安徽工业大学

毕业设计(论文)任务书

课题名称	基于 Python 的招聘数据采集与分析
学　　院	计算机科学与技术
专业班级	计 144 班
姓　　名	胡乐晴
学　　号	149074117

指导教师签字：_____

摘　要

随着互联网技术的发展，数据在生活中显得愈加重要，数据的价值不断爬升，甚至成为一种商品。然而如何高效的获取并整合数据进而分析数据，让巨量并存在冗余的数据可视化，从而对各领域的专业或相关人士提供指导意义成为了一个亟待解决的问题。Python 作为一门胶水语言，在数据分析领域有着得天独厚的优势，本课题以招聘信息的采集与分析为主导路线。探索数据的研究价值和实用意义。

使用 Pycharm 作为 IDE，Python3.5 作为开发语言，在众多第三方库的支持下，实现图形界面的编程，网络爬虫的应用以及数据可视化。

系统选择 Python 作为唯一开发语言，开发效率高，系统总体框架清晰，易于维护和扩展。

关键词：招聘信息；爬虫；数据可视化；Python

装

订

线

Abstract

With the development of various technologies of the Internet, data becomes more and more important in our life. The value of data keeps climbing up and even becomes a commodity.However, how to obtain and integrate data and make kinds of analysis for efficient data, let there be huge amount and redundancy of data visualization, to every field or related professionals to provide guidance has become a pressing problem.Python, as a glue language, has a unique advantage in the field of data analysis. This topic is dominated by recruitment information collection and analysis.Explore the research value and practical significance of data.

Using pycharm as the IDE,Python3.5 as the development language, and a number of third-party libraries as the support, implementation of graphical interface programming, network crawler and data visualization.

The system chose python as the only development language with high development efficiency, clear overall framework and easy maintenance and expansion.However, because python itself has a global interpreted lock, it cannot implement true multithreaded processing, and the efficiency is not satisfactory.

Keywords: Recruitment Information;The crawler;Data Visualization;Python

装

订

线

目 录

装 订 线

装

订

线

第 1 章 绪论

1.1 课题背景

1.1.1 数据的特征与发展

随着互联网的兴起，信息技术的不断进步，数据逐渐成为人们关心的话题。自古以来，数据便与人类息息相关，数据的价值可能在过去往往被人们忽略，但他的存在却是不可辩驳的事实。数据可能是声音、图像等具有连续性的模拟数据，亦或者是抽象的符号、文字等具有离散性的数字数据。

在计算机出现之前，限于技术水平，人们无法对数据进行大量的存储，所以也无法对大规模数据进行合理的整合与分析，记录与分析数据的最佳方式便是利用文字、符号进行书面记录和各种数学计算。而随着电脑的出现，互联网行业的兴起，数据可以以二进制形式存储在计算机中，以小体积存储介质存储大量数据，数据的存储期同时得以延长，才有了对数据采集与分析的研究。现如今，计算机存储能力飞速提升并已颇具规模，各种复杂算法接踵而出，近年来的数据量成指数型增长，数据已经具备规模巨大、类型繁多、价值密度低等特征。数据的价值不断增长，逐渐被商业化，来自网络、支付系统、智能手机或者其他途径的数据发展成为一项资产，具有巨大的商业价值。可见，人类已经迎来了大数据时代。

1.1.2 数据采集技术的发展与分类

数据的价值已经展现，数据采集的技术也在应对需求不断更新。数据采集必须应对各种苛刻的要求，考虑来自客观环境的限制因素，包括保证数据的准确性、数据量达到要求的数量级、数据采集对时间的消耗不能过大以及数据采集的成本问题。不同行业、领域对数据的采集有各种不同的专业手段，数学、物理、医学等领域的专业人士利用专业仪器进行数据的采集，而计算机领域使用较多的数据采集技术便是网络爬虫技术。

网络爬虫是一种自动获取网页 HTML 源码的脚本或程序，英文名为 Spider，有蜘蛛之意，HTML 意为超文本标记语言，是网页信息的主要载体。现代意义上，世界上第一个网络爬虫程序是 Archie，于 1990 年诞生，其出现的时候万维网尚未诞生。Archie 也可以说是第一个通用网络爬虫，使用者可以通过书写表达式进行查询。

互联网兴起之初，互联网规模较小，通用网络爬虫可以满足基本的信息查找。随着网络发展，互联网飞速发展，万维网成为信息的主要载体，网络爬虫技术不断发展。大批互联网公司开发的搜索引擎都是基于链式获取 Uniform Resource Location（url）的原理从互联网上获取网页源码，通过浏览器加以渲染，以大众易于浏览的形式呈现，一个页面中可以包含大量其他页面的 url，用户甚至可以以一个页面为起点，访问整个互联网的信息。通用的搜索引擎可以获取大量的数据，但是却缺乏针对性，不同行业人士所需数据有所差异，通用搜索引擎所返回的数据结果包含大量用户不关心的内

容，面对当今数据呈现的规模，必须有针对性的获取数据才能发挥数据的最大价值。

聚焦爬虫的工作流程比通用网络爬虫更为繁杂，为了过滤大量的与用户需求无关的网页的 url，聚焦爬虫包含大量算法分析过程，以从海量的数据堆中剥离出用户所关心的数据。这也正是爬虫程序员在挖掘数据的道路上最关心的一个方向，也是本文对于数据采集研究部分最关心的问题。

截至 2007 年底，Internet 上的网页数量已超过 160 亿，截至 2017 年底，国内 Internet 上的网页数量已经超过 2000 亿个，虽然其中存在大量重复的页面，但爆炸式的增长趋势已是客观存在的事实。人们在爬虫领域也开辟了新的道路，增量式网络爬虫，Deep Web 爬虫等相继出现。如果把互联网比作一片大海，网络爬虫就是那个大海捞针的探索器。网络爬虫技术作为计算机领域主要的数据采集技术，在数据领域至关重要。

1.1.3 数据分析的发展

数据分析是计算机科学与数理统计学的完美结合。计算机科学为分析者提供充足的技术支持，提供充足的数据源与分析工具，统计学为分析者提供良好的分析理论。两者结合发展，使如今的数据分析呈现多元化。其中，数据可视化更是一个精彩的亮点，以图形展示数据，无疑使数据更大众化。

1.2 研究意义

现在,传统的线下求职模式已逐渐转变为线上求职模式，大学生更倾向于通过招聘网站提交求职意向，招聘网站的出现使各公司信息更加透明化。本文通过采集招聘网站信息，进行整合、分析，形成可视化图形，为求职者提供参考，并作为数据分析研究的一个实例，将计算机编程技术与统计学加以结合应用。挖掘出数据的价值。其中，在数据采集之后，对数据进行进一步清洗，为数据分析提供完美契合的数据源具有关键性作用。

如今中国的招聘市场逐渐由互联网招聘模式统治，五八同城、前程无忧、51Job、智联招聘等招聘网站成为大学生和职场中人求职的主要途径，本课题通过对智联招聘网的信息爬取，准备数据分析所需的基本材料，然后做出多种形式的可视化图形，给求职者以指导意见。在这个人才竞争激烈的时代，让各行业人才更好的看清行业形势，衡量自身价值具有重要意义。

1.3 国内外发展趋势

为迎接大数据时代，国内知名企业，如阿里巴巴、百度、腾讯、网易都提供了相应的云服务，并通过数据分析取得了巨大的竞争优势。互联网企业也在大量引进大数据应用，数据分析的前景在国内已毋庸置疑。国外的各大企业，Apple、Yahoo 等，普遍利用数据分析进行企业战略部署。网络招聘公司也逐步注重通过数据分析对人才进行更合理化分配。如今，网络招聘已有属于自己的产业链及市场，等待着迎接更大的挑战。

第 2 章 开发工具综述

2.1 Pycharm

程序员在不适用集成开发环境的情况下，开发一个桌面应用程序是很困难的，Python 也不例外。本课题采用 Pycharm-community-2018.1.3 作为 Python IDE 进行开发。Pycharm 的优势在于支持所有版本的 Python 代码的语法报错。本课题是一个包含图形界面、爬虫、可视化分析的桌面应用程序，Pycharm 可充分发挥其优势。

2.2 Anaconda 和 Python 的第三方库

本课题选用的是 Anaconda3-4.1.0，是一个开源的 Python 发行版本，是数据分析领域最常用的 Python 发行版本。其功能强大处在于自带大量的科学包，不需要开发者自己导入常用的数据包，同“python”一词一样，Anaconda 具有蟒蛇之意，本课题中所用 Anaconda 版本自带 Python3.5。

Python 诞生于 1991 年，据说是荷兰人 Guido von Rossum 为了打发无聊的圣诞节而编写的一门编程语言，以其简单优雅的编程哲学而风靡全球。从 Python2.x 到 Python3.x 在语法上进行了了很多改变，本课题使用的是 Python3.5 版本。对于 Python 的第三方库，则不胜枚举，甚至有人说，Python 离开了它的第三方库便寸步难行，开发不出任何有意义的项目。本课题以 tkinter 库进行图形用户界面的编程，通过 Sqlalchemy 库与数据库进行连接并且可以在连接的基础上对数据库进行增、删、改、查一系列操作，Requests 库用于模拟浏览器发送 HTTP 请求从而获取页面响应，使用 BeautifulSoup 库结合正则表达式解析响应，获得所需的数据，利用 Pandas，Numpy 库进行数据处理，最后使用 Seaborn 和 Wordcloud 进行可视化分析。

2.3 正则表达式

正则表达式是指以程序员自己设计的特定语法匹配字符串的工具。正则表达式的用法很多，功能也很强大，但如果一个正则匹配稍有差池，那可能程序就处在永久的循环之中，正则表达式的使用需要使用者十分小心，Python 中的 Re 模块即为正则表达式模块，用于解析网页，获取需要的数据而除去不关心的数据。Beautiful Soup 则是更为强大的提取 HTML 和 XML 标签中内容的工具，本课题中使用到 Re 和 Beautiful Soup 两个库。

