哈尔滨工业大学

<<信息检索>>

实验报告

(2021 年度春季学期)

姓名：	刘璟烁
学号：	
学院：	
教师：	

实验一 网页文本的预处理

一、实验目的

本次实验目的是对信息检索中**网页文本预处理的流程和涉及的技术**有一个全面的了解，包括抓取网页、网页正文提取、分词处理、停用词处理等环节。本次实验所要用到的知识如下：

- 基本编程能力（文本处理、网页爬取等）
- 分词、停用词处理

二、实验内容

1.网页的抓取和正文提取

任务描述：

通过爬虫工具爬取网页（至少 1000 个，其中包含附件的网页不少于 100 个，多线程实现爬虫可加分），提取网页标题和网页正文，以及网页中的附件并保存附件到本地，然后将附件名称记录在 file_name 字段中，附件必须是文本文档（txt、doc、docx、xlsx 等）而不能是图片。网页正文和网页标题可以自行定义，但一般应该是网页中你最关注的内容。例如在一般的新闻网页上，就以新闻标题为网页标题，新闻内容为网页正文，而其他诸如导航栏、广告等都是不关心的内容。为了保证可读性，网页正文中不应该包含太多 HTML 标签（如<p>、<img>等），可以通过任何方法来去除掉这些标签。将爬取下来的数据保存为 json 格式，源程序文件命名为 craw.py。

2.分词处理、去停用词处理

任务描述：

将提取的网页文本进行分词和去停用词处理，并将结果保存。分词工具推荐使用由我校社会计算与信息检索研究中心开发的语言技术平台-LTP， LTP 的 Python 封装为 pyltp，这里是参考文档。停用词表采用由我校社会计算与信息检索研究中心发布的停用词表(stop_words.txt)。最后将经过分词和去停用词后的结果保存为 json 格式。提交完整程序(segment.py 或 segment 文件夹)和处理后文件的前 10 行(preprocessed.json)。

三、实验过程及结果

1. 实验关键步骤的实现

1) 网页抓取及正文提取

该部分的功能主要依靠网页爬虫工具 urllib2(在 python3.9 版本中为 urllib.request)、网页文本提取工具 Beautifu Soup 以及 json 文本处理语言自带 json 库来进行实现，具体步骤如下：

a. 爬取足量网页的 url：

我们首先用广度优先的方式利用 urllib2 爬取足量的 url 存入对应的 url 队列，（为了调试方便同时将 url 输出到文件 urls.txt 中），具体的实现方式为建立一个 queue 队列并初始化放入初始 url(这里我们选择 http://hitgs.hit.edu.cn),随后不断从队列中取出 url 并用 beutiful soup 解析内容得到含有超链接的 href 属性部分：

```
elements = bs.find_all('a')
for elem in elements:
    if 'href' in elem.attrs:
        new_url = elem.attrs['href']  # 获取href属性来获取url
        if new_url.startswith('http') and new_url not in urls:
            urls.append(new_url)
```

判断是否为 http 链接，将满足条件者加入队列中；

后续部分将直接从队列中取出 url 进行内容爬取；

b. 爬取网页内容：

从上述部分得到的的 url 队列中不断取出 url 对其所含文本内容进行爬取，并使用 beautiful soup 进行自定义的标题和正文的提取，经过对一定量网页的 html 代码分析，这里我们选择将 html 文件中 body 部分的元素<h1>-<h6>文本内容作为标题，并将元素<p>中的文本内容作为正文内容进行提取：

```
title = bs.find('body')  # 检查是否含有body部分
if title:
    title = title.find_all(re.compile('^h[1-6]$'))  # 检查是否含有元素h1-h6
    if title:
        title = title[0].getText().strip()  # 设为标题
        paragraphs = []  # 用于正文内容
        for cont in bs.body.select('p'):
            strin = "".join(cont.text.replace('\n','').split())
            if strin != '':
                paragraphs.append(strin)  # 存储正文
```

随后我们还需要检查网页是否含有附件，首先查找给定链接元素，并判断是否为要求的文本文档(这里我们包括了 xlsx、doc、docx 和 pdf):

```
attach_urls = bs.body.select('a[href]')
for new_url in attach_urls:
    new_url = new_url.attrs['href']
    if '.xlsx' in new_url or '.doc' in new_url or '.pdf' in new_url:
        has_attch = True
```

如果发现了满足指导书要求的附件，我们对仅仅提供相对路径的链接进行补齐，及提供其首

页的地址：

```
    if not new_url.startswith('http'):
        new_url = 'http://' + url.split('/')[2] + new_url
```

最后应用 urllib 库提供的方法将附件下载到指定目录 files 中；

c. 生成给定格式的 json 结果文件：

在上述进行网页内容提取的过程中，我们将每个合格网页（即含有标题、段落的网页）对应的内容按照指导书要求的格式存储到队列中，随后调用 json 库的 dump 方法转换格式并输出到 js_out_std.json 文档中：

```
stream = [json.dumps(res, ensure_ascii=False) for res in glo_result]
with open(json_path, 'w',encoding='utf-8') as js_file:
    js_file.write('\n'.join(stream))
```

d. 补充：多线程的实现

在继承类 CrawThread 中实现 run 方法，该方法不断从全局 url 队列中取出 url 并调用上述正文爬取方法 craw_web()，直到满足爬取网页的数量条件；

