**网站爬虫程序研究**

**摘要:** 在互联网信息化高速发展的现代，人们可以简单的获取大量数据，但也因为信息化的高速发展，能获取到真正想要并且有用的数据是不那么简单的，要花大量的时间对比筛选。

本课题是围绕51job前程无忧的职业信息，采用简洁高效的python语言与mysql数据库等技术开展的网站爬虫程序的研究。目的是呈现一个职业信息分析系统，帮助求职者获取准确的信息，根据大量数据做出最优的选择分析职业信息。最后实现在网页上展示结果。

整体设计包括前端和后端，先爬取数据，再进行一系列的数据清洗，导入数据库后进行分析，运用Echarts、词云、flash框架等技术对结果就业岗位、城市、待遇等信息进行结果展示，并且能引导用户搜索获得想要的信息。

**关键词：**网站爬虫；Python； Mysql；Flash

Web crawler research

Abstract**:**With the rapid development of Internet information in modern times, people can simply obtain a large number of data, but also because of the rapid development of information, it is not so simple to get really want and useful data, it takes a lot of time to compare and screen.This topic is to focus on the career information of 51job 51job, using simple and efficient python language and mysql database technology to carry out the research of website crawler. The purpose is to present a career information analysis system, to help job seekers to obtain accurate information, according to a large number of data to make the optimal choice analysis of career information. Finally, the results are displayed on the web page.The overall design includes the front end and the back end, first crawling data, then a series of data cleaning, analysis after the import of the database, the use of Echarts, word cloud, flash framework and other technologies to the results of jobs, cities, pay and other information results display, and can guide users to search for the information they want.

**Keywords:** Web crawlers; Python; Mysql; Flash

1. 绪 论

1.1 课题的目的与意义

就业作为久盛不衰的民生问题，在2020年，显得格外严峻。国家不断出台政策鼓励就业，对于应届生来说，压力不言而喻。从近几年来说,应届生都是逐年递增的,也就是说择业者的队伍逐年庞大。高校毕业生数量已大大超过社会就业岗位需求,高校毕业生就业压力达到前所未有的高度。而对于今年来说,情况更是严峻。受新冠肺炎疫情的影响,民生经济受到不同程度的冲击,直到今年三月,全国才开始陆陆续续的复工,有一定数量的中小企业甚至面临倒闭危机,失业比例同比明显增高,可见今年就业的困难程度。同时,由于疫情影响,各大高校积极开展各项促进就业的活动,但选择面还是不够全面,这能被动的接收到信息,而不能很好的结合自身优势与兴趣去选择就业。本课题的意义在于对职业信息的整合,通过多个方面切入将数据可视化,通过爬虫系统获取的数据以网页的形式让择业者能方便直观的了解到想要的内容,根据自己的期望做出最优的选择,也能了解到自己行业的多种可能性。

本课题的意义在于对职业信息的整合，通过多个方面切入将数据可视化，通过爬虫系统获取的数据以网页的形式让择业者能方便直观的了解到想要的内容，根据自己的期望做出最优的选择，也能了解到自己行业的多种可能性。

1.2 网站爬虫程序研究发展前景和展望

首先要说的是爬虫采用的语言——python。Python的优点包括但不限于以下几点：“优美”、“明确”、“简易”。所以近年来编程语言界内Python程序总是作为初学者的首选语言。这是因为在初学者开始学习Python时，由于它简明的特点所以入门相对于C、Java更加简单，入门后想要深入的学习，对于编写一些特别系统化的大程序也是可以实现的。我们常常听说的网络爬虫,其实并不是高深莫测的，甚至可以说是简单高效的，网络爬虫就是通过你的需求去制定相应的规则,用来欺骗网页服务器，模拟人类上网的行为，去自动的大量的获取网页上你想要的合法的信息的一种计算机程序。爬虫把这些合法的数据信息存储在本地,我们可以通过excel表、数据库等工具展现数据，有针对性地根据需求对数据进行分析研究。之所以爬虫技术会兴起，因为在一个信息化的时代里互联网上的海量数据是非常的有价值,通过运用爬虫这一技术,我们可以做到高效率并且较为准确的获取到数据，并且可以设定格式标准呈现数据，分析的数据结论运用在社会各领域中。结合上述对Python语言的分析,可以得知基于Python的网络爬虫实用性非常高，因为有非常多的现有的爬虫框架以及系统自带的工具包，相当大程度的降低了初学者入门的门槛,在使用时配合人灵活的正则表达式,丰富了数据抓取这一过程，体现了生动性。以下是对比分析各种语言下的爬虫：

Python：自带的请求与解析模块十分完善，功能丰富强大，scrapy框架更是庞大

PHP:在多线程方面表现欠佳

JAVA:代码量巨大，使用起来繁琐

C/C++:效率高;但是代码成型的效率不太高

得出本次python爬虫更适合本课题的研究，并且也符合数据分析的大趋势。

由于疫情的原因,今年变成了最难就业季,就业岗位信息每天不断的更新变化,应届毕业生每天需要进行不断的人工查询和繁琐的筛选操作,应届生们付出大量的时间精力,按照以往的趋势来说，线下是主流的找工作方式，但是今年是特殊情况，为了避免病毒传播线下招聘活动无法开展，这意味着择业者没有办法准确快速的获取到适合自己的招聘相关信息和资源,并且网络上存在大量钓鱼网站，或是发布过时的消息，伴随着的问题还包括了对工作信息的不当检索，并且对于大量的数据，不能进行全面的发掘,所以在线上检索时大家总是迷惘的，很多适合自己的潜在工作机会由于信息的数量繁多而遗漏，于是总是在错过机会的路上行走着。于是众多的线上招聘信息集合平台层出不穷,但良莠不齐并且还需大学生一一进行人工查询以及筛查操作,不仅费时费力,而且从用户体验角度来说,仍然不够自动化。

1.3 课题的主要内容和要求

将51job网的结构加以充分分析研究，明确就业信息的数据分布。系统需要具体实现的目标如下：

（1）对职业的职业名、要求学历、薪资和地点等数据进行爬取。

（2）导入数据库。

（3）按照数据进行可视化分析。

（4）从多个方面对数据进行提取总结。

（5）对于可视化结果分析全国就业信息概况。

（6）通过网页展示结果，引导用户查询获取结果。

1.4 论文的工作和安排

本论文的目的是基于51job前程无忧网的职业信息而采用Python爬虫、MySQ等技术L呈现一个职业信息分析系统，帮助用户获取就业信息，根据大量数据分析的结果，找到自己的定位与方向。

论文的设计分为一下几个部分：

第1章是绪论，简述网络爬虫程序研究就的开发背景，目的与意义。

第2章是网站爬虫程序研究中所用到的一些开发工具，还有一些相关技术的陈述介绍。

第3章是对网站爬虫程序研究的需求分析，首先介绍了爬虫程序设计目标，然后介绍了用户群体和市场分析，再从技术可行性去分析需求，最后是总结。

第4章是网站爬虫程序研究做了数据库的描述，并且介绍了给出数据表格。

第5章则描述了总体设计情况，介绍了实现本研究的两大框架。

第6章是讲解爬虫程序的代码，如何去实现，如何划分模块，还有对一些特殊结构的介绍。

第7章是测试程序，并且对数据可视化做出分析，还介绍了遇到的困难和解决办法。

1. 开发工具及相关技术

2.1 常用开发工具

2.1.1 Pycharm平台简介

PyCharm，是[Python](https://baike.baidu.com/item/Python/407313" \t "https://baike.baidu.com/item/PyCharm/_blank)的最热门的集成开发环境之一，Pycharm的使用广泛性是由于它具有大量的工具比如比如程序调试、智能联想提示、提示语法高亮、系统管理项目、点击跳转代码、单元测试等来提高Python编程时的效率。以下为Pycharm的特点：

1.编码协助

PyCharm的编辑器是可以配置的，具有一定的智能性。它提供编码补全功能，还支持智能的折叠代码和窗口的分割。从而时用户在编程时编码工作更加高效，并且不易出错。

2.项目代码导航

IDE有助于使用者快速的从一个文件导航移动到另一个文件。从一个方法通过其说明或使用方法可以跳过了类的层级。如果用户学习使用Pycharm内提供的快捷键，还会更快。

3.代码分析

用户在使用程中，可以通过PyCharm平台编码语言、错误处高亮强调、智能发现和单击代码来及时补充建议，以便优化编码。

4.集成版本控制

登录、注销、拆解视图和合并这一系列功能在统一的VCS的用户界面可以全部获取使用。

5.图形页面调试器

在pycharm中，自带了调试器，功能丰富全面。可以设置断点、步进，还能展现多方面视图、窗口和评估表达式来对python程序或者Django、Flask等应用程序进行调试。