第 3 章 系统需求分析与数据库设计

3.1 系统需求分析

本课题开发的是一个桌面应用程序，包含用户管理、数据采集与数据分析的功能。主要目的是抓取智联招聘网站上的招聘信息，存入本地 csv，txt 文件，同时导入到本地 Sql Server 数据库，作为储备数据源，最后通过对数据源的分析以可视化图形的形式呈现结果。由于一个桌面应用程序的开发需要先进行严谨的需求分析。所以将系统需求分析陈列如下：

（1）注册功能：使用一个桌面应用程序的前提是先拥有使用它的权限，用户可以通过注册一个账号从而获取使用权限。

（2）登录功能：拥有程序的使用权限之后，需要进行登录操作，对用户的个人信息进行验证，符合条件，才可以使用程序。

（3）重置密码功能：一旦忘记密码 ，可以通过其他途径重置密码，前提是重置密码提供的信息具有公正性，并且与数据库中的既存信息可以匹配。

（4）数据采集功能：数据采集功能分为两部分：选择界面部分和爬虫部分速度，采集的数据存为 csv 文件和 txt 文件，同时存入本地的 Sql Server 数据库。由于智联招聘所涵盖数据量巨大，启动爬虫时，用户应该自己选择一些限制条件，如职位名所包含的关键字，城市，地区等条件，这样获取的数据更加有针对性。当然，也可以不选择任何限制条件，进行全局搜索，爬取到数据。但由于智联招聘网站设置城市为必选参数，所以选择界面中的城市为必选项。无论其网站数据库多大，但总是有限的，所以页数参数也是必选项，不选会弹出提示信息，无法启动爬虫。

（5）薪资分析功能：以数据库中的指定行业指定地区的薪资数据作为数据源，进行分析，制成可视化柱形图，呈现给用户。

（6）词频统计功能：以数据库中的指定行业制定地区的任职要求，职位工作详情信息为数据源，进行数据分析，制成可视化的词云，词云的背景图可任意，默认为 Python 的图标图片，数据源中存在大量文本数据，将文本切分为有实际意义的词，排除与分析无关的停用词，根据剩余词出现的频率制成词云。词云是现代数据分析处理文本的流行方式。

（7）教育程度分析功能：学历通常是企业关心的一个问题，也是求职者求职时的一块敲门砖。通过对数据库中教育程度数据的提取、处理、可视化分析，制成饼图，呈现给用户。

（8）企业规模分析：企业规模是求职者关心的一个基本问题，通过对其分析，以折线图的形式呈现给用户。

（9）使用说明：介绍本程序的功能以及各分析结果的实际含义。

以上为本课题的需求分析，需求分析的制定让我清楚了整个软件的功能、流程，为软件开发理清了思路。本文侧重于实现软件的功能，在技术上，性能上有所突破，所以在交互性上要求不高。图形用户界面为用户提供了使用功能的入口，登录成功后可以使用 Spider 进行获取信息，爬虫运行过程中会遇到来自智联招聘的反爬虫机制

的阻碍，突破各种反爬成为本课题至关重要的技术难点。csv 文件与 txt 文件的存储上也会出现一些问题，文件名需要采用动态命名，根据搜索的关键字，以一定规则进行命名。验证程序的数据采集功能则需要通过观察 csv 文件，txt 文件的变化来判断，或者通过界面显示或者数据库中内容的增加进行判断。此外，爬取数据是一个需要等待的过程，要做到能够显示爬虫进度。

本程序可分为三大模块，用户管理模块，数据采集模块，数据分析模块。数据采集模块与数据分析模块都具有很好的扩展性。如果想爬取到另一网站数据，只需构造新的 url 地址，设计新的正则表达式即可。而利用现有的数据源可以做的数据分析类型则更加千变万化了。

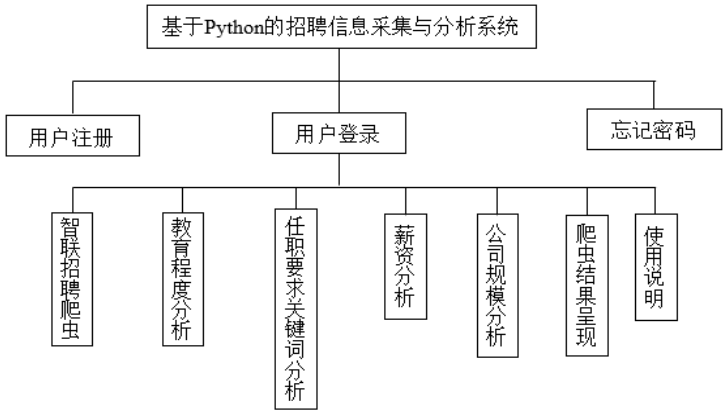

图 3-1 需求分析框架图

3.2 可行性分析

本课题开发之前需要对数据采集与分析系统需要用到的技术、经济、市场进行可行性分析：

（1）技术可行性分析：本系统使用 Python 的发行版 Anaconda 进行开发，需要使用到 Python 自带模块及 Python 第三方库。tkinter 作为 Python3.5 自带的用于图形界面编程的库可以满足本课题对于图形界面的编程要求，tkinter 在 python3 之后增加了 ttk 模块，自适应各种操作系统的图形界面风格，运用 PIL 库可以向界面中加入图片，使界面更加美观。Python 编程语言用来开发效率较高，能够在短时间完成一个逻辑严密的桌面应用程序。存储数据采用微软的 Sql Server 数据库，Sql Server 高效，并且在微软的支持下十分完善。本课题需要在数据库中建立三张表，使用 Navicat 这一数据库图形管理工具操作起来效果卓著。Python 的 Sqlalchemy 库提供 Sql 工具及对象关系映射工具，可以轻松使 Python 代码与 Sql Server 数据库对接，所以数据库的支持完全可以满足本课题的需求。Python 可以利用 Csv 库读写 csv 及 txt 文件，爬虫的结果数据可以通过 Windows 下自带的 Excel 与记事本工具查看。Pandas 与 Numpy 作为使用 Python 进行数据分析的主流科学包，功能强大，运用 Matplotlib 库和 Seaborn 库可进行数学建模并呈现出统计学图表。Jieba 与 Wordcloud 库配合可以绘出词云图。使用 Requests 获取网页之后，观察智联招聘网页 HTML 代码结构，合理设计正则表

达式，理论上都可以得到用户想要的数据，但是数据源网站都为应对爬虫，保护自身服务器而设计了反爬虫机制，这是本课题所要处理的重点和难点。总而言之，有了众多的第三方库的支持，加上对智联招聘网站代码的细致分析，本课题在技术方面是可行的。

（2）经济可行性分析：本课题基于 Python 语言，涉及正则表达式、众多第三方库，使用 Pycharm 社区版，Sql Server 数据库，这些都是开源的开发工具。同时，本课题作为我大学本科的毕业设计，学校给予了我充分的自由时间。综合这两方面因素，本次基于 Python 的数据采集与数据分析研究成本是非常低的。

（3）市场可行性分析：随着大数据时代的潮流，众多公司都是以数据为中心，数据挖掘也成为高薪产业，国家也发表文件声明支持大数据的发展。网络爬虫作为数据挖掘的常用工具具有长远前景，数据分析产业对未来的各行业都具有指导意义。智联招聘作为大型招聘网站，数据充足，于 1997 年建立，已有十几年历史，在其知名度的影响下，本课题以后的市场发展是可行的。

3.3 数据库设计

3.3.1 概述

本文分不同模块进行数据库设计，总共涉及到三张表，分别是用户信息表、地区信息表和招聘信息表。用户信息表存储用户的个人信息，在注册用户时向用户信息表插入提交的信息，登录时读取用户信息表，通过验证决定是否登录成功。地区信息表是为了分类采集与分类分析而建立的表，存储城市与地区的对应关系，城市与城市代码的对应关系，地区名与地区代码的对应关系。招聘信息表是一张用来存储所有招聘信息的表，爬虫的结果都存入这张表，并且作为数据分析的数据源。下面详细介绍数据库中三张表的结构设计。

3.3.2 数据库表设计

（1）用户信息表：用户信息表与注册功能、忘记密码功能、登录功能相关联。用户注册时会提交用户名、密码、邮箱三个字段的数据，并产生一个用户 id。用户名、密码字段用于登录使用，用户名、邮箱作为找回密码时的验证信息。由于登录的用户 id 具有唯一性，所以设为主键。

下图为用户信息表的 E-R 图：

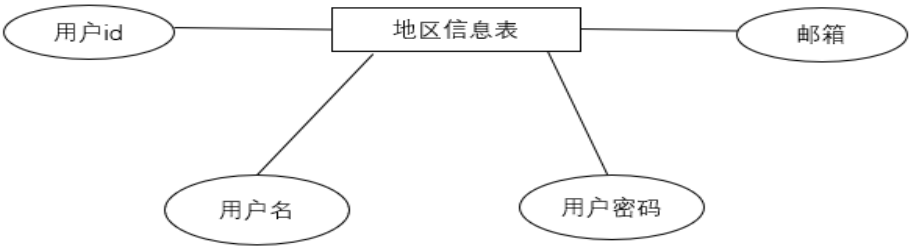

图 3-2 用户信息表 E-R 图

（2）地区信息表：爬虫地区分类，数据分析时对数据库中数据进行选择时需要使用，启动网络爬虫时需要根据被爬网站的 url 结构设计传入的参数，本课题中设计了城市、地区、关键字、页面数四个参数。由于智联招聘的 url 设计时，汉字形式城市名可直接作为字符串类型的参数，输入的关键字也可直接作为字符串类型的参数、页面数可直接作为整型参数，但智联招聘网为各个城市的不同地区设置了相应的编码。故为了分类采集的功能，建立了地区信息表。地区信息表包括五个字段，分别是城市、城市代码、地区、地区代码、城市代码与地区码拼接成的城市地区码作为唯一标识符，即本表的主键。

