南 京 财 经 大 学

2020—2021 学年第 1 学期

**基于python的豆瓣网读书爬虫设计 课程考察题目**

校区仙林校区专业年级 计算机科学与技术班级\_\_\_学号**\_\_\_1120201123** 姓名**\_**陈晓晓**\_**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 指 标 | 一 | 二 | 三 | 四 |  | 总分 |
| 分数 |  |  |  |  |  |  |

使用 python 开发完成以下程序的设计，提交系统设计文档，说明文档，使用文档，测试文档，并在 github 提交可运行的 python 源代码。要求同时提交纸质版项目材料，包括 github 首页情况，主要代码节选，项目文档，测试运行结果等主要部分，用于归档评分，并以本试卷作为封面，装订相关文档提交。

可选题目：

1. 基于 flask 的电子商务网站服务，要求提供用户注册，信息显示，用户发表评论，功能

完整，界面美观，可以在局域网能进行访问，如果部署在阿里云能够公网访问，加分。

1. 网络爬虫设计，针对某个网站的内容（如搜房网的房源信息，学校内学生成绩信息等）设计一套自动爬虫系统，并使用数据库对所爬取的数据进行保存与读取。爬取的数据要超过 1000 条以上
2. 基于 python 的移动开发，利用 python 开发一个基于 html5 的 app 服务，要能够实现二维码扫描，图片上传，信息查询等功能
3. 其他基于 python 的项目开发任务

评分方法：

1. 功能的完备性（50%）
2. 文档的完整性（30%）
3. 测试结果（10%）
4. Github 上的代码规范（10%）

基于python的豆瓣网读书爬虫设计

# 摘 要

随着社会经济的飞速发展,互联网应用已经成为人们日常生活中的一部分,及时准确地获得网络应用数据也变得越来越重要。网络爬虫作为一种通过对互联网数据进行精确信息抓取来满足用户对特定数据需求的计算机技术，对于信息的快速准确获取至关重要。为了搜集豆瓣网上的图书信息，给我们选书作为参考,本系统采用Python 语言。首先采集豆瓣网中书籍的页面,然后用 Beautiful Soup库解析出需要提取的待分析的数据并清洗，最后保存到数据库中。

**关键字**：网络爬虫 数据库

目录

[摘 要 2](#_Toc65533065)

[一、设计背景 4](#_Toc65533066)

[二、设计原理 4](#_Toc65533067)

[三、系统分析与设计 5](#_Toc65533068)

[3.1系统功能要求 5](#_Toc65533069)

[3.2系统设计 5](#_Toc65533070)

[四、系统模块结构设计 6](#_Toc65533071)

[4.1爬虫模块 6](#_Toc65533072)

[4.2数据库操作模块 6](#_Toc65533073)

[五、系统功能实现 7](#_Toc65533074)

[5.1爬虫模块 7](#_Toc65533075)

[5.1.1指定URL，进行浏览器伪装 7](#_Toc65533076)

[5.1.2获取初始页面，提取所需的信息，组成所要爬取的URL 8](#_Toc65533077)

[5.1.3获取信息，数据清洗，生成.xlsx文件 9](#_Toc65533078)

[5.2数据库模块 10](#_Toc65533079)

[5.2.1连接数据库 10](#_Toc65533080)

[5.2.2在数据库中建表 10](#_Toc65533081)

[5.2.3将清洗的数据导入数据库中 10](#_Toc65533082)

[六、不足与改进 11](#_Toc65533083)

[附录 11](#_Toc65533084)

# 一、设计背景

随着网络的迅速发展，万维网成为大量信息的载体，如何有效地提取并利用这些信息成为一个巨大的挑战。搜索引擎(Search Engine),例如传统的通用搜索引擎A1taVista, Yahoo!和Google等，作为一个辅助人们检索信息的工具成为用户访问万维网的入口和指南。

