电工导 C 第五次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 10 月 20 日

1　实验概览

本次实验在 Lab4 的基础上，对增添了搜索引擎的功能，使得其能够完成指定 site 搜索和图片搜索。

对于指定 site 搜索（也即组合查询），一般指查询两个或更多条件的查询方式，如指定域名，指定文件格式等。对于`lucene` 已经给出了相应的解决方案，我们只需要在建立查询时直接添加 site 的内容即可。

对于图片搜索，各大搜索引擎提供的方式并不一致。一般地，搜索网页标题和图片附近的文本内容可以做到较好地解决实验问题。

基于上述内容，本次实验需要进行以下两个任务：

- 在 Lab4 的基础上实现指定域名搜索功能。
- 爬取某一个特定的图片网站并将图片的描述文字作为索引从而实现一个图片搜索引擎。

2　实验环境

本次实验采用所需的实验环境如下：

- Docker 中的`sjtumic/ee208` 镜像
- Python3（使用 VSCode 编译）
- `BeautifulSoup4` 扩展以及`lxml` 扩展。
- `java` 环境及`lucene` 扩展（在 SJTU EE208 中已经给出）。

3　练习 1

3.1　问题重述与代码说明

练习 1 要求我们对 Lab4 的代码中加入调整网页的功能，使得其能够实现对网页的查询。我们只需要在查询的`contents` 以外添加一个`site` 的内容即可。相关代码如下：

```
1    doc.add(Field(”site”, url, t2))
```

同时，根据本次实验中已经给出的代码，我们需要对查询的代码进行一定的改动，对应代码如下：

```
    command_dict = parseCommand(command)
    querys = BooleanQuery.Builder()
    for k,v in command_dict.items():
        query = QueryParser(k, analyzer).parse(v)
        querys.add(query, BooleanClause.Occur.MUST)
    scoreDocs = searcher.search(querys.build(), 50).scoreDocs
```

其中parseCommand() 只需按照已给出代码复制并将查询改动为site 即可。对应的行代码如下：

```
    allowed_opt = ['site'] # allolwed_opt = ['title', 'author', 'language']
```

3.2 运行结果

```
Hit enter with no input to quit.
Query:上海交大 site:163.com

Searching for: 上海交大 site:163.com
50 total matching documents.
path: html/httpswww.163.comeduarticleHAHLEB3M00297VGM.html.txt
name: httpswww.163.comeduarticleHAHLEB3M00297VGM.html.txt
url: https://www.163.com/edu/article/HAHLEB3M00297VGM.html
title: C9、E9、华五、中九、五院四系、二龙四虎……这些黑话你懂吗？|志愿填报|大学_网易教育
------
path: html/httpswww.163.comeduarticleH9DILVH200297VGM.html.txt
name: httpswww.163.comeduarticleH9DILVH200297VGM.html.txt
url: https://www.163.com/edu/article/H9DILVH200297VGM.html
title: 清华调整强基计划方案：初试分省计算机考试，复试分区域|清华大学_网易教育
------
path: html/httpswww.163.comnewsarticleHASE7HSN00018AOQ.html.txt
name: httpswww.163.comnewsarticleHASE7HSN00018AOQ.html.txt
url: https://www.163.com/news/article/HASE7HSN00018AOQ.html
title: 谁是钱七虎？“消失”16年“修长城”，还在珠海搞出“天下第一爆”！|院士|教授|力学|中国工程院_网易新闻
------
path: html/httpswww.163.comdyarticleH89OK1H20530WJIN.html.txt
name: httpswww.163.comdyarticleH89OK1H20530WJIN.html.txt
url: https://www.163.com/dy/article/H89OK1H20530WJIN.html
Searching for: 上海 site:sina.cn
32 total matching documents.
path: html/httpssina.cn.txt
name: httpssina.cn.txt
url: https://sina.cn
title: 手机新浪网
------
path: html/httpssina.cnfrompage.txt
name: httpssina.cnfrompage.txt
url: https://sina.cn/?from=page
title: 手机新浪网
------
path: html/httpssina.cnreloadsina.txt
name: httpssina.cnreloadsina.txt
url: https://sina.cn/?reload=sina
title: 手机新浪网
------
path: html/httpssina.cnvt4.txt
name: httpssina.cnvt4.txt
url: https://sina.cn?vt=4
title: 手机新浪网
------
```

3.3 问题讨论

3.3.1 SearchFiles.py 的改动有何意义

通过查询query 的运行状态，我们发现，如果SearchFiles.py 不做改动，程序也可以正常运行且大体是按照域名分布的。但是如果不改动的话，不会完全因为不存在这一站点而不显示该结果，仅仅是调整了一下输出结果的优先级，这对于我们是不可容忍的。因此我们需要保证二者的优先级一致，即如果搜索的内容不在这个站点内，即使在其他站点出现，也不予以显示。

