大作业报告

第 25 组　　组长：杨宇博　　组员：陈硕　杨冠柔　臧浩男

目录

1.爬虫部分

1.0.1.选取爬取网站
1.0.2.爬取关键内容
1.0.3.商品信息分析与爬取实现关键(以苏宁为例)
1.0.4.实现结果展示
1.0.5.实验问题与改进

2. 索引建立部分

2.0 文字索引
2.0.1 主要思路
2.0.2 准备工作
2.0.3 具体代码实现
2.0.4 实验问题分析

2.1 图片索引的建立
2.1.1 图片索引建立原理概述
2.1.2 具体代码实现
2.1.3 待改进的地方

3. 搜索

3.0 文字搜索
3.0.1 文字搜索原理概述
3.0.2 文字搜索具体实现

3.1 以图搜图
3.1.1 以图搜图原理概述
3.1.2 以图搜图具体实现

4. 网页部分

4.0 flask 框架

4.0.1.flask 框架与内容概述

4.1 网站主页

4.1.1 主页前端实现

4.1.2 成果展示

4.2 网站商品品牌或类别过滤页

4.2.1 具体商品搜索结果

4.2.2 成果展示

4.3 搜索结果返回页

4.3.1 基本操作实现

4.3.2 前端结果展示实现

4.3.3 搜索结果展示

4.4.4 问题与改进

1.爬虫部分

1.0.1.选取爬取网站

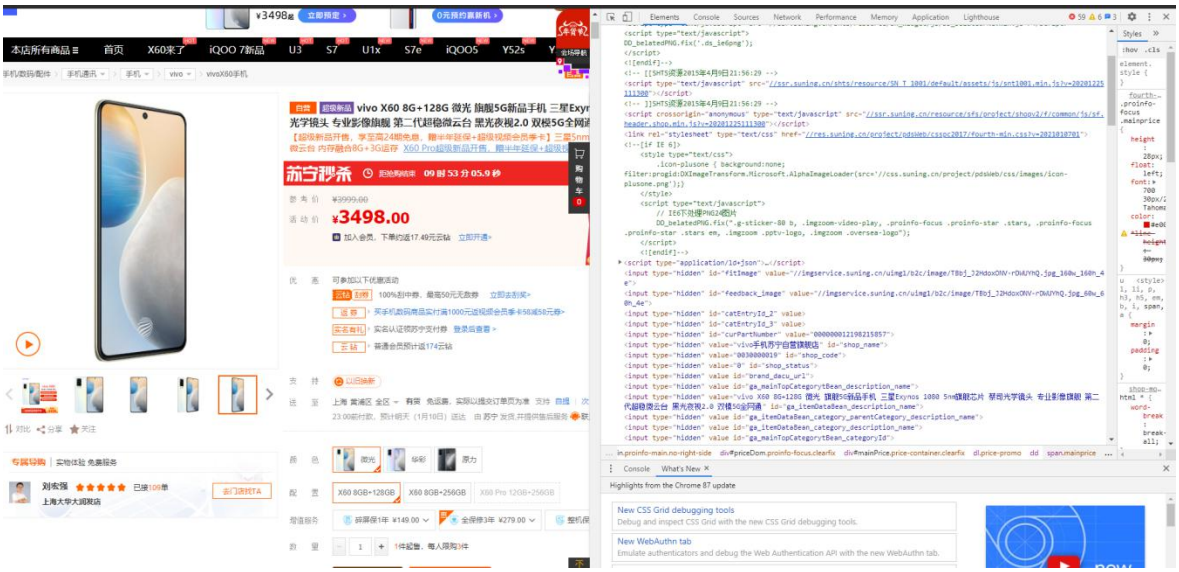
实验的开展，首先是从数据的获取出发，我们经过组内讨论决定采用三个网站为我们实验中信息的全部来源，分别为：当当、国美和苏宁。三种爬虫的实现方式分别交由不同的同学完成。在实验中，数据的主要来源为苏宁网站（https://www.suning.com/），由于爬虫的实现过程在面对前两者网站时差异并不明显，而在爬取苏宁网站的商品时操作较为复杂，我们决定在实验报告中仅展示小组对苏宁网站商品内容的爬取思路、过程与结果。

1.0.2.爬取关键内容

为完成搭建搜索引擎的目标，我们所需要的网页中的数据为：商品链接的 url，文件名，商品名称，商品关键词，商品图片，商品价格，商品评论数，商品好评率。

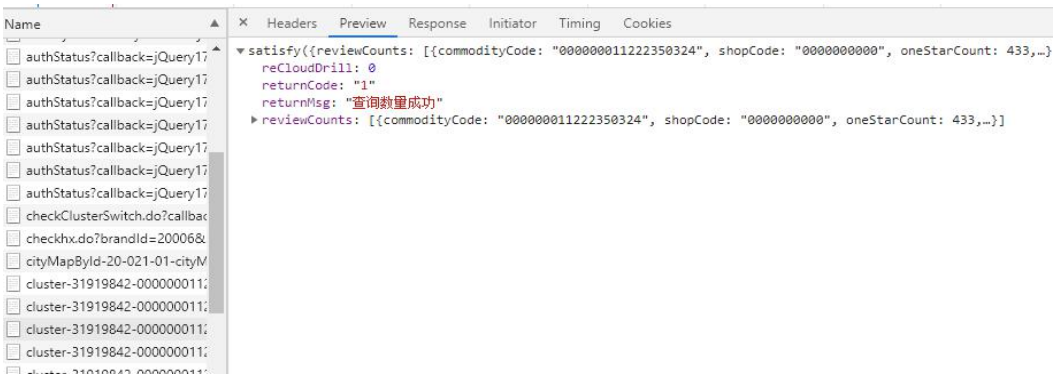
1.0.3.商品信息分析与爬取实现关键(以苏宁为例)

