要求：

1. **源代码一份(小于5MB)**
2. 软件设计报告一份(word)，字数不少于8000字，页数不少于25页。

**指导教师审阅意见及成绩评定：**

指导老师签章：

日 期： 年 月 日

目录

[一、 绪论 1](#_Toc70278494)

[二、 爬虫的背景 1](#_Toc70278495)

[三、 爬虫的分类 3](#_Toc70278496)

[四、 课程设计的目的及意义 5](#_Toc70278497)

[五、 设计环境 5](#_Toc70278498)

[六、 用户需求分析 5](#_Toc70278499)

[1. 系统名称 6](#_Toc70278500)

[2. 系统功能 6](#_Toc70278501)

[七、 系统总体设计 7](#_Toc70278502)

[1. 爬虫原理介绍 7](#_Toc70278503)

[2. 系统总体结构 8](#_Toc70278504)

[3. 爬虫模块的实现及代码 8](#_Toc70278505)

[4. 数据库模块的实现及代码 13](#_Toc70278506)

[5. Flask路由模块的实现及代码 14](#_Toc70278507)

[6. 数据可视化模版 19](#_Toc70278508)

[八、 系统测试 19](#_Toc70278509)

[1. 数据爬取测试 19](#_Toc70278510)

[2. 数据可视化测试 21](#_Toc70278511)

[九、 课程学习方法 25](#_Toc70278512)

[十、 心得体会 26](#_Toc70278513)

[十一、 参考文献 27](#_Toc70278514)

1. 绪论

本课题的主要目的是设计面向定向网站的网络爬虫程序，同时需要满足不同的性能要求，详细涉及到定向网络爬虫的各个细节与应用环节。

搜索引擎作为一个辅助人们检索信息的工具。但是，这些通用性搜索引擎也存在着一定的局限性。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。为了解决这个问题，一个灵活的爬虫有着无可替代的重要意义。

网络爬虫应用智能自构造技术，随着不同主题的网站，可以自动分析构造URL，去重。网络爬虫使用多线程技术，让爬虫具备更强大的抓取能力。对网络爬虫的连接网络设置连接及读取时间，避免无限制的等待。为了适应不同需求，使网络爬虫可以根据预先设定的主题实现对特定主题的爬取。研究网络爬虫的原理并实现爬虫的相关功能,并将爬去的数据清洗之后存入数据库，后期可视化显示。

**关键词：网络爬虫，定向爬取，json解析，sqlite3，数据可视化**

1. 爬虫的背景

网络爬虫（又称为网页蜘蛛，网络机器人，在FOFA社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。另外一些不常使用的名字还有蚂蚁、自动索引、模拟程序或者蠕虫。

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AltaVista，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，通过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，定向抓取相关网页资源的聚焦爬虫应运而生。聚焦爬虫是一个自动下载网页的程序，它根据既定的抓取目标，有选择的访问万维网上的网页与相关的链接，获取所需要的信息。与通用爬虫(general purpose web crawler)不同，聚焦爬虫并不追求大的覆盖，而将目标定为抓取与某一特定主题内容相关的网页，为面向主题的用户查询准备数据资源。聚焦爬虫工作原理以及关键技术概述

网络爬虫是一个自动提取网页的程序，它为搜索引擎从万维网上下载网页，是搜索引擎的重要组成。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。

聚焦爬虫的工作流程较为复杂，需要根据一定的网页分析算法过滤与主题无关的链接，保留有用的链接并将其放入等待抓取的URL队列。然后，它将根据一定的搜索策略从队列中选择下一步要抓取的网页URL，并重复上述过程，直到达到系统的某一条件时停止。

另外，所有被爬虫抓取的网页将会被系统存贮，进行一定的分析、过滤，并建立索引，以便之后的查询和检索；对于聚焦爬虫来说，这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

相对于通用网络爬虫，聚焦爬虫还需要解决三个主要问题：

(1) 对抓取目标的描述或定义；

(2) 对网页或数据的分析与过滤；

(3) 对URL的搜索策略。

截止到 2007 年底，Internet 上网页数量超出 160 亿个，研究表明接近 30%的页面是重复的；动态页面的存在：客户端、服务器端脚本语言的应用使得指向相同 Web 信息的 URL 数量呈指数级增长。 上述特征使得网络爬虫面临一定的困难，主要体现在 Web 信息的巨大容量使得爬虫在给定时间内只能下载少量网页。 Lawrence 和 Giles 的研究表明没有哪个搜索引擎能够索引超出 16%的Internet 上 Web 页面，即使能够提取全部页面，也没有足够的空间来存储。

