即自动化和智能化。只需要用户输入想要查看的小说的名字，程序便会自动在网站中寻找相应的小说，获取对应章节链接，然后对其呢绒进行爬取，然后在后台进行解析，最后生成一张云图。

用户可以根据喜好，自主选择是否保存图片。当输入Y、y时，图片保存到本地。

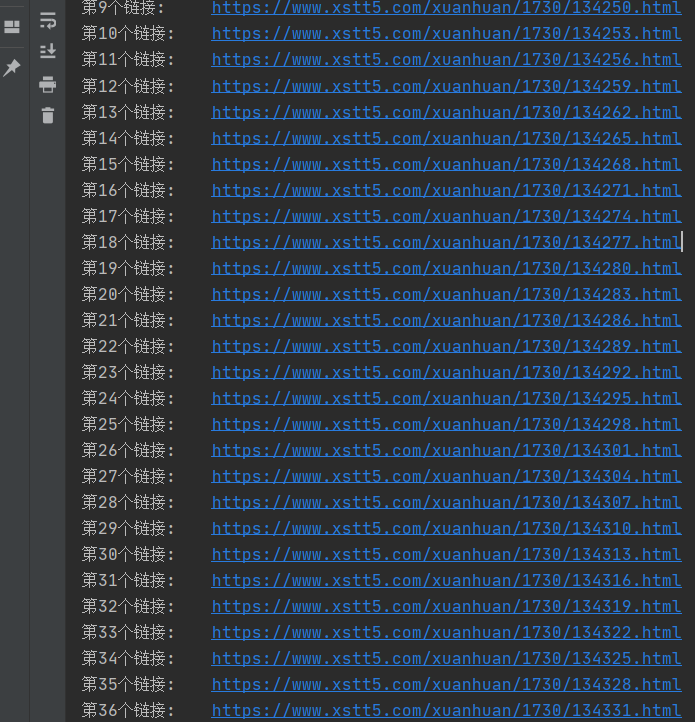
本程序因为是通过爬虫进行爬取，可能会出现小说搜索不到的情况，这里推荐已经通过测试的《斗罗大陆》、《绝世唐门》、《大主宰》、《武动乾坤》和《大主宰》。另外，可能会因为网络波动或者其他原因，导致访问失败的情况，请重新测试。最后，因为是爬取网站内容，当访问过急，可能会出现IP被封的情况，请注意使用。

