**基于python多进程实现POST请求方式网站（拉勾网）的数据爬取及可视化**

**工具：**

**Python3、pycharm、google浏览器**

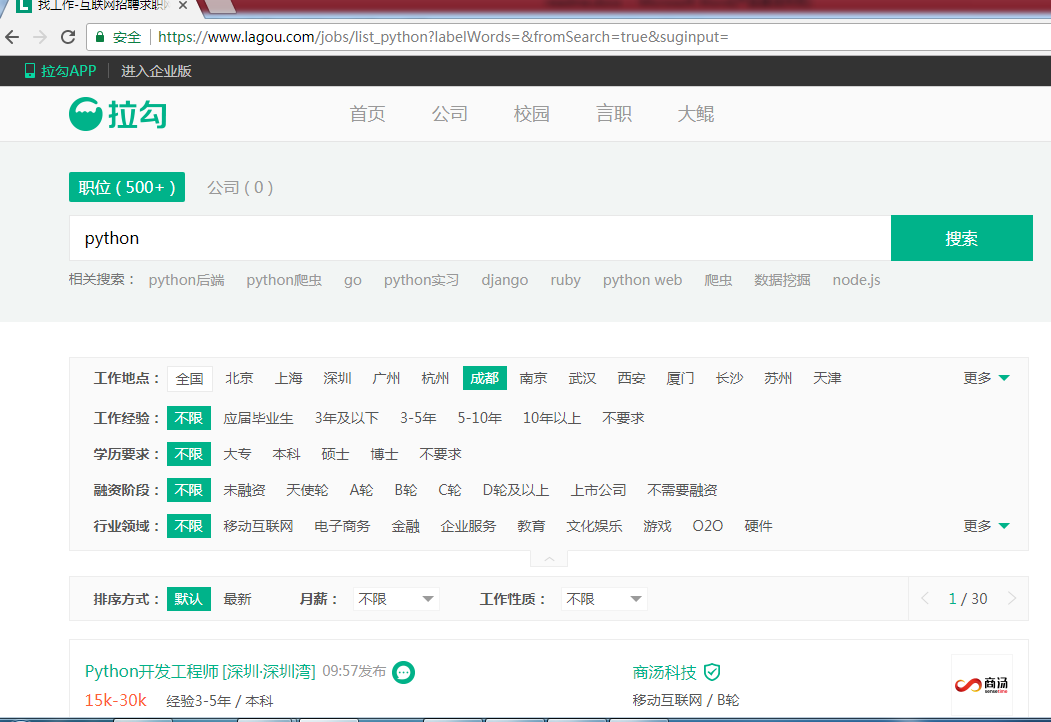
**需要安装的库：**

import requests *#请求网站*import pandas as pd *#处理爬取下来的数据格式*import pymysql *#连接mysql数据库*import uuid *#随机生成cookies的值，用于解决反爬*from multiprocessing import Pool *#进程池*from sqlalchemy import create\_engine *#连接数据库*import matplotlib.pyplot as plt *#绘图*import matplotlib as mob *#配置字体*import jieba.analyse *#分词*from wordcloud import WordCloud,ImageColorGenerator *#制作词云图*from PIL import Image *# 打开图片*import numpy as np *# 处理图片*import pylab as pl *#绘制横纵坐标图*