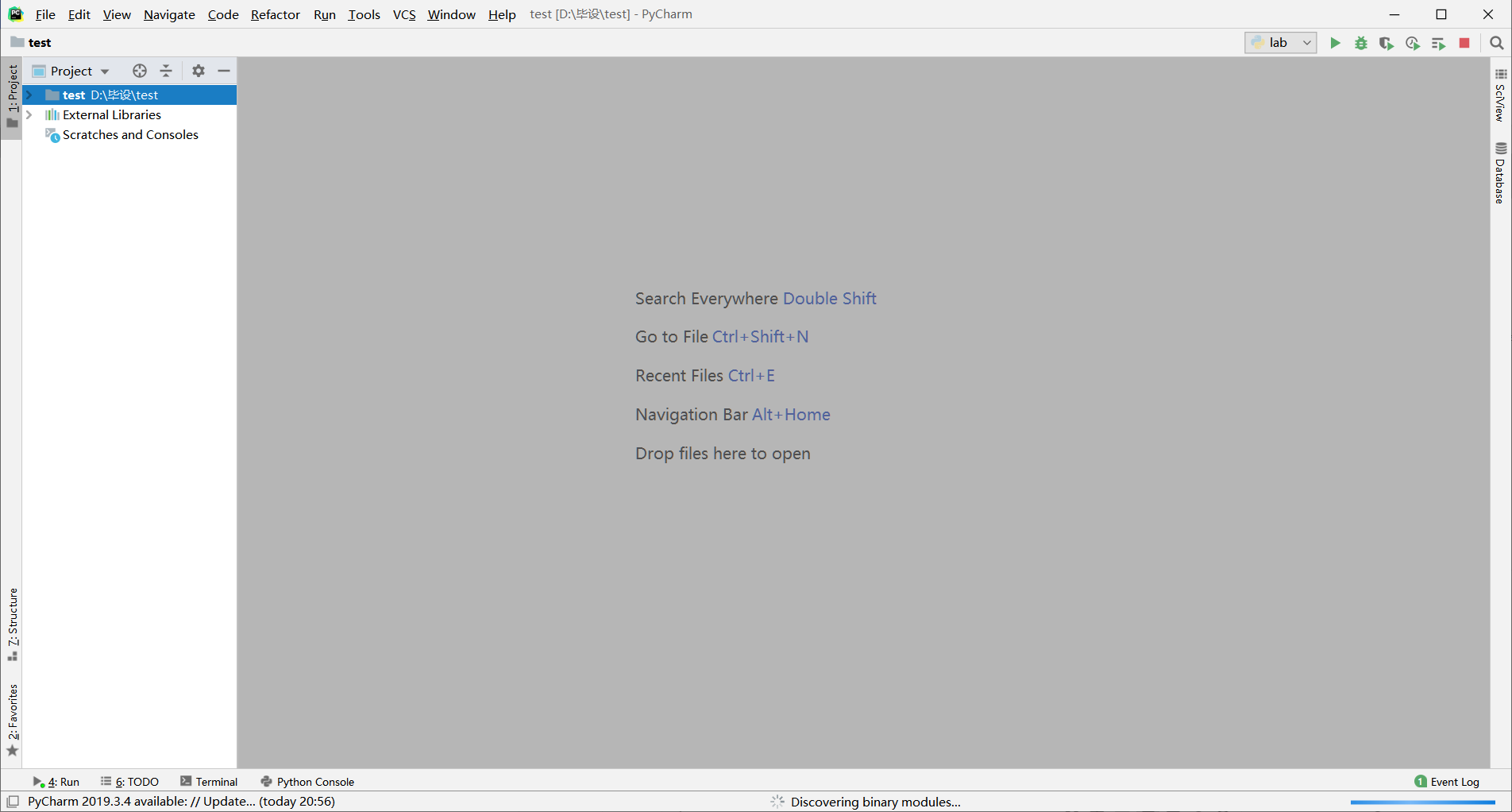
7.集成的单元测试

当使用者在一个文件夹内运行所有的测试项目。

8.可自定义&可扩展

Pycharm可以和一些键盘主盘绑定。比如说Textmate、NetBeans，、Eclipse & Emacs 还可以与Vi/Vim仿真插件绑定。所以说PyCharm是适用于Python编译的最受欢迎的集成开发环境之一。JetBrains设置了三种不同的IDE版本。开发人员可以自由的选择下载。它们分别是社区版，专业版和教育版。如果使用者想要降低成本，那么可以选择社区版，因为它是免费的，但是社区版相较于专业版来说，缺少了不少高级服务与功能。

Pycharm初始界面如图2.1所示。

图2.1 Pycharm初始界面

2.1.2 MySQL 简介

本次设计爬取到的数据都存入MySQL中。MySQL语言SQL语言，MySQL是一种关系型数据库管理系统。常应用于Web开发。 MySQL具有以下几种系统特性:

1.MySQL代码最大的特点是具有强大可移植性,在各种各样的主流编译器上可以运行。

2.MySQL支持非常多的操作系统类型，比如说 Mac OS、Linux、NovellNetware、OpenBSD、Solaris、Windows等。

3.MySQL可以为众多的编程语言提供API。

4.MySQL支持多线程，因此可以将CPU自由最大化的运用，不造成资源浪费。

5.优化了算法，提高了查询效率。

6.可以以一个应用程序的形式独立的存在网络中，又可以作为一个库应用再各种软件里。

7.可以展现多种编码语言。

8.提供各式各样的数据库连接途径。

9.自带了管理优化工具。

10.可以支撑并且处理大型复杂数据库。

11.支持多种存储引擎。

2.2 相关技术

2.2.1 爬虫

爬虫是一个模拟人类访问一个网站行为并且获取数据的程序。可以自动请求网页、并数据抓取下来，然后，通过所制定的既定规则来下载获取想要得到的数据。更生动地来描述，爬虫就好像是一个感应雷达。其基本操作是就是像人类手动去点击各种网站点击各种网页的各种链接，然后想要的内容就下载。

Scrapy是一种爬虫应用框架，主要可以爬取网页，然后把获取的信息结构化处理。在数据挖掘、信息处理等方向运用的非常之广泛。

Scrapy的主要优势之一——请求是[异步调度和处理](https://docs.scrapy.org/en/latest/topics/architecture.html" \l "topics-architecture)。这意味着Scrapy无需等待请求完成和处理，它可以同时发送另一个请求或执行其他操作。这也意味着即使某些请求失败或在处理过程中发生错误，其他请求也可以继续执行。

尽管这使您能够进行非常快速的爬网（以容错的方式同时发送多个并发请求），但是Scrapy还使您可以通过[一些设置](https://docs.scrapy.org/en/latest/topics/settings.html" \l "topics-settings-ref)来控制爬网的规则。你可以执行以下操作，例如设置每个请求之间的下载延迟，限制每个域或每个IP的并发请求数量，甚至[使用](https://docs.scrapy.org/en/latest/topics/autothrottle.html" \l "topics-autothrottle)尝试自动找出这些请求[的自动限制扩展](https://docs.scrapy.org/en/latest/topics/autothrottle.html" \l "topics-autothrottle)。

2.2.2 flask

Flask在2010年被Armin ronacher开发。他所采用的编程语言是 Python ，是一种轻量级的Web开发框架。  
　　Flask框架其实从本质上来看就是一个内核。它几乎全部功能都需要第三方来进行扩展，比如邮箱、认证、数据库等。Flask运用起来比较的灵活自由，难能可贵的是有众多第三方库供选择，即使没什么WEB开发经验，由于它的功能全面且简洁，很容易上手完成网站的开发。

2.2.3 chartsjs

chartsjs适用于设计人员和开发人员的简单而灵活的JavaScript图表。Chart.js可以与ES6模块，纯JavaScript和模块加载器一起使用。

一般配置：

1. [响应式](https://www.chartjs.org/docs/latest/general/responsive.html)：定义了适用于所有图表的响应式图表选项。
2. [设备像素比率](https://www.chartjs.org/docs/latest/general/device-pixel-ratio.html)：定义显示像素与渲染像素之间的比率。
3. [交互](https://www.chartjs.org/docs/latest/general/interactions/)：定义了反映悬停图表元素如何工作的选项。
4. [选项](https://www.chartjs.org/docs/latest/general/options.html)：可脚本化和可索引的选项语法。
5. [颜色](https://www.chartjs.org/docs/latest/general/colors.html):定义可接受的颜色值。
6. [字体](https://www.chartjs.org/docs/latest/general/fonts.html):定义各种字体选项。
7. [性能](https://www.chartjs.org/docs/latest/general/performance.html):为对性能敏感的应用程序提供了技巧。
8. 组态配置：该配置用于更改图表的行为。有一些属性可以控制样式，字体，图例等。

Chart.js 1.0中引入了此概念，以保持配置[DRY](https://en.wikipedia.org/wiki/Don't_repeat_yourself" \t "https://www.chartjs.org/docs/latest/configuration/_blank)，并允许跨图表类型全局更改选项，而无需为每个实例指定选项或为特定图表类型指定默认选项。

Chart.js使用图表类型默认值将传递给图表的选项对象与全局配置合并，并适当缩放默认值。这样，您可以像在单个图表配置中一样具体，同时仍在适用时更改所有图表类型的默认值。全局通用选项在中定义Chart.defaults.global。每种图表类型的默认值将在该图表类型的文档中进行讨论。

Chart.js带有内置图表众多，可以使用数据集fill选项从折线图或雷达图构建[面积图](https://www.chartjs.org/docs/latest/charts/area.html)。