为提高爬行效率，爬虫需要在单位时间内尽可能多的获取高质量页面，是它面临的难题之一。 当前有五种表示页面质量高低的方式[1]：Similarity（页面与爬行主题之间的相似度）、Backlink（页面在 Web 图中的入度大小）、PageRank（指向它的所有页面平均权值之和）、Forwardlink（页面在 Web 图中的出度大小）、Location（页面的信息位置）；Parallel（并行性问题）。

为了提高爬行速度，网络通常会采取并行爬行的工作方式，随之引入了新的问题：重复性（并行运行的爬虫或爬行线程同时运行时增加了重复页面）、质量问题（并行运行时，每个爬虫或爬行线程只能获取部分页面，导致页面质量下降）、通信带宽代价（并行运行时，各个爬虫或爬行线程之间不可避免要进行一些通信）。 并行运行时，网络爬虫通常采用三种方式：独立方式（各个爬虫独立爬行页面，互不通信）、动态分配方式（由一个中央协调器动态协调分配 URL 给各个爬虫）、静态分配方式（URL 事先划分给各个爬虫）。

1. 爬虫的分类

网络爬虫的分类，网络爬虫按照系统结构和实现技术，大致可以分为以下几种类型：通用网络爬虫（General Purpose Web Crawler）、聚焦网络爬虫（Focused Web Crawler）增量式网络爬虫（Incremental Web Crawler）、深层网络爬虫（Deep Web Crawler）。实际的网络爬虫系统通常是几种爬虫技术相结合实现的。

1. 通用网络爬虫

通用网络爬虫又称全网爬虫（Scalable Web Crawler），爬行对象从一些种子URL 扩充到整个 Web，主要为门户站点搜索引擎和大 型 Web 服务提供商采集数据。由于商业原因 ，它们的技术细节很少公布出来。通用网络爬虫的结构大致可以分为页面爬行模块 、页面分析模块、链接过滤模块、页面数据库、URL 队列初始URL集合几个部分。为提高工作效率，通用网络爬虫会采取一定的爬行策略。常 用的爬行策略有：深度优先策略、广度优先策略。

2. 聚焦网络爬虫

聚焦网络爬虫（Focused Crawler），又称主题 网络爬虫（Topical Crawler），是指选择性地爬行那 些与预先定义好的主题相关页面的网络爬虫[8]。和 通用网络爬虫相比，聚焦爬虫只需要爬行与主题相关 的页面，极大地节省了硬件和网络资源，保存的页面 也由于数量少而更新快，还可以很好地满足一些特定 人群对特定领域信息的需求 。聚焦网络爬虫和通用网络爬虫相比，增加了链 接评价模块以及内容评价模块。聚焦爬虫爬行策略实 现的关键是评价页面内容和链接的重要性，不同的方 法计算出的重要性不同，由此导致链接的访问顺序也 不同。主要的爬行策略包括：基于内容评价的爬行策 略、基于链接结构评价的爬行策略 、基于增强学习 的爬行策略、基于语境图的爬行策略。

3. 增量式网络爬虫

增量式网络爬虫（Incremental Web Crawler）是指对已下载网页采取增量式更新和只爬行新产生的或者已经发生变化 网页的爬虫，它能够在一定程度上保证所爬 行的页面是尽可能新的页面。增量式爬虫只 会在需要的时候爬行新产生或发生更新的页 面 ，并不重新下载没有发生变化的页面，可 有效减少数据下载量，及时更新已爬行的网 页，减小时间和空间上的耗费，但是增加了 爬行算法的复杂度和实现难度。增量式网络 爬虫的体系结构[包含爬行模块、排序模块、 更新模块、本地页面集、待爬行 URL 集以及 本地页面URL 集。

4. Deep Web 爬虫

Web 页面按存在方式可以分为表层网页（Surface Web和深层网页（Deep Web，也 称 Invisible Web Pages 或 Hidden Web）。表层网页是 指传统搜索引擎可以索引的页面，以超链接可以到达的静态网页 为主构成的 Web 页面。Deep Web 是那些大部分内容不能通过静 态链接获取的、隐藏在搜索表单后的，只有用户提交一些关键词 才能获得的 Web 页面。例如那些用户注册后内容才可见的网页 就属于 Deep Web。2000 年Bright Planet 指出：Deep Web 中可 访问信息容量是 Surface Web 的几百倍，是互联网上最大、发展 最快的新型信息资源 。Deep Web 爬虫体系结构包含六个基本功 能模块 （爬行控制器、解析器、表单分析器、表单处理器、响应 分析器、LVS 控制器）和两个爬虫内部数据结构（URL 列表、 LVS 表）