**原理：**

**通过修改提交给服务器的表单数据的值，来获取相应页码及搜索关键字的职位信息；并将获取到数据写入数据库后做可视化处理。**

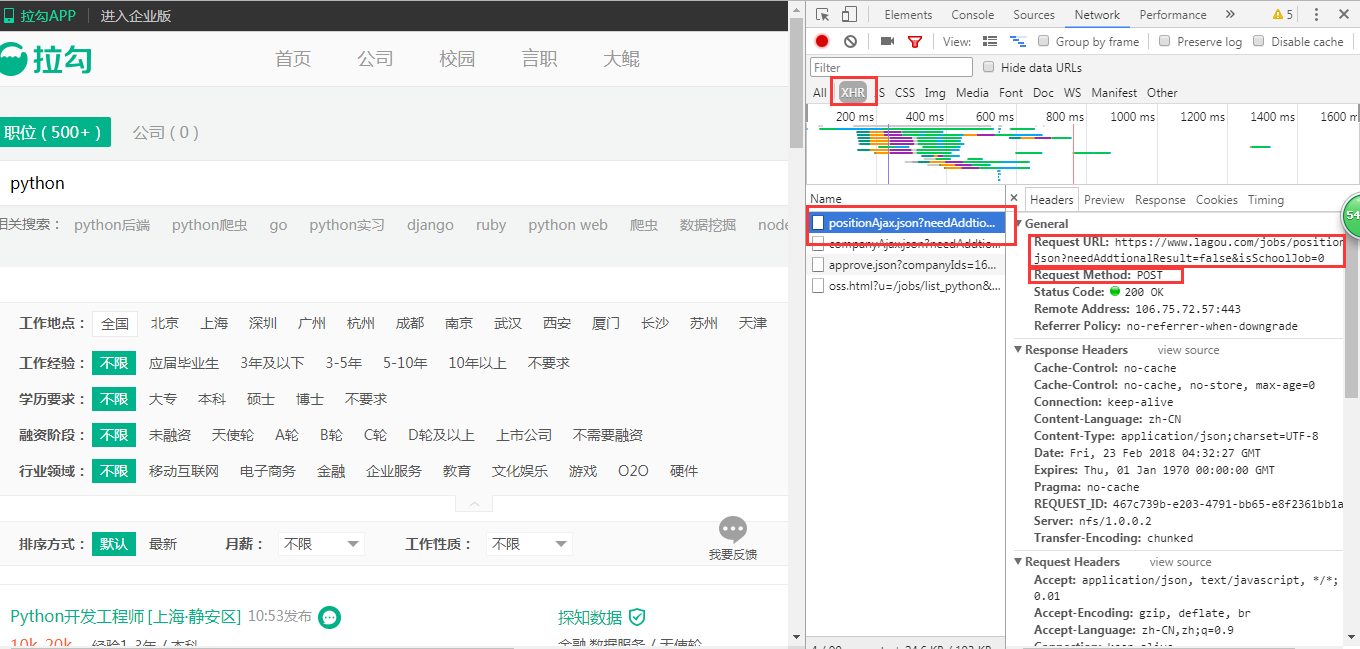
1. **分析网站**
2. **使用谷歌浏览器访问拉勾网主页，随便搜索一个职位名称。**



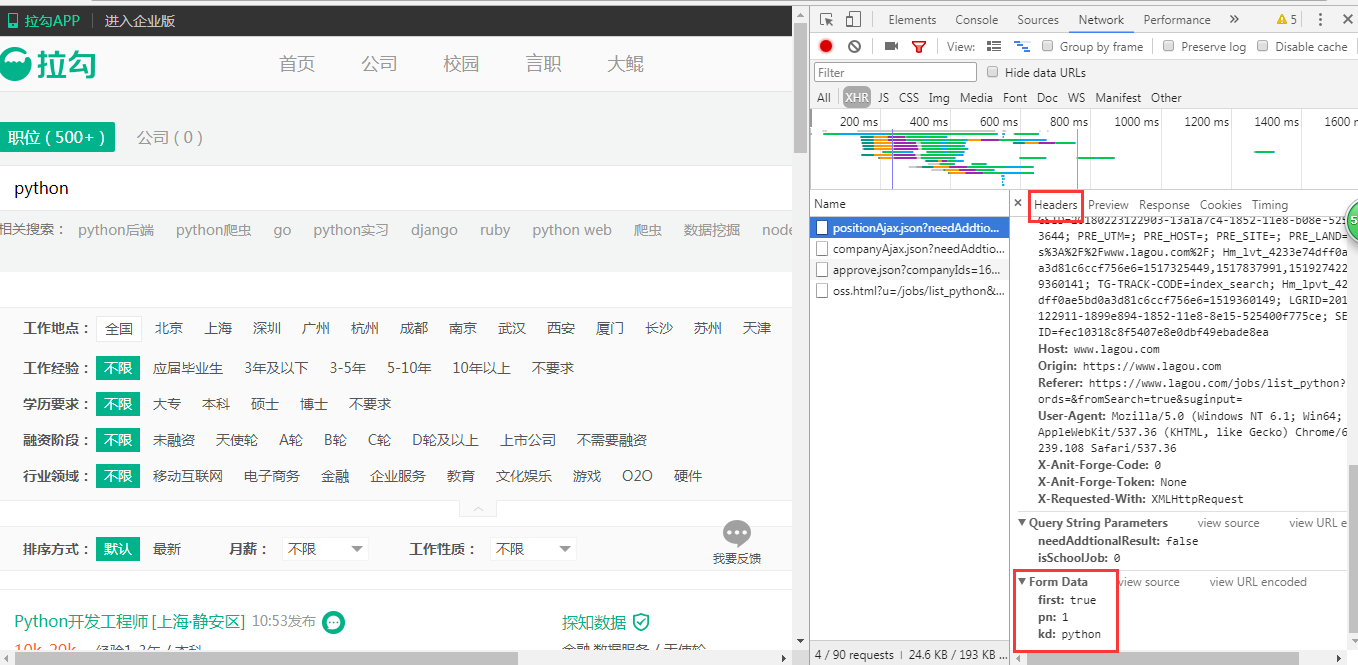
1. **打开开发者工具，选择Network进行抓包，通过抓包可以知道客户端与服务器之间进行怎样的一种交互。**



