html 文件检索系统开发文档三

精 82 张翀 2018010625

1. 增量开发说明

a) 第一次大作业中已经完成了第二次大作业的基本项：输入关键字，输出系统检索到的文档集；检索模型为布尔模型。且已制作了 GUI。

b) 第二次大作业完成了：搜索词自动纠错、搜索词自动补全、GUI 友好化、即时检索。

c) 本次大作业完成的内容：明网暗网区分，网页更新获取，流速限制，避免重抓，深度控制，优先级判断与处理，镜像网页剔除，GUI 修改。文档中以说明解决方案为主。

2. 实验环境与类库

a) 实验环境：jupyter notebook kernel python3.5

b) 新增功能用到的第三方类库：bs4、pyhanlp、selenium

c) 新增功能用到的 python 标准库：urllib、requests、re、time、heapq、hashlib

3. 功能实现

a) 爬虫综述

利用 selenium 实现对动态网页的访问，利用 BeautifulSoup 解析网页，获取新的 url。利用 md5 码采样进行网址去重，利用 pyhanlp 进行文本处理。以 it.sohu.com 和 www.elecfans.com 为起点，共动态爬取 802 个网页，平均用时约 2s/个，和 selenium 设置的动态加载时间上限基本一致。

selenium 爬虫设置如下所示：

```
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
chrome_options.add_argument('--ignore-ssl-errors')
chrome_options.add_argument('enable-automation')
chrome_options.add_argument('--window-size=1920,1080')
chrome_options.add_argument('--disable-extensions')
chrome_options.add_argument('--dns-prefetch-disable')
driver = webdriver.Chrome(chrome_options=chrome_options)
```

```
12. driver.set_page_load_timeout(2)
```

b) 明网暗网区分

根据定义，暗网即未被搜索引擎收录的网页。项目中以百度搜索引擎作为标准，利用其接口"site:网址"检查网页是否被百度收录。

具体实现通过 requests.get(url,headers)实现。其中，目标 url 为：

https://www.baidu.com/s?tn=02003390_14_hao_pg&ie=UTF-8&wd=site%3A+网址*+&rsv_spt=1&rsv_iqid=0xdb5b72850006626d&issp=1&f=8&rsv_bp=1&rsv_idx=2&ie=utf-8&tn=02003390_14_hao_pg&rsv_enter=0&rsv_dl=tb&rsv_sug3=6&rsv_sug1=6&rsv_sug7=101&rsv_btype=i&prefixsug=sdasd&rsp=0&inputT=1704&rsv_sug4=4371*

伪装用的头为：

'Connection': 'Keep-Alive',
'Accept': 'text/html, application/xhtml+xml, */*',
'Accept-Language': 'en-US,en;q=0.8,zh-Hans-CN;q=0.5,zh-Hans;q=0.3',
'Accept-Encoding': 'gzip, deflate',
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36'

这个目标 url 有一堆参数，非常复杂，是以 Hao123 作为入口、再人工进行了一系列搜索操作后再重搜索的路径，大量的参数即对应大量的搜索操作，具有高度的人工性，因此被反爬的概率极低（单线程无间隔连续访问 100 次都没有被封）。

根据访问得到的结果，利用 re.findall()判断，如果出现"安全验证"四个字就是被封了（使用该 url 后并没有遇到），出现"没有找到"四个字就是未被百度收录（暗网），其他情况就是被收录（明网）。

c) 网页更新获取

此处使用了小顶堆的数据结构，并定义了类 update_event。

update_event 类包含两个成员变量：更新时间 set_time 和更新目标 url，意义即为在 time.time()大于等于更新时间时对 url 进行访问（更新获取）。由于 time.time()表示 1970 纪元后经过的浮点秒数，此处直接取用其绝对值。

对于每一个刚访问完的网页，自动生成一个对应的 update_event(set_time,url)，插入到小顶堆中。小顶堆中的大小关系即 update_event.set_time 的大小关系。这样，其每次插入 update_event 之后，都可以自动把待更新的网页中预计更新时间最早的放到堆顶，而不会受到网页之间更新频率的差异的影响。