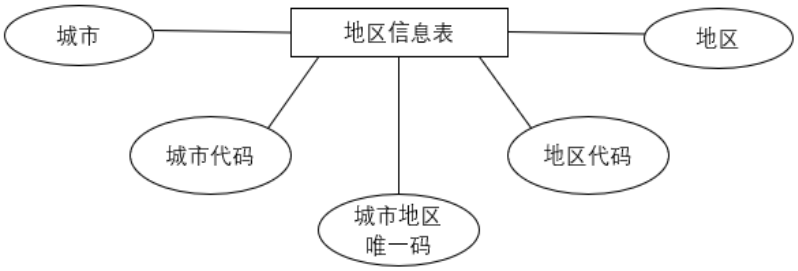

图 3-3 地区信息表 E-R 图

（3）招聘信息表：招聘信息表是本课题中至关重要的一张表。数据采集的结果保存在招聘信息表中，所有的数据分析都是基于这张表建立的。表中各字段与网站招聘信息各内容要求一一对应，由于职位详情内容较多，直接存在数据库中降低数据库整体效率，所以存储在具有一定命名规则的一系列 txt 文件中，招聘信息表中开辟一个字段存储各 txt 文件的文件路径。表中还设计了关键字字段，用于存储分类采集时提交的关键字，在后面的数据分析中会使用到。

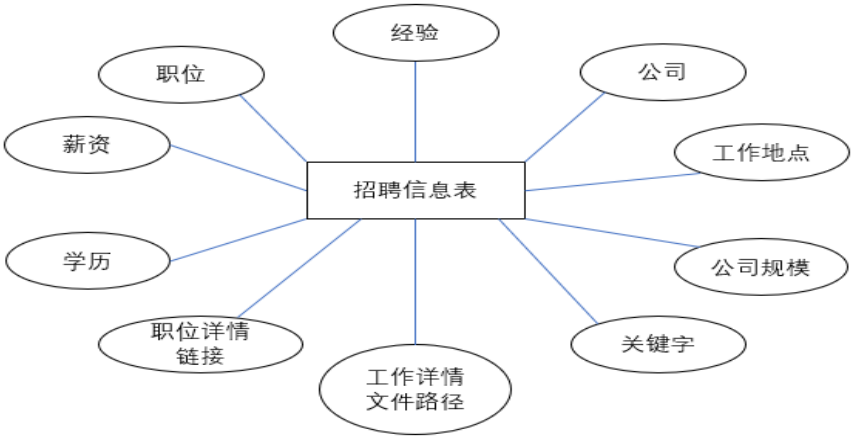

图 3-4 招聘信息表 E-R 图

3.3.3 数据库表结构

表 3-1 用户信息表（user_information）

字段名	数据类型	是否为空	是否主键	说明
user_id	Varchar(20)	Not null	是	用户 id
user_name	Varchar(20)	Not null	否	用户名
password	Varchar(20)	Not null	否	密码
e_mail	Varchar(40)	Not null	否	邮箱

表 3-2 地区信息表（region_information）

字段名	数据类型	是否为空	是否主键	说明

装 订 线

unique_code	Varchar(20)	Not null	是	城市地区唯一码
city	Nvarchar(20)	Not null	否	城市
city_code	Numeric(18,0)	Not null	否	城市代码
region	Varchar(20)	Not null	否	地区
region_code	Numeric(18,0)	Not null	否	地区代码

表 3-3 招聘信息表（recruit_information）

字段名	数据类型	是否为空	是否主键	说明
job_name	Nvarchar(80)	null	否	职位
company	Nvarchar(40)	null	否	公司
experience	Nvarchar(20)	null	否	工作经验
salary	Numeric(18,0)	null	否	薪资
location	Nvarchar(20)	null	否	工作地点
education	Nvarchar (10)	null	否	学历
scale	Nvarchar (20)	null	否	公司规模

装 订 线

job_url	Varchar (80)	Not null	是	职位详情链接
keyword	Varchar(80)	Not null	否	关键字
job_requirement	varchar(20)	null	否	职位详情

第 4 章 系统总体设计

4.1 开发环境与代码组织结构

（1）操作系统：Windows 10 家庭版。

（2）内存：4GB。

（3）Python 开发包版本：Anaconda3.4.1 开发包，自带 Python3.5。

（4）数据库：sql server 2008。

（6）本系统项目在 Pycharm 中创建，需要导入的 libraries 有：

Tkinter：编写图形用户界面；

Re：正则表达式库；

Csv：csv 文件操作库；

Requests：模拟浏览器发送 get 形式请求；

Sqlalchemy：对象关系映射工具；

PIL：标准图形库；

Matplotlib：python 的 2D 绘图库；

Numpy：python 的一种开源的数值计算扩展；

Jieba：分词功能；

Wordcloud：词云工具库。

（7）代码组织结构：

start.py：系统的入口，包含登录

signUp.py：注册功能

forgetPassword.py：忘记密码功能

mainPage.py：主页面，包含采集与分析两部分的功能的入口

sqlEngine.py：封装了与数据库的连接代码

mySpider.py：爬虫代码

salaryAnalysis.py：薪资分析代码

educationAnalysis.py：教育程度分析代码

scaleAnalysis.py：企业规模分析代码

wordAnalysis.py：职位要求常用词出现频率分析部分的代码

代码存放于 code 文件夹，另外创建文件夹 result file，存放爬虫完成后生成的 csv 文件与 txt 文件。创建文件夹 image 存储图形界面调用的图片，result image 文件夹存储分析结果图片。Stopwords 存放停用词表。

下列为文件和代码组织结构图：

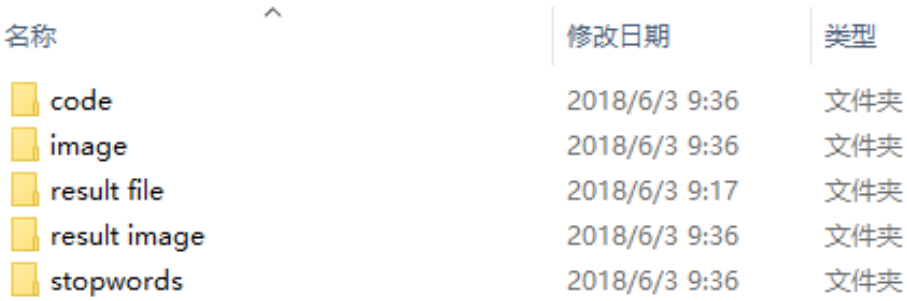

图 4-1 文件组织结构图

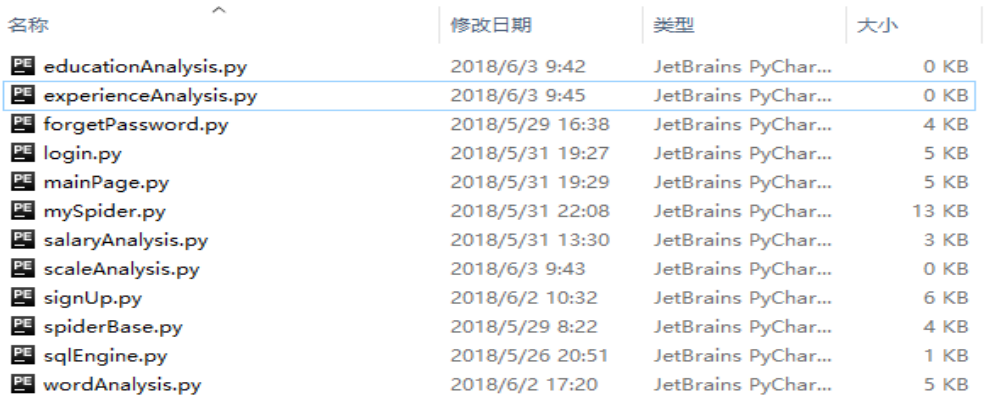

图 4-2 代码组织结构图

4.2 系统流程设计

基于程序的完整性与健壮性，设计该系统的流程图如下。首先打开招聘信息的采集与分析系统登录界面。注册一个新的账号，然后登录进入主界面，主界面相当于一个功能菜单。然后选择分类采集功能，选择城市，地区，输入关键字和页数，就可以开始采集了。爬虫完毕后，界面中会出现爬虫结果，同时爬虫结果存入数据库，如果对数据量不满意，可以重新选择爬虫目标，再次启动爬虫，获取更多的招聘信息。数据库中有了数据之后就可以对已有数据进行分析，选择要进行的分析类型。同采集数据一样，可以进行招聘信息的筛选，然后显示结果。

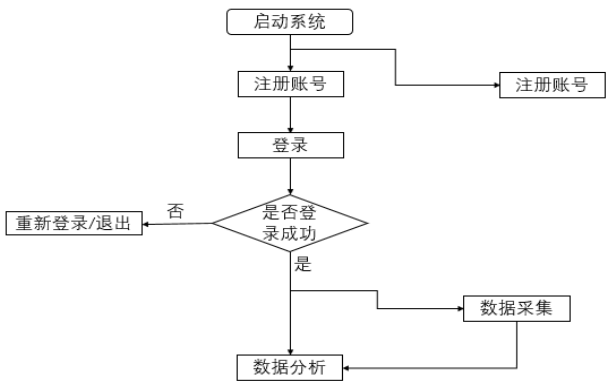

图 4-3 系统流程图

4.3 主要功能模块设计

4.3.1 数据采集模块

基于网络爬虫技术的数据采集模块是本课题的关键，数据采集是数据分析可以进行下去的前提条件，如果没有数据或者数据量不够大，数据分析也就没有任何现实意义。基于网络爬虫的重要性，对其功能流程进行总体设计。