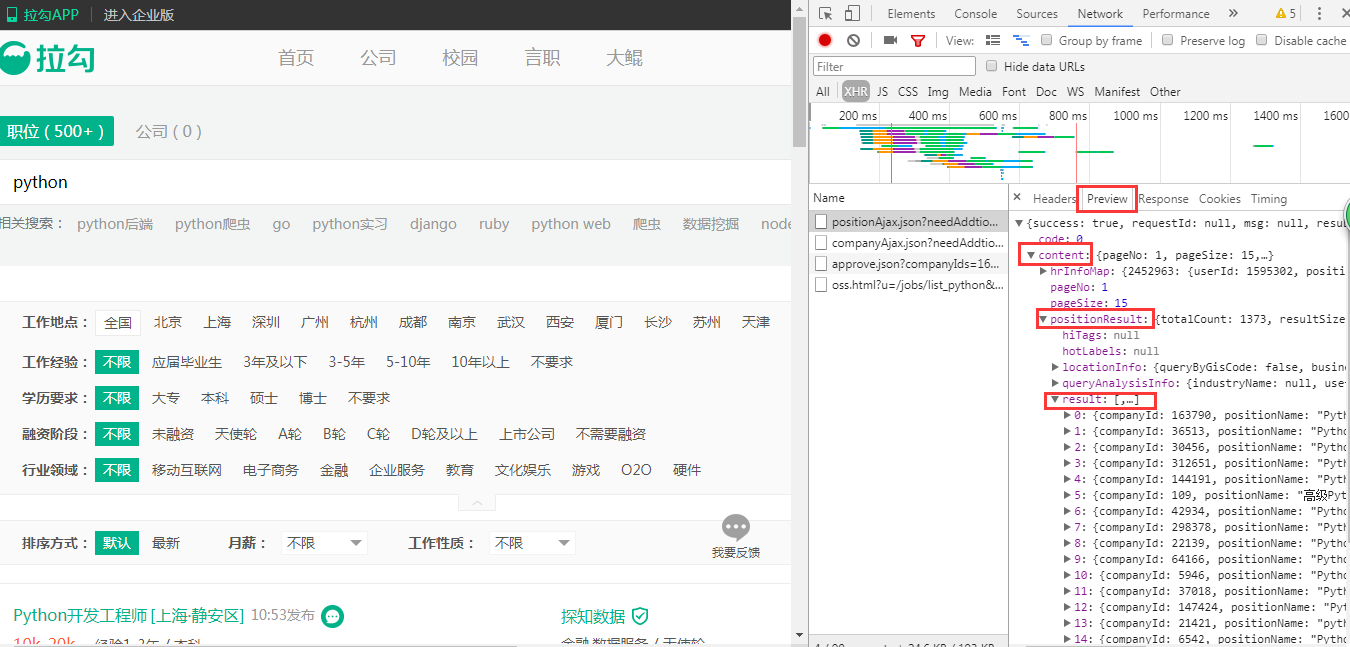
**3、刷新网页进行抓包，由于拉勾网的网络请求方式是POST，就是通过json格式的表单提交给服务器，服务器再返回对应的值。所以点击XHR按钮，可以看到提交给服务器的包，点击第一个包，可以看到request url是请求服务器地址，request method 是请求方式。**



