中国地质大学

信管专业就业岗位数据可视化分析报告
——基于 51job 网站的数据

指 导 教 师：朱镇

学 生 姓 名：徐嘉艺

班　　级：086191

学　　号：20191000960

专　　业：信息管理与信息系统

日　　期：2022.1.10

目录

1 实验介绍

1.1 实验目标

本次实验目的在于通过对 51job 招聘网站上数据的分析，探寻信息管理与信息系统专业的就业趋势，分析市场对信管专业的需求。本实验主要对与就业情况有关的以下内容进行分析：

（1）岗位数量分析

（2）岗位学历要求分析

（3）公司规模分析

（4）就业地区分析

（5）学历与薪资相关性分析

（6）职位类型与薪资水平相关性分析

（7）就业待遇分析

（8）公司规模与学历要求相关性分析

1.2 实验内容

本次实验将运用 Python 语言完成以下主要内容：

主要任务	任务内容
数据爬取	运用 Python 中的 requests 第三方库进行网页数据爬取，并存储到 MySQL 数据库中。
数据清洗	观察获得的数据并清楚实际分析需要的数据，运用 Python 语言筛选数据。
数据可视化	Python 相关第三方库（matplotlib）进行基础统计图形的绘制，能够对大数据进行初步分析并实现可视化编程。
可视化分析	了解信管专业相关岗位的地理分布、薪资水平、学历要求等信息，为之后找工作提供一定方向。

2 数据爬取

2.1 确定爬取内容

图 2-1 51job 网站首页

以 51job 招聘网站（https://www.51job.com/）作为数据获取平台，本实验选取了与信管专业相关的 30 个岗位（数据分析，产品经理，产品助理，交互设计，前端开发，软件设计，IOS 开发，业务分析，安卓开发，PHP 开发，业务咨询，需求分析，流程设计，售后经理，售前经理，技术支持，ERP 实施，实施工程师，IT 项目经理，IT 项目助理，信息咨询，数据挖掘，数据运营，网络营销，物流与供应链，渠道管理，电商运营，客户关系管理，新媒体运营，产品运营）。目的是对这些岗位进行相关内容的爬取。具体任务如下：

（1） 观察每个岗位在 51job 官网上关键词搜索内容的页数；

（2）根据页数爬取各个岗位的岗位名称、公司名称、公司规模、工作地点、薪资、工作要求、工作待遇等相关内容。

2.2 爬取工作开展

2.2.1 数据库准备

在数据库中新建表‘jobs’，用于存放爬取的岗位信息，各字段定义如图 2-2 所示：

字段 索引 外键 触发器 选项 注释 SQL 预览

名	类型	长度	小数点	不是 null	虚拟
岗位	varchar	255	0	☐	☐
更新时间	varchar	255	0	☐	☐
公司名称	varchar	255	0	☐	☐
公司类型	varchar	255	0	☐	☐
公司规模	varchar	255	0	☐	☐
工作地点	varchar	255	0	☐	☐
薪资	varchar	255	0	☐	☐
工作要求	varchar	255	0	☐	☐
工作待遇	varchar	255	0	☐	☐
当前爬取岗位	varchar	255	0	☐	☐

图 2-2 在数据库中建立新表 jobs

2.2.2 观察爬取内容

我们发现，例如：当搜索数据分析岗位时，尽管呈现的结果很多，但到了一定页数之后就与“数据分析”无关了，那么这部分内容要排除。所以本代码在运行之前需要使用者自行查看每个岗位的页数，然后自主进行输入。

例如，以关键词“数据分析”进行岗位信息搜索，如图 2-3，我们发现当页数为“34”时，提供的岗位信息名称开始包含非“数据分析”关键词，则我们只爬取前 34 页的数据。

图 2-3 从 34 页开始包含非“数据分析”关键词

2.2.3 爬取代码

```
#引入相关库
import requests
from bs4 import BeautifulSoup
import pymysql
```

```
#定义 spider()函数，用于获取每个 url 的 html
def spider(url):
    headers = { "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36"}
    try:
        rep = requests.get(url, headers=headers)
        rep.raise_for_status()
        rep.encoding = rep.apparent_encoding
        txt = rep.text
        return txt
    except:
        print("解析失败")
#定义 jiexi()函数，用于解析得到的 html
def jiexi(html, info, name):
    soup = BeautifulSoup(html, "lxml")
    text = soup.find_all("script", type="text/javascript")[2].string
#观察原始代码发现我们需要的数据在 engine_jds 后
    data = eval(str(text).split("=", 1)[1])["engine_jds"]
    for d in data:
    try:
        job_name = d["job_name"].replace("\\", "")    #岗位名称
    except:
        job_name = " "
    try:
        company_name = d["company_name"].replace("\\", "")    #公司名称
    except:
        company_name = " "
    try:
        providesalary_text=d["providesalary_text"].replace("\\", "")    #薪资
    except:
        providesalary_text = " "
    try:
        workarea_text = d["workarea_text"].replace("\\", "") #工作地点
    except:
        workarea_text = " "
    try:
        updatedate = d["updatedate"].replace("\\", "") #更新时间
    except:
         updatedate = " "
    try:
        jobwelf = d["jobwelf"].replace("\\", "") #工作待遇
    except:
        jobwelf = " "
    try:
        companyind_text=d["companyind_text"].replace("\\", "")    #公司类型
    except:
        companyind_text = " "
    try:
        companysize_text = d["companysize_text"].replace("\\", "") #公司规模
```

```
        except:
            companysize_text = " "
        try:
            at = d["attribute_text"]   #工作要求
            s = ''
            for i in range(0, len(at)):
                s = s + at[i] + ','
            attribute_text = s[:-1]
        except:
            attribute_text = " "
#将每一条岗位数据爬取的内容以及传入参数 name 作为一个列表，依次加入到 info 列表
        info.append( [name, job_name, updatedate, company_name, companyind_text, companysize_text,
workarea_text, providesalary_text, attribute_text, jobwelf])

#将数据存到 MySQL 中名为“51job”的数据库中
def save(data):
        db = pymysql.connect(           #连接数据库
host="127.0.0.1",           #MySQL 服务器名
user="root",                 #用户名
password="123456",          #密码
database="python 上机", #操作的数据库名称
charset="utf8"
)
        cursor = db.cursor()
#将数据保存到数据库表中对应的列
        for each_data in data:
            present_job = each_data[0]   #当前爬取岗位
            job_name = each_data[1]   #岗位
            updatedate = each_data[2]   #更新时间
            company_name = each_data[3]   #公司名称
            companyind_text = each_data[4]   #公司类型
            companysize_text = each_data[5]   #公司规模
            workarea_text = each_data[6]   #工作地点
            providesalary_text = each_data[7]   #薪资
            attribute_text = each_data[8]   #工作要求
            jobwelf = each_data[9]   #工作待遇
#创建 sql 语句
        sql = "insert into jobs(当前爬取岗位，岗位，更新时间,公司名称,公司类型,公司规 模,工作地
点,薪资,工作要求,工作待遇) values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

#执行 sql 语句
        cursor.execute(sql, [present_job, job_name, updatedate, company_name, companyind_text,
companysize_text, workarea_text, providesalary_text, attribute_text, jobwelf])
        db.commit()   #提交数据库
        cursor.close()   #关闭游标
        db.close()   #关闭数据库
if __name__ == '__main__':   #主函数
        name = input("请输入岗位名称")
#自行输入 30 个岗位名称搜索
```

```
    page = eval(input("请输入爬取页数"))
#在观察后输入数据页面爬取数据
    info = []
#空列表，传入 jiexi 函数用于存储每一条岗位的数据
    for i in range(1, page+1):
        url = "https://search.51job.com/list/000000,000000,0000,00,9,99," + name + ",2," + str(i) +
".html?"
        html = spider(url)
        jiexi(html, info)
save(info)
```