3.3.2 程序是如何对站点自动区分的

如果对于原本的程序，其实lucene 也可以自动识别内容，然而这样会出现前面的问题。设计parseCommand() 函数的目的是为了让各个部分各司其职，最后我们取交集即可。

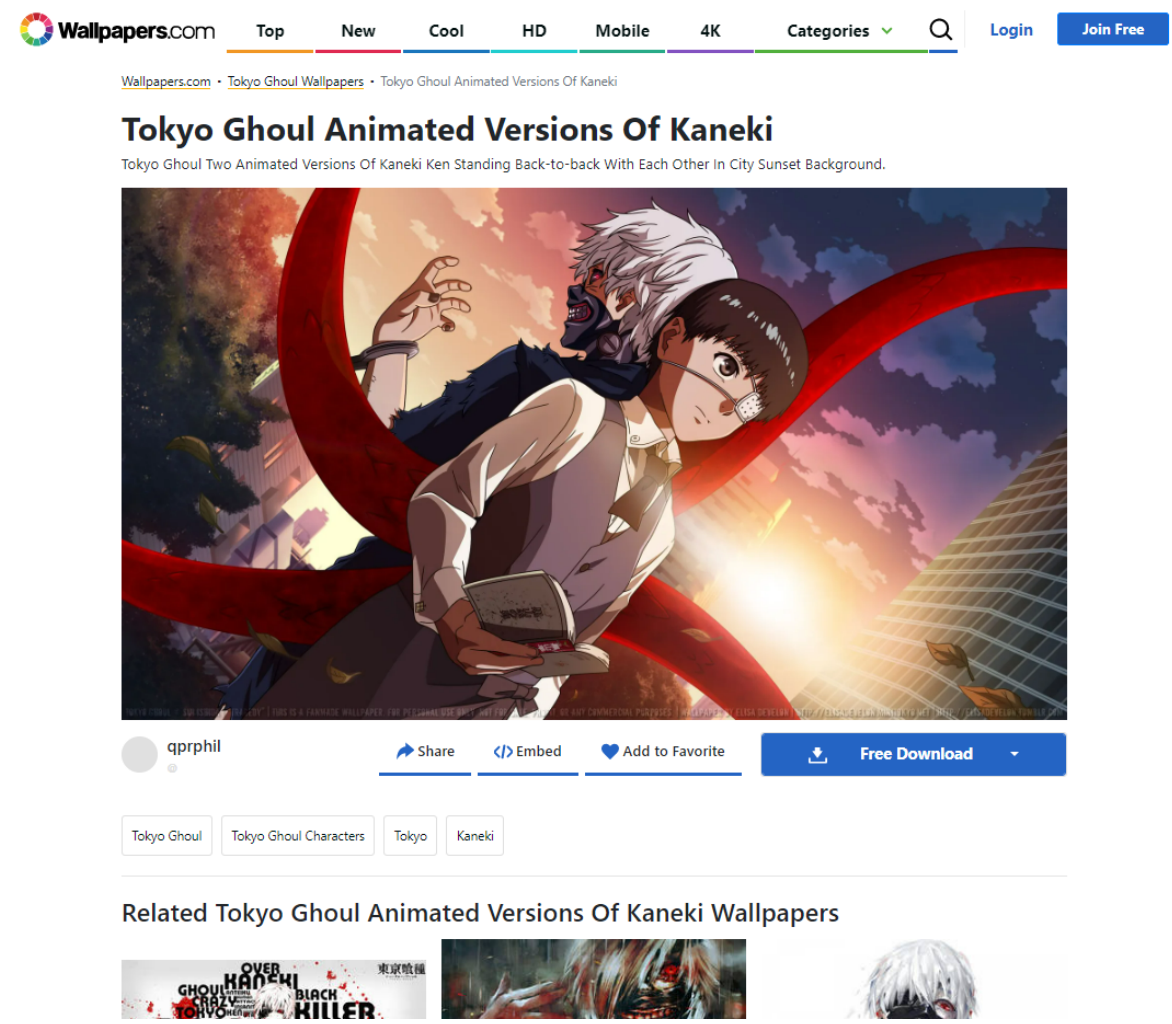
4 练习 2

4.1 问题重述与代码说明

我们在练习 2 中需要爬取一个图片网站，根据对应的文字信息返回图片内容。

我们选择了www.wallpapers.com 作为我们爬取图片的对象。www.wallpapers.com 是一个关于图片壁纸的网站，里面包含很多高清的地址。

我们首先设计对图片进行爬取的程序。观察该网站某张图片如下：



可以看到某张图片的网页分布为：标题 -内容 -图片 -标签 -推荐，因此我们只需要利用 Lab2 中的 tree 以及直接查找图片对应标签即可。

爬取链接的代码如下：

```
ef get_all_links(content, page):
    # 如果爬不到会返回None
    if content == None:
        return []
    links = []
    soup = BeautifulSoup(content, features="lxml")
    for link in soup.findAll('li',{'class' : 'card content'}):
        try:
            mylink = link.find("a").get("href","")

            if mylink[0] == "/":
                mylink = urllib.parse.urljoin(page,mylink) #连接链接
                links.append(mylink)
            else:
                links.append(mylink)
        except:
            print("Fail!")
    return links
```

