# **京东钢琴-搜索引擎作业实验报告**

姓名:吴永麒

学号:20200739020

## 前言：『音乐是比一切智慧、一切哲学更高的启示。』——贝多芬

我很喜欢贝多芬的这句名言，也在一点点的践行着崇高的“启示”，也是因此实验选题是和音乐有关的-钢琴。

我做的是一个用scrapy爬虫爬取京东上有关钢琴的主流商品信息并存储为本地文件，然后利用pyecharts可视化探究热门钢琴的品牌分布的一个项目。

## 实验目的: 爬取京东上有关主流钢琴信息并探究热门钢琴的品牌分布。

## 实验原理: scrapy爬虫，python，xpath，json,jsonpath,文件读写，urllib,pyecharts可视化

## 实验步骤: 实验步骤具体分为四个部分：01 环境配置说明，02 scrapy数据爬取部分，03 网页分析，04 pyecharts可视化

## 项目目录结构(下面有一个”venv”的虚拟环境目录,这个目录可以不用管的O(∩\_∩)O):

|  |
| --- |
|  |

## 注意事项(如下截图笔记有详细图文说明,老师可以点击图片放大查看^\_^):

|  |
| --- |
|  |

## 环境配置

## （Tip：以下环境是我运行时候的环境配置，老师测试的时候是可以不必要按照这个要求来进行测试的O(∩\_∩)O ）

操作系统：Windows 10

内存配置：16G

运行环境：PyCharm 2021.3.3

PyCharm编码设置（如下截图笔记有具体图文说明，图太小看不清的话可以放大看的）：

|  |
| --- |
|  |

## 二、scrapy数据爬取部分

###### 版本：

scrapy版本：2.6.3

###### 主要步骤：

###### 1.0 创建scrapy项目（如下截图）：scrapy startproject jd\_piano

|  |
| --- |
|  |

###### 2.0 创建完毕之后按照提示进入到spiders目录（如下截图）：

|  |
| --- |
|  |

###### 3.0 之后执行爬虫项目生成命令：

|  |
| --- |
|  |

###### 4.0 之后是需要到setting文件注释掉这个机器人协议：

|  |
| --- |
|  |

###### 5.0 测试运行爬虫程序（下图有具体的截图笔记图文说明）：

|  |
| --- |
|  |

###### 6.0 成功运行的话会打印这几行语句的：

|  |
| --- |
|  |

至此，scrapy项目初始化搭建完成。下面开始对我们爬取的网页进行观察解析。

## 三、网页分析

###### 1.0首先进入到京东官网，然后在分类栏找到乐器->钢琴分类。

###### 2.0 改为按照评论数排序，因为需要找的是主流的钢琴，所以评论数太少的就不需要了。

|  |
| --- |
|  |

###### 3.0 F12打开控制台，刷新，找到Type为xhr（ajax请求），并且名为list相关的文件:

|  |
| --- |
|  |

###### 4.0 查看所选中的内容是否对应（如下截图笔记有具体的图文说明）：

|  |
| --- |
|  |

###### 5.0 提取出来第一页的请求地址（如下截图笔记）：

|  |
| --- |
|  |

###### 6.0 依此类推，找出12页的数据：

第一页：[https://list.jd.com/listNew.php?cat=6233%2C18615%2C18686&psort=4&click=0](https://list.jd.com/listNew.php?cat=6233%2C18615%2C18686&psort=4&click=0s)

第二页：<https://list.jd.com/listNew.php?cat=6233%2C18615%2C18686&psort=4&psort=4&page=3&s=57&click=0>