在爬虫代码书写前，需要分析苏宁商品信息的结构，常见的苏宁网页商品信息如下图所示(html 文件)：



在根据上述网站以及其相关内容进行分析后，我们找到了实现苏宁爬虫的关键：

1.获取手机的信息：其中获取 clusterid 的步骤十分关键，在后续的操作中我们为了提取评论数与评论统计数据的 url 构造中都含有这个参数。



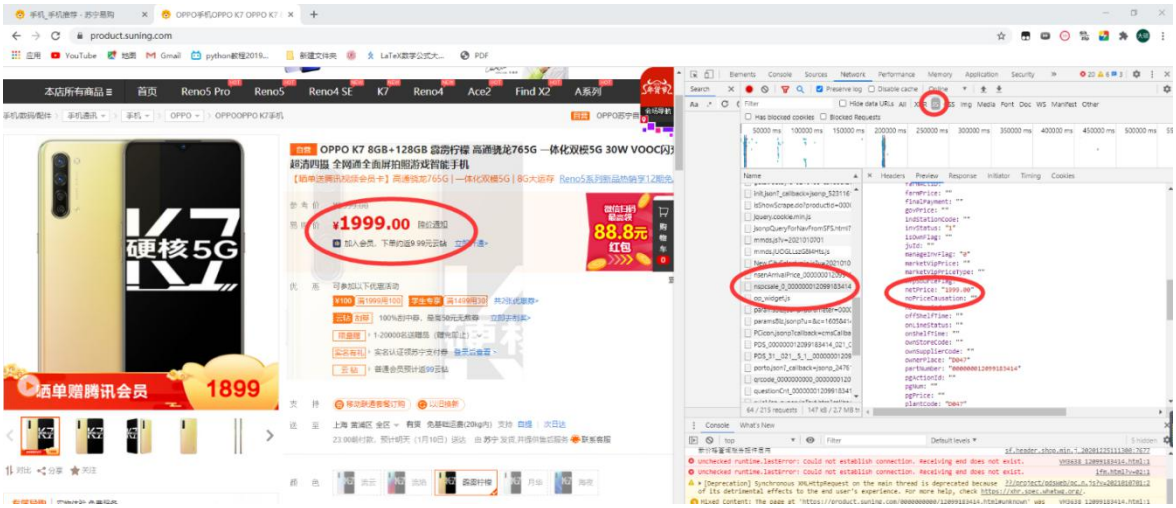
2.对于静态网页结构内容解析：对于搜索到的静态网页，我们为获取商品具体页面之后的内容，需要利用 Beautifulsoup 进行解析提取需要的信息。

上述操作的关键部分代码如下图所示：

```
1.  get_phone_data(self, html):
2.  soup = BeautifulSoup(html, 'lxml')
3.  li = soup.find_all("li")
4.  for i in range(len(li)):
5.  try:
6.      src = li[i].find_all("a", attrs={"target": "_blank"})[0].get("href")
```

使用如上述代码所示的操作可以获取到商品图片等相关数据，这些数据显然并未被保护。

而对于利用 JavaScript 函数保护的数据：比如手机的价格，我们通过检查元素分析（如下图所示）

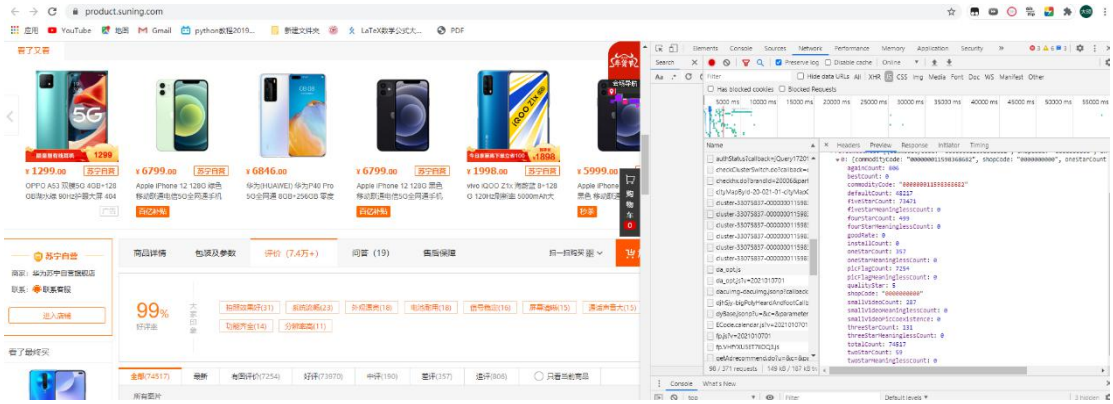


初始时无法直接搜索到网页中价格的存储位置，而实际上其存储地址如代码中所示：

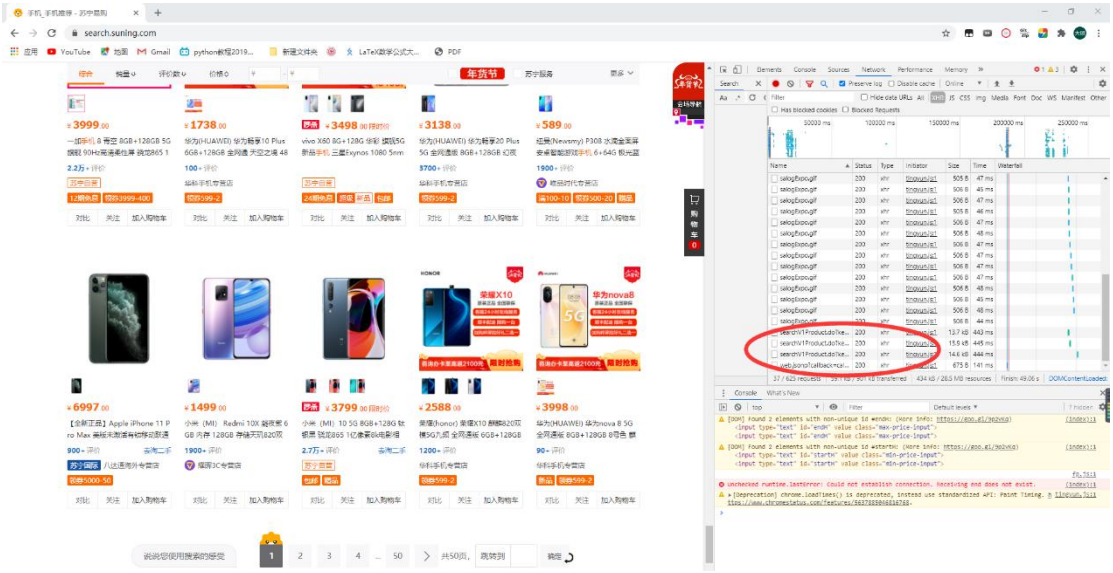
```
1.    price_src = "https://pas.suning.com/nspcsale_0_0000000" +/
2.                        key1 +str + key1 + "_" + key0 + str
```

因此在爬取过程中，我们需要做到的是对目标地址进行拼接，对于所需要的评论数，我们可以依照上述的方法进行拼接得到。

注意到，每个网站好评率的定义有较大的差异，这一点是我们在后续按照好评度排名统一方面的一个难点。对于我们所爬取的苏宁网站，其采取的是如下方法，即用户评论为四星到五星之间的评论结果定义为“好评“，因此我们小组再商讨后决定我们只爬取的不同好评数与总评论数，再对结果进行简单的比值计算，这样可以实现一定程度上各网站爬取结果的统一。在建立索引的时候，我们会依据这一比值以及好评数目与评论总数相结合定义一个相应的计算方法统一三个网站上的数据，其排名的顺序即为本次实验定义的 Rank 顺序。