爬取文档和图片的代码如下：

```
def add_page_to_folder(page, content):  # 将网页存到文件夹里，将网址和对应的文件名写
                                          入index.中txt
    index_filename = 'index.txt'  # index.中每行是txt网址' 对应的文件名'
    index = open(index_filename, 'a')
    filename = valid_filename(page)
    tree = etree.HTML(content)
# The rest is done by you:
    title = tree.xpath("/html/body/div[4]/main/div[1]/div/div[1]/h1")[0].text
    intro = tree.xpath("/html/body/div[4]/main/div[1]/div/div[1]/p[2]/text()")[0]
    pagesite = "https://wallpapers.com"+tree.xpath("/html/body/div[4]/main/div[1]/div/
                                                    div[2]/div[1]/div[1]/picture/source/
                                                    @srcset")[0]
    index.write(page.encode('ascii', 'ignore').decode() + "\t" + filename + "\t" +
                                                    pagesite + "\t" + title + "\t" + intro +
                                                     "\n")
    index.close()
```

应当注意的是，由于该网站的主页和图片页有所不同，因此我们的根文件应当是某一图片页。笔者采用的是东京喰种的某张图片（也即展示用图片）。

在此基础上，我们需要针对某一个图片进行文件索引的提取。直接将index.txt 各行分到各文件即可。

对于 Index 的建立和搜索引擎的建立，与 lab4 基本一致。此处不再重复。

4.2 运行结果

```
Hit enter with no input to quit.
Query:Pikachu
Searching for: Pikachu
8 total matching documents.
imgurl: https://wallpapers.com/images/hd/cute-pikachu-pokemon-ynbgo9vzeaemo126.webp
url: https://wallpapers.com/wallpapers/cute-pikachu-pokemon-ynbgo9vzeaemo126.html
title: Cute Pikachu Pokemon
--------
imgurl: https://wallpapers.com/images/high/ash-and-pikachu-pokemon-f55b1pbpq5kdu96b.webp
url: https://wallpapers.com/wallpapers/ash-and-pikachu-pokemon-f55b1pbpq5kdu96b.html
title: Ash And Pikachu Pokemon
--------
imgurl: https://wallpapers.com/images/hd/pikachu-with-pokemon-friends-uxxa8pvbvsqj0v3h.webp
url: https://wallpapers.com/wallpapers/pikachu-with-pokemon-friends-uxxa8pvbvsqj0v3h.html
title: Pikachu With Pokemon Friends
--------
imgurl: https://wallpapers.com/images/hd/pokemon-evolution-8b1ylsvrgjozt6aa.webp
url: https://wallpapers.com/wallpapers/pokemon-evolution-8b1ylsvrgjozt6aa.html
title: Pokemon Evolution
--------
imgurl: https://wallpapers.com/images/hd/pokemon-signs-iy1iod1dyvanxuju.webp
url: https://wallpapers.com/wallpapers/pokemon-signs-iy1iod1dyvanxuju.html
title: Pokemon Signs
--------
imgurl: https://wallpapers.com/images/hd/pokemon-characters-va6139eg5csznzmw.webp
url: https://wallpapers.com/wallpapers/pokemon-characters-va6139eg5csznzmw.html
title: Pokemon Characters
--------
imgurl: https://wallpapers.com/images/high/ash-misty-and-brock-pokemon-ticw7wac0on7ig5l.webp
url: https://wallpapers.com/wallpapers/ash-misty-and-brock-pokemon-ticw7wac0on7ig5l.html
title: Ash, Misty And Brock Pokemon
--------
Hit enter with no input to quit.
Query:Tokyo Ghoul
Searching for: Tokyo Ghoul
41 total matching documents.
imgurl: https://wallpapers.com/images/high/two-sides-tokyo-ghoul-fv8x53idw7ceunlv.webp
url: https://wallpapers.com/wallpapers/two-sides-tokyo-ghoul-fv8x53idw7ceunlv.html
title: Two Sides Tokyo Ghoul
--------