2.2.4 Ngnix

Nginx ，是众多Web 服务器最常见的一个服务器。Ngnix的架构风格是属于REST，通过URI统一资源描述符或以URL统一资源定位符来作为沟通的依据， HTTP 协议中有各种各样的网络服务，所以依靠HTTP来提供所需要的网络服务。Nginx 不仅是一个服务器，并且它还是一个反向代理服务器；它的特性是自由且开源的、并且拥有高效率。Nginx 是可以作为一个跨平台的服务器，可以在多种操作系统上运行，比如常见的 Linux、Windows、Mac OS，它都是一款必不可少的服务器。同时它具有极高的稳定性。Nginx作为一种HTTP 服务器，它的功能是可以对网站发布进行处理，同时Nginx 作为反向代理服务器时，它的功能是可以完成负载均衡。

当众多客户端给服务器发送请求时，服务器负责接收信息，并且根据既定规则分别发布给相应的后端业务处理服务器处理数据。客服发作为请求源头对于服务器来说是清楚明确的，但收到信息后再去请求服务器处理数据，我们对到底是给了哪台服务器是不清楚的的，所以此时Nginx 反向代理服务器的作用得以体现。

我们可以不用特意对客户端进行配置，是因为客户端对于代理的存在是无意识的感觉不到的，但是反向代理对于外界是公开明确的。访问者不知道到底访问的是什么。

作为服务器的反向代理，它的主要功能主要是适用于服务器集群的分散配置。反向代理的作用是不让外界获取到服务器的相关信息。

反向代理是为了确保内部网的安全，反向代理通常是作为公共访问地址使用，而Web服务器属于的是内部网络。通过反向代理服务器，来优化网站的负荷分散，也就是经常听到到负载均衡。体现了Nginx起到反向代理服务器的作用。到底是根据什么样的规则来要求进行分发的？分发相应规则可以通过不同的项目应用方案来控制吗？由客户端进行发送、由NGinx Reverse Proxy反向代里服务器所接收的请求信息的数量，我们称之为负载量。请求的数目根据一个既定规则被分发，并且对不同服务器的处理规则表现均衡。因此，将服务器根据规则接收到的请求进行分布式处理称为负载均衡。在实际的项目运用过程中，分为软硬件负载均衡两种类型。如果想追求数据稳定安全，那么通常选择硬件负载均衡。但如果考虑成本，则选择软件负负载均衡，是一种基于现有技术和主机硬件的消息队列分配机制。

2.2.5 PHPMYAdmin

PHPMyAdmin是PHP提供 在网页上管理[MySQL](https://www.mysql.com/)。使用者在PHP软件上进入PHPMyAdmin,然后登陆就能在WEB界面上进行常用数据库的操作，也可以在网页上输入SQL语言进行操作数据库。

PHPMYAdmin特征：

1. 直观的Web界面
2. 支持大多数MySQL功能：
3. 可以管理数据库，多表、图、字段和索引进行增添或者清除
4. 维护服务器，数据库和表，以及有关服务器配置的建议
5. 执行，编辑和标记任何 SQL语句，甚至是批处理查询
6. 管理MySQL用户帐户和权限
7. 管理存储过程和触发器
8. 从CSV和 SQL导入数据
9. 数据可以多种形式的文本格式导出，常见的有SQL和PDF
10. 管理多个服务器
11. 创建各种格式的数据库布局图形
12. 使用示例查询（QBE）创建复杂的查询

2.3 pycharm第三方库配置

attrs==19.3.0 Automat==20.2.0 certifi==2020.4.5.1  
cffi==1.14.0 chardet==3.0.4 constantly==15.1.0  
cryptography==2.9.2 cssselect==1.1.0 hyperlink==19.0.0idna==2.9  
incremental==17.5.0 lxml==4.5.0 mysqlclient==1.4.6 parsel==1.5.2  
Protego==0.1.16 pyasn1==0.4.8 pyasn1-modules==0.2.8 pycparser==2.20  
PyDispatcher==2.0.5 PyHamcrest==2.0.2 pyOpenSSL==19.1.0 queuelib==1.5.0  
requests==2.23.0 Scrapy==2.1.0 service-identity==18.1.0 six==1.14.0  
Twisted==20.3.0 urllib3==1.25.9 w3lib==1.21.0  
zope.interface==5.1.0click==7.1.2Flask==1.1.2  
Flask-Caching==1.8.0flask-sqlacodegen==1.1.7

Flask-SQLAlchemy==2.4.1 inflect==4.1.0 itsdangerous==1.1.0

Jinja2==2.11.2 MarkupSafe==1.1.1 SQLAlchemy==1.3.16 Werkzeug==1.0.1

1. 系统需求分析

3.1 爬虫程序设计目标

基于51job网搜索模块，爬取计算机相关的职业信息，包括前端、后端、人工智能等几大热门模块，获取每则招聘信息的职位、工作地点、薪资、学历要求、类别等信息，存入数据库中，从中分析出热门职业及其占比、热门职业的平均薪资和学历要求分布，利用数据可视化形象生动的用图表展示出来，用flask框架搭建一个可以搜索通过工作城市展现以上信息。以便于掌握整个计算机环境就业的大局势，选择最利于自己的方向，有目标的去获取求职信息。

3.2 用户群体与市场分析

就业一直是热门的民生话题，时代的发展带来了无穷的机遇，无穷的机遇意味着越来越多的职业涌现、越来越多的就业机会来临，但也意味着各行各业的人才也越来越多，越是热门的职业越是饱和的快。所以抓住机遇对每一个求职者来说都是重中之重的问题，而现在的就业信息琳琅满目，但是同时也是参差不齐，就业信息来源广而杂，各大招聘网站为了追求个性化，网站内容设计的花里胡哨，然后不仅视觉上格外的疲劳，还给了很多不良企业浑水摸鱼、鱼目混珠的机会，通过职业信息来“钓鱼”。而爬虫程序设计的目的在于，以统一的格式获取信息，再以简单直观的图表展示，利用了大数据的优势，了解就业的趋势，也能帮助就业者给自己合适的定位，再通过数据的分析，开展后续的求职之路。爬虫程序的研究起到一个导向作用，对于正在或潜在的求职者来说，可以帮助理清思路，找准定位。对于整个市场来说，去繁从简，精确的总结导向功能是在众多复杂的职位信息中的一股清流，拥有广阔的市场。

3.3 技术可行性

整个爬虫程序系统都可以在pycharm平台上完成，而且爬取内容都是网页合法合规所呈现的内容，整个设计包括爬虫，后端和前端。Pycharm功能强大，操作简单，为了达到想要的效果主要要运用到requests、Scrapy、 mysqlclient、flask等库，数据可视化可以通过charts、词云对存入数据库中的数据以饼状图、树状图等形式展现。

3.4 可行性分析总结

本设计仅仅是针对部分招聘信息的总结分析，如果将程序壮大，通过不断地完善最终成型的爬虫系统能够在各行各业中实现价值，根据所需要分析的数据进行定制化修改。比如：为学校分析学生每次考试的成绩分布，错题分布来制定相信的教学方案服务，为各大企业提供公司内部信息的获取分析，还可以对用户群体的需求分析提升企业服务机制，为商业性机构获取数据分析行情，为制定方案提供指导作用。根据以上的分析，网站爬虫程序研究系统具拥有十分强的可行性。

第4 章 系统数据库设计

4.1 MySQL数据库设计概述

对于任意一个系统来说，拥有一个合适的数据库结构无疑是至关重要的，它是一切程序的运行基础，是任何应用保持高性能高效率的前提。如果没有一个完整而工整的数据库结构，会让之后的一切工作都事倍功半，不经效率低下而且体验十分差，难以维护。无法保证数据的精简性，并且可能会有很多重复的数据存在，不仅对程序的完成有严重的阻碍作用，导致低性能，还可能对分析结果有多偏差，得到的结果不精确，那么所做的一切工作是没有意义的。若一个数据突然的改变，其他数据也要因此而改变时，操作起来是十分麻烦的。

本次设计的系统，爬虫所下载的内容存放在mysql数据表中，用PHP admin管理。每项数据的都是不可或缺的，一旦缺失，可能导致后续的程序无法执行，所以我们要养成备份的良好习惯，为设计顺利的进行提供了保障。

4.2 系统数据表设计

数据库详细信息如表4.1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 名字 | 类型 | 排序规则 | 备注 |
| job\_id | int(10) | 无 | 序列号 |
| job\_title | varchar(500) | utf8mb4\_general\_ci | 职位名称 |
| job\_degree | varchar(10) | utf8mb4\_general\_ci | 学历要求 |
| job\_city | varchar(10) | utf8mb4\_general\_ci | 工作地点 |
| job\_money\_min | float | 无 | 最小工资 |
| job\_money\_max | float | 无 | 最大工资 |
| job\_category | varchar(20) | utf8mb4\_general\_ci | 职位类别 |
| job\_url | varchar(1000) | utf8mb4\_general\_ci | 对应网址 |

表4.1 数据表图

第5章 总体设计

5.1 爬虫程序总体设计概述

5.1.1 总体设计结构

本次网站爬虫程序研究中，可分为数据获取模块、数据存取模块、后端模块、前端模块。前两个模块运用的是scrapy框架，前端和后端都运用flask框架。总体设计流程如下如图5.1：

数据存储

数据取得

数据爬取

前端展示

数据可视化

开始始

图5.1 总体设计流程图

数据获取模块：确定好爬取的目标和范围，选取指定url,获取职位的相关数据，包括职位名、薪资、学历要求、工作地点。

数据存取模块：引入Mysql第三方库，连接程序，将获取的数据存入Mysql,运用PHPadmin管理数据。

数据可视化模块如图5.2：

数据可视化功能模块

学历分布

热门职业薪资

热门职业

热门职业占比

图5.2 数据可视化模块

1. 打开网页默认的是深圳的职业信息，可以在搜索框内搜索想要查看的城市信息。
2. 展示热门职业分布的饼状图。
3. 展示热门职业薪资的树状图。
4. 展示学历分布的环形折线图。
5. 为了更直观的表现，采用词云的方式展示职位。

5.1.2 scrapy框架

爬虫流程：

1. 程序发URL给引擎。
2. 引擎发URL给调度器入队列
3. 调度去返回要爬取的URL
4. 引擎发生给下载器中间件，中间件爬取通过URL在Internet中获取数据
5. 爬取完毕返回response给引擎

6、获取response，处理完毕得到新的URL,引擎发送需要处理的数据到Pipeline.