2.2.4 爬取结果

爬取结果如图 2-4 所示。

岗位	更新时间	公司名称	公司类型	公司规模	工作地点	薪资	工作要求	工作待遇	当前
数据分析	44485	“前程无忧”5	互联网/电子	1000-5000,	上海-浦东新	(Null)	上海-浦东新	五险一金 补	数据
数据分析	44485	信宜智研瞻亿	专业服务(咨	少于50人	茂名	3-4万/月	茂名,3-4年经	(Null)	数据
数据分析	44485	深圳尊宝餐饮	餐饮业	1000-5000,	深圳-龙岗区	0.6-1万	深圳-龙岗区,	包住宿 全勤	数据
数据分析	44485	诺德睿行（广	快速消费品(	150-500人	广州-天河区	1-1.5万	广州-天河区,	五险一金 定	数据
数据分析	44485	字节跳动	互联网/电子	10000人以上	北京	2-5万/月	北京,3-4年经	六险一金 弹	数据
财经数据	44485	印孚瑟斯技术	计算机服务(i	1000-5000,	大连-高新园	3.5-4.5	大连-高新园	五险一金 补	数据
数据分析	44485	广东米客科技	互联网/电子	500-1000人	广州-番禺区	4.5-6千	广州-番禺区,	带薪年假 五	数据
高级数据	44485	上海数鸣人工	计算机软件	50-150人	上海-静安区	1.2-1.8	上海-静安区,	五险一金 周	数据
数据分析	44485	杭州玖欣物联	互联网/电子	50-150人	杭州-滨江区	2-2.5万	杭州-滨江区,	五险一金 弹	数据
数据分析	44485	济南德弘企业	专业服务(咨	50-150人	济南	0.5-1万	济南,无需经验	法定节假日	数据
数据分析	44485	裕力富食品	快速消费品(	150-500人	上海-闵行区	0.8-1.3	上海-闵行区,	周末双休 五	数据
数据分析	44485	深圳市蓝诺科	计算机软件	(Null)	深圳-福田区	1-1.5万	深圳-福田区,	交通补贴 餐	数据
发展部经	44485	上海美浮特生	制药/生物工	150-500人	上海-闵行区	1-1.5万	上海-闵行区,	五险一金 专	数据
数据分析	44485	广东合顺投资	信托/担保/抵	50-150人	广州-越秀区	0.8-1万	广州-越秀区,	(Null)	数据
数据分析	44485	深圳市凯信时	服装/纺织/皮	1000-5000,	深圳	6-9千/月	深圳,3-4年经	带薪年假 五	数据
数据分析	44485	上海项猎企业	专业服务(咨	50-150人	上海-浦东新	1.2-2.4	上海-浦东新	五险一金 专	数据
主数据分	44485	上海汉得信息	计算机软件	10000人以上	成都	1-1.8万	成都,2年经验	五险一金 补	数据
数据分析	44485	上海亚朵商业	多元化业务集	10000人以上	上海-闵行区	1.8-3万	上海-闵行区,	(Null)	数据
数据分析	44485	上海希源道具	家居/室内设	150-500人	上海-奉贤区	0.6-1万	上海-奉贤区,	做五休二 带	数据
Materia	44485	北京外企德科	专业服务(咨	(Null)	中山	0.8-1.2	中山,1年经验	五险一金 周	数据
数据分析	44485	安客诚信息服	计算机服务(i	150-500人	南通	0.8-1万	南通,1年经验	周末双休 带	数据
数据分析	44485	盛辉物流集团	交通/运输/物	5000-10000	福州	4-5.5千	福州,1年经验	(Null)	数据
运力数据	44485	英赋嘉(浙江)	交通/运输/物	150-500人	杭州	6-8千/月	杭州,1年经验	带薪年假 节	数据

图 2-4 爬取完成后的数据库结果

3 数据清洗

为了使数据结构更规范，便于我们处理，我们需要对爬取过的数据进行清洗。首先需要在数据库中新建表 after_clean 用于存放清洗后的数据，新建表的各个信息与图 2-2 相同。

数据清洗主要工作包括：

（1）匹配各个岗位；

（2）清洗工作地点；

（3）平均化各个岗位的薪资并统一量纲；

（4）精简工作要求。

3.1 匹配工作岗位

3.1.1 规则

在此我们将对我们爬取的 30 个岗位进行匹配，例如：在以关键词“电商运营”或“新媒体运营”搜索的岗位信息中包含大量具体电商或 新媒体平台名称的岗位名称，如“拼多多运营”“抖音运营”等，因此在这两类 岗位名称匹配时我们认为只要岗位名称中包含“运营”就算匹配成功。具体对每个岗位的处理方式将在代码中详细说明。

3.1.2 代码

```
# 连接 MySQL 数据库
db = pymysql.connect(
    host="127.0.0.1",
    user="root",
    password="123456",
    database="xjy",
    charset="utf8")

# 匹配工作岗位
def pipei():
    cursor = db.cursor()  # 获取操作游标
    cursor.execute("select * from jobs")  # 从 jobs 表中查询所有内容并保存
    results = cursor.fetchall()  # 接受全部的返回结果
    after_pipei = []  # 建立一个空列表，用来存储匹配后数据
    for each_result in results:
        if each_result[-1] == '物流与供应链':
            if '物流' in each_result[0] or '供应链' in each_result[0]:
                after_pipei.append(each_result)
        elif each_result[-1] == '新媒体运营' or each_result[-1] == '电商运营':
            if '运营' in each_result[0]:
                after_pipei.append(each_result)  # 由于在以关键词“电商运营”或“新媒体运营”搜索的岗位信息中包
                # 含大量具体电商或 新媒体平台名称的岗位名称，如“拼多多运营”
                # “抖音运营”等，因此在这两类 岗位名称匹配时我们认为只要岗位名
                # 称中包含“运营”就算匹配成功。
        elif each_result[-1] == '客户关系管理':
            if '客户关系' in each_result[0]:
                after_pipei.append(each_result)
        elif each_result[-1] == '安卓开发':
            if '安卓' in each_result[0] or 'Android' in each_result[0]:
                after_pipei.append(each_result)  # 由于在 很多公司的招聘岗位中“安卓”会以“Android”英文形式出现，
                # 因此，在以“安 卓开发”为关键词进行搜索时，我们认为只要包含“安卓”
                # 或“Android”开发 就算匹配成功。
        elif each_result[-1][:-2] in each_result[0] and each_result[-1][-2:] in each_result[0]:
            after_pipei.append(each_result)  # 剩余岗位需 要两个关键词都存在岗位名称中，例如包含“数据”或“分析”
```

```
            # 在以“数据分析”为关键词搜索的岗位名称种，我们就认为匹配成功。
    cursor.close()  # 关闭游标
    return after_pipei  # 返回匹配后的列表
```

3.2 工作地点筛选

3.2.1 规则

由于爬取到的工作地点信息格式不统一，有的地点只给出一级地名，有的给出了两级，我们将岗位对应的工作地点统一保留到给出的最高级别地名，例如“上海—普陀区”只保留“上海”。

3.2.2 代码

```
# 工作地点筛选
def split_city(data):
    after_split_city = []  # 建立一个空列表，用来存储匹配后数据
    for each_date in data:
        each_date_list = list(each_date)
        each_date_list[5] = each_date_list[5].split('-')[0]  # 将数据表中 工作地点列以'-'进行切割，选取第一个元素替换
        after_split_city.append(each_date_list)
    return after_split_city  # 返回筛除后的数据
```

3.3 清洗薪资内容

3.3.1 规则

由于爬取到的薪资信息大都是以区间数据形式出现，例如“5000-6000/月”，并且薪资单位不一致有“千/月”、“万/年”等形式，因此对薪资数据的清洗主要包括两个方面：