（1）通过选择界面，提交自己选择的爬虫范围。

（2） 根据选择器传入的参数为爬虫设计唯一资源标识符，客户端向服务器发送 HTTP 请求。

（3）将服务器上的 HTML 代码整个下载下来。

（4）设计正则表达式解析返回的网页，并且保存到 Sql Server 数据库的指定的表中。

爬虫过程中会遭遇到各种反爬机制，反爬机制的解决措施这部分内容会在详细设计中介绍。

4.3.2 数据分析模块

数据分析是呈现本课题的成果的模块，不仅要用到计算机科学，还要使用到统计学相关知识。本课题对招聘信息进行多方位分析。

（1）薪资分析：指定城市指定地区指定行业的薪资水平或者整个城市的薪资水平，由于不同城市的房价和物价不同，同样行业的薪资水平在地域上存在差异，有的甚至相差很大，所以对同一行业不在全国总体数据进行分析。

（2）任职要求关键字分析：各个职位都有任职要求、工作职责两部分描述。将这两部分文本下载下来。分析其中各词汇出现的频率，判断每个职位招人的关键点是什么，应聘该职位需要掌握的相关技术是什么。

（3）教育程度分析：对学历要求数据分析，分析出各学历层次人才所占比例。

（4）企业规模分析：对公司信息分析，显示各种规模的公司的数量并进行比较。

数据分析会随着数据源的规模增长而更在具有正确性，对人的决策更加具有指导意义。

第 5 章 系统详细设计

5.1 入口模块

5.1.1 模块功能介绍

进行命令行界面，到指定文件目录下，本课题中是 C:\Users\hulq\myProject 作为文件绝对路径。执行命令”python start.py”即可打开系统。其中包括登录、注册、忘记密码功能的入口。

图 5-1 登录界面

图 5-2 注册界面

图 5-3 忘记密码界面

装

订

线

5.1.2 模块流程图

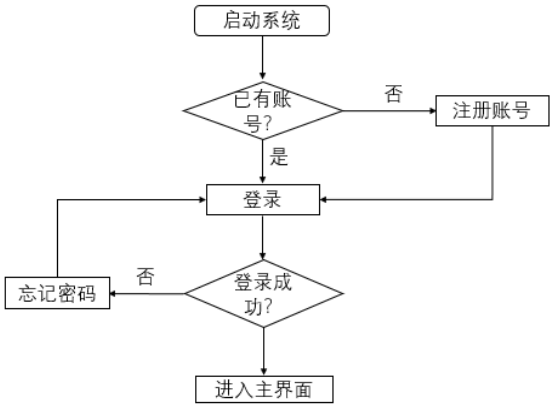

图 5-4 系统入口模块流程图

5.1.3 模块实现

（1）首先是图形用户界面的编程，使用的是 python3.5 自带的 tkinter 库。tkinter 中的 Tk（）函数创建了一个窗口，给这个窗口设置标题，大小，可调整度等基本参数。配合使用 label（标签）、entry（文本框）、button（按钮）、canvas（绘图）控件，基本的界面都可以完成。由于本课题重心不在 gui 编程，使用的图形界面较为简洁，实现难度等级为简单等级。

（2）登录功能：实现登录功能时，先从 entry 控件中获取用户信息，entry 控件 get（）方法可以捕捉到文本框中的文本，返回为 String 对象。如果存在信息缺失，通过 tkinter.messagebox.showinfo()函数可以弹出对话框，显示提示信息。然后从用户表

中读出已存在的用户信息，将用户登录时输入的信息与用户表匹配，如果该用户在用户表中存在，则登录成功，否则失败。

（3）注册功能：与登录类似，通过 entry 控件获取信息，然后与用户表进行比较，如果提交的用户名已存在，则证明该用户名已被使用过，用户名在本课题中是唯一的，所以注册失败，反之，提交的用户名为一个新的用户名，则将提交的数据存入用户表，得到一个新的用户记录。

（4）忘记密码：输入用户名和邮箱以及新密码，与用户表中的数据匹配，成功则更新用户表，重置密码。

5.2 爬虫模块

5.2.1 模块功能介绍

从主界面点击分类采集按钮，进入爬虫界面，选择城市、地区，输入关键字、页数后启动爬虫，开始采集，数据存储入库，显示爬虫进度。爬虫完毕后，可回到主界面，点击显示结果，查看本次爬虫结果。

图 5-5 爬虫界面

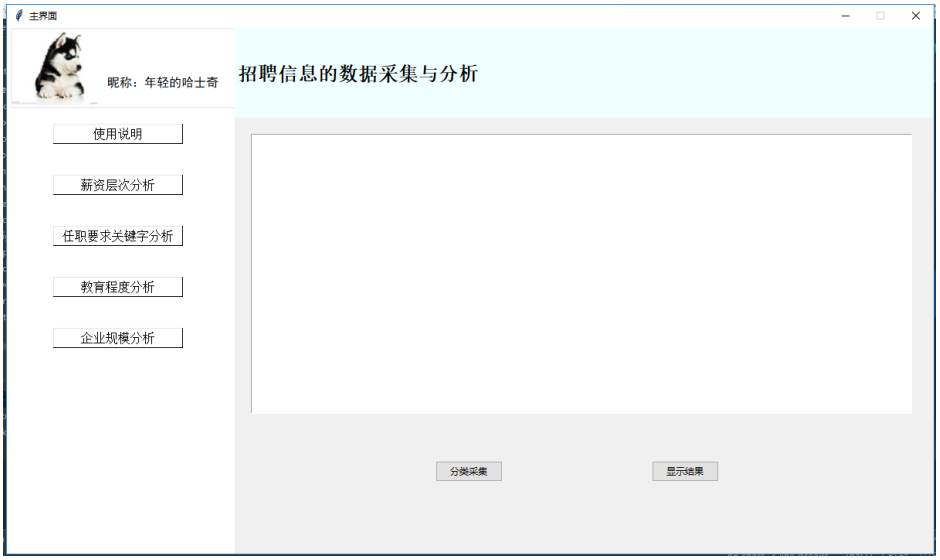

图 5-6 主界面

5.2.2 模块流程图

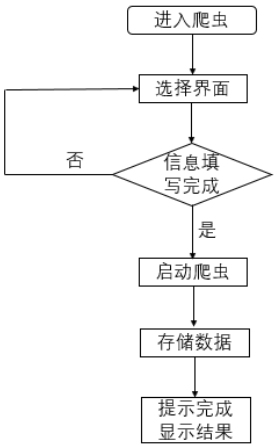

图 5-6 爬虫模块流程图

5.2.3 模块实现

首先实现选择器功能，创建两个下拉框控件、两个文本框控件，两个下拉框控件进行绑定，第一个下拉框中可以选择城市，第二个下拉框中可以选择城市中的地区。根据城市从数据库中的地区信息表中读出数据，动态更新第二个下拉框中的地区选项。下列为爬虫功能实现的详细设计：

（1）获取页面：取选择器中的值，将地区的值根据地区信息表替换为地区代码。城市、地区、地区代码四个字段作为参数。基本 url 加上参数构建成目标 url 并声明为purpose_url。基本url为'https://sou.zhaopin.com/jobs/searchresult.ashx?'。调用requests，发送 get 方式的 http 请求，获取一个 response，从中提取 text 类型的文本内容，这样就获得了一个页面。由于智联招聘的会检测请求是否来自于浏览器，如果不是来自浏览其会拒绝该请求。我为此设计了一个请求头 headers，在头部添加模拟火狐浏览器登录的用户代理信息。

（2）解析页面：获取一个页面之后，接下来的任务就是设计正则表达式解析页面，去除我们不关心的信息，留下我们需要的信息。经观察发现，第一个请求页面的只提供了基本的职位信息，包括职位名称、公司名称、薪资范围。不足以满足我的全部需求。为此需要从这个页面找到职位详情页面的 url 地址，进入职位详情页面寻找更多的信息。为此，设计正则表达式如下：

匹配职位详情地址和职位名称：

'<td class="zwmc".*?href="(.*?)" target="_blank">(.*?).*?'

匹配公司名称：'<td class="gsmc">.*? target="_blank">(.*?).*?'