Scrapy工作流程图如图5.3：

**Internet**

**调度器**

**Engine引擎框架核心**

**下载器downloader**

**普通文件Pipeline**

**爬虫程序**

图5.3 Scrapy工作流程图

Scrapy文件结构：

scrapy.cfg: 对整个框架的配置。

Spiders:顾名思义就是爬虫的package。我们创建的爬虫文件都会自动生成在该package下，可以看到之前创建的stack已经在这里了。

items.py:用来存放Item类的文件，Item类可以理解为数据的中转类，我们爬取网页后需要对其进行解析，并将解析后的数据进行存储分析。为了便于数据的迁移存储，我们可以将数据封装为一个Item类，然后再对Item类进行操作，这样可以避免很多不必要的错误。

middlewares.py:中间层文件，Scrapy自带的middleware可以分为spider middleware和downloader middleware两类，我们也可以自定义middleware类。我们爬取网页的网络请求和响应都会经过middleware进行处理，因此可以在这里做一些个性化的操作，比如设置用户代理，设置代理IP等。

piplines.py:用来处理保存数据的模块，我们爬取网页后解析生成的Item类会被传递到这里进行存储解析等操作。Scrapy提供了许多有用的Pipeline类来处理数据，我们也可以自定义Pipeline类进行处理。

Settings:Scrapy项目的配置文件，对整个项目进行设置。比如设置请求和响应的中间层，指定操作数据的Pipeline类等。

5.1.3 flask框架

本次设计最终呈现的页面是搜索对性的城市得到数据图表，搜索流程是浏览器接受到信息然后再发送请求，通过FLASK得到数据并加以处理，然后再响应请求返回到浏览器中，大致的流程如图5.4：

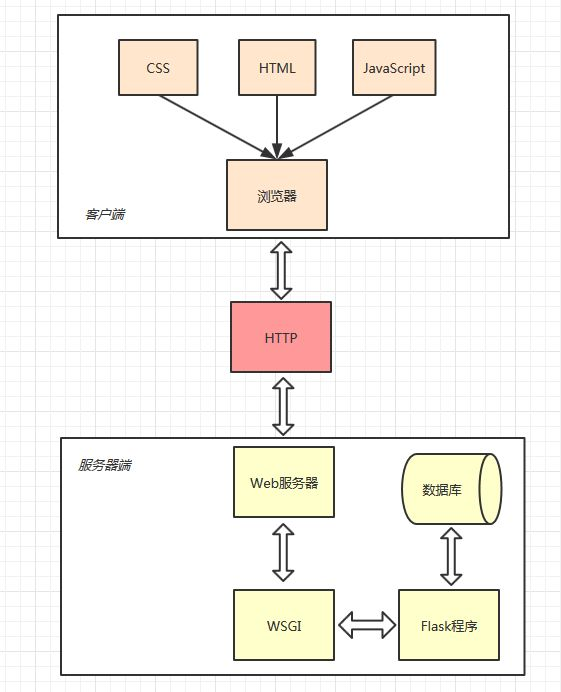


图5.4 Flask客户服务器结构

FLASK框架的工作流程大致流程如图5.5：

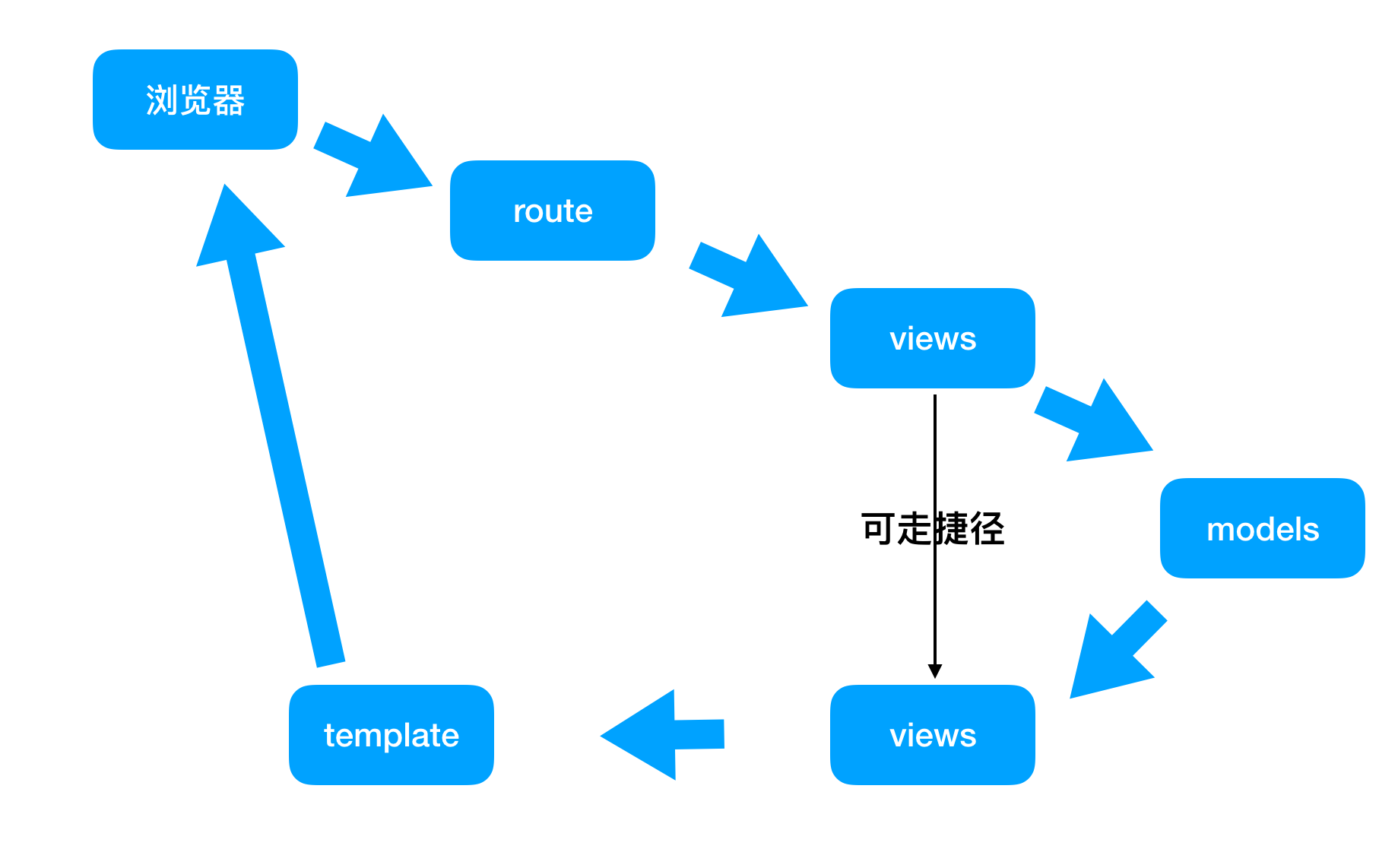


图5.5 Flask工作流程图

Flask框架结构如下：

首先是由app、tests、migrations3个目录和config.py、manage.py两个py文件组成的。

　　config.py：内含配置文档

manage.py ：是用来启动flask里的程序入口

还包含了main、static、templates三个二级目录，还包括了\_\_init\_\_.py、email.py、models.py

main文件夹：用来保存蓝本，此文件夹下\_\_inir\_\_.py文件里面创建蓝本，蓝本中定义的路由要通过注册程序激活。

views.py：用来保存程序的路由。

errors.py：用来处理错误。

forms.py：是存放表单定义。

Static：存放静态文件。

Templates：用来存放响应的html文件，mail子文件里面的用来保存发送邮件所需的.html和.txt文件。

\_\_inir\_\_.py：对app进行初始化操作。

email.py：包含send\_email()发送文件函数(异步)。

models.py：包含User和Role两个表定义

test\_basics.py：用来编写单元测试。

5.2 系统功能模块特点

每个模块中，按不同的功能都划分了对应的py文件，程序之间相互调用。再各级目录各模块中，添加了多个init\_\_py文件，实际上它相当于package的一个标志，IDE以这个文件来识别package,当我们想要从一层目录调用另一层目录中的py文件时，它的存在就像是一个结构化工具，让这个文件夹以一个软件包的形式被import。每个模块的代码力求工整化，以便于后续工作的开展，越是工整，再调试的时候越容易找到问题所在，正因为被划分的细致，当出错时，也是相对独立的，只要针对性的修改一个py文件中的内容。