（1）对区间数值取组中值。如“5000-6000/月”处理为“5500/月”；

（2）统一量纲。将薪资的单位统一为“千/月”，如“5500/月”处理为“5.5 千/月”，“24 万/年”处理为“20 千/月”。

3.3.2 代码

```
# 清洗薪资内容
def salary(data):
    after_salary = []  # 建立一个空列表，用来存储匹配后数据

    for each_data in data:
        print(each_data)
        if each_data[6] != None and each_data[6][-1] != '时' and each_data[6][-3] != '下' and each_data[6][-3] != '上' and each_data[6]!='1.5 千/月' and each_data[6]!='2 万/年':
            # 筛除缺失值，以小时 计费，给出的薪资表达为在“……以下”及“……以上”等难以计算数据的工作岗 位#统一量纲（单位:千/月）
                if each_data[6][-1] == '年':
                    each_data[6] = str((float(each_data[6].split(' 万')[0].split('-')[0]) + float(
```

```
                    each_data[6].split('万')[0].split('-')[1])) * 5 / 12) + '千/月'
            elif each_data[6][-1] == '天':
                each_data[6] = str(float(each_data[6].split('元')[0]) * 30 / 1000) + '千/月'
            elif each_data[6][-3] == '万':
                each_data[6] = str((float(each_data[6].split('万')[0].split('-')[0]) +
float(each_data[6].split('万')[0].split('-')[1])) * 5) + '千/月'
            else:
                each_data[6] = str((float(each_data[6].split('千')[0].split('-')[0]) +
float(each_data[6].split('千')[0].split('-')[1])) / 2) + '千/月'
            after_salary.append(each_data)
    return after_salary   # 返回平均工资后的数据
```

3.4 清洗工作要求

3.4.1 规则

由于爬取到的“工作要求”内容中一般包含“地点，经验，学历，招收人数”四项内容，在本次实验中只用到“经验”和“学历”两项内容，因此此步骤需要清洗掉其他两项内容。

3.4.2 代码

```
# 清洗“工作要求”
def job_attribute_text(data):
    for each_data in data:
        if len(each_data[7].split(',')) == 3:
            if '经验' in each_data[7].split(',')[1] or '在校生' in each_data[7].split(',')[1]:
                each_data[7] = each_data[7].split(',')[1] + ','
            else:
                each_data[7] = ',' + each_data[7].split(',')[1]
        elif len(each_data[7].split(',')) == 4:
            each_data[7] = each_data[7].split(',')[1] + ',' + each_data[7].split(',')[2]
        else:
            each_data[7] = ''
    return data
```

3.5 保存及主函数

3.5.1 内容保存

```
def save(data):
    cursor = db.cursor()
    for each_data in data:
        job_name = each_data[0]
        updatedate = each_data[1]
        company_name = each_data[2]
        companyind_text = each_data[3]
        companysize_text = each_data[4]
        workarea_text = each_data[5]
```

```
            providesalary_text = each_data[6]
            attribute_text = each_data[7]
            jobwelf = each_data[8]
            present_job = each_data[9]
            sql = "insert into jobs(当前爬取岗位,岗位,更新时间,公司名称,公司类型,公司规模,工作地点,薪资,工作要求,工作待遇) values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
            cursor.execute(sql, [present_job, job_name, updatedate, company_name, companyind_text, companysize_text,workarea_text, providesalary_text, attribute_text, jobwelf])
            db.commit()
        cursor.close()
        db.close()
```

3.5.2 主函数

```
# 定义主函数，用以执行以上四步数据清洗过程
if __name__ == '__main__':
    data = pipei()
    data1 = split_city(data)
    print("分开城市完成！")
    data2 = salary(data1)
    data3 = job_attribute_text(data2)
    save(data3)
```

3.6 清洗结果

岗位	更新时间	公司名称	公司类型	公司规模	工作地点	薪资	工作要求
阿里巴巴运营	08-23	广州赛美优品	快速消费品(	500-1000人	广州	7.5千/月	3-4年经验，
亚马逊运营助理	08-23	深圳市彩悦禾	互联网/电子	少于50人	深圳	7.5千/月	无需经验，大
天猫运营	08-23	广州市三横信	互联网/电子	150-500人	广州	9.0千/月	2年经验，大
亚马逊资深运营	08-23	上海千成实业	互联网/电子	150-500人	上海	12.5千/月	2年经验，本
淘宝天猫运营	08-23	广州名盟家具	家具/家电/玩	少于50人	广州	7.0千/月	3-4年经验，
拼多多运营	08-23	湖南承长文化	互联网/电子	50-150人	长沙	7.0千/月	1年经验，大
拼多多运营	08-23	上海毓悦网络	计算机软件	(Null)	上海	8.0千/月	1年经验，中
生活服务平台运	08-23	湖南友靠智能	计算机软件	少于50人	长沙	6.5千/月	1年经验，本
亚马逊欧洲站运	08-23	深圳市保凌影	电子技术/半	50-150人	深圳	12.5千/月	1年经验，大
日本产品运营经	08-23	四川省中国旺	酒店/旅游	50-150人	成都	5.25千/月	1年经验，
网站运营策划	08-23	深圳市国方润	快速消费品(	150-150人	深圳	6.5千/月	2年经验，中
天猫店长/天猫	08-23	浙江风橡飚贸	贸易/进出口	50-150人	杭州	9.0千/月	3-4年经验，
市场运营策划	08-23	上海云漉科技	计算机软件	50-150人	上海	12.5千/月	1年经验，本
亚马逊运营专员	08-23	深圳美泽保禾	互联网/电子	50-150人	深圳	6.0千/月	1年经验，大
天猫运营专员	08-23	深圳小米电子	互联网/电子	50-150人	深圳	7.0千/月	1年经验，大
平台运营专员	08-23	合盛硅业股份	石油/化工/矿	10000人以上	上海	8.5千/月	1年经验，本
第三方平台运营	08-23	广东丰尚商业	批发/零售	150-500人	佛山	8.0千/月	2年经验，大

图 2-5 清洗结果

4 数据可视化分析

4.1 岗位数量分析（柱状图）

4.1.1 分析思路

为了便于我们绘图及展示，在此首先将爬取到的 30 个岗位进行归类。我们将其分为九个大类：技术管理类、运维类、技术开发类、业务咨询类、技术支持类、数据运营类、市场职能类、产品运营类、数据管理类。具体细分操作会在代码中展示。

4.1.2 代码

```
#储存 30 种岗位
jobs = ['产品经理', '产品助理', '交互设计', '前端开发', '软件设计', 'IOS 开发', '业务分析', '安卓开发', 'PHP 开发', '业务咨询', '需求分析', '流程设计', '售后经理', '售前经理', '技术支持', 'ERP 实施', '实施工程师', 'IT 项目经理', 'IT 项目助理', '信息咨询', '数据挖掘', '数据运营', '数据分析', '网络营销', '物流与供应链', '渠道管理', '电商运营', '客户关系管理', '新媒体运营', '产品运营']
#创建一个空列表，用于存储每种岗位的数量值
count = []

#先在空列表中创建 30 个 0 元素
for i in range(len(jobs)):
    count.append(0)
for each_result in df['当前爬取岗位']:
    for i in range(0, 30):
        if each_result == jobs[i]:
            count[i] += 1
            continue
jobs_classification = ['技术管理类', 'IT 运维类', '技术开发类', '业务咨询 类', '技术支持类', '数据运营类', '市场职能类', '产品运营类', '数据管理类']
counts = [] #创建一个空列表，用于存储每小类岗位的数量值
for i in range(len(jobs_classification)):
    counts.append(0) #先在空列表中创建 9 个 0 元素

#根据大纲中给出的分类表，依据 30 种岗位数量值，分别计算出九小类岗 位的数量值
counts[0] = count[0] + count[1] + count[2]
counts[1] = count[17] + count[18]
counts[2] = count[3] + count[4] + count[5] + count[6] + count[7] + count[8]
counts[3] = count[9] + count[10] + count[11]
counts[4] = count[12] + count[13] + count[14] + count[15] + count[16]
counts[5] = count[21] + count[22]
counts[6] = count[23] + count[24] + count[25]
counts[7] = count[26] + count[27] + count[28] + count[29]
counts[8] = count[19] + count[20]

#绘制图片
```