第三页：。。。

###### 7.0 然后找出分页规律。但是很遗憾，没有像过去那么简单就找到分页规律，对于第二个参数s难以找出规律。。。只能够逐页的访问了。（如下图找规律失败详细过程）

|  |
| --- |
|  |

###### 8.0 将18页（收集500条左右就够了，所以需要18个请求）的url数据封装成json格式进行存储

|  |
| --- |
|  |

###### 9.0 items文件自定义brand数据结构:

|  |
| --- |
|  |

###### 10.0 品牌的xpath（这个只是粗略的信息，后面还会有提取的，因为京东反爬的缘故所以不能够直接爬取brand字段，只能爬取包含有这个字段的信息了。。）

|  |
| --- |
|  |

###### 11.0 到settings文件开启pipeline下载：

|  |
| --- |
|  |

###### 12.0 到pipeline编写打开文件，写入文件，关闭文件的执行代码：

|  |
| --- |
|  |

其中因为生成的是json数组,所以需要给全部json字符串用[ ]括起来同时每分隔一个json字符串需要加一个逗号',',同时还需要将单引号改为双引号

###### 13.0 还有需要到settings文件添加一个请求头伪装一下:

|  |
| --- |
|  |

###### 14.0 在piano文件定义变量page,值为1,同时将allowed\_domains改为京东域名.

|  |
| --- |
|  |

###### 15.0 piano主爬虫文件核心方法parse方法的代码逐行详细解析:

|  |
| --- |
|  |

**piano文件方法parse代码逐行解析:**

|  |
| --- |
| def parse(self, response):    *# 将之前封装的18页的json数据以”只读”和”utf-8”编码格式的方式加载出来* pj = json.load(open(file=r'./pages/pages.json', mode='r', encoding='utf-8'))   *# 这里通过xpath获取到每一页所有商品的li,这里的返回值是list列表* li\_list = response.xpath('//ul[@class="gl-warp clearfix"]//li')  # 遍历每一页的每一个li  for i in range(len(li\_list)):  # 通过xpath获取到有关brand的字段并提取第一个data值  brand\_ = li\_list[i].xpath('.//div[@class="gl-i-wrap"]//div[@class="p-name p-name-type-3"]//i/text()').extract\_first()  # 强转为字符串然后利用字符串的split方法按照空格分隔取第一个值,为什么要这样做## 呢?因为如果直接获取值放到json文件的话,有一些字段是单引号里面包着单引号(这会引起编译器语法错误检查),并且这些单引号包着单引号的数据都有一个规律-隔着空格,所以可以按照空格分开之后只需要获取分隔之后的第一个数据即可,第二个数据经过观察往往是不太中重要的数据可以直接舍弃掉.  brand = str(brand\_).split(' ')[0]  # 判断brand字段是否为空:None,事实上在实际爬取的时候不知道是什么原因会出现空数据None的情况的,这样的数据放到json文件里面也是不符合语法规范的,所以需要提前检测出来,如果brand字段不是None可以按照原来的值赋值,否则的话直接用’None’替换掉.  if brand:  brand = brand  else:  brand = 'None'  # 将提取到的brand字段数据传到items文件那边,所以写这行代码之前首先需要引# 入: from jd\_piano.items import JdPianoItem  piano = JdPianoItem(brand=brand)  # 将封装好的数据传输到pipelines进行处理  yield piano  # 判断页数是不是已经够了18页,够了的话就停  if self.page < 18:  # 获取pages.json文件对应页数的url  url = pj[self.page][str(self.page+1)]  # 页数+1  self.page = self.page + 1  # 再次发起下一页的请求调用parse方法作为回调函数执行业务代码  yield scrapy.Request(url=url,callback=self.parse) |