1. 课程设计的目的及意义

**目的:**

1．根据课堂讲授内容，学生做相应的项目设计，消化课堂所讲解的内容。

2.通过解决项目过程中遇到的问题积累调试程序的经验。

3.通过完成项目代码编写，逐渐培养学生的编程能力，用计算机解决实际问题的能力。

**意义：**

1.有助于加深我们对python网络程序设计这门课程的理解，我们在课堂上学的都是基础理论知识，对于如何用程序语言来描述所学知识还是有一定难度。通过课程设计，我们可以真正理解其内涵。

2.有利于我们逻辑思维的锻炼，程序设计能直接有效地训练学生的创新思维、培养分析问题、解决问题能力。即使是一个简单的程序，依然需要学生有条不理的构思。

3.有利于培养严谨认真的学习态度，在程序设计过程里，当我们输入程序代码的时候，如果不够认真或细心，那么可能就导致语法错误，从而无法得出运行结果。那么，这个我们反复调试，反复修改的过程，其实也是对我们认真严谨治学的一个锻炼。

1. 设计环境

操作系统：WIN10家庭中文版64位

CPU：AMD Ryzen 5 3550H 2.10GHz

内存：16GB

硬盘空间：512GB

开发工具：pycharm professional 2020.3，python3.9

1. 用户需求分析

网络爬虫是一个专门从万维网上下载网页并分析网页的程序。它将下载的网页和收集到的信息存储在本地数据库中以供搜索引擎使用。网络爬虫的工作原理是从一个或若干初始网页的链接开始进而得到一个链接队列。伴随着网页的抓取又不断从抓取到的网页里抽取新的链接放入到链接队列中，直到爬虫程序满足系统的某一条件时停止。它是搜索引擎的重要组成部分。

本系统主要实现对51job及前程无忧网站招聘信息的爬取及分析，利用echarts对爬取到的信息做可视化处理，可以帮助求职者更直观的了解某岗位的招聘市场情况。

## 1. 系统名称

基于Python 的51job网站招聘信息的爬取与分析系统

## 2. 系统功能

(1)网页下载功能

1. 能够下载[招聘网\_人才网\_找工作\_求职\_上前程无忧 (51job.com)](https://www.51job.com/)网页。
2. 构造HTTP请求中的GET请求。
3. 分析HTTP响应请求。

(2)网页分析功能

1. 提取搜索岗位结果页。
2. 提取岗位详情页链接。
3. 提取岗位详情页岗位名称。
4. 提取岗位详情页公司名称。
5. 提取岗位详情页工资。
6. 提取岗位详情页工作地点。
7. 提取岗位详情页工作经验要求。
8. 提取岗位详情页工作技能要求。

(3)数据清洗功能

* 1. 过滤工作技能要求中的特殊符号
  2. 提取工资数据中的有效数据
  3. 过滤掉广告链接
  4. 提取公司地点中的城市名
  5. 提取工作经验要求中的有效数据

(4)数据保存功能

1. 插入数据到数据库，并保存搜索的关键字

(5)数据可视化功能

* 1. 学历占比分析图
  2. 工资与工作经验关系图
  3. 各城市招聘数量统计图
  4. 各招聘公司对比图
  5. 各城市平均薪资待遇统计图
  6. 技能要求词云
  7. 数据库所有数据分页展示

1. 系统总体设计

## 1. 爬虫原理介绍

1、获取初始URL。初始URL地址可以有用户人为指定，也可以由用户指定的某个或某几个初始爬取网页决定。

　　2、根据初始的URL爬取页面并获得新的URL。获得初始的URL地址之后，首先需要爬取对应URL地址中的网页，爬取了对应的URL地址中的网页后，将网页存储到原始数据库中，并且在爬取网页的同时，发现新的URL地址，同时将已爬取的URL地址存放到一个URL列表中，用于去重及判断爬取的进程。

　　3、将新的URL放到URL队列中，在第二步中，获取下一个新的URL地址之后，会将新的URL地址放到URL队列中。

　　4、从URL队列中读取新的URL，并依据新的URL爬取网页，同时从新的网页中获取新的URL并重复上述的爬取过程。

5、满足爬虫系统设置的停止条件时，停止爬取。在编写爬虫的时候，一般会设置相应的停止条件。如果没有设置停止条件，爬虫会一直爬取下去，一直到无法获取新的URL地址为止，若设置了停止条件，爬虫则会在停止条件满足时停止爬取。