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False #用来正常显示负号
fig, ax = plt.subplots(figsize=(10, 7))#定义图片的大小
ax.bar(x=jobs_classification, height=counts, color='r')
ax.set_title("岗位数量柱状图", fontsize=15)
for x, y in enumerate(counts):
    plt.text(x, y+5, '%s' %y, ha='center') #为每根柱子加上数值
plt.show() #展示图片
```

4.1.3 可视化结果

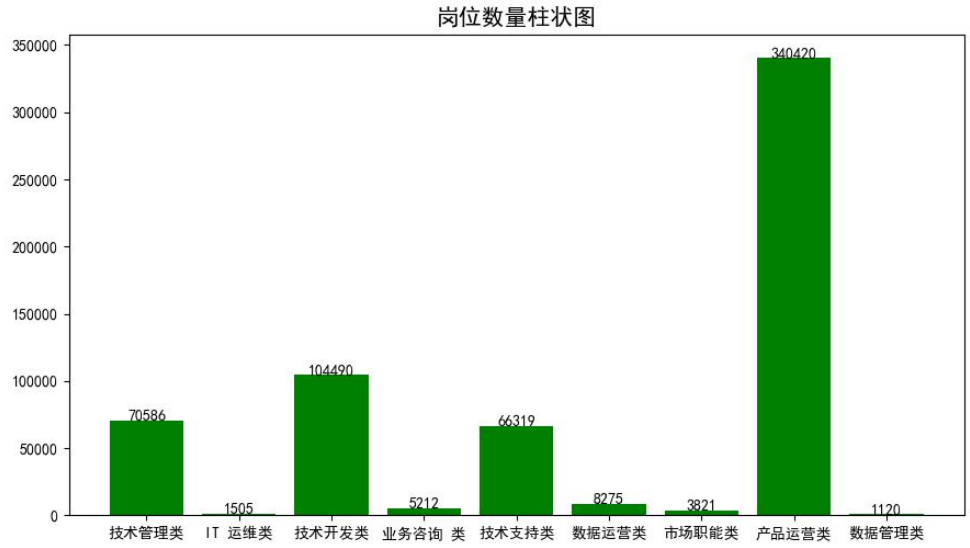

图 4-1 岗位数量柱状图

4.1.4 分析结论

从图中可以看出，九小类岗位的市场需求数量存在很大差异，得出的主要结论如下：

（1）产品运营类的岗位数量最多。这可能是因为这一类包括电商运营与新媒体运营这两大近年来发展迅猛的市场需求，这体现了信管专业就业的市场导向，符合互联网行业变化快的特征。

（2）可以看出信管专业市场需求的第二梯队主要集中在技术开发、技术管理和技术支持类。这三类的特点是对专业技术的要求很高，需要依靠所学的技术能力去解决企业的实际问题。

（3）市场对于信管专业在数据管理、IT 运维、市场职能等方面的需求较少。 岗位数量最少的是数据管理类，这可能是因为企业对数据挖掘和信息咨询岗位的定位在于少而精，导致需求量不多，但是要求高。

综上，可以发现信管专业就业主要有两个方向，一个是以近期较热门的电商运营与新媒体运营为代表的产品运营类岗位，另一个是以技术为主的技术开发、技术管理和技术支持类。因此，

同学们除了需要及时关注行业发展动态，捕捉市场需求的变化外，还需要不断提升自身的技术水平，以期在未来就业市场的激烈竞争中获取更大的竞争优势。

4.2 岗位学历要求分析（饼状图）

4.2.1 分析思路

在岗位学历要求分析中，我们主要使用“工作要求”这一表中的数据，通过索引切片提取出每个岗位的学历要求信息，然后使用字典进行聚类分组，从而统计出各学历的数值，进行饼图的绘制。在此，我们不考虑某些不要求学历的岗位。

4.2.2 代码

```
xl = {} #构建学历信息字典
for each_result in df['工作要求']:
    print(each_result)
    if str(each_result) != '无' and str(each_result).split('，')[1] !='':
        a = str(each_result).split('，')[1]
        if a in xl:
            xl[a] += 1
        else:
            xl[a] = 1
    if str(each_result) == '无':
        continue

count_xl = [] #创建学历计数列表
label_xl = [] #创建学历种类列表
for key in xl: #从字典取出对应值
    count_xl.append(xl[key])
    label_xl.append(key)

#绘制图片
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False #用来正常显示负号
plt.pie(x=count_xl,labels=label_xl,autopct='%1.1f%%',pctdistance=0.8,
        explode=(0.1,0.2,0.3,0.4,0.1,0.2,0.1,0))

#x 为基本数据，labels 为给各个部分添加标签的列表，autopct 显示各部分比例， 本例中调用%1.2f%%。
plt.title('学历要求饼状图') #为饼图添加标题
plt.show() #展示图片
```

4.2.3 可视化结果

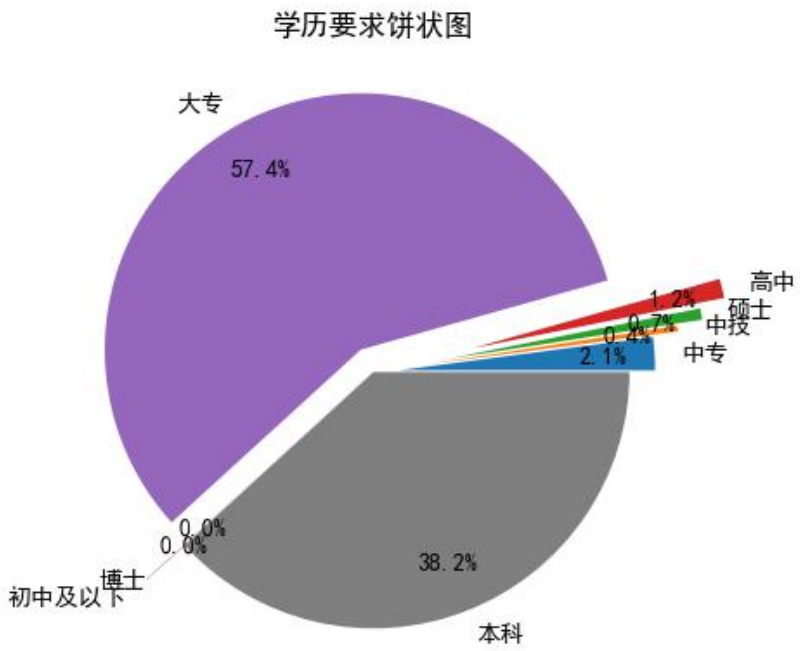

图 4-2 学历要求饼状图

4.2.4 分析结论

从图中可以看出，市场对信管专业同学的学历还是有一定要求的。

（1）学历要求主要是大专生和本科生，这两类学历占比超过了 95%。即对于找工作来说，本科学历是一个明显的门槛，拥有本科学历后可以选择就业市场上接近 99%的工作，就业前景十分良好。

（2）大专以下学历的岗位数量只占 3%左右，这体现了市场对信管专业同学的要求是起码需要经过专业的训练，是一个有门槛的行业。这也符合信管专业既包括专业技术学习，也包括管理理论学习的专业特色。

（3）硕士与博士的行业需求只占 1%左右。这首先与硕博士的市场供给有关，即就业市场上求职的硕博士并不多，导致需求也不多。其次，这也说明就业市场上只需要硕博士的高要求岗位并不多，绝大部分岗位对学历要求没这么高。

综上，对于就业来说，信管专业同学只要达到本科学历就可以胜任绝大部分工作，继续深造对就业选择数量的增加不多。

4.3 公司规模分析（条形图）

4.3.1 分析思路

对于公司规模的分析，我们采用条形图展示。遍历列表，对不同规模的公司聚类分组，统计数量。

4.3.2 代码