但是，这些通用性搜索引擎也存在着一定的局限性，如：

（1） 不同领域、不同背景的用户往往具有不同的检索目的和需求，通用搜索引擎所返回的结果包含大量用户不关心的网页。

（2）通用搜索引擎的目标是尽可能大的网络覆盖率，有限的搜索引擎服务器资源与无限的网络数据资源之间的矛盾将进一步加深。

（3）万维网数据形式的丰富和网络技术的不断发展，图片、数据库、音频、视频多媒体等不同数据大量出现，通用搜索引擎往往对这些信息含量密集且具有一定结构的数据无能为力，不能很好地发现和获取。

（4）通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询。

为了解决上述问题，网络爬虫应运而生。

网络蜘蛛( Web spider )也叫网络爬虫(Web crawler )，蚂蚁(ant)，自动检索工具(automatic indexer), 或者(在FOAP软件概念中)网络疾走(WEB scutter),是一种“自动化浏览网络”的程序，或者说是一种网络机器人。它们被广泛用于互联网搜索引擎或其他类似网站，以获取或更新这些网站的内容和检索方式。它们可以自动采集所有其能够访问到的页面内容，以供搜索引擎做进一步处理(分检整理下载的页面)，而使得用户能更快的检索到他们需要的信息。

# 二、设计原理

网络爬虫始于一张被称作种子的统一资源地址( URLs)列表。当网络爬虫访问这些统一资源定位器时，它们会甄别出页面上所有的超链接，并将它们写入一张"待访列表”，即所谓"爬行疆域" (craw1 frontier )。 此疆域上的统一资源地址将被按照一套策略循环访问。如果爬虫在他执行的过程中复制归档和保存网站.上的信息，这些档案通常储存，使他们可以被查看。阅读和浏览他们的网站上实时更新的信息，并保存为网站的“快照”。大容量的体积意味着网络爬虫只能在给定时间内下载有限数量的网页，所以要优先考虑其下载。高变化率意味着网页可能已经被更新或者删除。一些被服务器端软件生成的URLs (统一资源定位符)也使得网络爬虫很难避免检索到重复内容。基于目标数据模式的爬虫针对的是网页上的数据，所抓取的数据一般要符合一定的模式，或者可以转化或映射为目标数据模式。

# 三、系统分析与设计

## 3.1系统功能要求

作为一种便捷地收集网上信息并从中抽取出可用信息的方式，网络爬虫技术变得越来越有用。使用Python这样的简单编程语言，可以使用少量编程技能就可以爬取复杂的网站。本系统的功能实现了在后台爬取豆瓣网站上书籍，提取书籍信息并分类存储在数据库中。具体功能要求为：

（1）爬取豆瓣网上书籍信息

（2）提取书籍信息，包括：书名、评分、评价人数、作者和出版社

（3）将书籍分类保存在数据库中

（4）爬取数据在1000条以上

## 3.2系统设计



图 1系统设计图

系统设计围绕系统功能所展开，具体包括两个模块：爬虫模块与数据库操作模块。爬虫模块主要是通过对互联网数据进行精确信息抓取来满足特定数据需求。数据库模块主要用于存储数据，实现数据共享，减少数据的冗余度，提高数据的独立性，实现数据的集中控制等。系统设计图见图1

# 四、系统模块结构设计

## 4.1爬虫模块

爬虫模块采用传统的爬虫流程设计，从一个初始页面获得的重要组成。传统爬虫从一个或若干初始网页的URL开始，获得初始网页上的URL， 在抓取网页的过程中，不断从当前页面上抽取新的URL放入队列，直到满足系统的一定停止条件。

爬虫模块的流程主要有：

（1）指定URL，进行URL伪装

（2）获取初始页面，提取所需的信息，组成所要爬取的URL

（3）获取信息，数据清洗，生成.xlsx文件



图 2 爬虫模块设计图

爬虫模块是整个系统的核心模块，模块的设计详见图

## 4.2数据库操作模块

数据库是“按照数据结构来组织、存储和管理数据的仓库”。是一个长期存储在[计算机](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA/140338)内的、有组织的、可共享的、统一管理的大量数据的集合。将爬虫所得的数据清洗之后整理成Excel表导入数据库中，进行一系列的操作，可实现数据共享，减少数据的冗余度，提高数据的独立性，实现数据的集中控制等。