1. 系统详细设计

6.1 数据获取模块功能实现

6.1.1 获取网页信息

我们通过导入requests库来发送请求和接受响应，通俗的来说就是模拟一台主机去访问一个网站，然后获取它的网页源码，这一步的目的是为后续爬虫程序设计做准备，找到各项数据所对应的源码位置，找出结构特点，以此为模板去自动爬取大量内容。

import requests  
if \_\_name\_\_ == '\_\_main\_\_':  
 headers = {  
 'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) '  
 'Chrome/81.0.4044.122 Safari/537.36 '  
 }  
 rsp = requests.get(  
 "https://search.51job.com/list/030000,000000,0000,00,9,99,java,2,"  
 "1.html?lang=c&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&ord\_field=0"  
 "&dibiaoid=0&line=&welfare=",  
 headers=headers)  
 print(rsp.content.decode('gbk'))

以下是获取的部分内容如图6.1



图6.1 网页获取信息

6.1.2 爬虫配置

Setting.py对爬虫进行配置，首先我们设置ROBOTSTXT\_ OBEY = False不遵循robots.txt协议，但是默认情况下是遵循的，是因为 robots.txt 是属于Robot协议里的一个小的文件，通常web服务器都会将它保存，之所以保存它，是为了在搜索引擎爬虫爬取本网站信息时，明确告知本网站有一些目录中的部分网页时不能被你的爬虫获取信息的。当Scrapy爬虫启动后，第一件事就是爬到该网站下的robots.txt文件，然后根据该协议指定爬取范围。而本次实际并不是在做搜索引擎，并且获取的数据都是公开透明的，也没有用作商业用途，所以将这个配置项改成false,因为robots.txt恰好禁止了我们想访问的链接，程序就不得顺利的展开，达不到想要的数量从而导致程序结果的片面性。

ROBOTSTXT\_ OBEY = False

#取消cookies

COOKIES\_ ENABLED = False

#数据传输

ITEM\_ PIPELINES = {

'P0427. pipelines .JobPipeline': 300 ,

'P0427. pipelines .MysqlPipeline': 301

}

设置url.py,确定要爬取51job的范围。

urls = [  
 "https://search.51job.com/list/000000,000000,0100,00,9,99,%2520,2,1.html", 、

#后端开发  
 "https://search.51job.com/list/000000,000000,7300,00,9,99,%2520,2,1.html",

#人工智能  
 "https://search.51job.com/list/000000,000000,7700,00,9,99,%2520,2,1.html",

#移动开发  
 "https://search.51job.com/list/000000,000000,7200,00,9,99,%2520,2,1.html",

#前端开发  
 "https://search.51job.com/list/000000,000000,2700,00,9,99,%2520,2,1.html",

#测试  
]

定义抓取模型：  
class Job51Item(scrapy.Item):  
 #职位 城市 学历 薪资 类别  
 job\_title = scrapy.Field()  
 job\_city = scrapy.Field()  
 job\_degree = scrapy.Field()  
 job\_money\_min = scrapy.Field()  
 job\_money\_max = scrapy.Field()  
 job\_category = scrapy.Field()  
 job\_url = scrapy.Field()

6.1.3 启动爬虫获取数据

首先我们得到的网页数据，按照设置的规则进一步地提炼出我们想要的数据。parse函数的作用是获取二级链接，然后生成request请求交给下载器下载页面，回调给parse\_detail函数处理。parse\_detail从下载器返回的response里提取出要的数据，生成item交给管道（即pipelines）处理。  
　　pipelines读取到蜘蛛传过来的item对它做进一步的处理，储存到数据库

def parse(self, response):  
#获取当前页的所有职位详细介绍的页面连接，通过异步的方式交给parse\_ detail方法 解析  
 detail\_url = response.css(".dw\_table .el p span a::attr(href)").getall()  
 category = re.match(".{1}(.\*)招聘", response.xpath("/html/head/title/text()").get()).group(1)  
 for item in detail\_url:  
 yield Request(url=item, callback=self.parse\_detail, meta={"category": category})  
  
 logging.info("正在爬取 " + category + ' 第' + str(self.i) + "页")  
 self.i += 1  
# 获取下一页的链接，异步的方式交给自己解析，有点类似于递归  
 next\_url = response.xpath(  
 "// \*[ @ class = 'p\_in'] / ul / li / a[contains(text(), '下一页')] / @href").get()  
 if next\_url:  
 yield Request(url=next\_url, callback=self.parse)  
 else:  
 self.i = 1

获取学历信息：

job\_desc = [response.xpath("/html/body/div[3]/div[2]/div[2]/div/div[1]/p[2]/@title").get()](mailto:response.xpath(\"/html/body/div[3]/div[2]/div[2]/div/div[1]/p[2]/@title\").get())

获取职位信息：

item["job\_title"]=response.xpath("/html/body/div[3]/div[2]/div[2]/div/div[1]/h1/@title").get()

获取工资信息：

job\_money = response.xpath("/html/body/div[3]/div[2]/div[2]/div/div[1]/strong/text()").get()

获取地点信息：

item["job\_city"] = re.match("^([\u4e00-\u9fa5]{1,})", job\_desc).group(1)

由于每项职位信息的工资设定格式不一样，我们必须把工资信息解析成相同的格式再存入数据库，所以我们把单位统一成月薪制，以万为单位，并且由于薪资是有范围的，所以设置为最小薪资和最大薪资。代码实现如下：

money\_match = None  
item["job\_money\_min"] = None  
item["job\_money\_max"] = None  
if job\_money:  
 money\_match = re.match("(\d{1,}|\d{1,}.\d{1,})-(\d{1,}|\d{1,}.\d{1,})(.)\/(.)", job\_money)  
 if money\_match is None:  
 money\_match = re.match("(\d{1,}|\d{1,}.\d{1,})(.)\/(.)", job\_money)  
  
if money\_match:  
 if money\_match.lastindex == 4:  
 item["job\_money\_min"] = round(float(money\_match.group(1)), 1) \  
 if money\_match.group(4) == '年' or money\_match.group(3) == '万' \  
 else round(float(money\_match.group(1)) / 10, 1)  
  
 item["job\_money\_max"] = round(float(money\_match.group(2)), 1) \  
 if money\_match.group(4) == '年' or money\_match.group(3) == '万' \  
 else round(float(money\_match.group(2)) / 10, 1)  
 else:  
 item["job\_money\_max"] = item["job\_money\_min"] = round(float(money\_match.group(1)), 1) \  
 if money\_match.group(3) == '年' or money\_match.group(2) == '万' \  
 else round(float(money\_match.group(1)) / 10, 1)

将爬取的职位信息封装进Job51Item对象，异步交给管道处理：

item["job\_title"] = response.xpath("/html/body/div[3]/div[2]/div[2]/div/div[1]/h1/@title").get()  
item["job\_city"] = re.match("^([\u4e00-\u9fa5]{1,})", job\_desc).group(1)  
  
field = re.match(".\*([初高中大专本科硕士博]{2}).\*", job\_desc)  
item["job\_degree"] = field.group(1) if field else "不限"  
item["job\_category"] = response.meta.get("category", "")  
item["job\_url"] = response.url

我们设置一个main函数，作为爬虫程序的开始入口：

import os  
from scrapy.cmdline import execute  
import sys  
if \_\_name\_\_ == '\_\_main\_\_':  
 sys.path.append(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
 execute(["scrapy", "crawl", "51job", "-s", "JOBDIR=crawls/somespider-1"])

众多的编程语言中，所有的程序、项目都需要有一个开端，即入口。做常见的有C语言, C++,还有一些完全面向对象的编程语言，比如Java、 C#等。如C语言和C++要通过建立一个main函数来启动程序，也就是我们所说的入口，程序从main开始展开，不论main的位置在哪里。同时完全面向对象的编程语言类，Java和C#需要通过一个含有Main方法的主类来启动程序。

但Python是完全不同的,首先pythons是脚本语言，跟编译型语言不同在于，编译性语言是先把执行程序翻译成成二进制再开始执行，但python不是定位main函数的位置来执行的，而是动态从程序顶端按照顺序逐行逐行地运行，没有统一的入口，当做一个项目时，这样是极不方便的，因为每当引入一个模块，就要执行该模块，最后虽然有最想要的执行结果，但效率时极低的，因为执行次数随着引入模块而增加，有时候更是会显示混乱。所以我们引入if \_\_name\_\_ == '\_\_main\_\_'语句作为函数的入口，以实现只执行一次主要的程序，引入的模块加以辅助。

6.2数据存储模块设计

6.2.1 数据库配置

在setting.py中设置数据库的主机地址，数据库名，还有用户名和密码：  
MYSQL\_HOST = "127.0.0.1"  
MYSQL\_DBNAME = "51job"  
MYSQL\_USER = "root"  
MYSQL\_PASSWORD = "root"