3.对于利用 Ajax 异步加载的网页：对于商品，我们在搜索关键词后会得到带有一定量数据的页面，在分析网页结构时，我们发现在下拉滑动的时候，加 JS 页面中展示出了更多的商品信息，这样我们才能爬取足够多的有效信息。我们不难发现发现，苏宁在网页地址的简单变化上只是利用了异步，因此我们可以成功地获取商品搜索结果页，随后便可以开始爬取结果页中展示的所有商品的详情页：



4.其他的代码部分分析：首先初始化爬虫相关信息，然后利用字典的元素互异性实现网页存入文件夹的不重复。考虑到程序的健壮性，我们利用 try-except 结构，并利用 urllib.parse.quote 对输入的字符进行编码传参。

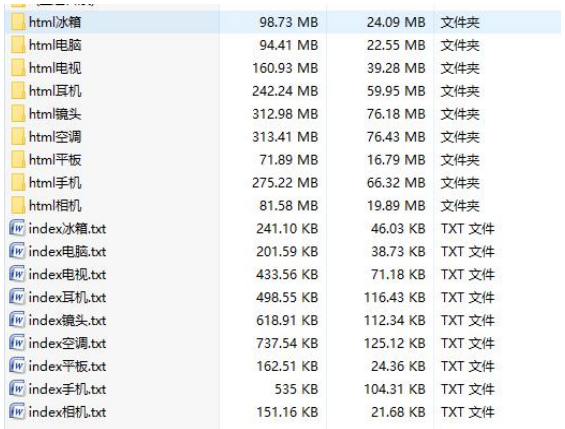
在搜索时我们采用了现定位后进行爬取的方法，即在爬取前先确定爬取的种类，如“手机”、“电脑”、“冰箱”等，在爬虫的搜索启动部分，只需要手动填入所需搜索的部分即可实现类似于手动填入搜索框并确定的效果。

完成了以上关键步骤的实现，对于苏宁部分的爬虫任务即可以用前文中提到的解决方法，结合本学期爬虫相关知识进行代码上的实现。详细的代码实现过程见文件中的（suning.py）代码。

1.0.4.实现结果展示

最后我们爬取了超过 8000 条苏宁的商品数据，接近 4000 的当当和国美的商品数据，共计爬取数据为 15000 逾条，基本上满足了我们实验目标，也在一定程度上能够很好的服务于后续程序中开展的索引建立以及搜索的实现。之后建立的索引是以苏宁商品信息为主要部分，其他两个网站穿插分析，这样也在一定程度上满足了搜索引擎的查全率和垂直搜索。

爬取的数据以 txt 文件形式进行存储，而 html 文件则统一存储在文件中。存储结果如下图所示：



html冰箱	98.73 MB	24.09 MB	文件夹
html电脑	94.41 MB	22.55 MB	文件夹
html电视	160.93 MB	39.28 MB	文件夹
html耳机	242.24 MB	59.95 MB	文件夹
html镜头	312.98 MB	76.18 MB	文件夹
html空调	313.41 MB	76.43 MB	文件夹
html平板	71.89 MB	16.79 MB	文件夹
html手机	275.22 MB	66.32 MB	文件夹
html相机	81.58 MB	19.89 MB	文件夹
index冰箱.txt	241.10 KB	46.03 KB	TXT 文件
index电脑.txt	201.59 KB	38.73 KB	TXT 文件
index电视.txt	433.56 KB	71.18 KB	TXT 文件
index耳机.txt	498.55 KB	116.43 KB	TXT 文件
index镜头.txt	618.91 KB	112.34 KB	TXT 文件
index空调.txt	737.54 KB	125.12 KB	TXT 文件
index平板.txt	162.51 KB	24.36 KB	TXT 文件
index手机.txt	535 KB	104.31 KB	TXT 文件
index相机.txt	151.16 KB	21.68 KB	TXT 文件

1.0.5.实验问题与改进

1.在爬取的过程中，发现爬虫因为访问速度过快，出现“您搜索的商品找不到了“这样类似的网站保护机制，测试时选择不同的代理 ip 进行爬取，我们选取了购买的代理 ip 地址，地址来源为芝麻 ip 网站。

将选取的 ip 地址存储在文件中，格式为 text 文档。

为了保证 ip 地址访问的有效性，我们采用随机数生成的方法来决定 ip 地址的选取，首先将 ip 的地址全部存入一个列表中，使用随机数种子生成相差较大的随机数，再调用我们 ip 列表中的 ip 地址，这样就可以一定程度上避免触发保护机制。

并添加进 request 爬虫的混淆池。当然我们也可以使用不同的 User-Agent，这里展示一种方法，具体实现的代码如下所示：

```
import random,requests
pro_list=[]
f = open('ip.txt','r')#ip 代理文件
for line in f.readlines():
    ipd = {}
    ipd["http"]=line
    pro_list.append(ipd)
f.close()
proxies = random.choice(pro_list)
res = requests.get(url, headers=self.headers,proxies=proxies)
```

2.注意到，我们的爬虫程序在访问到死链接或者陷阱商品网页时，爬虫会发生假死的现象，或者直接返回了空值，为避免这种方式的发生，我们在遇到这种情况时这直接利用 pass 过滤，这样能够保证一定的安全性，但偶尔会产生数据量过滤过多导致数据过少问题，我们不得不在某一类型的商品比原计划多爬取一部份文件进行预留，这导致了我们爬虫进度有一定的拖慢。

3.在爬取的过程中，我们也遇到了翻页失败的问题。苏宁采取的为 Ajax 加载的方式，这与其他的网站有较大的不同。在初次爬取时，我们忽略了这个问题，导致了爬取的商品数量过少并且偶尔在翻页时无法进行的情况。我们采用了模拟人工将页面调制最底端的来保证其异步加载时展示出完全的网页内容。对于翻页的步骤，通过跳转框来进行操作，输入循环的次数 i，模拟点击确定的操作，并通过高亮判断是否进入跳转页面，是否最终跳转至正确页面，将上一页获取的内容作为下一页的终止条件。

2.索引建立部分

2.0 文字索引

2.0.1 主要思路

通过 Lucene，根据已经爬取好的 txt 文件建立索引。

2.0.2 准备工作

（1）工具简述：
Lucene 是一个高效的，基于 Java 的全文检索库。Lucene 的 analysis 模块主要负责词法分析及语言处理而形成 Term。 Lucene 的 index 模块主要负责索引的创建，里面有 IndexWriter。Lucene 的 store 模块主要负责索引的读写。