imgurl: https://wallpapers.com/images/high/uta-poster-tokyo-ghoul-5au55ofs69mtd2ub.webp
url: https://wallpapers.com/wallpapers/uta-poster-tokyo-ghoul-5au55ofs69mtd2ub.html
title: Uta Poster Tokyo Ghoul
--------
imgurl: https://wallpapers.com/images/hd/aesthetic-touka-tokyo-ghoul-hd-xlrevfbba7c17v2n.webp
url: https://wallpapers.com/wallpapers/aesthetic-touka-tokyo-ghoul-hd-xlrevfbba7c17v2n.html
title: Aesthetic Touka Tokyo Ghoul Hd
--------
imgurl: https://wallpapers.com/images/hd/kaneki-and-touka-tokyo-ghoul-qabcbebjzrh5mqcw.webp
url: https://wallpapers.com/wallpapers/kaneki-and-touka-tokyo-ghoul-qabcbebjzrh5mqcw.html
title: Kaneki And Touka Tokyo Ghoul
--------
imgurl: https://wallpapers.com/images/hd/kaneki-vest-tokyo-ghoul-pq72wu55ot78hxxe.webp
url: https://wallpapers.com/wallpapers/kaneki-vest-tokyo-ghoul-pq72wu55ot78hxxe.html
title: Kaneki Vest Tokyo Ghoul
--------
imgurl: https://wallpapers.com/images/hd/scary-kaneki-tokyo-ghoul-lx1yju1bwzxukidf.webp
url: https://wallpapers.com/wallpapers/scary-kaneki-tokyo-ghoul-lx1yju1bwzxukidf.html
title: Scary Kaneki Tokyo Ghoul
--------
imgurl: https://wallpapers.com/images/hd/taunting-kaneki-tokyo-ghoul-hd-oeiamu3o7uksac3n.webp
url: https://wallpapers.com/wallpapers/taunting-kaneki-tokyo-ghoul-hd-oeiamu3o7uksac3n.html
title: Taunting Kaneki Tokyo Ghoul Hd
--------
imgurl: https://wallpapers.com/images/hd/kaneki-and-hide-tokyo-ghoul-mo8b6inbgvjy8ezm.webp
url: https://wallpapers.com/wallpapers/kaneki-and-hide-tokyo-ghoul-mo8b6inbgvjy8ezm.html
title: Kaneki And Hide Tokyo Ghoul
```

```
Hit enter with no input to quit.
Query:London
Searching for: London
3 total matching documents.
imgurl: https://wallpapers.com/images/high/london-night-cityscape-svyxpzgdis4ayl1m.webp
url: https://wallpapers.com/wallpapers/london-night-cityscape-svyxpzgdis4ayl1m.html
title: London Night Cityscape
--------
imgurl: https://wallpapers.com/images/hd/rainy-london-street-christmas-lights-gan90kfr8b8amcnf.webp
url: https://wallpapers.com/wallpapers/rainy-london-street-christmas-lights-gan90kfr8b8amcnf.html
title: Rainy London Street Christmas Lights
--------
imgurl: https://wallpapers.com/images/hd/4k-bright-city-lights-a94oba3aund01051.webp
url: https://wallpapers.com/wallpapers/4k-bright-city-lights-a94oba3aund01051.html
title: 4k Bright City Lights
--------
```

4.3 问题讨论

4.3.1 该模型的改进

该模型的最大问题在于仅爬取了图片的描述文字，但实际上描述文字仅为作者自己本人上传，与实际情况有所差异。比如一张同时包含皮卡丘、伊布、杰尼龟的照片，如果介绍中只有皮卡丘，那么伊布和杰尼龟就会被遗漏。

一些更准确的方法是搜索其推送相关内容，或是提取 tag。然而笔者在测试时发现该网站的推荐算法较为一般，容易推荐不想关内容。因此暂时维持原方案。

4.3.2 网页的图片显示问题

笔者在爬取网站时发现，该网站的图片分为`webp` 和`jpg` 两种形式。其中，`webp` 是图片在`web` 中显示的图片格式，也是我们爬取的格式。而`jpg` 格式由于网页限制，禁止对其进行爬取（爬取后显示无结果）。

5 拓展思考

真正的图片搜索引擎

我们设计的图片搜索引擎还是过于粗糙，一个正常的图片搜索引擎，分为以下几种情况：

1. Search By Metadata: 此即本次实验进行的内容，也即对图片附近文字进行搜索。缺点很明显，这样对于搜索准确性的影响非常大，结果也较差。
2. Search By Example: 此即专注于图片本身内容。通过对图片进行压缩，提取图片特征等。感知哈希碰撞是一个比较好用的方式 [1]。也可结合深度学习等场景对其进行扩充。
3. 可以对上述两种方式进行综合处理，这样能够综合二者的有点，当然也会继承二者的缺点。

参考文献

[1] Niu X, Jiao Y. An overview of perceptual hashing[J]. ACTA ELECTONICA SINICA, 2008, 36(7): 1405.