## 2. 系统总体结构

1. GUI模块：设计用户操作界面，由用户输入要爬取的岗位名和爬取的数量
2. askURL模块：指定一个URL,返回该网页的HTML代码
3. getLink模块：根据岗位搜索结果页的HTML代码，提取并返回有效的岗位详情页链接
4. getData模块：通过调用askURL模块得到岗位详情页的HTML代码，提取并返回有效的岗位信息
5. getWordCloud模块：用于生成词云
6. init\_db模块：初始化数据库
7. saveData模块：实现数据的数据库存储
8. app模块：实现路由选择

## 3. 爬虫模块的实现及代码

1. GUI模块

#GUI界面  
root = Tk()  
root.title('51job爬虫')  
root.geometry("550x750")  
root.resizable(0, 0)  
#输入爬取岗位  
tip = Label(root, text='请输入您要爬取的岗位：')  
inputkw = Entry(root)  
inputkw.place(relx=0.5, y=1, relheight=0.04)  
#指定爬取的数量  
tip2 = Label(root, text='请输入您要爬取的总页数：')  
inputnum = Entry(root)  
#开始按钮  
btn = Button(root, text='开始爬取', command=main)  
#爬取过程显示框  
txt = scrolledtext.ScrolledText(root)  
#布局  
txt.place(relx=0.05, y=120, relheight=0.8, relwidth=0.9)  
tip.place(relx=0.1, y=1, relheight=0.05)  
tip2.place(relx=0.1, y=40, relheight=0.05)  
inputnum.place(relx=0.5, y=40, relheight=0.04)  
btn.place(relx=0.4, y=80, relheight=0.04)  
root.mainloop()

1. askURL模块

#得到指定一个URL的网页内容  
*def* askURL(*url*):  
 head = { #模拟浏览器访问该服务器  
 "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36 "  
 }  
 request = urllib.request.Request(*url*,headers=head)  
 html = ""  
 *try*:  
 response = urllib.request.urlopen(request)  
 html = response.read()  
 # print(html)  
 *except* urllib.error.URLError *as* e: #捕获异常  
 *if* hasattr(e,"code"):  
 print(e.code)  
 *if* hasattr(e,"reason"):  
 print(e.reason)  
 *return* html

1. getLink模块

#获取每个岗位的详情页面链接  
*def* getLink(*url*,*keyword*):  
 jobLink = [] #存放每个职位的链接，格式[{"link":"...."},{},{}...]  
 html = askURL(*url*)  
 bs = BeautifulSoup(html, "html.parser")  
 data = bs.select("script[type='text/javascript']")[2]  
 a1 = data.string.split(" = ")[1]  
 res = json.loads(a1) #获取到动态数据  
 link = res["engine\_search\_result"] #获取到岗位信息(列表-字典)  
 *for* i *in* link: #i即每个岗位  
 *for* key,value *in* i.items():  
 *if* key == "job\_href" *and* re.match(r'https://jobs.51job.com/(.\*)',value) != *None*:  
 jobLink.append(value)  
 # print(value)  
 jobList.append({"link":value,"keyword":*keyword*})  
 *return* jobLink

1. getData模块

# 通过岗位详情页获得职位的具体信息  
*def* getData(*jobPage*):  
 jobHtml = askURL(*jobPage*) #获取详情页  
 bs = BeautifulSoup(jobHtml, "html.parser")  
 *for* job *in* jobList:  
 *if jobPage* == job["link"]:  
 #职位名称：  
 jobname = bs.select(".cn>h1")  
 *if* jobname[0]["title"] :  
 job["jobname"] = jobname[0]["title"]  
 *else*:  
 job["jobname"] = " "  
 txt.insert(END,jobname[0]["title"]+"----"+job["link"]+"\n"+"\n")  
 txt.see(END)  
 txt.update()  
 #公司名称：  
 companyname = bs.select(".cn>.cname>.catn")  
 job["companyname"] = companyname[0]["title"]  
 #工资(保存最低工资)：  
 salary = bs.select(".cn>strong")  
 findLSalary = re.compile(r'(\d+\.?\d\*?)-', re.S)  
 *if* salary[0].string:  
 L = re.findall(findLSalary, salary[0].string)  
 *if* L:  
 job["salary"] = L[0]  
 *else*:  
 job["salary"] = 0  
 *else*:  
 job["salary"] = 0  
  