匹配月薪：'<td class="zwyx">(.*?)</td>'

获取的职位详情地址作为新的 url，发送第二次 get 方式的 http 请求。这里选择 BeautifulSoup 库进行解析，提取出的内容分为八个标签，分别是职位月薪、工作地点、发布日期、工作性质、工作经验、最低学历、招聘人数、职位类别，筛选任职要求。

最终返回工作经验、学历、任职要求、公司规模的数据。至此，获取到了所有需要的数据。

（3）翻页功能：一个页面解析完成之后，需要进行翻页爬取下一个页面的内容。一般情况下，有两种方法实现翻页功能：从第一个页面中找到下一页的 url 或者寻找网站 url 的变化规律。智联招聘的 url 变化方式固定，本课题选第二个方法。

（4）字段处理：获取的数据有些字段需要进行清洗， 获取的职位名称中含有大量'<b>'字符，利用 replace('<b>', '')函数将其替换为空。薪资范围字段的值是一个字符串，取中间值作为薪资值，一些职位的薪资范围里面填写的是“面议”，碰到这中情况就在薪资分析时跳过这条数据。任职要求中存在大量的换行符与空格以及标点符号需要去掉。

（5）智联招聘的反爬虫机制及应对措施：智联招聘其数据本就具有公开性，反爬虫机制不多，在实现爬虫功能时仅遇见了一个反爬虫的保护措施。每次发送请求时会产生一个随机动态码，智联网在翻页时将这个动态码嵌入 url 中，如果不解决这个问题，每次只能爬取到第一个页面的数据。经观察发现，这个动态码写在前一个页面的 html 代码中，我设计了一个正则表达式，捕捉到这个动态码，每次执行翻页操作之前，更新动态码参数即可。

（6）写入文件：写入文件中保存只是为了方便自己检查爬虫的结果。调用 csv 库中的相应方法即可。

（7）存储数据到数据库：保存数据到数据库使用的是 Python 的 Sqlalchemy 模块，第一步是获取与本地的 Sql Server 数据库的连接。连接代码我封装在 sqlEngne.py 这个文件中。主要代码是：

engine=sqlalchemy.create_engine('mssql+pyodbc://{0}:{1}@{2}'.format(user,pwd,database))。

每次获取并处理完一条记录，就执行一次 sql 语句，把一条招聘信息写入数据库，逐条写入的好处是可以节省内存。

5.3 数据分析模块

5.3.1 模块功能分析

（1）薪资分布：选取数据库中的 recruit_information 表的 salary 字段，统计出各个薪资段的职位数，制作出柱状图呈现给用户。

（2）任职要求关键字分析：根据用户选择的条件在数据库中的 recruit_information 表中读出 job_requirement 字段，根据该字段存储的绝对路径找到相应的 txt 文件，读取 txt 文件中的文本信息，使用结巴分词进行文本分析，然后利用 wordcloud 统计各关键词的出现次数，一些无意义的词可以加到停用词表 stopwords.txt 文件中，统计时会跳过这些停用词，最后以词云图的形式展示出来。

（3）学历要求分析：选取数据库中的 recruit_information 表的 education 字段，统计出各个学历的职位数，计算出各学历层次所占百分比，制作出饼图呈现给用户。

（4）企业规模分析：选取数据库中的 recruit_information 表的 scale 字段，统计出各个企业规模的职位数，制作出折线图呈现给用户。

5.3.2 模块流程图

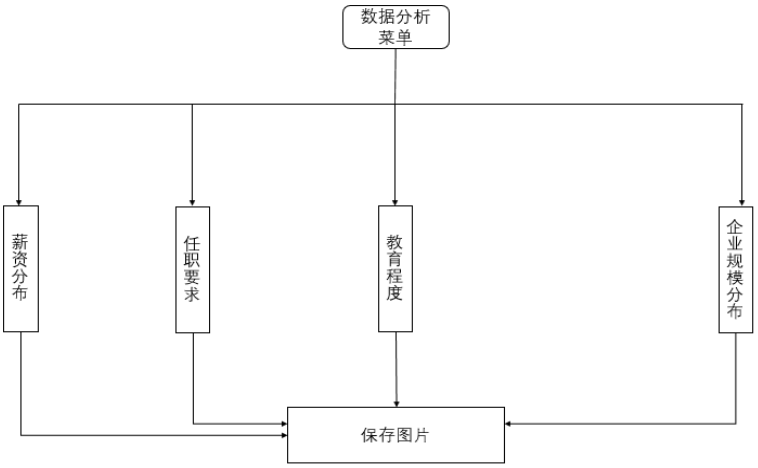

图 5.7 数据分析模块流程图

5.3.3 模块实现

数据分析时使用的选择界面与爬虫时使用的选择界面实现原理相同，在这里不加赘述。

（1）薪资分析：数据库中提取数据，以 2000 为间隔，0 为起始刻度，设置薪资等级，如 less than 2k、2k-4k，作为横坐标，职位个数作为纵坐标，制作成柱状图。

（2）任职要求关键字分析：读取 recruit_information 表，根据关键字选择 txt 文件。读出 txt 中的内容，返回一个 String 对象 comment，通过 jieba.lcut()的分词功能将 comment 切分为一个个词，并返回为一个 list 对象 segment，通过 pandas.DataFrame({'segment':segment})将 segment 转化为一个字典对象 words_df。同时，读取同级目录下的 stopwords.txt，返回一个字典对象 stopwords，words_df 与 stopwords 比较，去掉 words_df 中 stopwords 中含有的词，生成一个新的字典 words_stat。采用分组统计的方法统计出各词出现的次数，并按从大到小的规则排序。然后，设置词云的属性，包括：设置字体可以显示中文，背景颜色为白色，词云显示的最大词数为一百，设置背景图片，设置字体的 size 最大值为 100，设置图片默认的大小为宽等于 1000、高度等于 860、边缘宽度为 2，之后就可以生成词云。

（3）学历要求分析：学历层次分为不限、高中、中专、大专、本科。读取数据库中的 education 字段所有数据。计算出各学历层次的职位个数所占百分比，调用 matplotlib.pyplot 中封装好的 pie 方法生成饼图。

（4）企业规模分析：对数据库中所有企业规模数据进行整合，制成折线图。

5.3.4 分析结果的意义

（1）薪资分布：可以根据这一分析，看出某地区某一职业的薪资分布情况，判

装 订 线

断出该类型职位哪个薪资段的人数最多，达到月薪两万以上的职位数多不多，对衡量自身能力水平，给自己的价值进行合理定位具有指导意义。

（2）任职要求关键字分析：根据描述词出现的次数，改变词云中词的大小，可以让人一目了然，看出企业招聘新人时的注意点在哪里，可能是经验，也可能是技术栈，通过这一分析，求职者可以看出自己求职时需要注意的点，增加求职成功率。

（3）学历要求分析：如今社会学历的重要性在下降，企业更看重个人能力，但学历任然是求职的敲门砖，技术岗位对学历的要求一般在本科以上。

（4）企业规模分析：可以看出国内各种规模的企业的数量分布。

第 6 章 系统测试

6.1 测试目的

为了检测程序是否可用，爬虫的性能和数据分析结果的正确性，进行一系列系统测试。

6.2 测试过程与结果展示

6.2.1 基础模块测试

基础模块分为注册、登录、忘记密码模块。设计测试过程如下图：

注册成功界面：

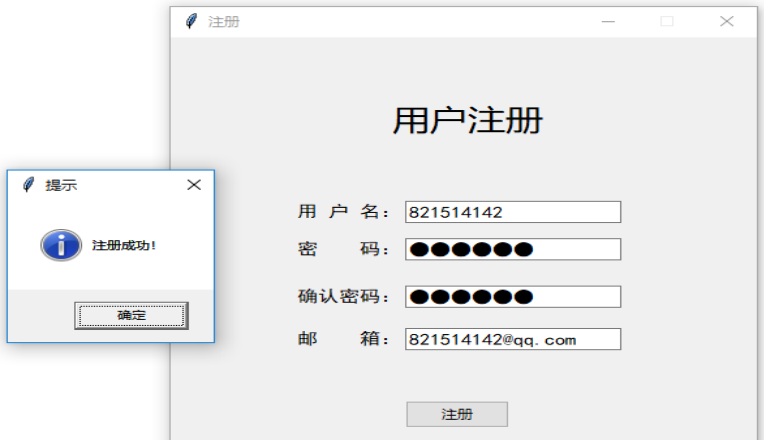

图 6-1 注册成功图

登录成功界面：

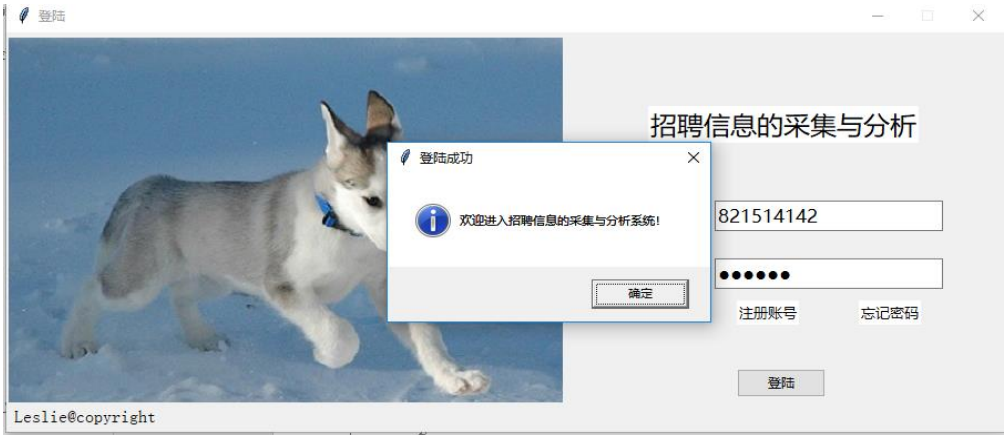

图 6-2 登录成功图

密码重置成功界面：

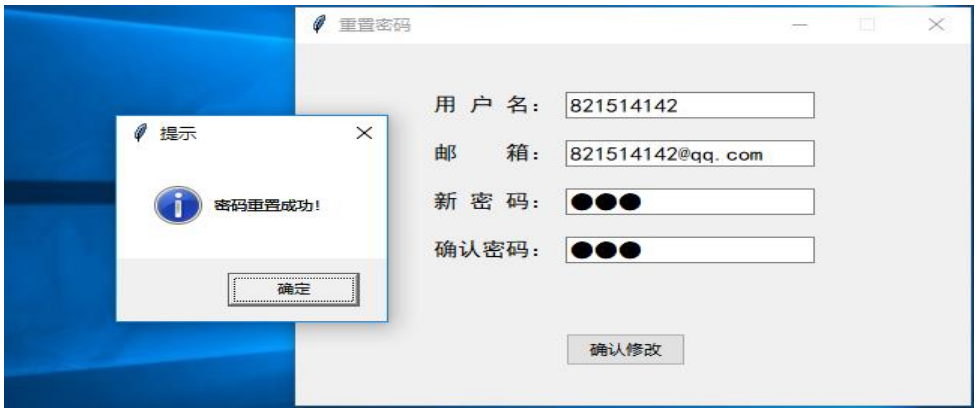

图 6-3 密码重置成功图

6.6.2 爬虫模块测试

登录成功后进入主界面，之后会看到有分类采集和显示结果两个按钮，显示结果按钮只能显示本次爬虫结果，如果未进行爬虫，会提示需要先进行爬虫，点击分类采集按钮，跳转到爬虫界面，选择好条件之后点击可以看到爬虫进度。爬虫完成后可以显示爬虫结果。以下为测试过程截图：