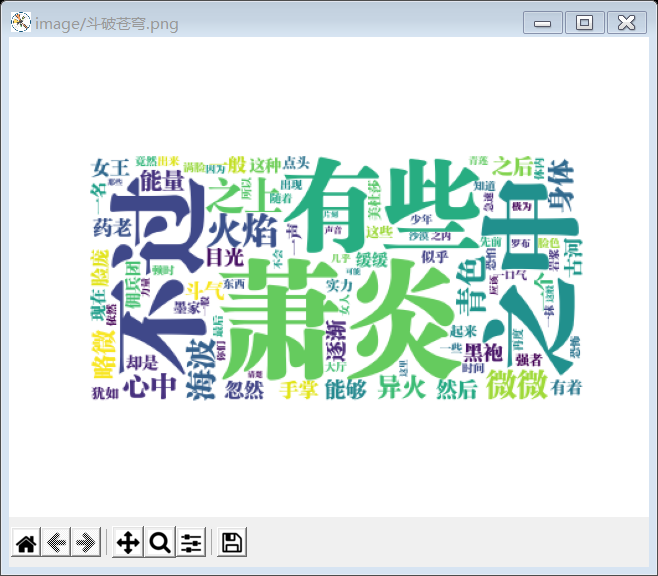
# （四）调试及测试

## 三国演义.txt - 记事本1. 测试数据：

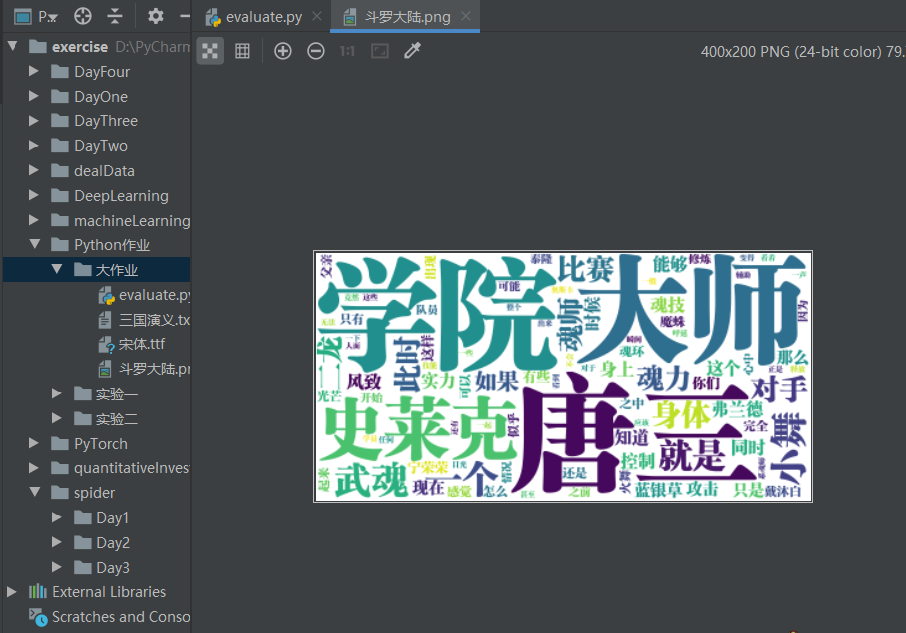
**本地小说**

## 2. 测试结果：

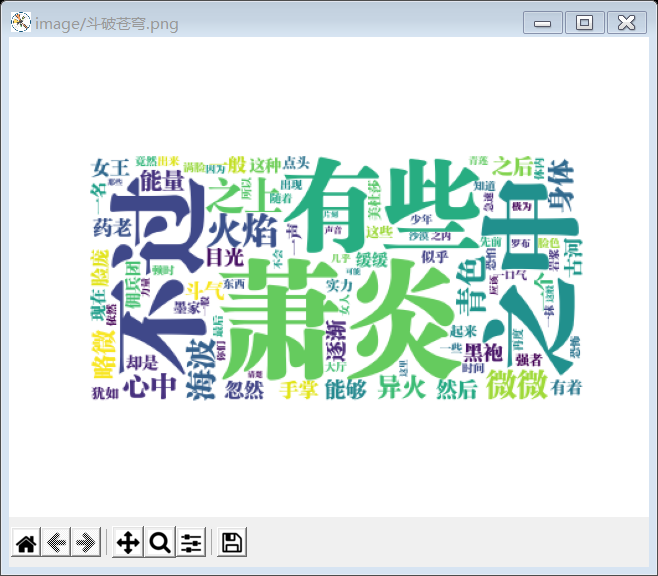
**输入小说**

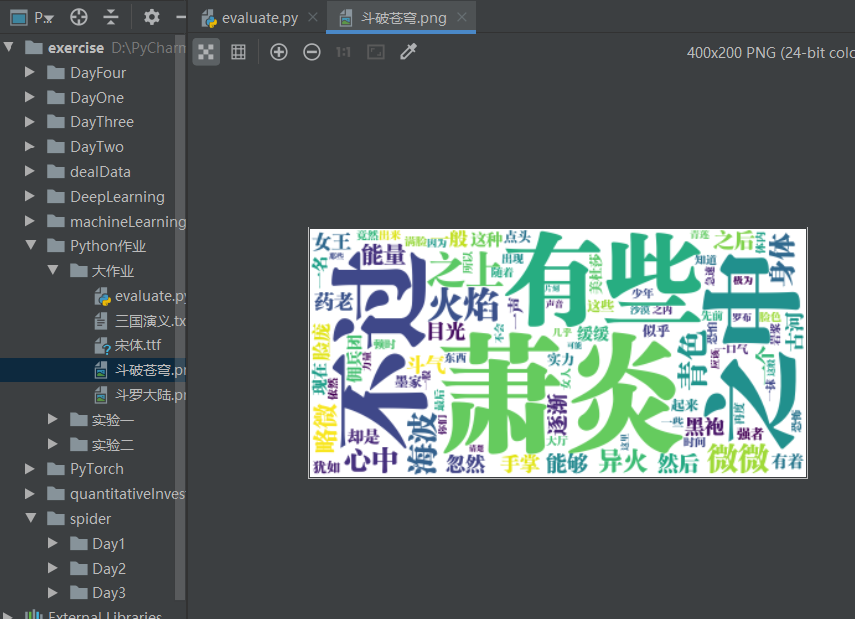
**章节链接**

**词语排名1  
**

**词云1**

**词云图片1**



**词云2**

**词云图片2**

## 3. 进一步改进：

（1）因为网络爬虫的速度取决于多方面的原因，原本打算用多线程进行爬虫，但后来因为时间和能力原因，导致了此想法的搁浅，希望在以后能实现此功能。

（2） 可以先自定义一些默认常见的小说文本，就可以避免爬取而耗费时间，目前常见小说库中只有《三国演义》，后续可能会添加其他小说。

（3）小说文本存放在本地不太方便，可以考虑存储到数据库中，我也有考虑到存储到腾讯的云数据库中，但因为时间问题，此想法暂时搁浅，以后会进行增加。

（4）程序应该可以和用户实现更多的交互，比如返回更多的小说信息，如作者简介，读友评价，评分……读友评价也同样可以做成一张词云图。与用户有更多的交互，可以给用户更好的体验。

（5）程序其实可以和网站进行配合，在网站上显示一些信息。我只能用html或者jsp做一些简单的网站，python暂时不会。

# (五) 感想

这道题是我的自拟选题，是根据我所掌握的Python知识所编写的，因为我在以前学过一些有关Python爬虫和深度学习的相关知识，所以我做的内容可能和其他同学略有不同，不过可以保证，均为自己编写，无抄袭现象。

一开始，在编写爬虫代码的时候，如果是爬取一些网站上面的内容还比较简单，但涉及到输入和点击的时候，普通的方法便不再适用，所以我采用了Web自动化工具selenium，其能模拟用户的操作，只需要通过id或class就能定位到组件，然后进行操作，并且我将其设置为了无头模式。在其中，我也遇到了一些问题，如点击之后网页进行了跳转，如果我不进行任何操作，则还是在原网页进行操作，我需要跳转导航栏。其次，find\_element\_by\_xpath()和find\_elements\_by\_xpath()看似只有一字之差，但实现的功能却有所差别，比如我定位到的地方有多个相同元素，我就需要使用后者来定位，然后其会返回一个列表，然后可以自己进行选择，而第一个函数只会返回第一个元素，这个坑当时想了许久。

然后，在编写页面解析代码的时候，首先应该配置一下headers，否则别人一看你就是爬虫，就会有相应的反扒措施。然后，对于xpath语句，我有些生疏，所以我又简要地学习了一下xpath，之后找元素就比较顺利了。当用etree.HTML()获取网页内容时，发现出现了乱码，根据经验，一定是中文字符的编码有问题，所以就回到原网页查看了header中的encoding，然后在代码中进行相应的修改，r.text.encode('iso-8859-1').decode('utf-8')。然后获得的文本中有部分空格和制表符，我通过replace(u'\xa0'\*4, u' ')进行修改。