（2）爬取的 txt 文件内容
我们从苏宁、当当、国美三个网站爬取电商信息，以下展示爬取结果的格式内容。下面展示一行内容为例，每一行中的内容用 tab 分隔。

1.苏宁
https://product.suning.com/0000000000/658519899.html
httpsproduct.suning.com0000000000658519899.html
容声(Ronshen)冰箱 BCD-251WKD1NY 容声（Ronshen）251 升 三门三开门 冰箱风冷无霜节能静音家用电脑控温 中门宽幅变温 BCD-251WKD1NY【价格 图片 品牌 报价】-苏宁易购容声苏宁自营旗舰店 品牌：容声(Ronshen) 颜色：沐光金 箱门结构：三门 面板类型：VCM 适用家庭：三口之家 整机质保年限：1 年
https://imgservice.suning.cn/uimg1/b2c/image/ODUx40Qe5Zm40jJgIp4Yww.jpg_800w_800h_4e2313.00 180813 0.9948289116379906

2.当当
http://product.dangdang.com/1516430471.html
httpproduct.dangdang.com1516430471.html
X1 Carbon 14 英寸原装笔记本电脑包+无线激光鼠标+ThinkPad HDMI 转 VGA 转换器
NU
http://img3m1.ddimg.cn/50/12/1516430471-1_k_2.jpg
288.00 102.0 100.0

3.国美
https://item.gome.com.cn/A0006634923-pop8013111940.html?search_id=sdw233vyv434
httpsitem.gome.com.cnA0006634923-pop8013111940.htmlsearch_idsdw233vyv434
vivo X30 6400 万拍照手机 50mm 专业人像镜头 智慧旗舰新品手机 全网通 5G 手机(曜石)
//gfs17.gomein.net.cn/T1T9LCBQLv1RCvBVdK_800_pc.jpg
2766.0 3 100

2.0.3 具体代码实现
（1）创建一个 IndexFiles 类，它有几个参数，store 是索引文件所存放的位置，analyzer 用来对文档进行词法分析和语言处理的。三个网站爬下来的内容有细微区别，所以分别调用 indexDocsdd，indexDocsguomei，indexDocssn 三个函数建立 index。

```
class IndexFiles(object):
    def __init__(self, root1, root2,root3, storeDir):
        if not os.path.exists(storeDir):
            os.mkdir(storeDir)
        store = SimpleFSDirectory(Paths.get(storeDir))
        analyzer = WhitespaceAnalyzer()#在jieba分词之后使用空格分词器
        analyzer = LimitTokenCountAnalyzer(analyzer, 1048576)
        config = IndexWriterConfig(analyzer)
        config.setOpenMode(IndexWriterConfig.OpenMode.CREATE)
        writer = IndexWriter(store, config)

        self.indexDocsdd(root1,writer)
        self.indexDocsguomei(root2,writer)
        self.indexDocssn(root3, writer)

        ticker = Ticker()
        print('commit index')
        threading.Thread(target=ticker.run).start()
        writer.commit()
        writer.close()
        ticker.tick = False
        print('done')
```

（2）以 indexDocsguomei 为例解释具体代码内容。

1. 定义 Field 的相关属性 t1 和 t2，t1 不分词并且完全储存内容，t2 分词但不完全储存内容。

```
t1 = FieldType()
t1.setStored(True)
t1.setTokenized(False)
t1.setIndexOptions(IndexOptions.NONE)

t2 = FieldType()
t2.setStored(False)
t2.setTokenized(True)
t2.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS)
```

2. 打开 guomeiindex.txt，提取相应信息。

·正常的一个商品的信息应该有七个内容，所以要舍弃不足七个内容的爬取失败的商品。同样对于苏宁和当当的 index 也要有相应的判断和调整。比如当当的一个商品的信息占用了两行，就与国美和苏宁不同。

· rank=float(pinglunshu)*0.8+float(pinglunshu)*float(haopinglv)*0.2*0.01

rank 是我们通过给评论数，和好评数赋一定的权重得到的一个排序指标。用来反应商品的评价。

·shuxing= line1[2]

text = ' '.join((jieba.cut(shuxing)))

通过 txt 可以对商品 url，图片 url，价格，评论总数，好评率直接建立索引，但是需要对第三条内容使用 jieba 分词器分词，来获取商品名称、商品属性和关键词。

```
file = open(root,'r',encoding='gb18030',errors='ignore')
for line in file.readlines():
    try:
        line1 = line.split('\t')
        if len(line1)!=7:
            continue
        else:
            img_url = 'http:'+line1[3]
            product_name = line1[2]
            html_url = line1[0]
            price=line1[4]
            pinglunshu=line1[5]
            haopinglv=line1[6][:-1]
            rank=float(pinglunshu)*0.8+float(pinglunshu)*float(haopinglv)*0.2*0.01
            shuxing= line1[2]
            text = ' '.join((jieba.cut(shuxing)))
```

3. 创建一个 Document 代表我们要索引的文档。将不同的 Field 如商品 url、商品名称、图片 url、价格、评论等等加入到文档中。不同类型的信息用不同的 Field 来表示。IndexWriter 调用函数 addDocument 将索引写到索引文件夹中。

```
print(c,'adding',html_url)
c+=1
doc = Document()
doc.add(Field("name", product_name, t1))
doc.add(Field("url",html_url,t1))
doc.add(Field("img_url",img_url,t1))
doc.add(Field("price",price,t1))
doc.add(Field("pinglunshu",pinglunshu,t1))
doc.add(Field("haopinglv",haopinglv,t1))
doc.add(Field("rank",rank,t1))

if len(text) > 0:
    doc.add(Field("contents", text, t2))
else:
    print("warning: no content in %s" % filename)
writer.addDocument(doc)
```

4.运行类的初始化函数，建立索引，将结果存至“index”文件夹中。

```
IndexFiles('dangdangindex.txt','guomeiindex.txt','snindex.txt',"index")#建立索引
```

5.辅助函数：提取价格、评论数等数字信息时，除去无关字符，只保留数字和小数点。

```
def is_digit(n):
    return n in "0123456789."

def ch(m):
    a=filter(is_digit,list(m))
    a=''.join(list(a))
    return a
```

2.0.4 实验问题分析

建立索引遇到的问题，主要是三个 index.txt 实际上有所区别，比如存储格式不同，当当的一个商品占用两行，其他的占用一行。或者对于好评率属性，当当和国美是没乘 0.01 的结果，比如好评率为 100，而苏宁则是已经乘了 0.01 的结果，如 0.98。对于这些细节要十分注意，不然会导致可笑的错误。另外也要注意舍弃信息有误的商品，比如属性条目不全的，信息将发生错位。