6.2.2 数据入库

from MySQLdb import cursors  
from scrapy.exceptions import DropItem  
from twisted.enterprise import adbapi  
  
import logging  
  
  
class JobPipeline:  
 def process\_item(self, item, spider):  
 if item["job\_money\_min"] and item["job\_money\_min"] > 100:  
 raise DropItem("过滤")  
 return item  
  
  
class MysqlPipeline:  
  
 def \_\_init\_\_(self, db\_pool):  
 self.db\_pool = db\_pool  
  
 @classmethod  
 def from\_settings(cls, settings):  
 db\_parm = dict(  
 host=settings["MYSQL\_HOST"],  
 db=settings["MYSQL\_DBNAME"],  
 user=settings["MYSQL\_USER"],  
 passwd=settings["MYSQL\_PASSWORD"],  
 charset="utf8",  
 cursorclass=cursors.DictCursor,  
 use\_unicode=True  
 )  
 return cls(adbapi.ConnectionPool("MySQLdb", \*\*db\_parm))  
  
 def process\_item(self, item, spider):  
 query = self.db\_pool.runInteraction(self.do\_insert, item)  
 query.addErrback(self.handle\_error, item, spider) # 处理异常  
 return item  
  
 def do\_insert(self, cursor, item):  
 sql = """  
 insert into job(job\_title,job\_degree,job\_city,job\_money\_min,job\_money\_max,job\_category,job\_url)  
 values (%s,%s,%s,%s,%s,%s,%s)  
 """  
 cursor.execute(sql,  
 (item["job\_title"], item["job\_degree"], item["job\_city"], item["job\_money\_min"],  
 item["job\_money\_max"], item["job\_category"], item["job\_url"]))  
  
 def handle\_error(self, failure, item, spider):  
 logging.warning("入库失败：" + item["job\_url"])

6.3 后端模块设计

6.3.1 连接数据库

1.数据库设置：

DEBUG = False  
SQLALCHEMY\_ECHO = True  
SQLALCHEMY\_DATABASE\_URI = 'mysql://root:root@127.0.0.1:3306/51job?charset=utf8mb4'  
SQLALCHEMY\_TRACK\_MODIFICATIONS = False  
SQLALCHEMY\_ENCODING = "utf8mb4"

1. 封装数据

数据封装进一个对象在DTO目录里实现。Data Transfer Object，也就是数据传送对象。它是一种软件运用系统，其作用是：设计模式之间数据传输。以便于我们实现搜索功能，数据传输的目的是将数据封装进对象后，对象从我们存入数据的数据库中搜索数据。它们之间的关系就像是存储器和访客。

class JobModel:  
 \_job\_set = []  
 \_is\_chang = True  
  
 def \_\_init\_\_(self, name, query\_job):  
 if len(query\_job) == 0:  
 return  
 self.job\_list = query\_job  
 self.length = len(query\_job)  
 self.name = name  
 self.average = None  
 self.degree = {  
 "不限": 0,  
 "中专": 0,  
 "高中": 0,  
 "大专": 0,  
 "本科": 0,  
 "硕士": 0,  
 "博士": 0  
 }  
 JobModel.\_job\_set.append(self)  
 JobModel.\_is\_chang = True  
  
 total = 0.0  
 for job in query\_job:  
 m\_min = job.job\_money\_min if job.job\_money\_min else 0.0  
 m\_max = job.job\_money\_max if job.job\_money\_max else 0.0  
 total += round((m\_min + m\_max) / 2, 1)  
 self.total\_degree(job.job\_degree)  
  
 self.average = round(total / self.length, 1)  
  
 def total\_degree(self, key):  
 if self.degree.get(key) is not None:  
 self.degree[key] += 1

1. 数据库ORM模型，通过导入SQLAlchemy第三方库，它在连接数据库时，设定一个URl,它必须时特定的，也称为统一资源定位符，来实现数据库连接功能。这个URI是有一串有特殊意义的字符串组成的，这写字符串的意思涵盖了其连接数据库的一切必要的信息。也就是说，通过ORM模型对Mysql进行操作，这样就不用谢Sql语言了，比较简单，不用来回切换，不易出错。

def job\_model\_factory(city, fields: dict):

if city != "全国":

for word, query in fields.items():

JobModel(word, Job.query.filter(and\_(Job.job\_city == city, Job.job\_title.like("%" + query + "%"))).all())

else:

for word, query in fields.items():

JobModel(word, Job.query.filter(Job.job\_title.like("%" + query + "%")).all())

return {

"city": city if city else "全国",

"hot": JobModel.get\_hot\_job(),

"all": [[item, (index + 1)\*2] for index, item in enumerate(JobModel.get\_all\_job())]

}

6.3.2 查询模块

我们设计一进入网页便是以深圳为参考城市，然后引导用户查询想知道的城市：

@view.route("/")  
def index():  
 city = request.args.get("city")  
 city = city if city else "深圳"  
  
 data = cache.get(city)  
 if data is None:  
 data = job\_model\_factory(city, FIELDS) if cache.get(city) is None else cache.get(city)  
 cache.set(city, data)  
 JobModel.close()

6.4 前端模块设计

6.4.1 数据可视化

本次作图前端的图形库在chartjs库里实现：

1.热门职业分布的饼状图

2.热门职业薪资的树状图

3.学历分布的环形折线图

4.词云

indow.chartColors = {  
 red: 'rgb(255, 99, 132)',  
 orange: 'rgb(255, 159, 64)',  
 yellow: 'rgb(255, 205, 86)',  
 green: 'rgb(75, 192, 192)',  
 blue: 'rgb(54, 162, 235)',  
 purple: 'rgb(153, 102, 255)',  
 grey: 'rgb(201, 203, 207)'  
};  
  
window.randomScalingFactor = function () {  
 return (Math.random() > 0.5 ? 1.0 : -1.0) \* Math.round(Math.random() \* 100);  
};  
  
  
var pieConfig = {  
 type: 'pie',  
 data: {  
 datasets: [{  
 data: [  
 400, 600, 100, 800, 600  
 ],  
 backgroundColor: [  
 window.chartColors.red,  
 window.chartColors.orange,  
 window.chartColors.yellow,  
 window.chartColors.green,  
 window.chartColors.blue,  
 ],  
 label: 'Dataset 1'  
 }],  
 labels: [  
 "Java工程师",  
 "Orange",  
 "Yellow",  
 "Green",  
 "Blue"  
 ]  
 },  
 options: {  
 responsive: true,  
 maintainAspectRatio: false,  
 title: {  
 display: true,  
 text: '深圳热门职位'  
 }  
 }  
};  
  
var pieCtx = document.getElementById("chartjs-pie-chart").getContext("2d");  
var pieCtx2 = document.getElementById("chartjs-pie-chart2").getContext("2d");  
window.myPie = new Chart(pieCtx, pieConfig);  
window.myPie = new Chart(pieCtx2, pieConfig);

6.4.2 html设计

1.提示错误页面

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>{{ data["city"] }}</title>  
</head>  
<body>  
 <h1>找不到数据</h1>  
</body>  
</html>

1. 模板继承页面

把前端网页的一些公用代码抽取到父模块base.html中，以便于子模块继承套用，即将代码尽可能的简明，并且方便修改。

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
 {% block title %}<title></title>{% endblock %}  
 <link href="https://cdn.bootcdn.net/ajax/libs/twitter-bootstrap/4.4.1/css/bootstrap.min.css" rel="stylesheet">  
 <script src="https://cdn.bootcdn.net/ajax/libs/Chart.js/2.9.3/Chart.js"></script>  
 <script src="https://cdn.bootcdn.net/ajax/libs/wordcloud2.js/1.1.1/wordcloud2.min.js"></script>  
 <style>  
 body {  
 background-color: #f5f5f5;  
 }  
 </style>  
 {% block css %}{% endblock %}  
</head>  
<body>  
{% block body %}{% endblock %}  
{% block js %}{% endblock %}  
</body>  
</html>

1. 搜索主页

首先定义好css，以便于之后的设计，然后设置搜索框，再引用static里的js文件具体实现数据可视化。

以下是搜索框设计：

{% block body %}  
 <div class="container d-flex flex-column align-items-center">  
 <form action="/" method="get">  
 <div class="form-group-lg mt-2" style="width: 600px">  
 <div class="input-group mb-3">  
 <input type="text" class="form-control" placeholder="输入城市搜索职位信息"  
 aria-label="Recipient's username"  
 aria-describedby="button-addon2" name="city" value="{{ data["city"] }}">  
 <div class="input-group-append">  
 <button class="btn btn-primary" type="submit" id="button-addon2">搜索</button>  
 <a href="/?city=全国" class="btn btn-primary ml-3">全国数据</a>  
 </div>  
 </div>  
 </div>  
 </form>  
  
 <div class="show d-flex flex-row align-items-center flex-wrap">  
 <div class="canvas">  
 <canvas id="chartjs-pie-chart"></canvas>  
 </div>  
 <div class="canvas">  
 <canvas id="chartjs-radar-chart"></canvas>  
 </div>  
 <div class="canvas" style="margin-top: 100px">  
 <canvas style="width:auto" id="chartjs-line-chart"></canvas>  
 </div>  
 <div id="canvas" style="margin-top: 100px">  
 <canvas id="word" width="500px" height="300px"></canvas>  
 </div>  
 </div>  
 </div>

