1. **需求分析**

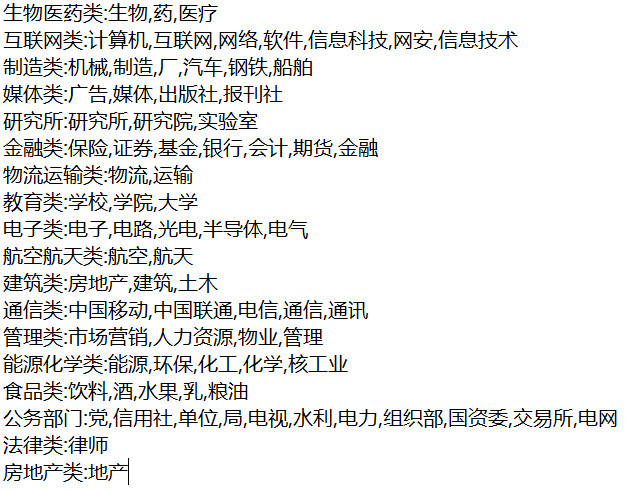
本次作业要求我们抓取北京邮电大学、西安电子科技大学、电子科技大学，自2021年9月1日以来的就业帖子，并对抓取下来的相关数据进行处理、分析，并最终得出结论。

实现这些需求所需要的python相关知识有爬虫、文件操作、数据处理等。

1. **模型设计**

设计的雇主分类保存在文件夹下的”分类.txt”

具体如下：



共有18个分类，在程序中以字典的形式存储。字典元素的键表示分类名称，值为一个列表，里面存有一个或多个表示雇主特征的关键词，以字符串形式存储。

若获取的公司名称数据中有与某个雇主分类相关的关键词，则该条帖子就会被打上相应的雇主标签，以此来达到雇主分类的目的。

由于雇主类型及不同雇主所对应的关键词是由自己设计，因此难免有覆盖不全面的问题存在。

1. **系统设计**
2. **数据抓取**

三校均采用selenium进行数据抓取（由于selenium要对网页进行渲染，数据爬取速度会比较慢）

***北邮：***

**浏览次数的获取**

该网站是动态网页，因此翻页不能用URL来实现，又因为对浏览次数的获取要到每个帖子的详情页进行，若在一个标签页中进行“定位标题元素→点击，进入详情页→获取浏览次数→返回上级页面”的操作的话，返回的页面会是网站的第一页。又因不能用URL来定位页面，故会丢失上次爬虫的进度。

针对这个问题的解决方法是在定位到每个帖子的标题元素后，获取它的详情页的URL。

|  |
| --- |
| url1**=**page.find\_element(By.XPATH,"//div[@class='infoItem collected recmmonded'][{}]/div[@class='left']/a[@class='tit']".format(i)).get\_attribute('href') |

在一个新的标签页中打开该URL，并将句柄切换到当前窗口

|  |  |
| --- | --- |
| 1  2  3  4 | js **=** 'window.open("{}");'.format(url1)  page.execute\_script(js)  page.implicitly\_wait(5)  page.switch\_to.window(page.window\_handles[**-**1]) |

在完成数据获取后，关闭该标签页并将句柄切换到上一标签页

|  |
| --- |
| page.close()  page.switch\_to.window(page.window\_handles[0]) |

这样便可以在不丢失首页面爬取进度的前提下获取浏览次数的信息

**所需数据的正则匹配**

对于职位数量、日期、浏览次数均采用正则表达式来进行匹配

例：定位到的元素文本为：  
*发布企业：中国医药健康产业股份有限公司   日期：2022-01-09   浏览次数：223*

则日期的正则表达式为：“(?<=日期：)\d\d\d\d-\d\d-\d\d” 浏览次数为：“(?<=次数：)\d\*”

当日期为"2021-08-31"时，退出循环

**翻页**

由于无法通过URL进行翻页，故采用点击页面中“下一页”元素的方式来实现

**程序简述：**

Get\_data()函数实现数据的爬取，while循环设为死循环,for循环读取每个页面中16个帖子详情页的数据，当日期为"2021-08-31"时，跳出循环。返回四个列表，分别为：