2.1 图片索引的建立

2.1.1 图片索引建立原理概述

由于本次大作业需要实现以图搜图的功能，因此除了要对爬取到的商品信息建立索引之外，还需要对每个商品对应的图片信息建立索引，将商品的图片信息存储到 lucne 建立的索引中。由于图片不能直接存入索引，因此我们决定使用本学期所学的图像的特征提取的一些方法，将图片信息转换为能够存入 lucene 索引中的类型来建立图片索引。如果只提取单一的图像特征可能会使搜索的结果不准确，而 sift 提取特征又很慢，因此我们选择了在 LSH 课程中学习的图像特征提取方法，同时对图像的颜色直方向量特征和灰度直方向量特征进行提取，将两个向量合并成为一个 Hamming 码并将这个 Hamming 码存入索引。当搜索时，只需要对想要搜索的目标图片调用同样的计算函数，得到其 Hamming 码，就可以对其在事先建立好的图像索引中进行搜索了。

2.1.2 具体代码实现

首先也是最重要的就是计算图片特征向量进而得到 Hamming 码的函数。这里我们定义了三个函数，分别用于提取图片的色彩直方特征向量，提取图片的灰度直方特征向量和通过直方向量计算图像的 Hamming 码。

（1）计算色彩梯度向量

将整个图像平分为左上，右上，左下，右下四个区域，分别计算每个区域内蓝色，绿色和红色 BGR 三种颜色所占的比例：

```
h,w,c = img.shape
for k1 in range(2):
    for k2 in range(2):
        sum_blue = sum_green = sum_red = 0
        for i in range(int(k1*h/2),int((k1+1)*h/2)):
            for j in range(int(k2*w/2),int((k2+1)*w/2)):
                sum_blue += img[i,j][0]
                sum_green += img[i,j][1]
                sum_red += img[i,j][2]
        total_energy = sum_blue + sum_green + sum_red
        blue_rate = float(sum_blue)/float(total_energy)
        green_rate = float(sum_green)/float(total_energy)
        red_rate = float(sum_red)/float(total_energy)
```

将计算出的 4 个区域内共 12 个数字加入到一个 list 中，在加入之前将每个比例乘

10 倍得到 0-9 中的一个数字，最终得到的 12 维度向量 p 就是我们从图像中提取出的颜色直方特征向量：

```
        p.append(blue_rate)
        p.append(green_rate)
        p.append(red_rate)
for i in range(12):
    p[i] = int(10*p[i])
```

（2）计算灰度梯度向量：

我们爬取的商品图片其中很多都是白底色的，也就是说图像中会有很大一部分的灰度都是 255。为了使每张图片的灰度向量具有其特殊性，我们同样将其设置为一个 12 维向量，并在计算这个向量时使用较为特殊的处理。

首先用两重 for 循环遍历整个图像矩阵，由于灰度范围是 0-255，而我们要将其压缩为 12 维向量，因此我们对每个像素的灰度值除以 22 再进行下取整，这样就得到的范围就是 0-11，共 12 个维度。每计算出一个像素的灰度值，就把事先设置好初始值全部为 0 的 12 维 list 中对应的位置对应的数值加一。

```
g = [0 for i in range(0,12)]
h,w = img.shape
for i in range(h):
    for j in range(w):
        id = img[i,j]
        g[int(id/22)] += 1
```

由于我们想通过图像含有的灰度的比例来表征图像，因此我们还需要计算 12 维灰度向量各个元素的和（后面用 sum 代指），再将向量中每个元素的值替换为其除以这个和得到的比例。但是有时图像的灰度会比较集中在某一个范围，这是因为图像大多为白底色且白底色所占面积较大，所以其它灰度除以 sum 时大多情况为 0。因此为了更好的表现图像的灰度特征，也就是非白底色的部分，我们在对向量进行乘 10 处理时（为了得到 0-9 之间的整数，用于计算 Hamming 码），如果发现向量的某个元素所占 sum 的比例小于 0.1，则改为将其乘 100。这样能更好地体现图像非白底部分的灰度特征：

```
total = sum(g)
for i in range(0,len(g)):
    if(g[i]/total<0.1):
        g[i] = int(g[i]/total*100)
    else:
        g[i] = int(g[i]/total*10)
```

（3）得到图像的特征 Hamming 码：

传入该函数的参数是一个向量，这个函数的功能是将向量转换为一段编码。具体就是将向量中 0-9 的每个数转换成对应的 4 位二进制数，数据类型位字符串，再将向量中每个数字得到的长度为 4 的字符串相加在一起得到整个的字符串。

定义了提取图像特征的方法就可以开始建立图像索引了。

（4）图像索引的建立

爬虫从苏宁，国美和当当爬取下来的商品信息分别存在三个 txt 文档中。我们对这三个文档每一个分别设立一个 index 类中建立索引的方法，这里以苏宁的为例。

由于我们再爬取的时候并没有直接将图片存入本地，而只在本地存储了商品图片对应的地址，因此我们决定在建立索引时每遍历到存在本地的一个商品信息，就先将其图片 url 对应的图片下载到本地，对这张图片使用之前定义的色彩直方向量计算函数，灰度直方向量计算函数以及 Hamming 码计算函数得到图片的特征值，将其与商品的名称，网页 url，商品图片 url 存入索引。最后在存入一个图片之后将其从文件夹中删除，以便进行下一个图片的存入。

打开 txt 文档并提取商品信息：

```
id_file = open(root,'r')
length = len(id_file.readlines())
id_file.close()
id_file = open(root,'r')
for i in range(int(length)):
    try:
        line1 = id_file.readline().split('\t')
        if len(line1)!=8:
            continue
        img_url = line1[4]#图片地址
        product_name = line1[2]#商品名称
        html_url = line1[0]#商品的html的url
```

下载图片：

```
urllib.request.urlretrieve(img_url,'pic/1.jpg')
```

提取图片特征码：

```
img1 = cv2.imread('pic/1.jpg',cv2.IMREAD_COLOR)
img2 = cv2.imread('pic/1.jpg',cv2.IMREAD_GRAYSCALE)
vector_p = get_p(img1)
vector_g = get_gray(img2)
hamming_code1 = get_Hamming(vector_p)
hamming_code2 = get_Hamming(vector_g)
hamming_code = hamming_code1 + hamming_code2
```

将信息加入索引并删除图片：

```
doc = Document()
doc.add(Field('name',product_name,t1))
doc.add(Field('imgurl',img_url,t1))
doc.add(Field('htmlurl',html_url,t1))
doc.add(Field('hamming',hamming_code,t2))
writer.addDocument(doc)
os.remove('pic/1.jpg')#删除图片
```

2.1.3 待改进的地方

由于对于图片特征码的计算方式比较特殊，也就是对搜索结果和目标图片匹配的要求较为严格，因此搜索时的结果虽然比较准确，但是返回结果较少。由于考虑到使用 sift 特征提取的方法计算较慢，所以最终我们还是选择了使用更加简便的图像颜色和灰度特征提取。但是通常这两种特征并不能反映图像的全貌，更加准确且高效的图像搜索应该是使用机器学习训练的模型来提取图像的向量并对其进行索引。但是我们尝试后发现机器学习模型的结果与 lucene 的搜索并不是很符合，因为 lucene 是搜索是在索引中寻找和输入的目标值完全匹配的数据，而之前课程中用到的模型是通过计算向量距离来判断图像匹配程度的。希望在下次进行类似的项目时能够尝试使用机器学习模型来进行图像特征值提取。