1. 测试与结果分析

7.1 系统测试

7.1.1 测试的目的

对于测试的目的是指发现软件中的错误和缺陷并加以改正，它对于一个项目的开发来说是非常重要的，不仅仅是发现错误的过程，而且还可以不断地改进和完善，在错误中找进步，在进步中找创新。

软件测试是一个开发过程的必经过程，关系着整个项目的质量。我们找到错误或不足，首先不急于去改进，先记录下来，避免下一次遇到耗费不必要的时间，并且把错误归好类，然后总结原因，提出改善方案，然后逐步逐条的去修改完善，不要追求一蹴而就。在这样循序渐进的过程中，改错的同时积累经验，可以在基础的功能上更进一步，一步步的将程序饱满生动起来。

7.1.2 测试流程

1.执行main.py文件，完成scrapy部分。如图7.1所示：

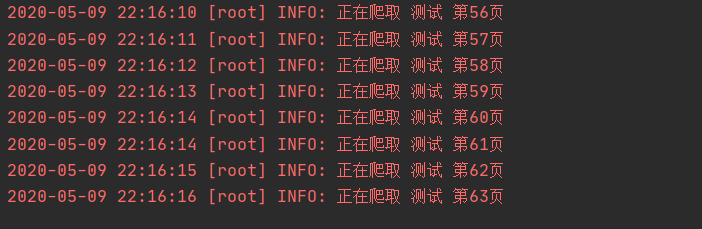


图7.1 爬虫爬取流程图

2.登陆PHP admin检查数据是否入库。如图7.2所示：



图7.2 登陆PHPadmin

数据成功入库，如图7.3所示

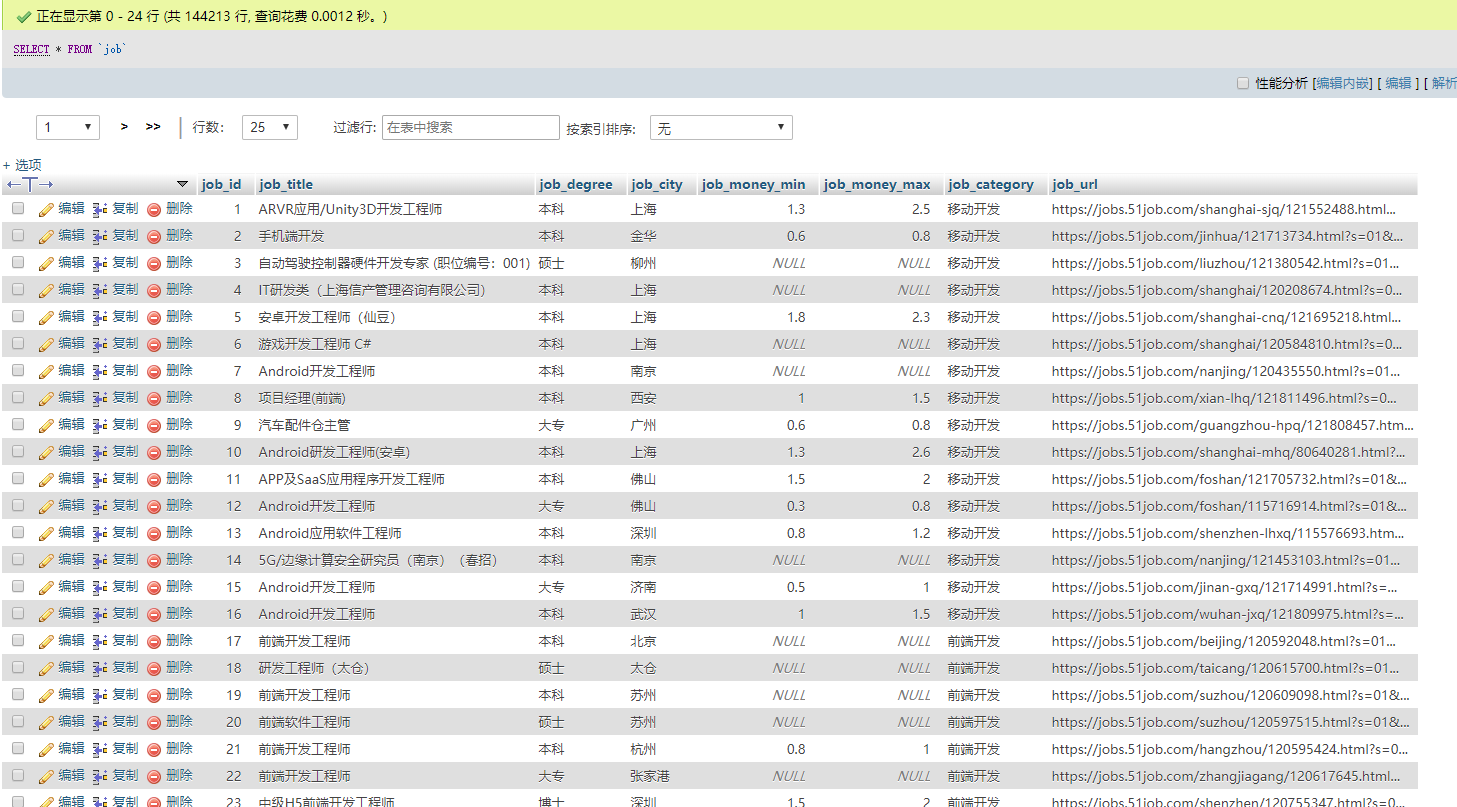


图7.3 数据成功入库

1. 执行manager.py文件，完成flask部分。如图7.4所示：

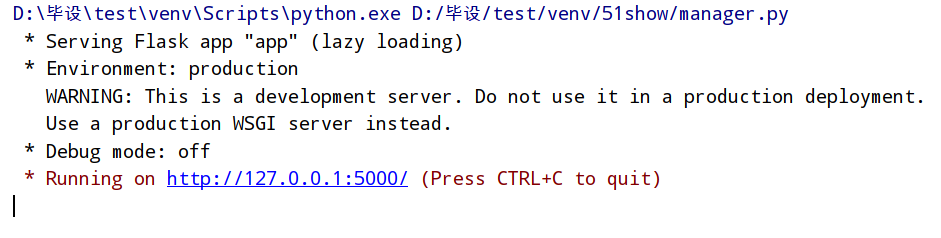


图7.4.启动Flask

4.本地访问前端，显示主页。如图7.5所示：

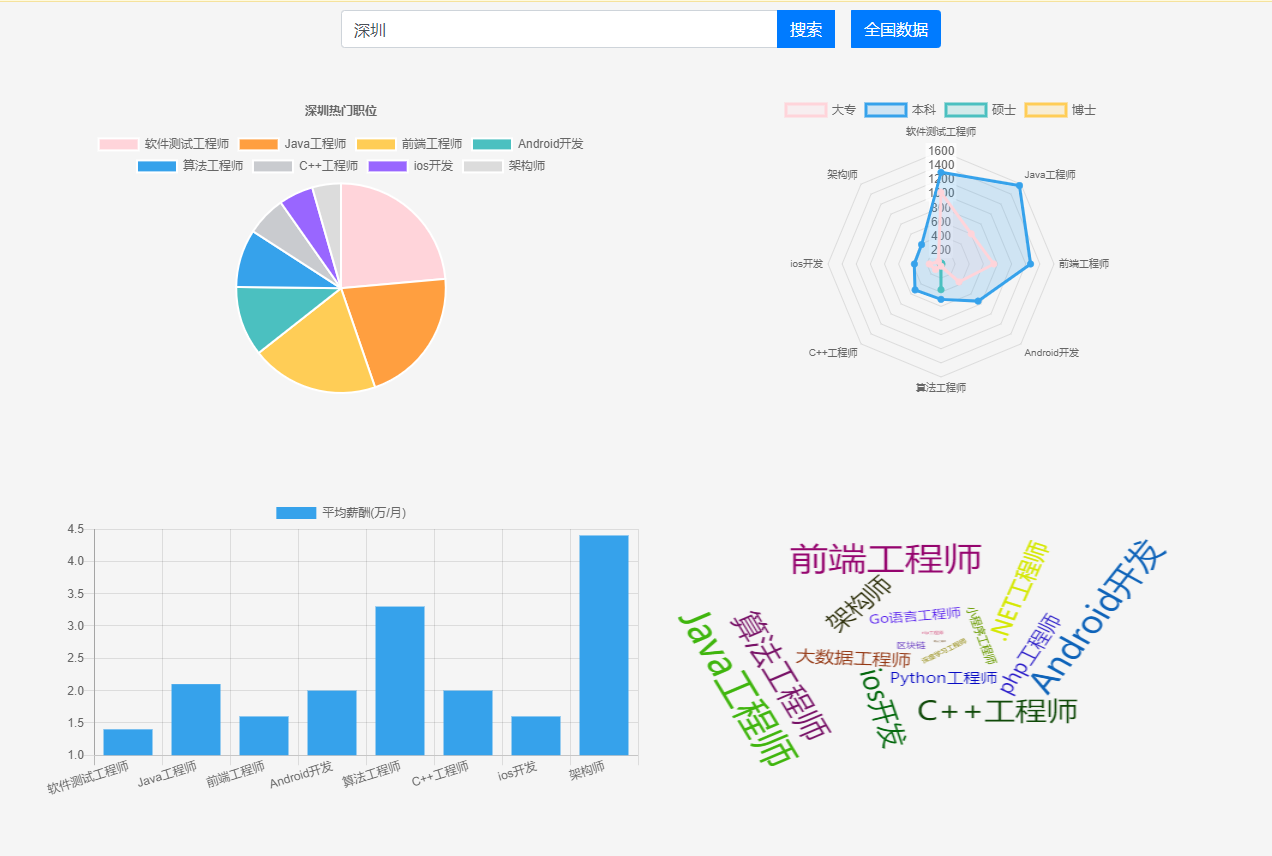


图7.5 首页效果图

5.输入想要查询城市，如图7.6所示：



图7.6 查询结果页面显示图

7.1.3 系统中存在的问题及解决方法

问题及解决方法如表7.1所示：

|  |  |
| --- | --- |
| 问题 | 对策 |
| 'scrapy' 不是内部或外部命令，也不是可运行的程序或批处理文件。 | 这是因为没有把Anaconda加入到环境变量中。在安装scrapy时，会将scrapy.exe文件安装到Scripts文件夹下面，如果没有设置环境变量，将无法找到scrapy.exe文件。 |
| ImportError: DLL load failed: 找不到指定的模块。 | 先升级pip,再重新导入第三方库，如果没有作用，再把相关的whl文件下载到本地，在python下的site-package里找到文件移入项目的site-package中。 |
| Request.url is not modifiable, use Request.replace() instead | 不能直接在process\_response直接修改url，需要调用方法：request.\_set\_url(url) |
| <twisted.python.failure.Failure <class 'OpenSSL.SSL.Error'> | 错误原因一般是http网站使用了https代理，更换http代理即可，此异常经常发生在重定向后的页面，可以禁止重定向，allow\_redirects=False |

表7.1 问题及解决方法

7.2 测试结果分析

本次程序主要目的是通过爬虫程序得到大量数据，然后加以分析，接下来以得到的全国数据为例，按照热门职业、热门职位薪资分布、学历分布三个模块来分析。全国热门职业图如图7.7所示。

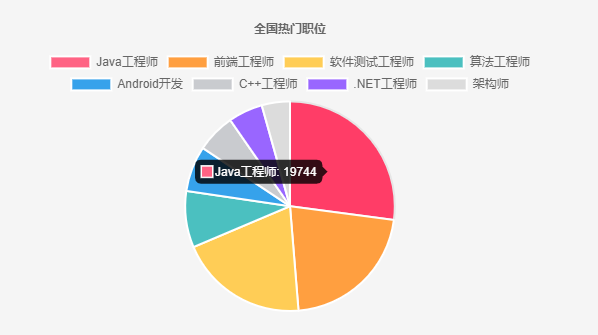


图7.7 全国热门职业图

全国计算机相关热门八大职业分别是前端工程师、软件测试工程师、算法工程师、安卓开发、架构师、IOS开发、.Net工程师、C++工程师，Java工程师。其中Java工程师占比最大，Java被誉为世界第一编程语言，从市场需求方面来分析，因为编程语言Java的应用是非常的广泛，前端后端都是主流地位，同时也是企业选作开发软件编程语言的首位，我们知道的是企业级的软件在市场上几乎是垄断级的，除了一些系统软件，几乎都是企业软件，它所涉及到的层面包括了财务资源管理、需求分析等软件。Java语言 在web开发中的地位也是甚高，中国乃至世界熟知的京东、淘宝的后台系统Java在其中举足轻重。Java自身的特性跨平台兼容性已经是非常大的优势，并且作为众多初学者的首选，所以整个市场中Java工程师的需求师非常高的，