Title = [] # 招聘主题

Company = [] # 用人单位

Date = [] # 发表日期

View\_count = [] # 浏览次数

Position\_num=[] #职位数量

元素定位采用Xpath的方式。变量s,t用于定位“下一页”元素（因为在不同页中“下一页”所属xpath不同）。

最后将四个列表写入BEIYOU\_1.csv文件中。

***西电：***

大体和北邮差不多，有几个不同点：

**翻页**

由于西电是静态网页，因此翻页能采用URL的形式，但由于西电浏览次数仍要到详情页中获取，因此同样采用标签页切换的方法。

**循环跳出的条件：**

由于西电最后一个帖子日期仍在8月31日之前，要将西电的所有帖子都遍历一遍。因此循环跳出的条件变为是否已经遍历到最后一页，因此在开始爬取之前先找到最后一页的数字：

|  |
| --- |
| Last\_page **=** re.findall("(?<=page=)\d\*",page.find\_element(By.XPATH, "//\*[@id='yw0']/li[9]/a").get\_attribute('href'))[0] |

当当前页为最后一页时，跳出循环。

**每页中帖子的个数：**

由于西电最后一页中帖子个数不定，因此先用find\_elements方法获取帖子数量，再循环

|  |
| --- |
| Item\_Num**=**page.find\_elements(By.XPATH,"//div[@class='left']/div[@class='job']/div[@class='name']/a")  **for** i **in** range(1,len(Item\_Num)**+**1): |

***成电：***

相较于前两个，成电最为简单，由于成电没有浏览次数这个数据，因此不需要进到详情页来获取数据。仅需一页页地爬取，到"2021-08-31"时跳出循环即可。

1. **数据处理**

**1、**

要求：

* 1. 去掉招聘主题两端的空格等无关字符（如空格、tab、换行等）（有可能有，有可能没有）。
  2. 统一把日期处理为：YYYY-MM-DD的格式。
  3. 统一把访问次数为空的置为0，字符串统一转为整数格式。职位数量如果为空，置为1，统一转为整数。

实现：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | reader **=** csv.reader(rf)  **for** i, row **in** enumerate(reader):  **if** i !**=** 0: #从csv文件第二行开始读取，即不读表头          title, company, date, view\_Count,position\_num **=** row          Title.append(title.strip())          Company.append(company.strip())          Date.append(date) #日期本身即为YYYY-MM-DD的格式          View\_Count.append(int(view\_Count))  **if** position\_num**==**'':              Position\_num.append(1)  **else** :Position\_num.append(int(position\_num)) |

数据处理完写入对应CSV文件即可

**2、**

要求：

第4个标签页，表头字段为：序号（数字序号）、招聘主题、雇主类型。其中，招聘主题来自于前三个标签页去重之后的结果，雇主类型由完成作业的人自行设定，如中国移动雇主类型可以设为“电信运营商”，BAT可以设为“互联网平台企业”等，自己总结，自己设定，存入汉字即可。

实现：

首先，构造字典。Classify函数读入分类.txt文件，并将其存入全局变量字典tag中，字典的键为雇主类型，值为一个列表，列表中存放雇主所对应的一个或多个关键词(str)。

|  |
| --- |
| **def** Classify():      file **=** open("分类.txt", encoding**=**"UTF-8")  **while** True:          text **=** file.readline()  **if** **not** text:  **break**  **else**:              tag[text.split(":")[0]] **=** text.split(":")[1].split(",")  **for** item **in** tag:  # 去掉换行符          t **=** tag[item][**-**1]          tag[item][**-**1] **=** t.strip()          tag\_num[item]**=**[0,0,0] #个数,浏览次数,招聘职位个数          #tag\_num为一个字典类型全局变量，键为雇主类型，值为一个列表:[个数，浏览次数，招聘职位个数]          #在此定义tag\_num，便于后续数据处理      file.close() |