提示未进行爬虫：

图 6-4 提示图

显示爬虫进度：

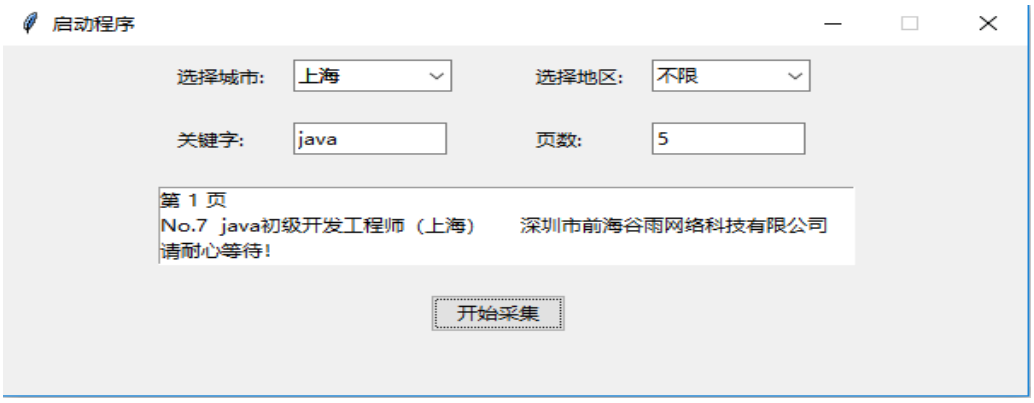

图 6-5 爬虫进度图

显示结果：

图 6-6 爬虫结果显示图

6.6.3 数据分析测试

选择指定分析，程序从数据库中调数据进行分析。

点击薪资分析按钮出现下面选择界面，选择条件如下：

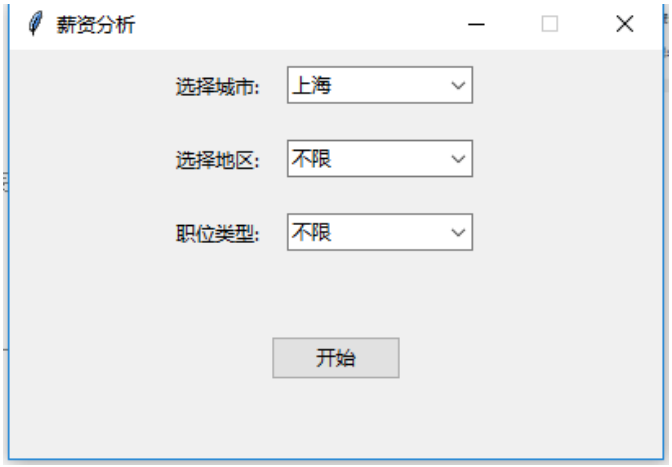

图 6-7 选择界面

点击开始，展示分析结果：

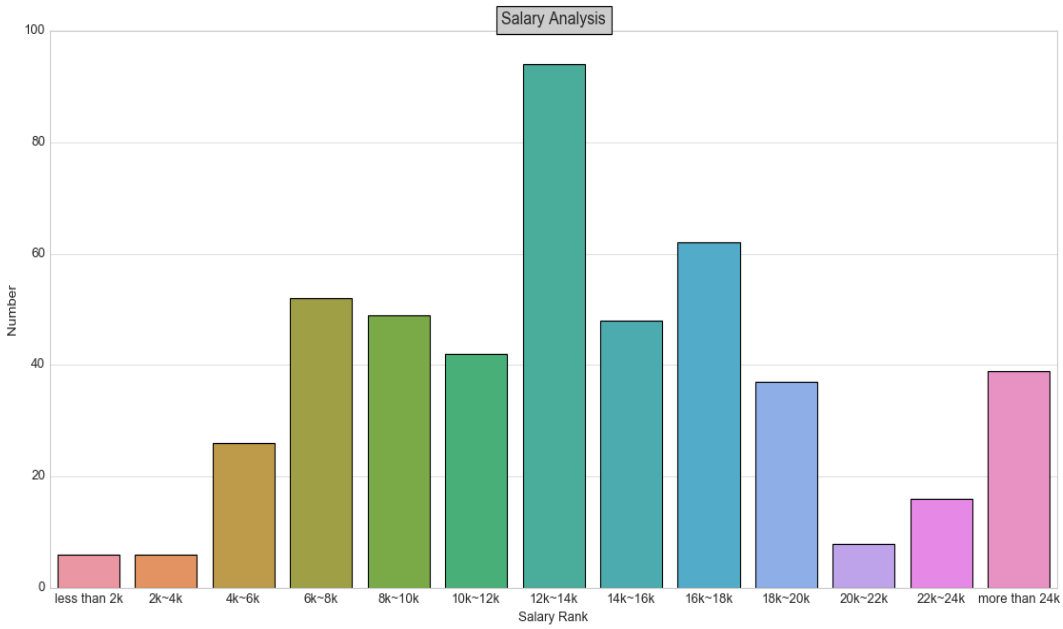

图 6-7 薪资分析柱状图

任职要求关键字分析：

图 6-8 任职要求关键词词云

教育程度分析：

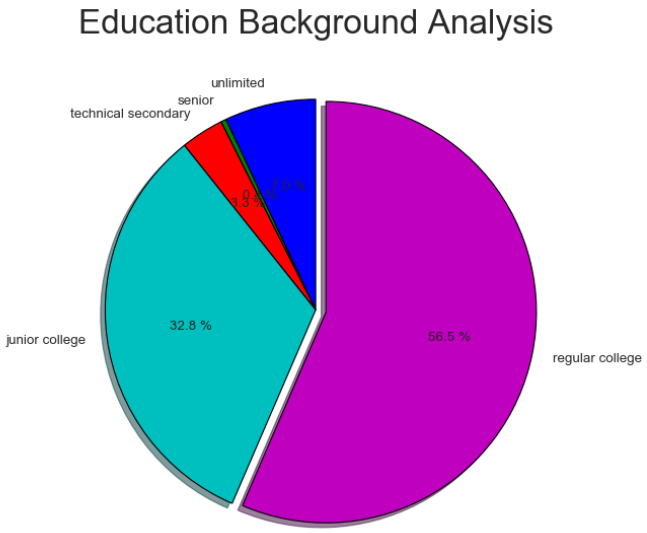

图 6-9 教育程度分析饼图

企业规模分析：

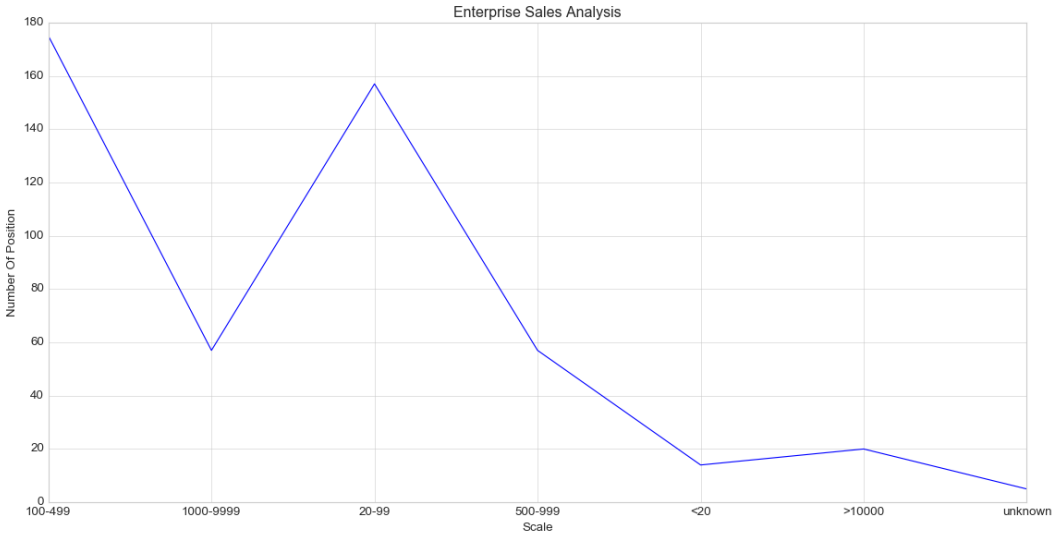

图 6-10 企业规模分析折线图

装 订 线

结论

通过本课题的研究，让我对 Python 语言有了更深入的了解，体会到了 Python 第三方库的强大。对图形界面的编程有了一定的了解，为以后编写更美观，交互性更好的图形界面打下了基础。经过对 Sql Server 的应用，发现这个数据库对字符串的编码较其他数据库更加细致，其中对字符串的定义分添加了 nvarchar 类型，nvarchar 定义所有字符都占两个字节，利用好这一特点可以提高数据库的性能。

爬虫是一个搜集数据的利器，编写爬虫过程中有一些常见的问题。对于本课题中所爬取的智联招聘网站，首先，存在代理请求的问题，采用设置 user-agent 的方式处理，本文模拟火狐浏览器发送 HTTP 请求。

在存储数据时，写数据进库是会出现 warning 模块的警告，虽然数据存储成功，但对数据库的操作方式还需要进一步改进。正则表达式是一个非常实用的工具，在以后解析页面时，是否能够合理的设计正则表达式是一个很关键的问题。

通过对招聘信息的研究，发现当前国内互联网行业发展火热，人工智能人才稀缺，人工智能相关领域薪资水平很高。互联网职位大多分布在一线城市和二线城市，公司规模在 1000 人以上的大多在北京上海等一线城市。

参考文献

[1]邵丹.基于大数据技术的社交网络招聘研究.[J]中国战略新兴产业，2018.