```
# 生成公司规模的用于计数的字典
gm={}
for e in df['公司规模']:
    print(e)
    if str(e) != '':
        if str(e) in gm:
            gm[str(e)] += 1
        else:
            gm[e] = 1
    if str(e) == '':
        continue

# 生成列表
count_gm = []
label_gm = []
for key in gm:
    count_gm.append(gm[key])
    label_gm.append(key)
print(count_gm)
print(label_gm)

# 绘图。
fig, ax = plt.subplots()
b = ax.barh(range(len(label_gm)), count_gm, color='#7999CC')

# 为横向水平的柱图右侧添加数据标签。
for rect in b:
    w = rect.get_width()
    ax.text(w, rect.get_y() + rect.get_height() / 2, '%d' %
            int(w), ha='left', va='center')

# 设置 Y 轴纵坐标上的刻度线标签。
ax.set_yticks(range(len(label_gm)))
ax.set_yticklabels(label_gm)

# 不要 X 横坐标上的 label 标签。
plt.xticks(())
plt.xlabel('数量')   #x 轴名称
plt.ylabel('规模')   #y 轴名称
```

```
plt.title('公司规模', loc='center', fontsize='15',
          fontweight='bold', color='black')

plt.show()
```

4.3.3 可视化结果

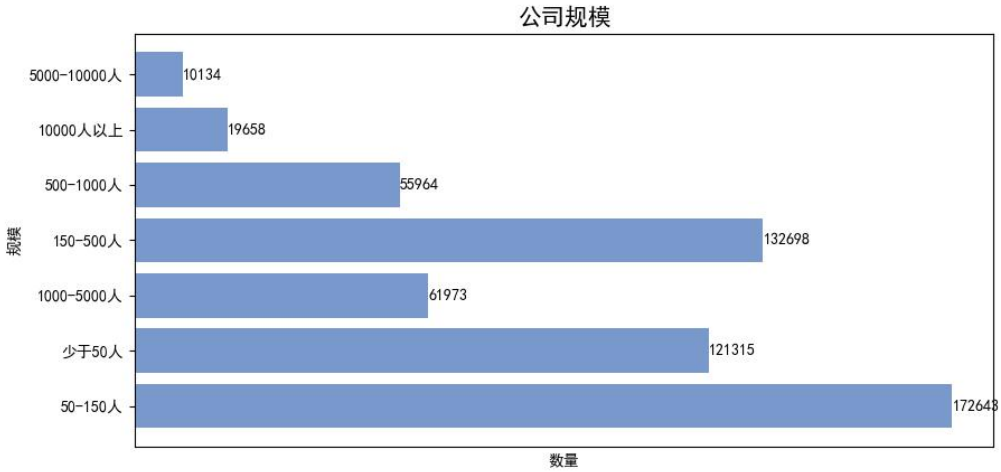

图 4-3 公司规模条形图

4.3.4 分析结论

从整体上可以看出：

（1）信管专业的就业需求主要集中在公司规模小于 500 人的中小微企业。

（2）公司规模在 50 人以下的企业提供的就业需求是第三多的。这说明大量的初创微型企业对信管专业同学有着需求，富有冒险精神与创业热情的同学在就业选择时可以着重考虑这类企业，日后可能就成为某个独角兽企业。

（3）公司规模在 5000 人以上的大厂对信管专业同学的需求只占总需求的 6%左右。这说明信管专业同学毕业后想进入大厂的难度很高，竞争十分激烈。

总的来说，中小微型企业占据了大多数信管专业人才的需求。

4.4 就业地区分析（热力图）

4.4.1 分析思路

对于就业地区的分析，我们采用热力图的形式展现。热力图是通过密度函数进行可视化用于表示地图中点的密度的热图，以特殊高亮的形式显示热度较强的地理区域。我们首先提取每个岗

位的工作地点，然后将其按照省份/直辖市进行分组，最后生成按照省份/直辖市分类的热力图。（之所以在这里进行按省分组的原因是如果按照城市直接进行绘图的话，绘制出来的热力图会呈现一大片红色，不太直观。

4.4.2 代码

```
# 生成城市的用于计数的字典
cs = {}
for city in df['工作地点']:
    print(city)
    if str(city) in cs:
        cs[city] += 1
    else:
        cs[city] = 1
# 生成列表
count_cs = []
label_cs = []
for key in cs:
    count_cs.append(cs[key])

    label_cs.append(key)
print(count_cs)
print(label_cs)
aa = [list(z) for z in zip(label_cs, count_cs)]

geo = ( Geo().add_schema(maptype="china")
        .add( "岗位-城市数量分布热力图(信息管理与信息系统专业)", # 图题
              aa,
               type_=ChartType.HEATMAP, # 地图类型
            )
        .set_series_opts(label_opts=opts.LabelOpts(is_show=False)) # 设置 是否显示标签
        .set_global_opts( visualmap_opts=opts.VisualMapOpts(max_=260), # 设置 legend 显示的最大值
            )
        )

geo.render(r"D:\XJY\大三上\专业课程\大数据实训-卢辉\python 数据可视化实验\citiesheatmap.html") #以 html 类型保存，名称为 cities_heatmap
```

4.4.3 可视化结果

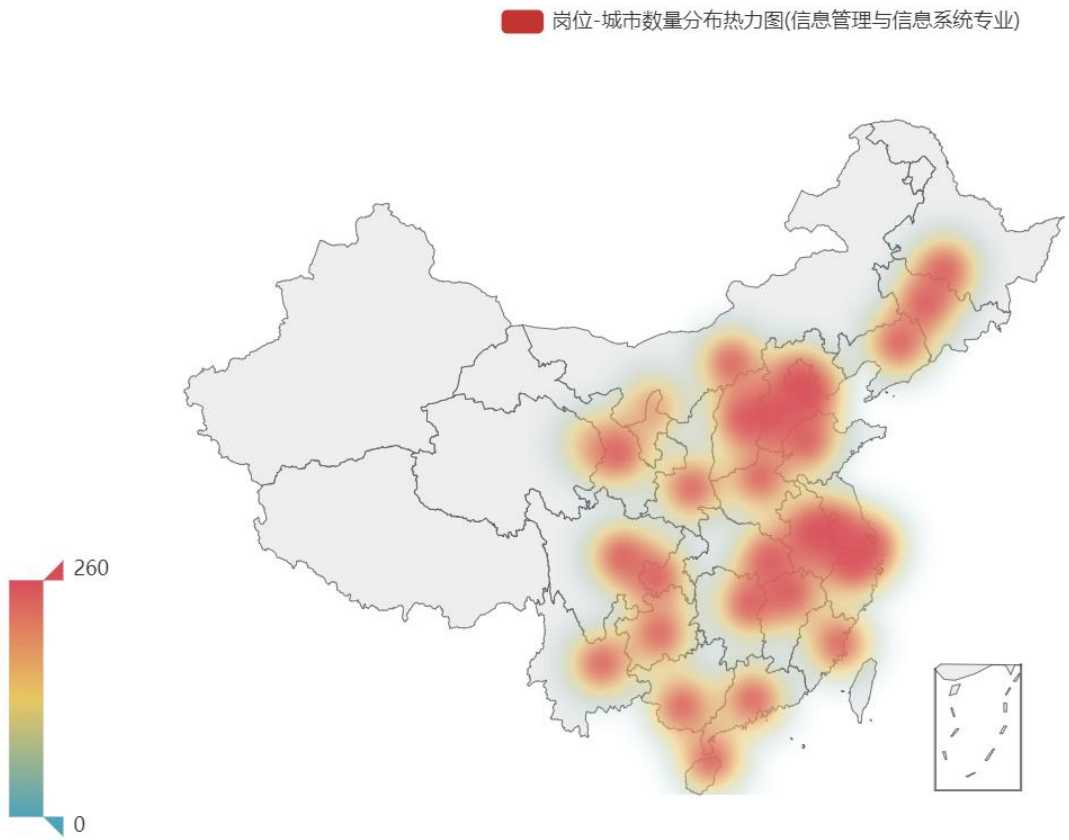

图 4-4 岗位-城市数量分布热力图

4.4.4 分析结论

从图中可以看出，信管专业的市场需求主要分布在东南沿海以及除西北地区外的各个省会城市。 其主要特征如下：

（1）以上海为中心的长三角地区的岗位数量明显多于其他各城市，这说明信管专业的岗位需求与经济发展水平有着明显的正向关系，同学们想要获得更好的职业发展机会应该优先考虑一线沿海城市。

（2）武汉、成都、西安、长沙等内陆地区的的岗位数量虽不及一线沿海城市，但对信管专业的同学也有一定的需求，若来自这些地方的同学想回家乡工作，各自的省会城市也能提供不错的发展机会。

（3）西北地区提供的信管专业岗位数量很少，这说明信管专业的就业还是有一定的地区聚集效应。

4.5 学历与薪资的相关性分析（箱线图）

4.5.1 分析思路

对于学历与薪资的相关性分析，我们采用箱线图。箱线图是一种利用数据中的五个统计量：上边缘、下边缘、中位数和两个四分位数来描述数据的一种方法，它可以粗略地看出数据是否具有对称性，分布的分散程度等信息，特别可以用于对几个样本的比较。在这里我们首先抽取出学历的类别，然后将类别作为字典的键，之后统计每个学历类别的所有薪资（进行单位转换），从而生成箱线图。

4.5.2 代码