2) 分词处理，去停用词处理

a. 加载停用词列表：

从 SCIR 提供的 stop_words.txt 中加载停用词到列表中：

```
def get_st_words(file_path):
    with open(file_path,'r') as stp_file:
        stop_words = set(stp_file.readlines())
    return stop_words
```

b. 读取 json 文件并逐条进行分词、去停用词处理

加载 ltp 模型，并循环便利每条提取的内容，对 title、paragraph 字段的内容进行分词、去停用词处理：

```
list = [json.loads(line) for line in input_jsf]
for ele in list:
    if ele['title'] and ele['paragraphs']:
        seged_title,seged_paras = [], []
        for token in ltp.seg([ele['title']])[0][0]:
            if token not in stop_words:
                seged_title.append(token)
        for token_lst in ltp.seg(ele['paragraphs'])[0]:
            for token in token_lst:
                if token not in stop_words:
                    seged_paras.append(token)
        result.append({"url": ele['url'], "title": seged_title, 'paragraphs':seg
```

随后将处理结果的前十条写回结果文件 preprocessed.json 即可。

2. 算法运行结果

1) 网页抓取及正文提取

根据实验指导书的要求，对于 1000+个网页（包括 100+带有附件的网页）进行爬取并按照给定格式正文提取所生成的结果 json 文件(js_out_std.json)部分截图如下：

注：同时生成了便于审查的附件(js_out.json)，此附件未按行存储，因此并非报告要求的按行存储格式，仅用于审查，部分结果截图如下：


```
{
    "url":"http://keyan.hit.edu.cn/\n",
    "title":"通知公告",
    "paragraphs":[
        "Copyright©2018andAllRightsReserved",
        "|地址：哈尔滨市南岗区一匡街2号2H栋5楼|邮编：150080|服务电话：0451-86418144"
    ],
    "file_name":[
        "files/9adbf461-b355-40d1-a19b-967bef9dae29.doc"
    ]
}
{
    "url":"http://yz.chsi.com.cn/ \n",
    "title":"中国研究生招生信息网",
    "paragraphs":[
        "研招复试北京外国语大学：“两分钟”应急方案保障远程复试",
        "教育评论用党史学习激励青年立业报国",
        "就业资讯抢抓关键冲刺期促进毕业生顺利毕业尽早就业"
    ],
    "file_name":[]
}
{
    "url":"http://ru.hit.edu.cn/\n",
    "title":"Ректор ХПУ Чжоу Юй поздравляет с Китайским Новым годом!",
    "paragraphs":[
        "Китайско-российскийсовместныйкампусбудетследоватьориентациивобучении:«Интернационали
        "ОбъединениеусилийэлитныхуниверситетовРоссиииКитаявподготовкевысококвалифицированныхка
        "ОбучениевХПУstudyathit@hit.edu.cn",
        "РаботавХПУjobs@hit.edu.cn",
        "Международноесотрудничествоiea3@hit.edu.cn"
    ],
    "file_name":[
        "files/1599015181211681.pdf"
    ]
}
```

2) 分词处理，去停用词处理

对于输出结果，我们选取对文章的正文及标题进行分词、去停用词处理的 json 文件的前十行，生成的结果文件(preprocessed.json)截图如下：

但经过检查，在抓取部分生成的 json 文件的前十行内容中 file_name 对应的属性值均为空，均没有附件。由于后续实验可能对附件有所需求，本次实验额外生成了含有附件的是个网页对用的 json 结果文件(with_file_preprocessed.json)作为备用，截图如下：

四、实验心得

1. 实验收获：

本次实验完成了对超过一千个网页文本的内容爬取，在完成实验的过程中，通过对网页的 html 文本分析对大量网页的组成结构有了一定的了解，从而能够根据其结构解析出不同类型的文本内容，比如标题、正文等；

除此之外，还学习并应用了爬虫工具、网页内容解析工具、分词工具的使用技巧，包括下载附件、抽取内容等，进而帮助日后对所需内容的高效获取及处理；

最后，通过对多线程爬虫的实现也有效的提高了网页爬取的效率，大大的减少的完成爬取所需要的时间。

2. 遇到的问题及解决方法：

- 如何分别获取网页标题和正文：

在完成实验的过程中，需要对所爬取的网页的正文和标题进行抽取，而对标题和正文的定义是模糊的，同时即使定义了正文和标题，由于 html 文本的规范问题，也无法保证能抽取到

相应的内容。

解决：通过对学校官网的通知类网页结构观察，发现通知的标题一般在 body 部分的元素<h1>-<h6>中，而正文一般在元素<p>中，因此我们直接从相应部分抽取内容；而对于不符合该规范的网页，我们直接进行舍弃；

- 如何避免多线程执行过程中对网页的重复爬取：

解决：首先在每个线程获取 url 的时，我们将从 url 队列弹出 url 的语句块周围调用对象锁进行保护，并在读取后释放，从而有效避免多线程执行过程中对网页的重复爬取。