 #msg包括工作地点、经验要求、学历要求、招聘人数、发布时间：  
 msg = bs.select(".ltype")  
 msg = msg[0]["title"].split("|") #msg形成列表形式  
 #工作地点  
 job["place"] = ''.join(msg[0].split("-")[0].strip())  
 #工作经验  
 ex = ''.join(msg[1].split())  
 findExp = re.compile(r'(\d+)', re.S)  
 exp = re.findall(findExp,ex)  
 *if* len(exp) == 0:  
 job["experience"] = 0  
 *else* :  
 job["experience"] = exp[0]  
 #学历要求  
 job["edu"] = ''.join(msg[2].split())  
 #工作描述  
 jobMsgList = bs.select(".job\_msg") # 工作描述  
 jobMsgStr = ""  
 *for* str *in* jobMsgList:  
 jobMsgStr = jobMsgStr + str.text  
 jobMsgStr = jobMsgStr.replace(';','\;') #过滤特殊符号  
 jobMsgStr = jobMsgStr.replace('\\', ' ')  
 jobMsgStr = jobMsgStr.replace('\'', ' ')  
 jobMsgStr = jobMsgStr.replace(':', ' ')  
 jobMsgStr = jobMsgStr.replace(',', '\,')  
 job["info"] = ''.join(jobMsgStr.split())

1. getWordCloud模块

conn = sqlite3.connect('51job.db')  
cur = conn.cursor()  
sql = 'select info from job'  
data =cur.execute(sql)  
text = ""  
*for* item *in* data:  
 text = text + item[0]  
cur.close()  
conn.close()  
  
cut = jieba.cut(text)  
string = ' '.join(cut)  
print(len(string))  
img = Image.open(r'.\static\img\python.jpeg')  
img\_array = np.array(img)  
wc = WordCloud(  
 background\_color = 'white',  
 mask = img\_array,  
 font\_path = "msyh.ttc"  
)  
wc.generate\_from\_text(string)  
fig = plt.figure(1)  
plt.imshow(wc)  
plt.axis('off')  
# plt.show()  
plt.savefig(r'.\static\img\pythonword.jpeg',dpi=200)

## 4. 数据库模块的实现及代码

1. init\_db模块

#初始化数据库  
*def* init\_db(*dbpath*):  
 sql = '''  
 create table job  
 (  
 id integer primary key autoincrement,  
 link text,  
 keyword text,  
 jobname text,  
 companyname text,  
 salary text,  
 place text,  
 experience text,  
 educate text,  
 info text  
 )  
 '''  
 conn = sqlite3.connect(*dbpath*)  
 cursor = conn.cursor()  
 cursor.execute(sql)  
 conn.commit()  
 conn.close()

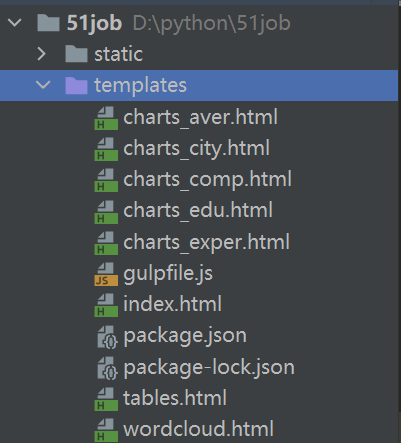
1. saveData模块

#保存数据(sqlite)  
*def* saveData(*jobList*,*dbpath*):  
 conn = sqlite3.connect(*dbpath*)  
 cursor = conn.cursor()  
 *for* data *in jobList*: #data即每条字典  
  
 data = list(data.values()) #把data字典转成列表  
 *for* index *in* range(len(data)):  
 data[index] = "'"+str(data[index])+"'" #转成引号格式  
 sql = '''  
 insert into job (  
 link,keyword,jobname,companyname,salary,place,experience,educate,info  
 )values(%s)'''%",".join(data) #将data列表以逗号分隔  
 cursor.execute(sql)  
 conn.commit()  
 cursor.close()  
 conn.close()

## 5. Flask路由模块的实现及代码

@app.route('/')  
*def* index():  
 datalist = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select count(\*) from job'''  
 num = cur.execute(sql1)  
 *for* item *in* num:  
 datalist.append(item)  
 sql2 = '''select keyword from job where id = 1'''  
 keyword = cur.execute(sql2)  
 *for* item2 *in* keyword:  
 datalist.append(item2)  
 cur.close()  
 conn.close()  
 *return* render\_template("/index.html", data=datalist)  
  