3.搜索

3.0 文字搜索

3.0.1 文字搜索原理概述

lucene 的文字搜索是将输入的目标信息在经过选用的分词器进行分词后，在事先建立好的倒排索引表里面进行搜索，并将搜索结果按照 lucene 自带的计算相关度，即搜索返回结果的得分的方法将搜索结果降序排序。

3.0.2 文字搜索具体实现

由于大多是 lucene 自带的功能，故文字搜索的实现较为简单，这里简略说明。对于输入的文字还是先进行 jieba 分词，再将分词的结果用空格分词器进行分词。为了实现通过价格和商品评价进行排序，对于搜索的返回结果我们不直接输出，而是将其存入一个 list 中，同时分别以不排序（即默认返回顺序，lucene 中的搜素结果评分降序），价格升序，价格降序和评价降序调用 sort（）方法得到 4 个 list，并按照网页前端用户的点击交互决定展示出哪个 list（默认展示的是相关度排序）。

```
return_list=[]
for i, scoreDoc in enumerate(scoreDocs):
    doc = searcher.doc(scoreDoc.doc)
    if float(doc.get("price"))<0:
        continue
    return_list.append((doc.get("name"),doc.get("url"),doc.get("img_url"),doc.get("price"),doc.get("pinglunshu"),doc.get("haopinglv")
    doc.get("rank") ))
    # 列表·
return_to_sort = return_list[:]
return_to_sortbyrank = return_list[:]
return_to_sort_down = return_list[:]

return_to_sort.sort(key=price)#按照商品价格升序排序
return_to_sort_down.sort(key=price,reverse=True)#按照商品评价降序排序
return_to_sortbyrank.sort(key=rank,reverse=True)#按照商品价格降序排序
return return_list,return_to_sort,return_to_sortbyrank,return_to_sort_down
```

3.1 以图搜图

3.1.1 以图搜图原理概述

将之前建立图像索引部分提及的三个最终能得到图像特征码的函数在以图搜图时同样用来计算目标图像的特征码，之后使用 lucene 自带的搜索功能进行特征码的匹配，并将搜索到的返回结果展示在网页前端。

3.1.2 以图搜图具体实现

html 的 input 标签中有一个"type"，为了能上传图片，我们将以图搜图的 input 标签的 type 设置为 file。在上传文件时，实际相当与我们输入了文件名称。

通过 flask 框架，我们将 zhu.html（主页）中得到的这个文件名称传参给 im.html，也就时图像搜索的返回结果页（Flask 以及搜索结果页面的排版等实现在报告的第四部分），也就是 app.py 中的 pic_results 函数。由于在进行搜索之前我们已经将待搜索的目标图片都存放在了一个叫做 pic 的文件夹中，现在已经得到了具体的文件名，我们就可以得到待搜索图片的路径，并通过 opencv 的 imread 将其读入并进行处理：

```
<label>商品信息</label>
<input type="file" name="keyword">
<input type="submit" name="">
```

```
#分别以灰度和色彩形式读入图片
img1 = cv2.imread('pic/'+str(keyword),cv2.IMREAD_COLOR)
img2 = cv2.imread('pic/'+str(keyword),cv2.IMREAD_GRAYSCALE)
```

之后就是计算 Hamming 码，并通过 Hamming 码在已经建立好的图像索引中搜索目标：

```
#得到搜索图片的Hamming码
vector_p = get_p(img1)
vector_g = get_gray(img2)
hamming_code1 = get_Hamming(vector_p)
hamming_code2 = get_Hamming(vector_g)
hamming_code = hamming_code1 + hamming_code2
```

最后再将搜索返回结果传递给 pic_results.html 页面，就可以在前端进行搜索结果的展

示了。

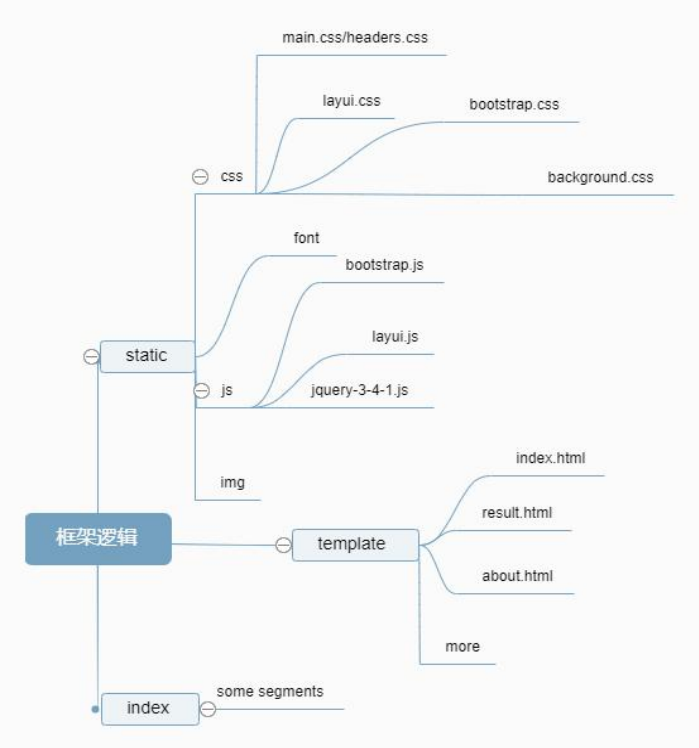
5. 网页部分

4.0 flask 框架

4.0.1.flask 框架与内容概述

根据本学期学习的 web 搭建的知识，我们决定采用 flask 这一 python 的 web 框架，flask 框架整体偏向于后端，它通过装饰器定位到 URL，并启用端口进行调试。

整体 flask 逻辑如下图所示：



我们需要在 app.py 中定义不同的路由与视图函数。在最终，实现创建程序实例，在使用 WSGI 协议的基础上启用 web 服务器，把接受端请求转交程序对象处理。