平均薪酬图如图7.8所示：

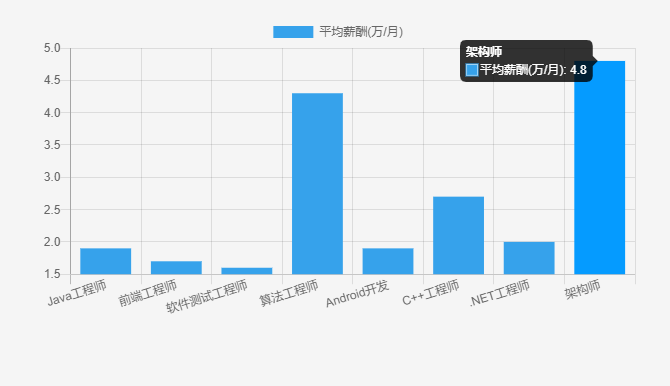


图7.8 平均薪酬图

虽然Java工程师是最热门的职业，但它的薪资方面却不是比较高的。算法工程师和架构师是比较高的。在最近的调查研究中发现，架构师的平均薪酬已经达到了4.8万/月，稳居热门职业中平均薪酬第一的宝座。为什么架构师的工资会这么高呢，我们首先得知道构架师是什么。架构师，就是一个软件的总体设计师，打个比方的话，他就类似于建筑学里的结构设计师，是软件开发与集成的构造者。架构师作为软件职业中的一门新兴行业，主要的工作职责就是在一个软件的开发创建过程中，收集客户的需求与意见，把其转化为规则合理的开发计划，并设计出整个软件的架构，指挥团队里的各个职位进行工作，完成这个开发计划。 架构师之所以薪酬高，可能与他的团队地位息息相关。架构师可以说是团队的领导级人物，他不仅能快速的发现一个问题的要点并及时做出合理的决定来解决问题，还能在各种不同的角度来看待问题思考问题以掌握全局，架构师这种出色的洞察力与执行力就已经让很多人难以企及，其突出的团队作用更是决定了他的含金量不是一般职业所能达到的。

学历分布图如图7.9所示：

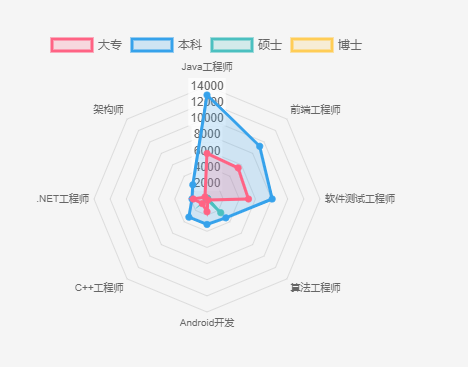


图7.9 学历分布图

从图中我们得知，在整个市场中本科的需求量是最大的，专科的需求量甚至超过了硕博，但仔细分析其中原因，发现并不是计算机行业不需要高学历，而是由于计算机行业大部分的职位是技术性的，所以门槛比其他职业的学历门槛要低一些，包容性更高一些。但是结合薪资，发现架构师和算法工程师是高学历占比比较大的，因为是需要学术和技术相结合的，而且计算机高端人才相当大一部分是在一线做学术研究的，所以市场需求没有体现出来。但是越是高精尖的人才越是稀缺，也不必担忧就业问题。

总 结

在家隔离的这几个月里，为了完成毕业设计，必须提高自己的自制力，而且选题选的是我自学的部分，没有课堂只是参考，所以更要加倍专注。我摸索出了一套适合自己的学习方法，坚持早睡早起，早上九点起来后就开始学习网课，下午开始跟着视频操作，晚上构思程序，根据所学的知识填补代码，疫情不能出去，那就在家里不断学习充实自己，疫情带来的负面心情，也会被书中的知识所冲淡。学习总会有疲惫期，但是通过同学和老师的鼓励还是会重新燃气热情，特别是在调试过程中，总会出现一些书上找不到的BUG,我会请教老师和同学，有时候逛逛论坛，发现有很多计算机从事者特别愿意分享他们的经验，并且会提醒注意事项，我想这就是热爱一门行业把，对待BUG，似乎有强迫症般的偏执，直到问题彻底解决了，我才会罢手。由于是晚上写代码，，或许一晚上一事无成，就为了学习一个函数，学会一种方法。所以总是不能坚持早睡，睡觉之前总是要玩一下手机，这一下就经常到了一点，导致白天经常精神看书没有精神，后来我就狠下心来，把手机放在我手够不到的地方，慢慢的我也适应了没有手机的日子，也习惯了早睡早起的生活，说实话，早睡早起真的是个很好的习惯，不仅精神好，而且对身体健康也很有益。说起来,python确实比较容易上手，给人的感觉就是说白话一样的翻译出来，但是为了实现我想要的功能，导入了许多第三方库，这时候我才发现，python之所以看起来简单，是因为拥有众多的第三方库，它们就像一件一件的小工具，但是另一方面，如果不了解它，就不会去用，就会用很大的力气可能敌不过一个import,于是，我每次写到一个功能是，先会查查有没有相应的库文件，然后再去学习它，找到自己想要的函数。

毕业设计是对大学的总结，不仅仅是技术，还有学习能力，大学里一直缺乏着动手精神，做项目也是推一步走一步，而且再也不去练习，好像白学了一样，做项目时大家分工明确，而毕业设计是一个人系统性的完成一个小项目，难度明显的增加了，但是时间更长，更能体现四年来的积累。这次的设计不是一个句号，而是未来计算机道路上一块小小的垫脚石，是它激发了我的热情，点燃我的斗志，明白了编程所带来的巨大成就感。