@app.route('/index.html')  
*def* home():  
 datalist = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select count(\*) from job'''  
 num = cur.execute(sql1)  
 *for* item *in* num:  
 datalist.append(item)  
 sql2 = '''select keyword from job where id = 1'''  
 keyword = cur.execute(sql2)  
 *for* item2 *in* keyword:  
 datalist.append(item2)  
 # sql3 = 'select info from job'  
 # data = cur.execute(sql3)  
 # text = ""  
 # for item in data:  
 # text = text + item[0]  
 # cut = jieba.cut(text)  
 # string = ' '.join(cut)  
 cur.close()  
 conn.close()  
 *return* render\_template("/index.html", data=datalist)  
  
  
@app.route('/charts\_edu.html')  
*def* charts\_salary():  
 benke = []  
 shuoshi = []  
 dazhuan = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select count(\*) from job where educate = "本科"'''  
 ben = cur.execute(sql1)  
 *for* i *in* ben:  
 benke.append(i)  
 sql2 = '''select count(\*) from job where educate = "大专"'''  
 da = cur.execute(sql2)  
 *for* i *in* da:  
 dazhuan.append(i)  
 sql3 = '''select count(\*) from job where educate = "硕士"'''  
 shuo = cur.execute(sql3)  
 *for* i *in* shuo:  
 shuoshi.append(i)  
 cur.close()  
 conn.close()  
 *return* render\_template("/charts\_edu.html",benke = benke,dazhuan = dazhuan,shuoshi=shuoshi)  
  
  
  
@app.route('/charts\_exper.html')  
*def* charts\_exper():  
 salary = []  
 exper = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select salary,experience from job where salary != '0' order by salary asc,experience asc'''  
 # sql2 = '''select place,count(place) from job group by place order by count(place) desc'''  
 res = cur.execute(sql1)  
 *for* i *in* res:  
 salary.append(float(i[0])\*10000)  
 exper.append(i[1])  
 cur.close()  
 conn.close()  
 *return* render\_template("/charts\_exper.html",salary=salary,exper=exper)  
  
  
  
@app.route('/charts\_city.html')  
*def* charts\_city():  
 keyword = []  
 city = []  
 num = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select keyword from job where id = 1'''  
 sql2 = '''select place,count(place) from job group by place order by count(place) desc'''  
 kw = cur.execute(sql1)  
 *for* i *in* kw:  
 keyword.append(i)  
 ci = cur.execute(sql2)  
 *for* j *in* ci:  
 city.append(j[0])  
 num.append(j[1])  
 cur.close()  
 conn.close()  
 *return* render\_template("/charts\_city.html", keyword=keyword, city=city, num=num)  
  
  
  
@app.route('/charts\_comp.html')  
*def* charts\_comp():  
 keyword = []  
 company = []  
 num = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql1 = '''select keyword from job where id = 1'''  
 sql2 = '''select companyname,count(companyname) from job group by companyname order by count(companyname) desc'''  
 kw = cur.execute(sql1)  
 *for* i *in* kw:  
 keyword.append(i)  
 cn = cur.execute(sql2)  
 *for* j *in* cn:  
 company.append(j[0])  
 num.append(j[1])  
 cur.close()  
 conn.close()  
 *return* render\_template("/charts\_comp.html", keyword=keyword,company = company,num = num)  
  
  
  
@app.route('/charts\_aver.html')  
*def* charts\_aver():  
 city = []  
 salary = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql = '''select place,AVG(salary) from job where place in ('北京','上海','广州','深圳','杭州','武汉','成都','南京','苏州','东莞','西安','佛山','长沙','汕头') group by place order by AVG(salary) desc'''  
 res = cur.execute(sql)  
 *for* item *in* res:  
 city.append(item[0])  
 salary.append(float(item[1])\*10000)  
 *return* render\_template("/charts\_aver.html",city = city,salary = salary)  
  
  
  
@app.route('/tables.html')  
*def* tables():  
 datalist = []  
 conn = sqlite3.connect("51job.db")  
 cur = conn.cursor()  
 sql = '''select \* from job'''  
 list = cur.execute(sql)  
 *for* item *in* list:  
 datalist.append(item)  
 cur.close()  
 conn.close()  
 *return* render\_template("/tables.html",datalist=datalist)  
  
  
  
@app.route('/wordcloud.html')  
*def* wordcloud():  
 *return* render\_template("/wordcloud.html")