同时注意到，开发的搜索引擎需要有一定的可维护性，写出有序的框架逻辑有助于我们对 web 进行后续的维护与修改。

本报告部分主要阐释对 result 结果返回页面的排序处理和对 layui 模板渲染的简单分析。

4.1 网站主页

4.1.1 主页前端实现

主页主要由三个部分组成：左边的折叠面板，展示商品分类。右上方的搜索框和右下方的背景图片。

1. 折叠面板：

折叠面板主要使用 layui-v2.5.6 的 css 和 js 代码逻辑块进行渲染。

```
<style type="text/css">
        /* 左边栏 */
        .sidebar{
            float: left;/*侧边栏居左，改为right可令侧边栏居右*/
            top:60px;
            bottom: 0px;
            width: 250px;
        }
```

```
<!-- 左栏 -->
<div class="content">
    <div class="sidebar">

    <div class="layui-collapse">

            <div class="layui-colla-item">
                <h2 class='layui-colla-title'>
                    <a href="http://127.0.0.1:8081/result2?keyword=手机&Search=提交">手机</a> </h2>
                <div class='layui-colla-content layui-show'>
                <a href="http://127.0.0.1:8081/result2?keyword=华为手机&Search=提交">华为</a>  
                <a href="http://127.0.0.1:8081/result2?keyword=苹果手机&Search=提交">苹果</a>  
                <a href="http://127.0.0.1:8081/result2?keyword=小米手机&Search=提交">小米</a> 
                <a href="http://127.0.0.1:8081/result2?keyword=vivo手机&Search=提交">vivo</a>  
                <a href="http://127.0.0.1:8081/result2?keyword=三星手机&Search=提交">三星</a>

                </div>
            </div>
```

```
<script>
//注意：折叠面板 依赖 element 模块，否则无法进行功能性操作
layui.use('element', function(){
  var element = layui.element;

  //…
});
</script>
```

·class sidebar 将折叠面板定位到网页左部，使用 layui 框架， 包裹的是一个折叠条目，<h2 class='layui-colla-title'></h2> 包裹的是条目标题， 包裹的是条目内容。此外，折叠面板可以嵌套使用。

·每一个词条通过超链接，通过 keyword 传参，链接到具体类别的页面，即 result2.html。

2. 背景图片：

通过调节各种参数，将背景图片定位在想要的位置。

```
<div class="layui-body">

  <style>
    body{
        background-image: url("../a/shop1.png");
        background-repeat: no-repeat;
        background-attachment:fixed;
        background-position:70% 100%;
        background-size:55%;
        background-origin:content-box;
    }

    </style>
```

4.1.2 成果展示

4.2 网站商品品牌或类别过滤页

result2 页面由搜索框和具体商品搜索结果组成，并有一个返回主页的功能。

4.2.1 具体商品搜索结果：

· 借鉴之前 lab7 的内容，通过规定主 div 和每个子 div 的 width 大小，实现在一行输出 5 个搜索结果。

```
#big {

width:100%;

}

#small {

float: left;
width: 20%;
box-sizing: border-box;
padding: 10px;
min-width: 150px;

}
```

·container 容器使网页左右留白。

```
#container {

padding-left:50px;
padding-right:100px;

}
```

·通过循环语句，将商品的图片、名称、价格、评论数等显示出来。

```
<div id="container">
        <div id="big">
        {% for line in result[0] %}

            <div id="small">
                <img src={{line[2]}} alt="texthhh" width="300" height="300" >  <br>
                <a href={{line[1]}} style="color:rgb(0, 102, 255)">{{line[0]}}.....</a><br>
                实际价格：{{line[3]}}<br>评论数：{{line[4]}}<br>好评率：{{line[5]}}<br></td>

                <br><br><br>

            </div>

        {% endfor %}
        </div>

</div>
```

2.通过带链接的图片，可以返回主页。

```
<a href='/zhu'><img src="static/img/home1.jpg"  width="100" height="100" ></a>
```

4.2.2 成果展示

4.3 搜索结果返回页

4.3.1 基本操作实现

对于结果页面，我们采取了 layui-v2.5.6 的 css 和 js 代码逻辑块进行渲染。

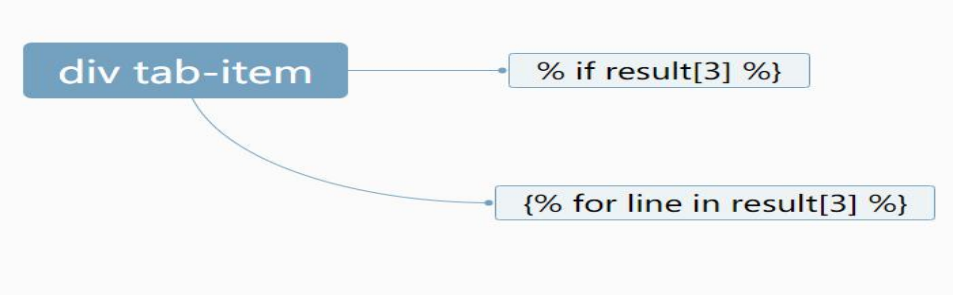
1.Headers 头部文件：

基本的 css 渲染调用 layui.css 与额外补充的 headers.css 与素材网站的 common.css 文件。然后因为主要的分页布局仍然需要渲染，所以在 style 样式表单中继续补充。

2.Body 主体部分：

用 div 标签与 box 模型。主要分为三部分，首先是头部包裹，利用窗口滑动的元素样式，右上角设置首页与小组信息相关，并充分利用 ul，li 标签来美化页面。然后运用语句 div class=" container"，设计 item 容器标签。

参照大多数搜索引擎的样式，为满足用户多次搜索，设计相关搜索框，这里只说明文字部分的搜索框制作，图像搜索部分在后文中进行说明。同时，利用 item-select 属性，选择采用 li 标签的形式并绑定 layui.js 的点击事件。其中每个 li 标签下的内容每次先进行判断是否为空后再决定是否要输出在前端，进行判断是否输出的逻辑图像如下所示：



当判定为可输出形式的时候，我们采用 layui-tab-item 的 style 样式表渲染。因为分类展示中最后的部分数据缺失但仍要在前端展示其内容，选择在前端显示 noinput 的特定字符串。然后每个小 item 部分利用 flask 的数据传递方式与语法，循环展示结果，

接下来进行编写 JavaScript 代码。首先定义 layui 的使用，随后进行设置弹层与元素的操作。为了测试是否成功加载，用 layui.msg 在前端先显示 hello world，这可以使用户明确自己已经成功切换了 li 标签，所以设置元素监听 tab 切换函数并展示切换了几次。最后实现隐藏底部信息。

代码显示如下所示：

```
1.  //监听 Tab 切换
2.  lement.on('tab(demo)', function(data){
3.   layer.tips('切换了 '+ data.index +':  '+ this.innerHTML, this, {
4.     tips: 1 }););
```

4.3.2 前端结果展示实现

其中展示结果的部分，我们设计并展示了 lucene 相关度排序的默认排序结果，并按照商品的价格升序和降序进行一定程度上的排序，并在结果页上进行最大数目为 50 的显示。事