[2]李鑫. 基于 Python 的软件测试自动化平台[D]. 太原科技大学，2014.

[3]郭涛，黄铭钧.社区网络爬虫的设计与实现[J].智能计算机与应用.2012(04).

[4](美)James Payne.Python 编程入门经典[M].2009.

[5]田利云.面向学位论文的致谢分析和生成研究[M].大连理工大学.2017.

[6]刘杨.融合网络数据摄取的企业关系网络平台的设计与实现[M].山东大学.2017.

[7]刘江锋，基于领域的网络爬虫研究与实现[M].电子科技大学.2017.

[8](法)A. Garrido-Fernández; A. Cortés-Delgado; A. López-López,Tentative application of compositional data analysis to the fatty acid profiles of green Spanish-style Gordal table olives.[J]. Food Chemistry 2018.6.15

[9]刘晶晶.面向微博的网络爬虫研究与实现[D].复旦大学，2012.

[10]周立柱，林玲.聚焦爬虫技术研究综述[J].计算机应用，2014.

[11] Barry.Head First Python.2012

装

订

线

致谢

本次毕业设计和论文是在我的指导老师安徽工业大学的姜太平老师的悉心指导下完成。老师对我的毕业设计工作十分关心，初步完成时的系统功能缺失严重，经过老师的分析在最后的时间对系统进行了完善。老师十分关注我的工作进展，使我可以按时完成毕业设计与论文，同时老师对论文要求严格，在后期进行了多次改善。在此过程中，姜太平老师科学的教学方法和严谨的工作态度深深影响了我，感谢姜太平老师。

同时，在大学四年，各专业课老师的用心教导，使我在基础课学习期间打下了一定基础，感谢那些在我们学习道路上不可缺少的老师们。

装

订

线

附录

Code：

登录功能：

```
def login_in(self):
    # 获取输入的用户信息
    user_name = self.user_name.get()
    password = self.password.get()
    #获取用户表信息
    engine = se.sql_engine()
    sess = sessionmaker(bind=engine)
    session = sess()
    info = session.execute('select user_name,password from dao.login_info')
    # 临时变量
    x = True
    for i in info:
        if user_name == i[0] and password == i[1]:
            # 登录成功
            showinfo(title='登录成功',message='欢迎进入爬虫小程序！')
            x = False
            break
            if x:
                showinfo(title='登录失败',message='请核对你的用户名和密码！')
            else:
                # self.root.quit()
                self.root.destroy()
                mainPage.MainPage()
```

注册功能：

```
def sign_in(self):
    # 获取输入的值 v 代表 value
    v_user_name = self.user_name.get()
    v_password = self.password.get()
    v_check_password = self.check_password.get()
    v_e_mail = self.e_mail.get()
    # 提示
    if v_user_name == '':
        msg_u = '用户名不为空！\n'
    else:
        msg_u = ''
        self.i+=1
    if v_password == '':
        msg_p = '密码不为空！\n'
    else:
        msg_p = ''
        self.i+=1
    if v_check_password == '':
        msg_cp = '确认密码不为空！\n'
    else:
        msg_cp = ''
        self.i+=1
    if v_e_mail == '':
        msg_e = '邮箱不为空!\n'
    else:
        msg_e = ''
```

```
        self.i+=1
    if v_password != '' and v_check_password != '' and v_password != v_check_password:
        msg_ck = '确认密码输入有误！'
    else:
        msg_ck = ''
        self.i+=1
    if self.i == 5:
        #条件控制
        con = False
        # 创建会话
        engine = se.sql_engine()
        sess = sessionmaker(bind=engine)
        session = sess()
        # 读取用户信息表
        names = session.execute('select user_name from dao.login_info')
        for name in names:
            name = name[0]
            if v_user_name == name:
                con = True
                break
            else:
                continue
            # print(con)
            if con:
                showinfo(title='提示',message='用户已存在！')
            else:
                # 将用户信息存入数据库
                try:
                    session.execute("insert into dao.login_info (user_name,password,e_mail)
                    values ('{0}','{1}','{2}')".format(v_user_name,v_password,v_e_mail))
                    session.commit()
                    showinfo(title='提示',message='注册成功！')
                    self.i = 0
                    self.si.destroy()
                except:
                    showinfo(title='提示',message='输入格式有误！')
                    self.i = 0
                self.i = 0
            else:
                showinfo(title='提示',message=msg_u+msg_p+msg_cp+msg_e+msg_ck)
                self.i =0
    忘记密码功能：
    def forget_password(self):
        # 获取输入的值 v 代表 value
        v_user_name = self.user_name.get()
        v_e_mail = self.e_mail.get()
        v_new_password = self.new_password.get()
        v_check_password = self.check_password.get()
        # 创建 session 对象
        engine = se.sql_engine()
        sess = sessionmaker(bind=engine)
        session = sess()
        if v_user_name != '' and v_e_mail != '' and v_new_password != '' and v_check_password != '':
            # 检查用户名与邮箱是否匹配
            emails = session.execute("select e_mail from dao.login_info where
            user_name='{0}'".format(v_user_name))
```

```
            if v_e_mail == emails.first()[0]:
                if v_new_password == v_check_password:
                    session.execute("update dao.login_info set password = '{0}' where user_name
                    = '{1}'".format(v_new_password,v_user_name))
                    session.commit()
                    self.root.destroy()
                    showinfo(title='提示',message='密码重置成功！')
                else:
                    showinfo(title='提示',message='信息输入有误！')
            else:
                showinfo(title='提示',message='请确认你的信息完整！')
```