然后，在编写解析文本代码的时候，首先用jieba.lcut()进行精确模式的分词，然后用老师上课讲的，用字典和counts.get(word, 0)+1的方法，保存每个词语出现的次数。然后用将的sorted方法、lambda表达式和reverse按字典值进行降序排列。

最后，在编写生成云图代码的时候，因为以前matplotlib库用的比较多，所以我并没有遇到什么问题，只需要注意记得要调用show方法，否则无输出，这是我第一次用的时候犯的错误。

总而言之，这次的Python大作业让我将以前学习的只是进行了一次有效的梳理。我能够更加熟练地运用列表的sorted方法，我也更加熟悉字符串的一些操作。对于api的调用，我也能更加得心应手。对于爬虫的步骤，我又进行了一遍梳理，更加熟练。此次也新学了jieba库和wordCloud库，觉得做出来的东西也挺有意思。然后就是matplotlib库，这个库很重要，在诸多场合都有作用，比如当深度学习跑完了，进行损失函数的展示等。

最后，这次大作业让我增加不少知识，感谢老师的教导。

# 附录：

import jieba  
import matplotlib.pyplot as plt  
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator  
from selenium.webdriver.common.keys import Keys  
from selenium import webdriver  
from selenium.webdriver.chrome.options import Options # 引入Chrome的配置  
import time  
import requests  
from lxml import etree  
  
  
def parse\_url(text\_name="斗罗大陆"):  
 *"""  
 爬取指定小说的所有章节链接* ***:param*** *text\_name: 小说名称* ***:return****: 章节链接  
 """* # 配置  
 chrome\_options = Options()  
 chrome\_options.add\_argument("--headless") # 把Chrome设置为无界面模式  
 chrome\_options.add\_argument("--disable-gpu")  
 chrome\_options.add\_argument("--window-size=1920,1080") # 设置全屏  
 driver = webdriver.Chrome(options=chrome\_options)  
 # driver = webdriver.Chrome()  
  