###### 16.0 最后,运行爬虫框架代码

|  |
| --- |
|  |

# pyecharts生成扇形图查看钢琴的品牌分布

**版本: pyecharts: 1.9.0**

**generate\_pic.py文件代码逐行解析:**

|  |
| --- |
|  |

|  |
| --- |
| from pyecharts import options as opts from pyecharts.charts import Pie from urllib import request from lxml import etree  import json import jsonpath  # 加载我们之前写入的pianos.json文件 json\_p = json.load(open(file=r'../pianos/pianos.json', mode='r', encoding='utf-8')) # jsonpath直接提取brand字段的数据 json\_brand = jsonpath.jsonpath(json\_p, expr='$..brand') # 请求头伪造 headers = {  'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36' } # 爬取京东上有关钢琴的总共123个品牌字段的url地址，这里的品牌和上面爬取到的brand字段# 是不同的，上面的brand字段是每一个商品的具体说明，这里爬取的品牌是钢琴分类栏那边爬取## 的所以是不一样的数据 url = 'https://list.jd.com/list.html?cat=6233%2C18615%2C18686&psort=4&psort=4&page=1&s=1&click=0'  # 请求头构造函数定制  request01 = request.Request(url=url, headers=headers) # 发起请求 resp01 = request.urlopen(request01) # etree解析 brand\_ = etree.HTML(resp01.read().decode('utf-8')) # 通过xpath获取钢琴所有的品牌字段信息 brand\_l0 = brand\_.xpath('//div[@class="sl-v-logos"]//a/@title') # 定义列表存储钢琴品牌数据 brand\_l = [] # 往列表添加钢琴品牌数据 for l in range(len(brand\_l0)):  # 提取钢琴的品牌信息,但是因为爬取到的钢琴字段有’()’之类的干扰,所以需要去掉,按##### 照’(’分隔只需要第一个数据即可  brand\_l.append(str(brand\_l0[l]).split('（')[0]) # 定义others字段为0,用来对其他钢琴品牌进行计数 others = 0 # 循环动态创建变量,变量名为钢琴品牌的字段,初始值是0 for i in range(len(brand\_l)):  locals()[str(brand\_l[i])] = 0 # 遍历brand字段数据 for b in range(len(json\_brand)):  # 遍历钢琴品牌字段  for j in range(len(brand\_l)):  # 利用brand字段字符串的find方法逐项匹配每一个品牌字段,大于等于0说明匹配成功  if str(json\_brand[b]).find(brand\_l[j]) >= 0:  # 匹配成功之后往对应的变量加一  locals()[brand\_l[j]] = locals()[brand\_l[j]] + 1  # 退出循环  break  # 判断是否会出现匹配了所有爬取到的品牌但是没有一个匹配到的,也就是其他品牌了  if j == (len(brand\_l) - 1):  # 计算其他品牌加一  others = others + 1 # 定义列表封装品牌数最高的10个品牌(这是个品牌是我自己筛选出来的,就不写排序的代码了)以及others brand\_list = [  brand\_l[0],  brand\_l[1],  brand\_l[3],  brand\_l[4],  brand\_l[9],  brand\_l[10],  brand\_l[16],  brand\_l[17],  brand\_l[24],  brand\_l[26],  'others' ] # 定义列表封装对应的局部变量的数值到列表中(这个数值是每个品牌的数量) num\_list = [  locals()[brand\_l[0]],  locals()[brand\_l[1]],  locals()[brand\_l[3]],  locals()[brand\_l[4]],  locals()[brand\_l[9]],  locals()[brand\_l[10]],  locals()[brand\_l[16]],  locals()[brand\_l[17]],  locals()[brand\_l[24]],  locals()[brand\_l[26]],  others ] # 这一部分是pyecharts的代码,是到pyecharts官网的扇形图部分直接粘贴的,所以就没有详细# 讲了^\_^,只是改了几个参数代入即可 c = (  Pie(init\_opts=opts.InitOpts(height='800px'))  # 这里的brand\_list和num\_list是放入我们上面自己处理好的的数据  .add("", [list(z) for z in zip(brand\_list, num\_list)])  .set\_colors(["blue", "green", "aqua", "red", "pink", "orange", "purple"])  # 改一下标题  .set\_global\_opts(title\_opts=opts.TitleOpts(title="京东钢琴品牌分布扇形图",pos\_top='50px'))  .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))  .render("pie\_set\_color.html") ) |

最后,运行这个generate\_pic.py文件,即可在pyecharts文件夹下面看到生成的pie\_set\_color.html文件,然后在浏览器打开这个html文件即可看到可视化的结果了

|  |
| --- |
|  |

结果图效果:

|  |
| --- |
|  |