csv4函数实现上述需求，详见代码及注释

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | **def** csv4(Title\_CD,Company\_CD,Title\_XD,Company\_XD,Title\_BY,Company\_BY):      Title**=**[]      Company**=**[]      Item**=**{}  **for** i **in** Title\_XD,Title\_CD,Title\_BY:          Title.extend(i)  **for** i **in** Company\_XD,Company\_CD,Company\_BY:          Company.extend(i)  **for** i **in** range(len(Title)):          Item[Title[i]]**=**[Company[i],'']      # 利用字典的特性进行去重，当标题名称相同时，用后面的来覆盖前面的  **for** i **in** Item: #以公司名称为标准对雇主类型分类  **for** tag\_name **in** tag:  **for** item **in** tag[tag\_name]:  **if** len(re.findall(item, Item[i][0])) !**=** 0:                      # 当匹配到相应的关键词时，打上标签                      Item[i][1]**=**tag\_name  **break**      with open("CAT\_A.csv", "w", newline**=**'', encoding**=**"utf-8") as f:          writer **=** csv.writer(f)          writer.writerow(["序号", "招聘主题", "雇主类型"])          i**=**1  **for** item **in** Item:              writer.writerow([i,item,Item[item][1]])              i**+=**1      f.close() |

**3、**

要求：

最受北邮学生关注的招聘TOP20、最受西电学生关注的招聘TOP20、最受成电学生关注的招聘TOP20（以浏览次数决定排序）。

北邮招聘职位总数、招聘职位数量TOP10（简单按帖子主题就行，不需要解析企业名称）

实现：

|  |
| --- |
| data**=**[]  with open('XD\_D.csv', 'r+',encoding**=**"utf-8") as f:      reader **=** csv.reader(f)  **for** i,line **in** enumerate(reader):  **if** i!**=**0:              data.append(line)      data.sort(key**=lambda** item: int(item[4]), reverse**=**True)      #以第四列数据（浏览次数）为依据进行降序排序  f.close()  with open("最受西电学生关注的招聘TOP20.csv", "w", newline**=**'',encoding**=**"utf-8") as csvfile:      writer **=** csv.writer(csvfile)      writer.writerow(["序号", "招聘主题", "用人单位", "发布日期", "浏览次数"])  **for** i **in** range(0,20):          writer.writerow(data[i])  csvfile.close() |

北邮学生关注的招聘TOP20，北邮招聘职位总数、招聘职位数量TOP10与上述代码逻辑与形式基本相同，故不在此贴出。招聘职位总数、招聘职位数量TOP10只需将sort中的key参数改为：key=lambda item: int(item[5])即可。

由于成电浏览次数缺失，因此成电不统计该项。

**4、**

要求：

最受北邮学生关注的雇主类型TOP10、最受西电学生关注的雇主类型TOP10、最受成电学生关注的雇主类型TOP10。

北邮招聘职位所属雇主类型TOP10。

实现：

首先，将每个帖子的主题，招聘公司，浏览次数，职位数量存到以列表的形式存到列表Post中（二维列表）

Post.append([Title[i],Company[i],View\_Count[i],Position\_num[i]])