实上我们发现，对于不同价格区间的商品，按照 lucene 处理后价格升降序，显示出的商品的符合度却不尽相同。比如在搜索电脑这一商品时，价格升序展示的更多是电脑配件，而降序却更符合人们搜索这一商品的本意，当然，搜索结果已经建立在有相关度基础上，因此偏差范围可以接受，也符合一般的商品网页搜索结果的显示。

同时我们设计并展示了好评度排序，这里采用的是我们在前文中提到的 rank 这一变量(rank 好评度原理在建立索引时已经阐述)。在展示的页面，我们可以看到，rank 值与评论数正相关，并且这一 rank 更多的显示了人们购买此商品的意愿，符合一定范围内的从众心理。

在制作结果展示网页时，我们运用了成型的模板进行渲染，让网站的实现更加美观。对于最重要的商品结果的展示页面上，我们定义了一个<table>的标签，将从索引文件中获取到的名称、连接、商品价格、好评数和好评率等信息都存放在了表格中，形成一个成体系的较为规整的展示结构，并将表格的 border 参数调整为 0，使得在不显示表格的情况下显示出整齐的文字形式。

```
<table class="table table-striped table-hover" width="100%" border="0"
        cellpadding="0"style="table-layout:fixed;word-wrap:break-word;word-break;break-all;">
```

为了实现用户的多次搜索，我们在上方保留了搜索框，将其调整至页面居中，方便用户进行二次的输入。

显示中最重要的部分是实现用户对于按照何种方式进行排序的选择。将四个排序方式的元素（按默认排序、按价格正序、按价格逆序、按 rank 值排序）放入一个元组中。设置标签使得用户在单击想要采用的排序方式时，调用相应的排序进行输出。

在直接显示的搜索结果界面上，我们直接对商品的默认排序进行输出。这样有利于狡猾和分析搜索结果。

```
<div class="layui-tab-item layui-show" id="data">
```

在搜索结果页到商品页的页面跳转上，我们直接采用了页面直接跳转到对应商品页的方式，而针对不同的用户习惯的分析，我们并没有采用直接创建新标签的方式打开网页。同时，在用户位于浏览器上实现“退回”按键的时候，我们仍可以退回到上一个标签的网页，这个网页将仍为用户前一次搜索时的网页。

最后是底部信息展示，展示最终项目时间，展示小组信息，效果类似于“about”中的内容，主要使用 layui 的渲染，让网页整体呈现简练。

下面主要分析 headers.css 文件的内容，网站头部设计成浮动的，对于每一个包裹的 div 盒子的内容，设置左右与距离顶部距离底部的内边距。对于搜索框的边框渲染，使其颜色明显淡化。设置 ul/li 标签的边框属性与内部字体大小颜色样式。对于网页主要的 container 部分采用对 item 标签逐一调整高度。最后设置底部背景，选择常规的模式。

利用详情页的框架，在关于页面将小组成员信息展示。有了详情页的整体框架，实现起来毫不费力。

对于网页结果搜索和排序的路由注册和视图函数的分析如下：在定义 result 和 imagesearchresult 的视图函数中，调用搜索函数，为使得搜索引擎方便维护，因此单独编写搜索函数的脚本文件之后直接调用。以搜索文字为例，将 keyword 传参脚本，在程序的主要部分进行搜索，利用 jieba 分词器，注意，由于更多是分词器接受一个字符串作为输入，将这个字符串拆分成独立的词或语汇单元（token 可能会丢弃一些标点符号等字符），然后输出一个语汇单元流（token stream），因此 lucene 采用空格分词器。

在索引的基础上，将我们想展示的商品名称，地址，图片（图片地址链接），价格，以及评论数好评率与自定义的 rank 值。因为将搜索结果以类对象的形式保存到 data 列表中，

所以我们可以很好地利用 python 内置的 sort 函数轻松地对我们的搜索结果进行排序再把 data 列表返回给前端即可，对于 python 的 sort 函数，我们可以设置排序的参照 key，和升序 reverse=False 降序 reverse=True，这个排序的 key 需要自定义，这里返回 float 值。

```
def price(elem):
    return float(elem[3])
```

注意，我们并不能直接对返回的 data 进行排序，在 python 中列表的浅拷贝和深拷贝不同，因此我们需要先将 data 的地址传入新的地址，值不变传入新的列表

```
return_list=[]
for i, scoreDoc in enumerate(scoreDocs):
    doc = searcher.doc(scoreDoc.doc)
    if float(doc.get("price"))<0:
        continue
    return_list.append((doc.get("name"),doc.get("url"),doc.get("img_url"),
        doc.get("price"),doc.get("pinglunshu"),doc.get("haopinglv"), doc.get("rank") ))
    # 列表·
return_to_sort = return_list[:]
return_to_sort.sort(key=price)
```

以同样的形式我们定义了多个排序函数以备使用，并利用 return 返回一个元组，这样之后对元组的每个 value 进行判断是否为空就可以进行其他的排序

```
return return_list,return_to_sort,return_to_sortbyrank,return_to_sort_down
{%if result[0]%} <!--对于其他的判断，同理-->
```

最后，我们在 app.py 中对我们的搜索稍微美化封装一下，并及时判断是否需要初始化并绑定线程，在搜索后及时销毁线程来保证搜索的进一步进行。

```
result = mySearchFilenotpic.run(searcher,analyzer,keyword)
del searcher
vm_env.detachCurrentThread()#销毁线程
```

4.3.3 web 成果展示

搜索华为手机：

4.3.4 实验问题与改进

1、 对于网页的静态文件路径，以相对路经导入时必须注意层级关系，否则调试时会出现 not found 的错误，错误代码 404；当成功启动和成功展示结果后，我们在本地 terminal 终端可以很明显的看到状态码 200

2、网页设计成自由拉伸的样式方便元素填充

3、整体的运行效率尚为可观，但对于图片检索，因为需调用 form 的 filetype 和向后端进行图片检索，效率有待提高，后文叙述。

4、基于 python 的 web 后端开发主流模型框架有 Django 和 Flask 等，相对来说，我们所采用的 flask 框架用于我们的轻量级搜索引擎已经足够，并且有很大的想象空间。当然，对于不同的服务，有一定规模时，选取 django 也不失为好方法。

5、机器可读性强的静态语言，更有利于各种自动化的支持。而我们所采用的动态类型语言更方便上手和简单操作。

6、Java 中更侧重提到 MVC,而在 Python 中则是 MTV。

7、Solr 和 Lucene 不同：Lucene 本质上是搜索库，不是独立的应用程序，而 Solr 是。Lucene 专注于搜索底层的建设，而 Solr 专注于企业应用。Lucene 不负责支撑搜索服务所必须的管理，而 Solr 负责。Solr: Solr 是 Lucene 面向企业搜索应用的扩展。因此对于简单的搜索和返回，我们采用 Lucene 足够。