## 6. 数据可视化模版



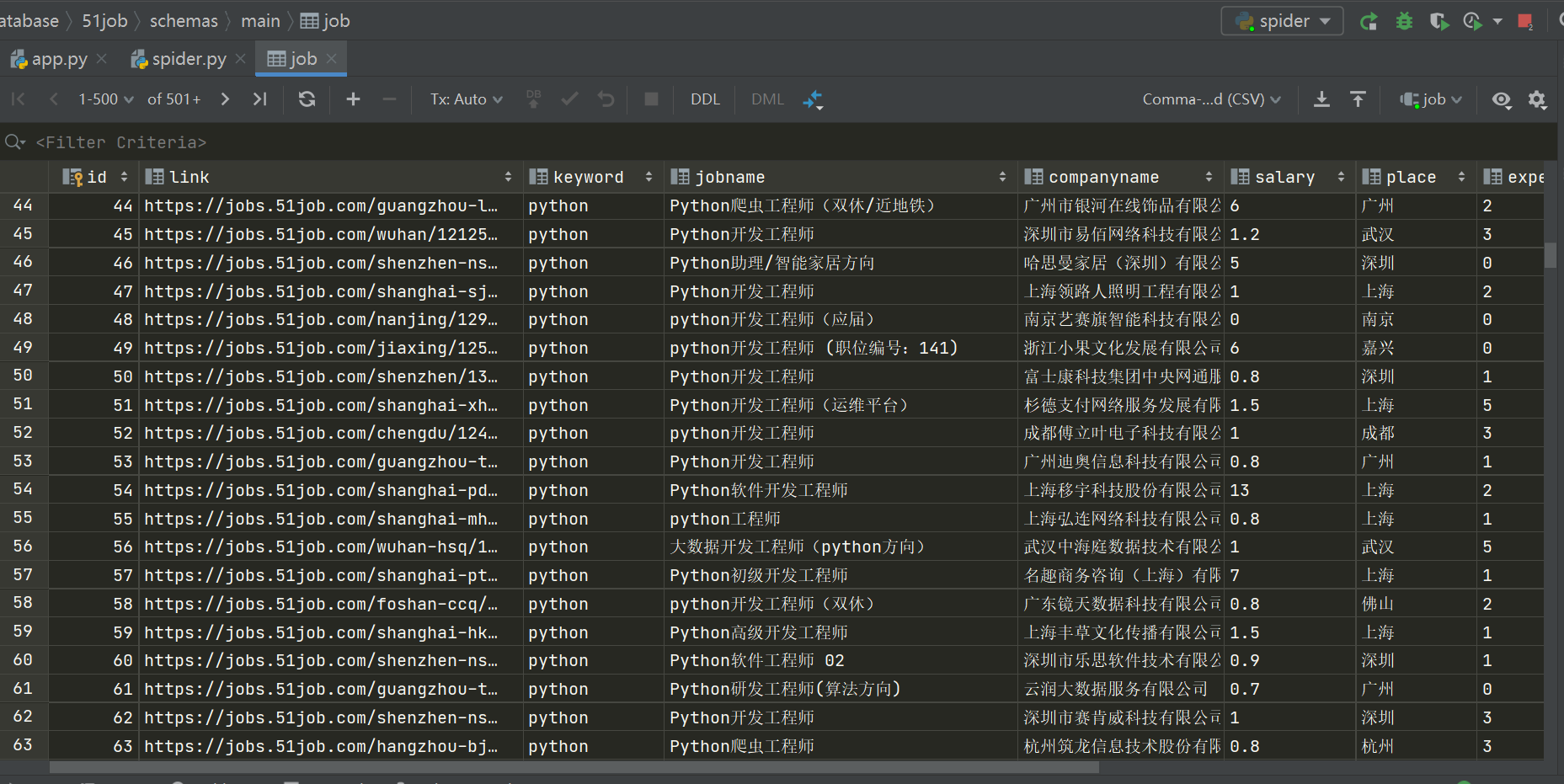
1. 系统测试

## 1. 数据爬取测试

输入爬取岗位及爬取页面的数量，点击“开始爬取”，爬取过程中文本框输出爬取到的部分信息及岗位名称和详情页链接：



数据成功保存到数据库：



## 2. 数据可视化测试

1. 首页数据渲染：



1. Echarts分析图表：











1. 数据库信息分页展示：



1. 招聘词云展示：



1. 课程学习方法

1、课前认真预习，把下一节要上的内容先自己看一遍，把不懂的地方打个标记，等上课的时候可以请教老师。理解书上的例子，搞懂每个语句的含义，用法。看完后尝试着不看例子自己编写程序，编完以后再跟书上进行比较，看自己的程序还有哪方面的不足，再进行改正。想要真正学好python光看书是不够的，一定要自己多动手。  
　　2、上课认真听讲积极思考，提高上课效率。老师上课讲述的都是课程的重点和难点，听好课程可以达到"事半功倍"的目的。  
　　3、课后复习。把每天的上课内容回家再温习一遍，独立完成课后的作业，如果有时间，最好能找点课外书籍、课外习题什么的来巩固所学的内容。  
　　4、主动上机，多多实践，提高动手能力。通过上机实践来验证所学的理论，通过在机器上查找错误来巩固概念，通过提高上机能力来促进理论学习，开阔编程思路，提高反映速度，提高分析问题解决问题的能力，锻炼心理素质，提高心理承受能力。  
　　学习python的初期重点要放在掌握语言的语法和规定上，一定要养成良好的编程习惯，平时写程序注意语法规范格式控制，格式规范了，出了错误也容易找到出错的地方，这是python的基础;后期的重点要转移到掌握编程的思路和基本算法上来。课后多做习题，找一些经典例子尝试自己编写，写程序切忌半途而费，想到一个思路，就要按自己思路把它写完，就算错了，也可以让自己了解为什么这样做会错?错在哪里?会出什么结果?以后自己也很难在这个地方犯同样的错误。当学到一个新的知识点，就试着把它运用到以前做的习题上，不但巩固了以前学的内容，也更加能加强新知识的理解。

1. 心得体会

1.什么是爬虫

在学习爬虫之前只对爬虫有个概念性的认识。通过向服务器发送请求获取服务器传回信息，再根据其提取所需的信息。原理虽然简单，但是涉及的细节非常多，从一个坑爬出来又掉进另一个坑。

2.post和get

post和get是两种向服务器发送请求的方式，在写爬虫时，涉及到同服务器交互肯定是离不了这两种方法的。其中get请求用于一系列对服务器数据没有影响的操作，如获取html，检索等，而post一般是用于对服务器数据有影响的功能，如提交表单等。如果只是简单爬取信息，一般用get方法。除此之前，get方法的安全性较Post方式要差些，如果需要提交包含机密信息的话，建议用Post提交方式，但get的执行效率比post方法好。