数据存入数据库的具体实现流程为：

（1）连接数据库；

（2）在数据库中建表；

（3）将清洗的数据导入数据库中。



图 3数据库模块设计图

数据库模块用于在数据库中存放清洗的数据，便于对数据进行后续的处理。数据库模块设计详见图

# 五、系统功能实现

此次系统实现是采用python3.6实现系统功能，所使用的软件为pycharm，爬虫所用的第三方库主要有bs4,urllib。

## 5.1爬虫模块

### 5.1.1指定URL，进行浏览器伪装

URL是统一资源定位符，对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的URL, 它包含的信息指出文件的位置以及浏览器应该怎么处理它。爬取指定页面时，需要获取指定页面的URL，才能获得所需的信息。具体代码见图4



图 4 指定URL

现在多数网站都设置了反爬虫机制，对陌生来源会禁止访问。为了能够顺利获取所需信息，我们需要对浏览器进行伪装，具体见图

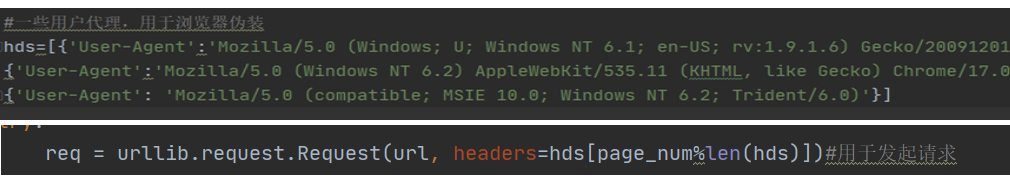


图 5 浏览器伪装代码

短时间内访问网站次数过多，会限制爬虫，被访问网站封禁。在这里设置了延时，每访问一次会暂停一段时间。具体见图



图 6 延时代码

### 5.1.2获取初始页面，提取所需的信息，组成所要爬取的URL

我们平时要搜索的信息通常都要经过好几个页面跳转才能得到，虽然这当中的跳转我们感受不到。因此,在实际的应用中,设计的网络爬虫应能够顺着超链接大面积地爬行万维网(web),并且将网页信息抓取下来。所以，从网页上抓取超链接是网络爬虫自主扩大访问范围的必要操作，是网络爬虫自主和自动进行的。

那么爬虫程序是如何遍历互联网，把网页全部抓取下来的呢?互联网可以看成一个超级大的“图”，而每个页面可以看作是一个“节点”。页面中的链接可以看成是图的“有向边”因此，能够通过图的遍历的方式对互联网这个超级大“图”进行访问。