爬虫功能：

```
def get_one_page(city, keyword, region, page,sg):
    # 获取网页 html 内容并返回
    paras = {
        'jl': city,                  # 搜索城市
        'kw': keyword,               # 搜索关键词
        'fl': 530,
        'isadv': 0,                  # 是否打开更详细搜索选项
        'isfilter': 1,               # 是否对结果过滤
        'sm': 0,
        're': region,                # region 的缩写，地区，2005 代表海淀
        'sg': sg,
        'p': page                    # 页数
    }
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
        like Gecko) Chrome/63.0.3239.132 Safari/537.36',
        'Host': 'sou.zhaopin.com',
        'Referer': 'https://www.zhaopin.com/',
        'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
        g,*/*;q=0.8',
        'Accept-Encoding': 'gzip, deflate, br',
        'Accept-Language': 'zh-CN,zh;q=0.9'
    }
    url = 'https://sou.zhaopin.com/jobs/searchresult.ashx?'
    try:
        # 获取网页内容，返回 html 数据
        response = requests.get(url, params=paras, headers=headers)
        print(response.url)
        # 通过状态码判断是否获取成功
        if response.status_code == 200:
            return response.text
        return None
    except RequestException as e:
        return None
def parse_one_page(html):
    # 解析 HTML 代码，提取有用信息并返回
    # 正则表达式进行解析
    pattern = re.compile('<td class="zwmc".*?href="(.*?)" target="_blank">(.*?)</a>.*?' # 匹配
    职位详情地址和职位名称
    '<td class="gsmc">.*? target="_blank">(.*?)</a>.*?' # 匹配公司名称
    '<td class="zwyx">(.*?)</td>', re.S) # 匹配月薪
    # 匹配所有符合条件的内容
    items = re.findall(pattern, html)
```

```
        for item in items:
            job_name = item[1]
            job_name = job_name.replace('<b>', '')
            job_name = job_name.replace('</b>', '')
            salary_average = 0
            temp = item[3]
            if temp != '面议':
                idx = temp.find('-')
                # 求平均工资
                salary_average = (int(temp[0:idx]) + int(temp[idx+1:]))//2
            # html = get_detail_page(job_url)
            # print(html)
            yield {
                'job': job_name,
                'job_url': item[0],
                'company': item[2],
                'salary': salary_average
            }
    # 获取动态 sg
    def getSg(html):
        '''
        针对智联招聘的反爬虫设置的动态码，解决方法
        '''
        sg = ''
        # 正则表达式进行解析
        pattern = re.compile('<div class="pagesDown".*?sg=(.*?)&', re.S)
        # 匹配所有符合条件的内容
        items = re.findall(pattern, html)
        for item in items:
            sg = item
        return sg
    def get_detail_page(url):
        # 获取职位详情页 html 内容并返回
        headers = {
          'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
          like Gecko) Chrome/63.0.3239.132 Safari/537.36',
          'Host': 'jobs.zhaopin.com',

          'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*
          /*;q=0.8',
          'Accept-Encoding': 'gzip, deflate',
          'Accept-Language': 'zh-CN,zh;q=0.9'
        }
        try:
            # 获取网页内容，返回 html 数据
            response = requests.get(url, headers=headers)
            # 通过状态码判断是否获取成功
            if response.status_code == 200:
                return response.text
            return None
        except RequestException as e:
            return None
    def get_job_detail(html):
        requirement = ''
        # 使用 BeautifulSoup 进行数据筛选
        soup = BeautifulSoup(html, 'html.parser')
```

```
        # 找到<ul class="terminal-ul clearfix">标签
        for ul in soup.find_all('ul', class_='terminal-ul clearfix'):
            # 该标签共有 8 个子标签，分别为：
            # 职位月薪|工作地点|发布日期|工作性质|工作经验|最低学历|招聘人数|职位类别
            lis = ul.find_all('strong')
            # 工作经验
            years = lis[4].get_text()
            # 最低学历
            education = lis[5].get_text()
        # 筛选任职要求
        for terminalpage in soup.find_all('div', class_='terminalpage-main clearfix'):
            for box in terminalpage.find_all('div', class_='tab-cont-box'):
                cont = box.find_all('div', class_='tab-inner-cont')[0]
                ps = cont.find_all('p')
                # "立即申请"按钮也是个 p 标签，将其排除
                for i in range(len(ps) - 1):
                    requirement += ps[i].get_text().replace("\n", "").strip()    # 去掉换行符和空格
        # 筛选公司规模，该标签内有四个或五个<li>标签，但是第一个就是公司规模
        scale=soup.find(class_='terminal-ul clearfix
        terminal-company mt20').find_all('li')[0].strong.get_text()
        return {'years': years, 'education': education, 'requirement': requirement, 'scale': scale}
def write_csv_file(path, headers, rows):
    '''
    将表头和行写入 csv 文件
    '''
    # 加入 encoding 防止中文写入报错
    # newline 参数防止每写入一行都多一个空行
    with open(path, 'a', encoding='gb18030', newline='') as f:
        f_csv = csv.DictWriter(f, headers)
        f_csv.writeheader()
        f_csv.writerows(rows)
def write_csv_headers(path, headers):
    '''
    写入表头
    '''
    with open(path, 'a', encoding='gb18030', newline='') as f:
        f_csv = csv.DictWriter(f, headers)
        f_csv.writeheader()
def write_csv_rows(path, headers, rows):
    '''
    写入行
    '''
    with open(path, 'a', encoding='gb18030', newline='') as f:
        f_csv = csv.DictWriter(f, headers)
        # 如果写入数据为字典，则写入一行，否则写入多行
        if type(rows) == type({}):
            f_csv.writerow(rows)
        else:
            f_csv.writerows(rows)
def write_txt_file(path, txt):
    '''
    写入 txt 文本
    '''
    with open(path, 'a', encoding='gb18030', newline='') as f:
        f.write(txt)
```

```
def main(city, keyword, region, pages,sg):
        # 主函数
        csv_filename = 'zl_' + city + '_' + keyword + '.csv'
        txt_filename = 'zl_' + city + '_' + keyword + '.txt'
        headers = ['job', 'years', 'education', 'salary', 'company', 'scale', 'job_url']
        salaries = []
        write_csv_headers(csv_filename, headers)
        for i in range(1, pages):
            # 获取该页中所有职位信息，写入 csv 文件
            job_dict = {}
            # 返回一个 response.text
            html = get_one_page(city, keyword, region, i, sg)
            # 解析 response.text,返回一个发生器
            try:
                items = parse_one_page(html)
            except:
                continue
            for item in items:
                # 获取详细信息，返回一个 response.text
                try:
                    d_html_2 = get_detail_page(item['job_url'])
                    job_detail = get_job_detail(d_html_2)
                    job_dict['scale'] = job_detail.get('scale')
                    job_dict['years'] = job_detail.get('years')
                    job_dict['education'] = job_detail.get('education')
                    job_dict['job'] = item.get('job')
                    job_dict['salary'] = item.get('salary')
                    job_dict['company'] = item.get('company')
                    job_dict['job_url'] = item.get('job_url')
                    print(job_dict)
                    # 对数据进行清洗，将标点符号等对词频统计造成影响的因素剔除
                    pattern = re.compile(r'[一-龥]+')
                    filterdata = re.findall(pattern, job_detail.get('requirement'))
                    print(filterdata)
                    write_txt_file(txt_filename, ''.join(filterdata))
                    write_csv_rows(csv_filename, headers, job_dict)
                except:
                    continue
            # 获取动态码
            sg = getSg(html)
        showinfo(title='提示',message='数据爬取完毕！')
class Spider(object):
    def __init__(self):
        self.engine = se.sql_engine()
        self.sess = sessionmaker(bind=self.engine)
        self.session = self.sess()

        self.list = ['不限']
        self.root = tkinter.Tk()
        self.root.title('启动程序')
        self.root.resizable(0, 0)

        # window size and center
        self.window_width = 600
        self.window_height = 150
        self.screen_height = self.root.winfo_screenheight()
```

```
        self.screen_width = self.root.winfo_screenwidth()
        self.x = (self.screen_width - self.window_width) / 2 + 200
        self.y = (self.screen_height - self.window_height) / 2
        self.root.geometry("%dx%d+%d+%d" % (self.window_width, self.window_height, self.x,
self.y))
        '''
        创建控件
        '''
        # label
        self.lp_city = ttk.Label(self.root, text='选择城市:')
        self.lp_region = ttk.Label(self.root, text='选择地区:')
        self.lp_keyword = ttk.Label(self.root,text='关 键 字:')
        self.lp_pages = ttk.Label(self.root,text='页数:')
        #  文本框
        self.keyword = ttk.Entry(self.root,width=12)
        self.pages = ttk.Entry(self.root,width=12)
        self.number = StringVar()
        self.cityChosen = ttk.Combobox(self.root, textvariable=self.number, width=10,
        state='readonly')
        self.cityChosen['values'] = ('上海', '北京', '广州', '深圳', '杭州', '苏州')
        self.cityChosen.current(0)
        self.cityChosen.bind('<Button-1>', self.change)
        self.regionChosen = ttk.Combobox(self.root, width=10, state='readonly')
        self.regionChosen['values'] = ('不限', '嘉定', '杨浦', '浦东新区', '青浦', '黄浦', '闸北', '崇
        明县', '静安', '虹口', '长宁', '普陀', '闵行', '徐汇', '金山', '宝山', '松江', '奉贤')
        self.regionChosen.current(0)
        self.start = ttk.Button(self.root, width=10, text='开始采集', command=self.start_spider)
        '''
        布置控件
        '''
        self.lp_city.grid(column=0, row=0, sticky=NW, padx=100, pady=12)
        self.lp_region.grid(column=0, row=0, sticky=NW, padx=310, pady=12)
        self.lp_keyword.grid(column=0, row=1, sticky=NW, padx=100, pady=12)
        self.lp_pages.grid(column=0, row=1, sticky=NW, padx=310, pady=12)
        #
        self.cityChosen.grid(column=0, row=0, sticky=NW, padx=170, pady=10)
        self.regionChosen.grid(column=0, row=0, sticky=NW, padx=380, pady=10)
        self.keyword.grid(column=0, row=1, sticky=NW, padx=170, pady=10)
        self.pages.grid(column=0, row=1, sticky=NW, padx=380, pady=10)
        #
        self.start.grid(column=0, row=2, sticky=NW, padx=250, pady=10)
        # self.root.mainloop()
    def change(self, event):
        region = self.session.execute("select region from dao.region where city_province =
        '{0}'".format(self.cityChosen.get()))
        for i in region:
            self.list.append(i[0])
        self.regionChosen['values'] = tuple(self.list)
        self.regionChosen.update()
        self.root.update()
        self.list = ['不限']
    def start_spider(self):
        v_city = self.cityChosen.get()
        region = 0
        if self.regionChosen.get() == '不限':
            region = ''
```

```
                else:
                    items = self.session.execute("select region_code from dao.region where region =
                    '{0}'".format(self.regionChosen.get()))
                    for i in items:
                        region = i[0]
                keyword = self.keyword.get()
                if self.pages.get() == '':
                    showinfo(title='提示', message='请填写页数!')
                else:
                    pages = int(self.pages.get())+1
                # print(v_city,region,keyword,pages)
                sg = ''
                main(v_city, keyword, region, pages, sg)
                self.root.update()
                self.root.destroy()
```

薪资分析：

```
        def find_file(self):
            path = self.openfile()
            try:
                sal = read_csv_column(path, 3)
            except:
                showinfo(title='提示', message='请核对你的数据源文件！')
                self.root.destroy()
            # 撇除第一项，并转换成整形，生成新的列表
            for i in range(len(sal) - 1):
                # 工资为'0'的表示招聘上写的是'面议',不做统计
                if not sal[i] == '0':
                    self.salaries.append(int(sal[i + 1]))
            self.root.destroy()
            plt.hist(self.salaries, bins=10 ,)
            plt.show()
```

任职要求关键词分析：

```
        def find_file(self):
            txt_filename = self.openfile()
            content = read_txt_file(txt_filename)
            print(content)
            print(type(content))
            segment = jieba.lcut(content)
            print(segment)
            print(type(segment))
            words_df = pd.DataFrame({'segment':segment})

            stopwords = pd.read_csv("stopwords.txt",index_col=False,quoting=3,sep="
",names=['stopword'],encoding='utf-8')
            words_df = words_df[~words_df.segment.isin(stopwords.stopword)]

            words_stat = words_df.groupby(by=['segment'])['segment'].agg({"计数":numpy.size})
            words_stat = words_stat.reset_index().sort_values(by=["计数"],ascending=False)

            # 设置词云属性
            color_mask = imread('background.jfif')
            wordcloud = WordCloud(font_path="simhei.ttf",     # 设置字体可以显示中文
                                  background_color="white",          # 背景颜色
                                  max_words=100,                     # 词云显示的最大词数
                                  mask=color_mask,                   # 设置背景图片
                                  max_font_size=100,                 # 字体最大值
```

```
                                    random_state=42,
                                    width=1000, height=860, margin=2,)
generate_from_frequencies 函数
        word_frequence = {x[0]:x[1]for x in words_stat.head(100).values}
        word_frequence_dict = {}
        for key in word_frequence:
            word_frequence_dict[key] = word_frequence[key]

        wordcloud.generate_from_frequencies(word_frequence_dict)
        ## 从背景图片生成颜色值
        image_colors = ImageColorGenerator(color_mask)
        ## 重新上色
        wordcloud.recolor(color_func=image_colors)
        ## 保存图片
        wordcloud.to_file('output.png')
        plt.imshow(wordcloud)
        plt.axis("off")
        self.root.destroy()
        plt.show()
```

装

订

线