```
li = df[['薪资','工作要求']] #取出薪资和工作要求两列
list_xg = []
for indexs in li.index:
    list_xg.append(li.loc[indexs].values[0:])
xueli = []
salary = []

#进行数据处理
for each in list_xg:
    print(each)
    if each[1] != '无' and each[1].split('，')[1] != '' and float(each[0].split('千/月')[0]) < 40: #筛除学历的缺失值以及部分异常值
        xueli.append(each[1].split('，')[1]) #晒除薪资的缺失值
        salary.append(each[0].split('千/月')[0]) # 对学历去重复值，并转换为列表形式，得到学历类型列表 xueli_after_quchong
        xueli_after_quchong = list(set(xueli))
        xueli_after_quchong = ['在校生/应届生' if each == '在校生\\/应届生' else each for each in xueli_after_quchong]

#创建一个包含学历类型数个空列表的列表
final = []
for each in range(len(xueli_after_quchong)):
    final.append([]) #将不同类型学历的所有薪资先由字符串类型转换为浮点数类型，通过 index 索引，放进 final 列表中对应的空列表，得到每种学历对应的薪资列表 final
for each in xueli_after_quchong:
    for i in range(len(xueli)):
        if xueli[i] == each:
            final[xueli_after_quchong.index(each)].append(float(salary[i])) #画图
#画图
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号
plt.boxplot(final,labels=xueli_after_quchong)
plt.title('学历-薪资水平箱线图', fontsize=15)
```

```
plt.ylabel('薪资(单位：千/月)', fontsize=12)
plt.show() #展示图形
```

4.5.3 可视化结果

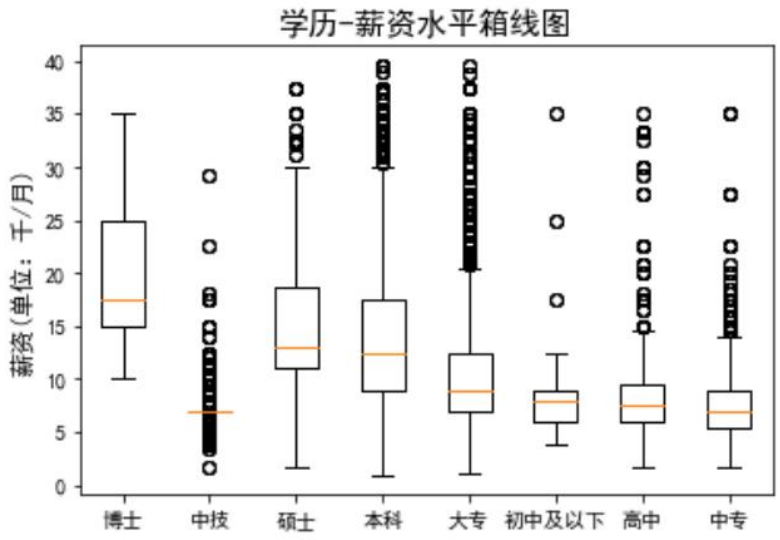

图 4-5 学历-薪资水平箱线图

4.5.4 分析结论

从图中可以看出，市场对于不同学历的求职者给与的薪资水平存在的差异如下：

（1）随着学历水平的提高，薪资水平也有着相应的提高，从中位数来看，博士生的薪资是要明显高于其他学历求职者的。

（2）对于学历要求为本科和硕士的岗位，其薪资水平十分接近，这说明可能对于信管专业学生来说，本科学历已经足以胜任大部分工作。

（3）学历要求为大专的岗位异常值最多，这说明该学历的薪资分歧严重，这可能是以为某些小企业对于数据分析人才的急缺，导致它们在降低学历要求的同时给出了更高的薪资待遇。

（4）学历要求为高中、中技、中专的岗位数较少，因此其各数据几乎重叠且数值较低，不具代表性。

总的来说，除去一些极端情况，学历和薪资还是存在一定正相关关系的。

4.6 职位类型与薪资水平相关性分析（桑基图）

4.6.1 分析思路

对于职位类型和薪资水平的相关性分析，我们采用桑基图。它是一种特定类型的流 程图，图中延伸的分支的宽度对应数据流量的大小，所有主支宽度的总和应与所有分出去的分支宽度的总和相等，保持能量的平衡。在本次实验中，我们还是先对各种岗位进行分组，然后统计每个类别中不同薪资的岗位数量，从而生成桑基图。桑基图可以让我们更直观地发现不同岗位的薪资水平的差异和集中的薪资区间。

4.6.2 代码

```
from pyecharts.charts import Sankey
#取出所需数据列
dx = df[['当前爬取岗位','薪资']]
list_dx = []
for indexs in dx.index:
    list_dx.append(dx.loc[indexs].values[0:])
#根据实验课程大纲中的表格分类信息，将 30 种岗位分为九小类
post = ['技术管理类', '技术开发类', '业务咨询类', '技术支持类', 'IT 运维类', '数据管理类', '数据运营类', '市场职能类', '产品运营类']
T1 = ['产品经理', '产品助理', '交互设计']
T2 = ['前端开发', '软件设计', 'IOS 开发', '业务分析', '安卓开发', 'PHP 开发']
C1 = ['业务咨询', '需求分析', '流程设计']
C2 = ['售后经理', '售前经理', '技术支持', 'ERP 实施', '实施工程师']
C3 = ['IT 项目经理', 'IT 项目助理']
D1 = ['信息咨询', '数据挖掘']
D2 = ['数据运营', '数据分析']
P1 = ['网络营销', '物流与供应链', '渠道管理']
P2 = ['电商运营', '客户关系管理', '新媒体运营', '产品运营']
clasify = [T1,T2,C1,C2,C3,D1,D2,P1,P2]
post1 = []
for each_post in post:
    for i in range(7):
        post1.append(each_post) #将九小类岗位重复 7 次，放进 post1 列表中，用于后续计算数值和画图
#薪资水平划分
salary = ['5 千/月以下','5-10 千/月','10-15 千/月','15-20 千/月','20-25 千/月 ','25-30 千/月','30 千/月以上']
#为了能对应九小类职务，将每种水平薪资取 7 次
salary1 = salary*9 #构建一个包含 7*9 个元素的列表，初始值为 0，用于存储职务与薪资水平 的一对一岗位数量
count = []
for i in range(7*9):
    count.append(0) #计算 count 列表种每个元素的值，即职务与薪资水平的一对一岗位数量
```