本系统获取的目标URL采用深度优先遍历策略，网络爬虫从起始页开始，一个链接一个链接跟踪下去，处理完这条线路之后再转入下一个起始页，继续跟踪链接。具体过程见图

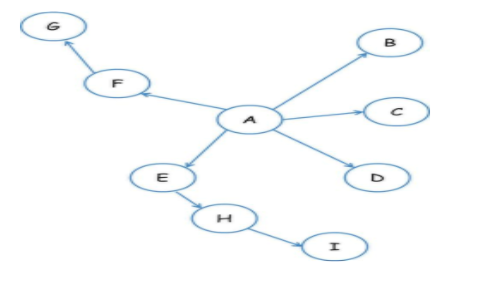


图 7深度优先遍历示意图

遍历的路径: A-F-G E-H-I BCD，具体实现见图



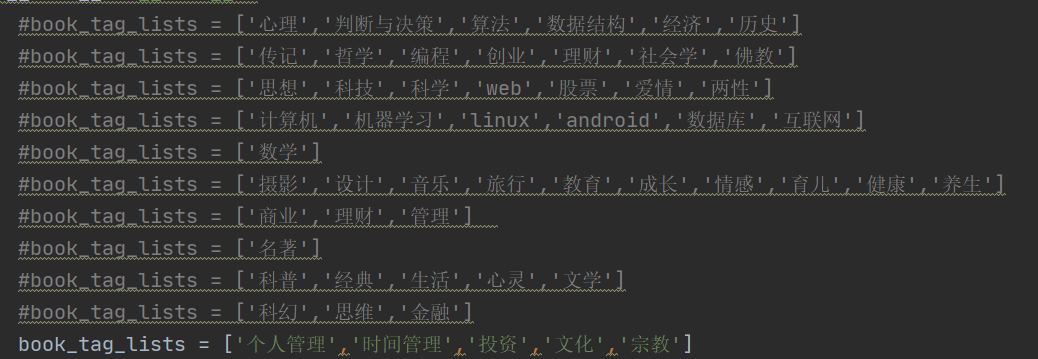


图 8 目标URL

将初始页面的URL与book\_tag\_List列表中的元素以及页数进行拼接得到最终的目标URL

### 5.1.3获取信息，数据清洗，生成.xlsx文件

获取信息采用的是bs4第三方库，通过将爬取的页面保存，利用BeautifulSoup框架，解析HTML页面，完成对数据的清洗。将数据清洗之后生成.xlsx文件，用于信息的存储。

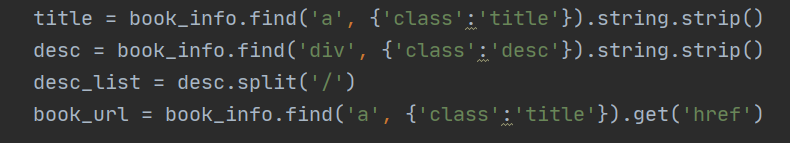


图 9 解析HTML代码

BeautifulSoup框架是用于解析获取的HTML页面，通过调用find函数，选择HTML页面中需要的信息进行整理保存，从而得到我们所需的信息。详见图9

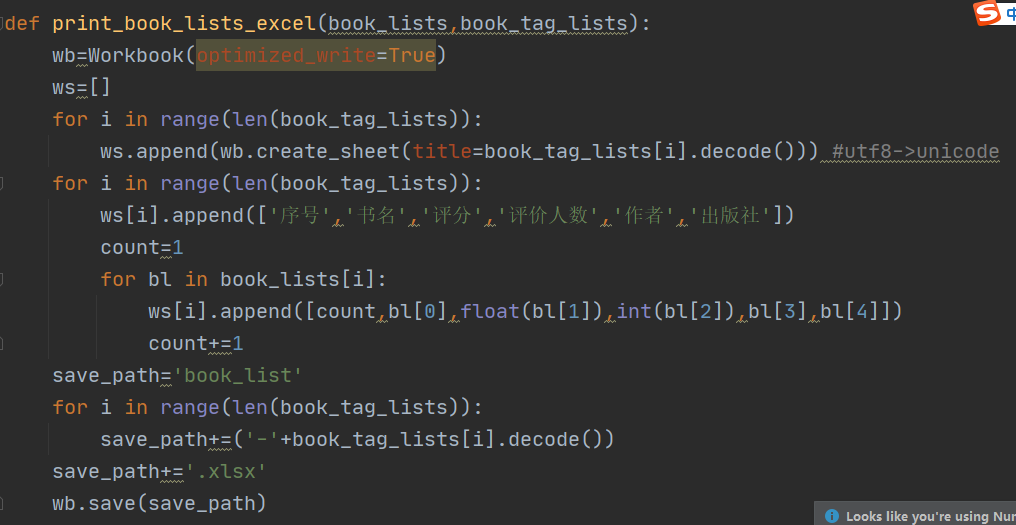


图 10 生成.xlsx文件代码

生成.xlsx文件，首先需要将获取的信息分类整理，如图10所示，将获取的信息分为'序号','书名','评分','评价人数','作者','出版社'这六类，表的名称以book\_tag\_List列表名来命名，将获取的数据分类保存成.xlsx文件。

## 5.2数据库模块

将数据存放在.xlsx文件中，信息过于分散，不利于信息的集中处理，还会造成数据的冗余等问题，为了能够更好地处理数据，我们将数据保存在MySQL数据库中，方便对获取的数据进行后续的操作。