**4、在头部文件headers最下面有个farm data，就是提交给服务器的表单数据。Pn是页码，拉勾网只会返回30页的数据，kd是关键字，这样只要将需要查找的职位的关键字传入这个表单的kd里，就可以返回对应的数据。**

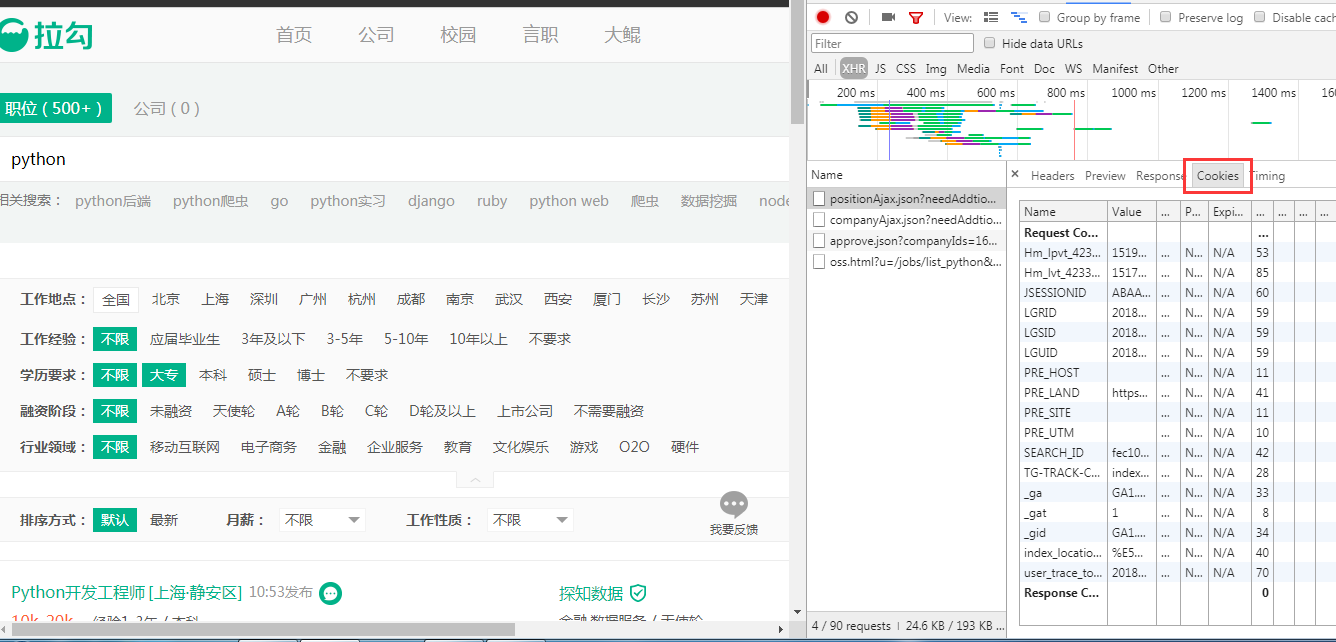


**5、点击preview(这里是服务器返回的数据)，在红框字段里一层层点开，在result里可以发现需要爬取的数据，即职位的相关信息都在这里。POST方法返回的数据格式为json格式，在处理数据时调用json包就可以获取到数据了。**



1. **反爬处理**

**拉勾网的反爬手段向头文件识别之类的肯定有的，主要的反爬手段还是访问频率过快服务器就会中断连接。这样反爬手段解决方法有通过代理ip访问网站，但这样方法需要去代理ip网爬取ip，还要验证ip的可不可用，有效期时间够不够。有点麻烦，因此在这里使用另外一种相对便捷的处理方法。**