```
for each_result in list_dx:
    for i in range (9):
        if each_result[0] in clasify[i] and each_result[1]!='':
            if float(each_result[1].split('千/月')[0]) <5:
                count[i*7] += 1
            elif 5 <= float(each_result[1].split('千/月')[0]) < 10:
                count[i*7+1] +=1
            elif 10 <= float(each_result[1].split('千/月')[0]) < 15:
                count[i*7+2] +=1
            elif 15 <= float(each_result[1].split('千/月')[0]) < 20:
                count[i*7+3] +=1
            elif 20 <= float(each_result[1].split('千/月')[0]) < 25:
                count[i*7+4] +=1
            elif 25 <= float(each_result[1].split('千/月')[0]) < 30:
                count[i*7+5] +=1
            elif 30 <= float(each_result[1].split('千/月')[0]):
                count[i*7+6] +=1

# 整理数据
df_3 = pd.DataFrame({'职位': post1, '薪资': salary1, '数量': count})
# 把所有涉及到的节点去重规整到一起，即把“职位”列的'数据分析','产品经理 ', '产品助理', '交互设计', '前端开发', '软件设计', 'IOS 开发'和“薪资”列中的'5 千/ 月以下','5-10 千/月','10-15 千/月','15-20 千/月','20-25 千/月','25-30 千/月','30 千/月以 上'以列表内嵌套字典的形式去重汇总
nodes = []
for i in range(2):
    values = df_3.iloc[:, i].unique()
    for value in values:
        dic = {}
        dic['name'] = value
        nodes.append(dic)

    # 定义边和流量，用 Source-target-value 字典格式，能清晰描述数据的流转情况
linkes = []
for i in df_3.values:
    dic = {}
    dic['source'] = i[0]
    dic['target'] = i[1]
    dic['value'] = i[2]
    linkes.append(dic)
c = (
    Sankey()
    .add(
        "sankey",
        nodes,
        links,
        linestyle_opt=opts.LineStyleOpts(opacity=0.2, curve=0.5, color="source"),
        label_opts=opts.LabelOpts(position="right"),
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="职位-薪资桑基图"))
```

.render(r"D:\XJY\大三上\专业课程\大数据实训-卢辉\python 数据可视化实验\sankey2_base.html")
)

4.6.3 可视化结果

如果出现打开桑基图的 html 文件无法显示的情况，可以去 echarts 官网下载桑基图的对应文件 echarts.min.js，官网地址：https://echarts.apache.org/zh/builder.html。然后更改 html 源码中的 src 地址标签。具体操作可以参考资料：https://blog.csdn.net/m0_56595556/article/details/115309091。

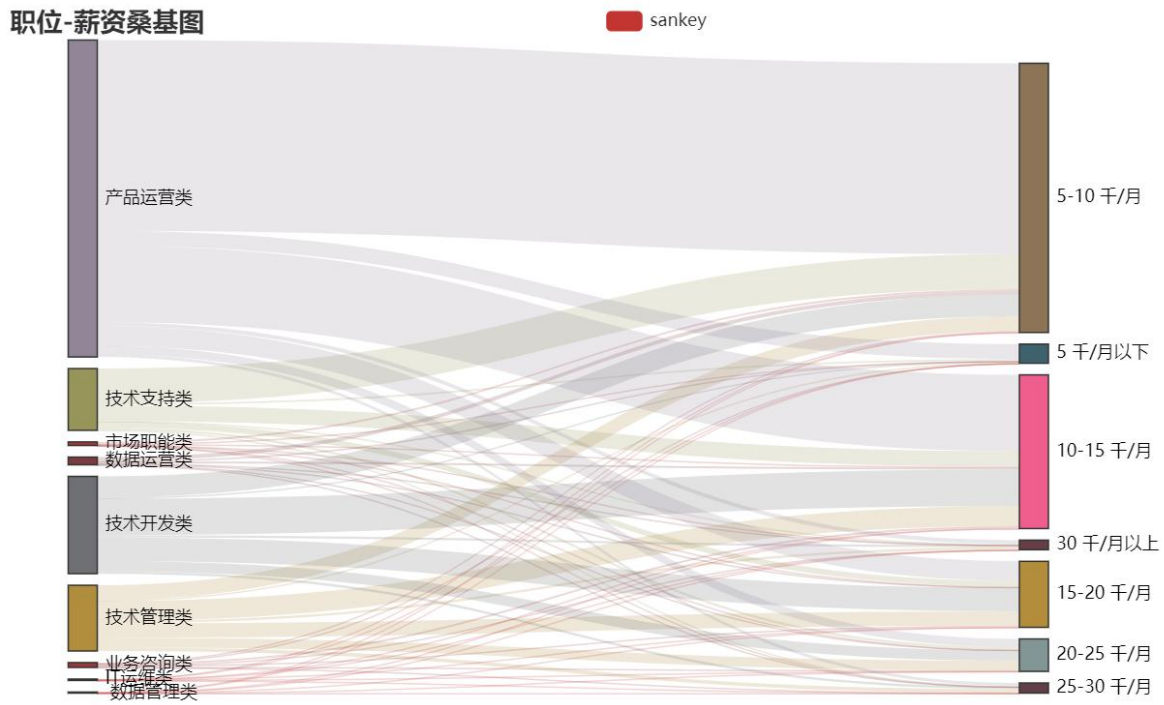

图 4-6 职位-薪资桑基图

4.6.4 分析结论

从 Sankey 图中可以很清晰的看出这九小类岗位类型大致的薪酬水平大致的分布情况和主要的分布区间。

（1）信管专业的总体薪资水平主要集中在 5-10k/月、10-15k/月和 15-20k/月这三个区间，每月 30k 以上的岗位数量非常少。

（2）市场需求较大的产品运营类、技术支持类、技术开发类和技术管理类岗位的薪资主要集中在 5-10k/月和 10-15k/月两个区间水平，并且这几类岗位中还有不少岗位的薪资能达到每月 15-20k/月的薪资水平。

总体来说，各种岗位类型的薪酬水平结构是比较稳定的。无论从事何种岗位工作，其薪酬水平分布都不会过于极端。

4.7 就业待遇分析（词云）

4.7.1 分析思路

由于原数据中的就业待遇内容比较杂乱，所以针对就业待遇，我们采用文本分析的方法，根据词频大小来制作词云图。从而可以查看被提及频次最多的就业待遇是哪些。

4.7.2 代码

```
import jieba
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
#用来描绘所有岗位的工作待遇热词共现
txt = ''
for each_result in list_dy:
    txt = txt + str(each_result[0])
dy = df[['工作待遇']]
list_dy = []
for d in dy.index:
    list_dy.append(dy.loc[d].values[0:])
#用来描绘所有岗位的工作待遇热词共现
txt = ''
for each_result in list_dy:
    txt = txt + str(each_result[0])
# 统计词频的字典
word_freq = {}

# 装载停用词
with open(r"D:\XJY\大三上\专业课程\大数据实训-卢辉\python 数据可视化实验\实验所需输入文件\stopwords.txt", "r", encoding='utf-8') as f1: # 读取我们的待处 理本文
    txt1 = f1.readlines()
stoplist = []
for line in txt1:
    stoplist.append(line.strip('\n'))
#切分、停用词过滤、统计词频
for w in list(jieba.cut(txt)):
    if len(w) > 1 and w not in stoplist:
        if w not in word_freq:
            word_freq[w] = 1
        else:
            word_freq[w] = word_freq[w] + 1
word_freq.pop('nan')
word_freq.pop('nannan')
#指定背景模式图片
```