### 5.2.1连接数据库

在将数据存入MySQL数据库之前，需要对数据库进行连接，具体实现见图11

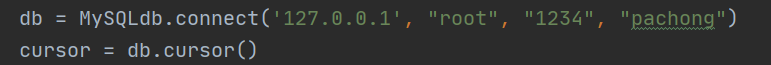


图 11 连接数据库代码

### 5.2.2在数据库中建表

在连接上数据库之后，我们需要将在数据库中建表，完成对数据的存放。具体实现见图12



图 12 数据库建表代码

### 5.2.3将清洗的数据导入数据库中

建表之后，便将清洗的数据导入到数据库中。最终效果见图13

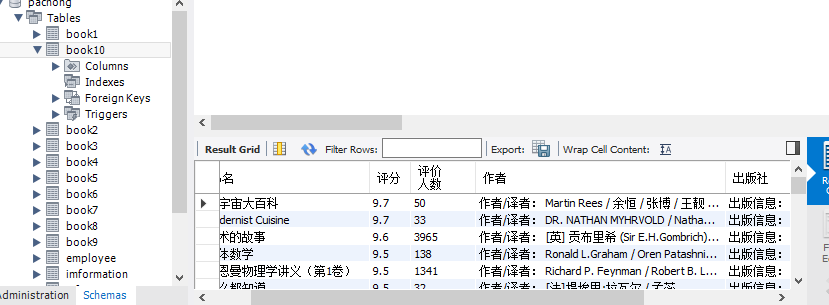


图 13 效果图

# 六、不足与改进

本系统在后台自动爬取了豆瓣网上读书页的所有图书，爬取信息5000条以上，将清洗后的数据存入数据库中，便于之后对信息的操作。基本实现了系统的功能。在此次系统实现过程中，遇到了很多问题，比如怎样建立爬虫框架，清洗数据，怎样将数据存入数据库中等问题。在学习了很多资料之后，我将整个系统拆分成若干个模块，逐一将模块实现，最后将各个模块连接起来，最终完成了系统的设计。

但是本系统仍有许多不足之处，本系统所采用的的爬虫框架为最单线程的爬虫框架，可以尝试多线程爬虫来提高系统效率。总的来说，我从此次系统的设计中学到了很多关于爬虫的知识，收获颇丰，在此次系统设计实现中受益良多。

# 附录

import sys

import time

import urllib

import urllib.request

import importlib,sys

import requests

import numpy as np

from bs4 import BeautifulSoup

from openpyxl import Workbook

import lxml

importlib.reload(sys)

#一些用户代理，用于浏览器伪装

hds=[{'User-Agent':'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.6) Gecko/20091201 Firefox/3.5.6'},\

{'User-Agent':'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.12 Safari/535.11'},\

{'User-Agent': 'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Trident/6.0)'}]

def book\_spider(book\_tag):

page\_num=0;

book\_list=[]

try\_times=0

while(1):

#url='http://www.douban.com/tag/%E5%B0%8F%E8%AF%B4/book?start=0' # For Test

url='http://www.douban.com/tag/'+urllib.parse.quote(book\_tag)+'/book?start='+str(page\_num\*15)

time.sleep(np.random.rand()\*5)

#Last Version

try:

req = urllib.request.Request(url, headers=hds[page\_num%len(hds)])#用于发起请求

source\_code = urllib.request.urlopen(req).read()

plain\_text=str(source\_code)

except (urllib.request.HTTPError, urllib.request.URLError) as e:

# urllib.request.HTTPE

print (e)

continue

soup = BeautifulSoup(plain\_text)

list\_soup = soup.find('div', {'class': 'mod book-list'})

try\_times+=1;

if list\_soup==None and try\_times<200:

continue

elif list\_soup==None or len(list\_soup)<=1:

break # Break when no informatoin got after 200 times requesting

for book\_info in list\_soup.findAll('dd'):

title = book\_info.find('a', {'class':'title'}).string.strip()

desc = book\_info.find('div', {'class':'desc'}).string.strip()

desc\_list = desc.split('/')

book\_url = book\_info.find('a', {'class':'title'}).get('href')

try:

author\_info = '作者/译者： ' + '/'.join(desc\_list[0:-3])

except:

author\_info ='作者/译者： 暂无'

try:

pub\_info = '出版信息： ' + '/'.join(desc\_list[-3:])

except:

pub\_info = '出版信息： 暂无'

try:

rating = book\_info.find('span', {'class':'rating\_nums'}).string.strip()

except:

rating='0.0'

try:

#people\_num = book\_info.findAll('span')[2].string.strip()

people\_num = get\_people\_num(book\_url)

people\_num = people\_num.strip('人评价')

except:

people\_num ='0'

book\_list.append([title,rating,people\_num,author\_info,pub\_info])

try\_times=0 #获取有效信息时设置0

page\_num+=1

print ('Downloading Information From Page %d' % page\_num)

return book\_list

def get\_people\_num(url):

#url='http://book.douban.com/subject/6082808/?from=tag\_all' # 用于测试

try:

req = urllib2.Request(url, headers=hds[np.random.randint(0,len(hds))])

source\_code = urllib2.urlopen(req).read()

plain\_text=str(source\_code)

except (urllib2.HTTPError, urllib2.URLError) as e:

print (e)

soup = BeautifulSoup(plain\_text)

people\_num=soup.find('div',{'class':'rating\_sum'}).findAll('span')[1].string.strip()

return people\_num

def do\_spider(book\_tag\_lists):

book\_lists=[]

for book\_tag in book\_tag\_lists:

book\_list=book\_spider(book\_tag)

book\_list=sorted(book\_list,key=lambda x:x[1],reverse=True)

book\_lists.append(book\_list)

return book\_lists

#永久性存储，存储为.xlsx文件

def print\_book\_lists\_excel(book\_lists,book\_tag\_lists):

wb=Workbook(optimized\_write=True)

ws=[]

for i in range(len(book\_tag\_lists)):

ws.append(wb.create\_sheet(title=book\_tag\_lists[i].decode())) #utf8->unicode

for i in range(len(book\_tag\_lists)):

ws[i].append(['序号','书名','评分','评价人数','作者','出版社'])

count=1

for bl in book\_lists[i]:

ws[i].append([count,bl[0],float(bl[1]),int(bl[2]),bl[3],bl[4]])

count+=1

save\_path='book\_list'

for i in range(len(book\_tag\_lists)):

save\_path+=('-'+book\_tag\_lists[i].decode())

save\_path+='.xlsx'

wb.save(save\_path)

if \_\_name\_\_=='\_\_main\_\_':

#book\_tag\_lists = ['心理','判断与决策','算法','数据结构','经济','历史']

#book\_tag\_lists = ['传记','哲学','编程','创业','理财','社会学','佛教']

#book\_tag\_lists = ['思想','科技','科学','web','股票','爱情','两性']

#book\_tag\_lists = ['计算机','机器学习','linux','android','数据库','互联网']

#book\_tag\_lists = ['数学']

#book\_tag\_lists = ['摄影','设计','音乐','旅行','教育','成长','情感','育儿','健康','养生']

#book\_tag\_lists = ['商业','理财','管理']

#book\_tag\_lists = ['名著']

#book\_tag\_lists = ['科普','经典','生活','心灵','文学']

#book\_tag\_lists = ['科幻','思维','金融']

book\_tag\_lists = ['个人管理','时间管理','投资','文化','宗教']

book\_lists=do\_spider(book\_tag\_lists)

print\_book\_lists\_excel(book\_lists,book\_tag\_lists)

#数据库存储

from sqlalchemy import create\_engine

import pandas as pd

#将Excel表格中的数据放到MySQL中

path = r'./book\_list-科普-经典-生活-心灵-文学.xlsx'

data=pd.read\_excel(path)

#这里是对data.columns进行处理

list = data.columns

new\_list = []

for x in list:

new\_list.append(''.join(x.split()))

data.columns=new\_list

engine = create\_engine('mysql+pymysql://root:1234@127.0.0.1:3306/pachong?charset=utf8')

#if\_exists三个模式：fail，若表存在，则不输出；replace：若表存在，覆盖原来表里的数据；append：若表存在，将数据写到原表的后面。默认为fail

sql\_chengfen = data.to\_sql(name='book10', con=engine, index=False,if\_exists='replace')