**解决原理：在cookies这里，可以看到有很多值组成。其中JSESSIONID、user\_trace\_token、LGUID、SEARCH\_ID、LGSID、LGRID这字段的值是由随机数据数组组成，即只要修改这几个字段值，就会组成不同的cookies。访问一页数据就换一个新的cookies，这样服务器就会重新统计访问频率，就不会中断连接了。注意其他的cookies字段需要按照抓包时候的值填写。**

**解决方法：**

**安装uuid库，这个库可以随机生成cookies数组。**

import uuid

*# 随机生成uuid*def get\_uuid():  
 return str(uuid.uuid4())

cookie = "JSESSIONID=" + get\_uuid() + ";"  
"user\_trace\_token=" + get\_uuid() + "; LGUID=" + get\_uuid() + "; index\_location\_city=%E6%88%90%E9%83%BD; "  
"SEARCH\_ID=" + get\_uuid() + '; \_gid=GA1.2.717841549.1514043316; '  
'\_ga=GA1.2.952298646.1514043316; '  
'LGSID=' + get\_uuid() + "; "  
"LGRID=" + get\_uuid() + "; "  
headers = {'cookie': cookie, 'origin': "https://www.lagou.com", 'x-anit-forge-code': "0",  
 'accept-encoding': "gzip, deflate, br", 'accept-language': "zh-CN,zh;q=0.8,en;q=0.6",  
 'user-agent': "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36",  
 'content-type': "application/x-www-form-urlencoded; charset=UTF-8",  
 'accept': "application/json, text/javascript, \*/\*; q=0.01",  
 'referer': "https://www.lagou.com/jobs/list\_Java?px=new&city=%E6%88%90%E9%83%BD",  
 'x-requested-with': "XMLHttpRequest", 'connection': "keep-alive", 'x-anit-forge-token': "None",  
 'cache-control': "no-cache", 'postman-token': "91beb456-8dd9-0390-a3a5-64ff3936fa63"}

**这样就可以解决拉勾网所有的反爬手段啦。注意要在request.post（）中加上头文件headers**

1. **多进程处理**

**为什么要使用多进程，因为按照平常的速度，爬取一页数据需要1秒，30页就30秒，太慢了，‘快、准、狠’是程序的最高要求。在这里我使用20个进程，爬取30页共450条数据仅需2-3秒。**

**Python3的多进程实现主要依赖进程池库multiprocessing 的Pool方法。**

**关注python多进程的详解可浏览下面两个网页**

[**https://www.cnblogs.com/Tour/p/4564710.html?from=singlemessage&isappinstalled=0**](https://www.cnblogs.com/Tour/p/4564710.html?from=singlemessage&isappinstalled=0)

[**https://thief.one/2016/11/24/Multiprocessing-Pool/?utm\_medium=social&utm\_source=wechat\_session&from=singlemessage&isappinstalled=0**](https://thief.one/2016/11/24/Multiprocessing-Pool/?utm_medium=social&utm_source=wechat_session&from=singlemessage&isappinstalled=0)

**由于我这里需要传入页码跟关键字两个参数到主函数里，所以使用了异步进程池。**

from multiprocessing import Pool *#进程池*

page = [i for i in range(1, 31)]#创建一个1-30的页码列表  
pool = Pool(processes=20)#进程池数，一次最多可执行20个进程  
for i in page:  
 pool.apply\_async(get\_one\_page, args=(i,str(keyword)))#args是需要传入get\_one\_page函数里的多个值  
  
pool.close()  
pool.join()

1. **词云图wordcloud的安装**

**这个库按照平常的pip install 安装基本是不会成功的 ，可参考下面的链接安装**

[**http://blog.csdn.net/co\_zy/article/details/73922213**](http://blog.csdn.net/co_zy/article/details/73922213)

**我安装的时候就不只是链接里提到的问题，还报错说我缺少几个文件，不过都可以在**

[**https://www.lfd.uci.edu/~gohlke/pythonlibs/#wordcloud**](https://www.lfd.uci.edu/~gohlke/pythonlibs/#wordcloud)

**这里找到，只要找到报错里的文件下载下来安装上去就没问题了。**

**其他的就不写啦，都在代码里，有详细的注释。**