随后在函数tag\_match中实现上述需求

|  |
| --- |
| **def** tag\_match(Post,schoolname):      tmp\_tagNum**=**copy.deepcopy(tag\_num)      ICT**=**[]  **for** i **in** range(0,len(Post)):  **for** tag\_name **in** tag:  **for** item **in** tag[tag\_name]:  **if** len(re.findall(item,Post[i][1]))!**=**0:#匹配关键词                      tmp\_tagNum[tag\_name][0]**+=**1                      #统计该雇主类型数量                      tmp\_tagNum[tag\_name][1]**+=**Post[i][2]                      #统计该雇主类型对应浏览次数  **if** (tag\_name**==**"互联网类")**or** (tag\_name**==**"通信类"):#统计ICT行业                          ICT.append([Post[i][0],Post[i][1],Post[i][2],Post[i][3]])#主题,浏览次数,职位个数  **if** (schoolname**==**"北邮"):#若为北邮，统计职位数量                          tmp\_tagNum[tag\_name][2]**+=**Post[i][3]  **break**      \_ICT(ICT,schoolname)      sorted\_tagNum **=** sorted(tmp\_tagNum.items(), key**=lambda** item:(int(item[1][1]),int(item[1][0])),reverse**=**True)      #雇主类型数量，浏览次数两重排序，雇主类型数量优先，相同时浏览次数高的排在前面      with open("最受{}学生关注的雇主类型TOP10.csv".format(schoolname), "w", newline**=**'',encoding**=**"utf-8") as f:          writer **=** csv.writer(f)          writer.writerow(["雇主类型", "数量","浏览总数","排名"])  **for** i **in** range(0,10):              writer.writerow([sorted\_tagNum[i][0],sorted\_tagNum[i][1][0],sorted\_tagNum[i][1][1],i**+**1])      f.close()  **if** (schoolname**==**"北邮"):#北邮招聘职位所属雇主类型TOP10          sorted\_tagNum\_ByPositionNum **=** sorted(tmp\_tagNum.items(), key**=lambda** item: (int(item[1][2])), reverse**=**True)          #按招聘职位数量排序          with open("北邮招聘职位所属雇主类型TOP10.csv", "w", newline**=**'', encoding**=**"utf-8") as f:              writer **=** csv.writer(f)              writer.writerow(["雇主类型", "招聘职位", "排名"])  **for** i **in** range(0, 10):                  writer.writerow([sorted\_tagNum\_ByPositionNum[i][0], sorted\_tagNum\_ByPositionNum[i][1][2], i **+** 1])          f.close() |

**5、**

要求：

最关注ICT行业的招聘主题TOP10（在三个高校都出现，并且按浏览次数、职位个数，两重排序）

实现：

在tag\_match(详见上一条）函数中我们获得了与ICT行业有关的帖子的主题、浏览次数、职位个数，同样是以[主题，浏览次数，职位个数]的形式存在列表ICT里（二维列表）