 driver.get("https://www.xstt5.com/")  
 # print(driver.page\_source)  
 # 1. 在输入框中输入小说名  
 driver.find\_element\_by\_id("bdcs-search-form-input").send\_keys(text\_name)  
 # 2. 点击搜索按钮  
 # 注意find\_elements\_by\_xpath返回的列表才能索引  
 driver.find\_elements\_by\_xpath("//div[@class='search']/form[@id='searchform']/input")[3].click()  
 # 3. 跳转到第二个页面  
 driver.switch\_to.window(driver.window\_handles[1])  
 # 4. 点击小说名  
 driver.find\_element\_by\_xpath("//div[@class='sbintro']/h3/a").click()  
 # 5. 跳转到第三个页面  
 driver.switch\_to.window(driver.window\_handles[2])  
 # 6. 访问每个章节  
 content\_list = driver.find\_elements\_by\_xpath("//div[@id='chapterList\_1']/ul/table/tbody/tr/td/ul/li/a")  
 # print(len(content\_list))  
 # 7. 获取每个章节的文章内容  
 url\_list = []  
 for content in content\_list[66:88]:  
 url\_list.append(content.get\_attribute("href")) # 获取文章链接, 通过driver返回, 不能直接取出, 而需要用get\_attribute()函数或text属性  
  
 return url\_list  
  
  
def parse\_content(url\_list):  
 texts = ""  
 headers = {  
 "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.25 Safari/537.36 Core/1.70.3676.400 QQBrowser/10.4.3505.400"  
 }  
 i = 0  
 for url in url\_list:  
 print(f"第{i}个链接: \t", url)  
 i += 1  
 # 发送请求, 获取响应  
 r = requests.get(url)  
 # print(r.text.encode('iso-8859-1').decode('utf-8'))  
 # 解析内容  
 tree = etree.HTML(r.text.encode('iso-8859-1').decode('utf-8')) # 进行gbk->utf-8的编码转换, 否则出现乱码  
 # 获取文本  
 content = tree.xpath("//div[@id='main']/div[@class='zw']/text()")  
 text = "".join(content).replace(u'\xa0'\*4, u' ') # 对文本无关数据进行处理  
 # print(text)  
 texts += text  
  
 return texts  
  
  
def parse\_text(texts):  
 # text = open(text\_path, "r", encoding="utf-8").read()  
 words = jieba.lcut(texts) # 精确模式, 返回一个列表  
 counts = {} # 用字典来存储词语及出现次数  
  
 for word in words:  
 if len(word) == 1:  
 continue  
 else:  
 counts[word] = counts.get(word, 0) + 1  
  
 items = list(counts.items()) # 以元组的形式组成列表  
 # print(items)  
 items.sort(key=lambda x: x[1], reverse=True) # 按字典中值的大小降序排列  
  
 # 取出出现次数排名前10的词语  
 for i in range(10):  
 (word, count) = items[i]  
 print(f"{word:<5}{count:>5}")  
  
 return items[:100]  
  
  
def cloud\_text(text, text\_name):  
 # 选择屏蔽词，不显示在词云里面  
 stopwords = set(STOPWORDS)  
 # 可以加多个屏蔽词  
 stopwords.add("我们")  
 stopwords.add("没有")  
 stopwords.add("自己")  
 stopwords.add("他们")  
 stopwords.add("已经")  
 stopwords.add("不是")  
 stopwords.add("虽然")  
 stopwords.add("什么")  
 stopwords.add("便是")  
 wc = WordCloud(  
 background\_color="white",  
 max\_words=100,  
 max\_font\_size=100,  
 stopwords=stopwords,  
 font\_path="宋体.ttf" # 注意: 原始字体不支持中文, 不引入字体将会是空白  
 )  
 result = [i[0] for i in text] # 获取元组列表的第一个元素  
 names = " ".join(result) # 转化为字符串  
 # print(names)  
 wc.generate(names)  
 plt.figure(f'image/{text\_name}.png')  
 plt.imshow(wc)  
 plt.axis("off")  
 plt.show()  
 choice = input("是否保存图片(Y/N)?")  
 if choice == "Y" or choice == "y":  
 wc.to\_file(f'{text\_name}.png')  
  
  
def main():  
 text\_name = input("请输入文章路径: ")  
 url\_list = parse\_url(text\_name)  
 texts = parse\_content(url\_list)  
 text = parse\_text(texts) # 解析文本  
 cloud\_text(text, text\_name)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()