3.编码方式

这是跌进的第一个坑。源于对一个网页爬取时，不管怎么样都失败，浪费了将近一天时间才发现是编码混乱的原因造成。目前大多数中文网站采用GBK，GB2312，UTF-8这三种编码方式进行编码，详细的编码原理可查阅其他资料。通常在获取一个网页的html代码后都会先根据其编码方式进行转码，下面用python对从path获取的网页根据utf-8的方式进行转码。

html=urllib.request.urlopen(path).read().decode("utf-8")

4.正则表达式

在获取html代码后，就可以利用正则表达式进行信息提取了，虽然大部分信息都可以通过正则表达式进行提取，但如果需要对代码本身的结构进行提取和分析就需要用到lxml库了。

5.Chrom浏览器

Chrome浏览器对于写爬虫非常有用，Chrome自带的开发者工具能帮助分析网页的源码、资源和数据包。特别是要写表单提交一类功能时，Chrome自带的抓包功能非常好用，通过分析包结构可以实现各类复杂表单的模拟提交。

**总结爬虫要点：**

1.在刚拿到一个网页时,先确立自己要爬取的数据,记下来,方便一步步完成；

2.不要使用一个请求头过多次数,可以适当的加一些随机使用；

3.根据url拿到页面元素是很重要的一点,每次拿到都要输出,对比页面源代码是否正确；

4.分析页面元素取内容的时候一定要细心,达到绝对精确,其实也是方便后面的工作；

5.思路要活,多种取标签内容的方式都可以尝试,遇到不规范的标签会出现bug,不要慌,让程序跑起来,找到该标签,寻找共同规律,实在不行,就异常捕获,把该标签的提取方式单独写在except里；

6.字符串的处理,尽量处理的干净一点,看着也舒心；

7.要熟悉HTTP请求的方式,以及拿到的数据是否该用json格式化；

8.要学会分析页码规律,以及事先想好网站的数据更新方式,比如一些下拉刷新的数据,在url上是无法体现的,这时候就要分析网站发出的请求。

1. 参考文献

[1]苟英，张小华，高博，Python网络编程从入门到精通，北京大学出版社，2018

[2]蒋宗礼，赵钦，肖华，等．高性能并行爬行器．计算机工程与设计，2006

[3]张三峰，吴国新．一种面向动态异构网络的容错非对称DHT方法．计算机研究与发展，2007

[4]余锦，史树明．分布式网页排序算法及其传输模式分析．计算机工程与应用，2004

[5]沈贺丹，潘亚楠．关于搜索引擎的研究综述．计算机技术与发展，2006

[6]张敏，高剑峰，马少平．基于链接描述文本及其上下文的Web信息检索．计算机研究与发展，2004