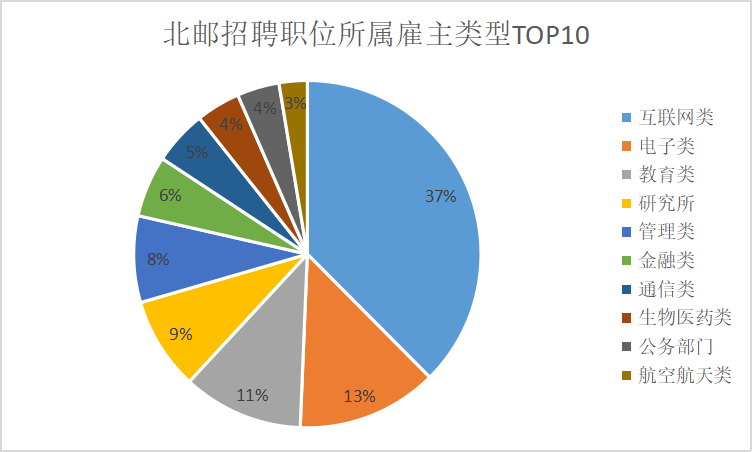
在函数\_ICT中实现上述需求：

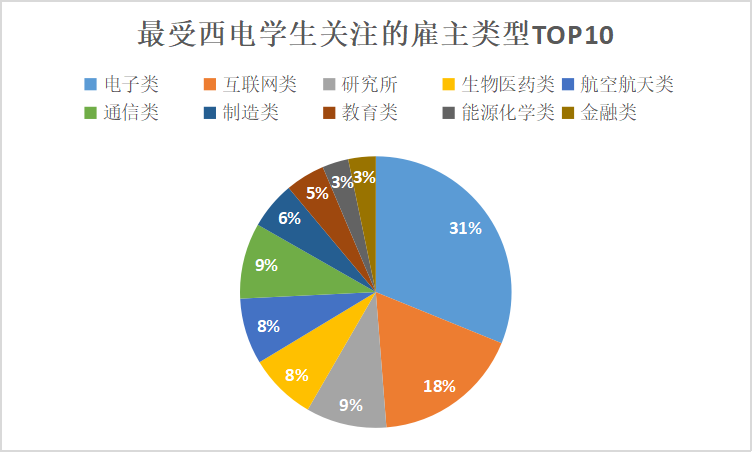
|  |
| --- |
| **def** \_ICT(ICT,schoolname):      ICT.sort(key**=lambda** item:(int(item[2]),int(item[3])),reverse**=**True) #按浏览次数、职位个数，两重排序      with open("{}最关注ICT行业的招聘主题TOP10.csv".format(schoolname), "w", newline**=**'', encoding**=**"utf-8") as f:          writer **=** csv.writer(f)          writer.writerow(["招聘主题", "招聘公司","浏览次数", "职位数量","排名"])  **for** i **in** range(0, 10):              writer.writerow([ICT[i][0], ICT[i][1], ICT[i][2],ICT[i][3],i **+** 1])      f.close() |

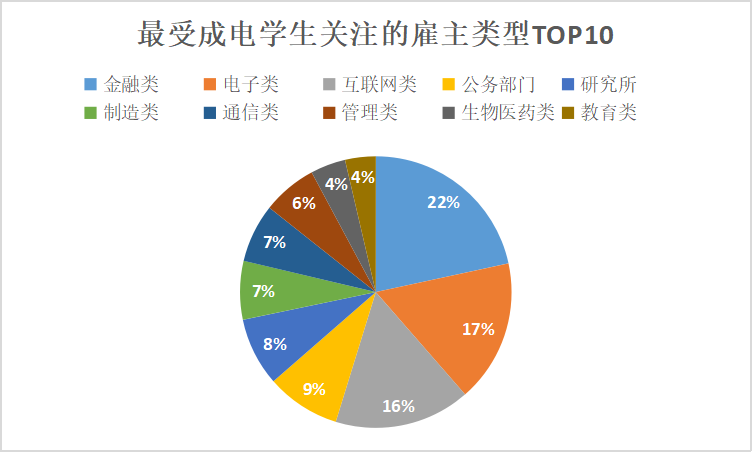
1. **数据分析**

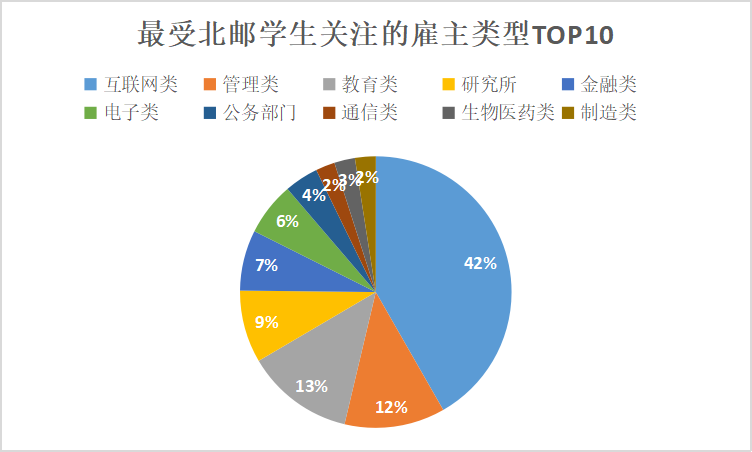
各详细数据详见文件夹中对应名称的csv文件

可视化图表如下（由于成电没有浏览次数数据，因此部分可视化图表没有意义，在此不列出）：









1. **作业小结**

本次作业对我而言是一次挑战，同时也是一个学习的机会。在做作业的过程中我遇到了许多困难，例如对北邮动态网页的无从下手、对csv文件操作的不熟练等，但我通过自己摸索、上网找资料等方式不断学习，最终将这些困难逐个击破，在这个过程中我既得到了代码能力的提升，又获得了满满的信心和成就感。因此，虽然做这个作业花费了我很长的时间和精力，但我认为这一切都是值得的。