```
backgroud_Image = plt.imread("D:/XJY/大三上/专业课程/大数据实训-卢辉/python 数据可视化实验/地大.jpg")
# 构造 WordCloud 对象
wc= WordCloud(
    background_color='white',
    max_words=100,
    collocations=False,
    width=1000,
    height=1000,
    font_path='simhei.ttf',
    mask=backgroud_Image)
# 调用方法生成词云图
wc = wc.generate_from_frequencies(word_freq)
# 保存图片
wc.to_file('WordCloud.png')
plt.imshow(wc)
plt.axis("off")
plt.show()
```

4.7.3 可视化结果

图 4-7 工作待遇词云图

4.7.4 分析结论

从图中可以非常清晰直观地看出岗位提供的各种福利待遇和一些基本的待遇类型。

（1）字体最大的是专业培训、绩效奖金和年终奖金，说明这些待遇是公司提供的基本福利待遇，一般公司都会提供。

（2）餐饮、通讯、交通、体检、弹性和全勤奖等的字体也较大，大多数的公司也会给予员工这些福利待遇，可以较基本待遇更吸引求职者。

（3）还有一些待遇较好的公司还会提供公积金、包吃、社保、班车、下午茶、住房补贴、一金五险等等福利，但这种待遇在图中的字体较小，说明这并不是普遍存在于各个公司，要视公司而定。这些福利待遇将更多地吸引要求较高福利待遇的求职者并起到积极的激励作用。

总体而言，公司一般都会提供专业培训、绩效奖和年终奖这些基本福利待遇，较好的公司还会提供更多更吸引人的福利。

4.8 公司规模与学历要求相关性分析（桑基图）

4.8.1 分析思路

本节内容主要希望探究公司规模与学历要求的相关性，在此同样采用桑基图的形式展现。首先统计各个公司规模与学历要求的对应关系，再逐一计数，最后生成桑基图。

4.8.2 代码

```
import pandas as pd
from pyecharts.charts import Sankey
from pyecharts import options as opts
#读取数据
f = open(r"D:\XJY\大三上\专业课程\大数据实训-卢辉\python 数据可视化实验\51Job_清洗完成.csv")
df = pd.read_csv(f)
#取出所需数据列
dx = df[['公司规模','工作要求']]
list_dx = []
for indexs in dx.index:
    list_dx.append(dx.loc[indexs].values[0:])
#取出公司规模不为空的元素
#取出工作要求不为空的元素，并只取出学历一值
list_new = []
for i in list_dx:
    print(i)
    if str(i[0]) != '' and str(i[1]) != '无' and str(i[1].split('，')[1]) != '':
        list_new.append([i[0],i[1].split('，')[1]])
    else:
        continue
for j in list_new:
    if str(j[0]) == 'nan':
        print(j)
        list_new.remove(j)
xl = ['中专','中技','硕士','高中','大专','初中及以下','博士','本科']
gm = ['50-150 人','少于 50 人','1000-5000 人','150-500 人','500-1000 人','10000 人以上','5000-10000 人']
```

```
#节点
nodes=[]
for con in xl:
    dic={}
    dic['name']=con
    nodes.append(dic)
for con in gm:
    dic={}
    dic['name']=con
    nodes.append(dic)
linkes = []
type = []
for a in xl:
    for b in gm:
        type.append([b, a])

    # 定义边和流量
linkes = []
for i in type:
    dic = {}
    count = 0
    for j in list_new:
        if j == i:
            count += 1
    dic['source'] = i[0]
    dic['target'] = i[1]
    dic['value'] = count
    linkes.append(dic)
c = (
    Sankey()
    .add(
        "sankey",
        nodes,
        linkes,
        linestyle_opt=opts.LineStyleOpts(opacity=0.2, curve=0.5, color="source"),
        label_opts=opts.LabelOpts(position="right"),
    )
    .set_global_opts(title_opts=opts.TitleOpts(title="公司规模-学历桑基图"))
    .render(r"D:\XJY\大三上\专业课程\大数据实训-卢辉\python 数据可视化实验
\sankey3_base.html")
)
```

4.8.3 可视化结果

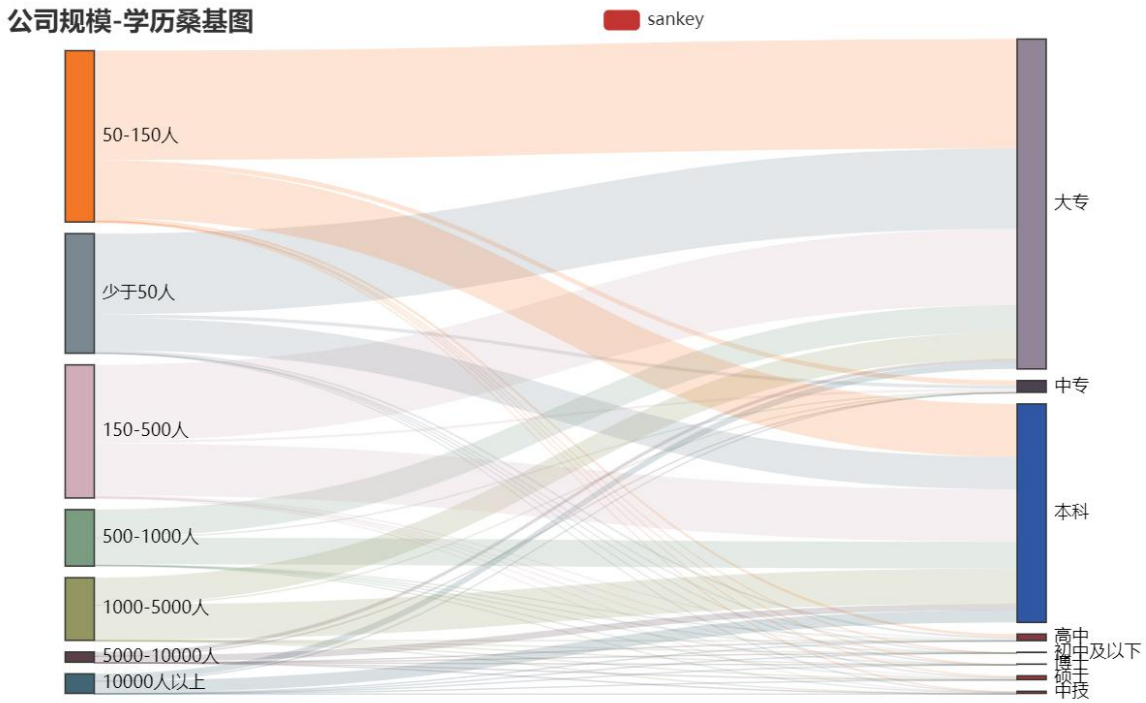

图 4-8 公司规模-学历桑基图

4.8.4 分析结论

从得到的公司规模-学历桑基图中可以观察到，公司规模与其学历要求还是存在一定正相关的关系的，主要结论如下：

（1）规模小于 500 人的公司有超过一半的学历要求都为大专。可以看出，大专学历的主要就业公司规模还是为中小型企业为主。

（2）规模超过 500 人的公司有超过一般的学历要求都为本科。所以可以得出，拥有本科学历之后，去规模较大的公司就职的机会还是比较大的。

（3）工作要求为博士的岗位数量较少，看不出较明显的就业趋势。然而，对于硕士来说，去 150-500 人规模的公司数量远远超过其他公司规模的数量。

总体来说，信管专业的公司规模与学历存在着一定正相关关系，但是关系不是非常大。