此外，还定义了一个双端队列 waiting_list，每当堆顶元素的 set_time 小于等于当前的 time.time()时，就把其放入 waiting_list。双端队列数据结构主要服务于优先级处理，具体的放入方式之后说明。

d) 流速限制

每爬取完一个网页，time.sleep()一段时间即可。

e) 避免重抓

使用 md5 码，避免同一网址短时间内重复爬取；至于同一网页不同网址，用镜像判断去除。

考虑到 md5 码全长 128 位，占用存储空间巨大，必须进行采样。项目中，将网址转化为 md5 码后，用先 16 进制表示，转化为字符串后取下标 0:32:5 的 7 位，便成为了长度为 7 的字符串，再视作一个 16 进制表示，转化为十进制。这样，用一个长度 16^7、值为 0-1 的 list 就可以表示某网址是否已经被记录。

f) 深度控制

项目中通过两种方法，限制爬取深度：

1. 对整体而言，使用广度优先策略进行爬取，避免了深度过深；
2. 对单个网页而言，通过标签树，只爬取从根节点开始的前五层中的 url，避免过深爬取。

g) 优先级处理

首先，要求 url 开头无“http(s)://”，结尾无“/”，提出以下 url 的优先级计算方式：

优先级值=4×“/”的个数+1×“?”的个数+1×“=”的个数+1×“#”的个数-3×“index”的个数

可见，优先级值越低，网页越接近根节点，优先度越高。

接下来，对之前所提及的双端队列 waiting_list 进行说明。在爬取到一个新的 url 或是小顶堆的堆顶网页需要更新时，先将 url 插入 waiting_list 的最右端。之后，根据优先级，对 waiting_list 进行稳定排序，使得优先级值越低的越靠左，同一优先级下先进入的排在左边。在当前网页爬取完之后，取出 waiting_list 中最左端的元素进行爬取。

换言之，原则为：同一优先级先进先出，不同优先级以值为序。

h) 镜像网页剔除

对于一个网页，获取源码之后，利用 BeautifulSoup.get_text()获取非标签文本，再利用 pyhanlp.HanLP.extractKeyword 提取若干个关键词（项目中设置为 60 个）。

对于两个网页，设其关键词集为 P、Q，则定义其相似度为：

$$sim(P,Q)=\frac{|P\cap Q|}{|P\cup Q|}$$

相似度高于某一阈值（项目中为 0.9），即视为镜像网页。将已录入的网页的关键词

用字典储存，将新网页的关键词和字典中的每一组关键词进行对比，发现与其中某组相似度过高则可剔除新网页。

此处的相似度不仅仅可以用于镜像网页的判断，也可以用于某一网页更新程度的判断。如果相似度较高，说明某网页更新程度较低，可以考虑降低其更新频率。这种方案与 I-Match 算法相似，但我认为，pyhanlp 的类似于 Pagerank 算法的特征值提取法，比简单地去除高频、低频词要合理。

i) GUI 修改

基本基于任务 1、2 实现的 GUI。由于网页格式不同于之前的论文数据库，加之引入了中文，对 GUI 程序进行了一些修改，主要在于初始化部分。由于词库数量剧增，在初始化过程中生成索引需要长达十几分钟的时间。因此，事先生成了 csv 格式的索引，在初始化过程中花费约 20s 读取即可。

4. GUI 使用说明

见另一篇 PDF 文档《GUI 使用说明 Ver3.0》。

5. 重要类、变量、函数说明

源码文件为 1 爬虫源码.ipynb，各py对应的是GUI。源码参数略改以重跑供展示。

类：

url_class: 包含成员整数 rank（优先级值）和字符串 url。即将 url 和其优先级值绑定起来。比较基于 rank。

update_event:包含成员浮点数 set_time 和 url_class 类的对象 url，比较基于 set_time。同时还包含成员函数 revisit，表示在双端队列 waiting_list 中再加入该 url 以便下次访问。

变量：

driver:selenium 爬虫

head:requests 访问百度所用头

md5s:用去标记网址是否加入过 waiting_list 的长 16^7 的 0-1 值列表

waiting_list:即之前所述双端队列

heap:即之前所述小顶堆

link_wait_num:已经进入过 waiting_list 的网址数

link_finish_num:已经被访问过并入库的网址数，若有更新计算多次。

max_num:link_finish_num的最大值，同时也设定为link_wait_num的最大值的1/2

url_kw:用于保存网址的关键词的字典，格式为{url 的七位 md5 码采样：≤60 个关键词}

abandon : 如果变为1，代表当前网址和已有网址相似度过高，废弃。在demo中会把30s后重爬不更新的直接废弃（毕竟只是示意版，不可能让程序跑个好几天去等更新）

函数：

baidu_find(url):检查百度是否收录某 url，作为明网暗网判断依据

md5_(s):根据网址字符串 s 生成采样七位 md5 码

heap2wl():从 heap 中取出更新时间到了的放到 waiting_list 里

gettitle(soup):从 BeautifulSoup 对象中提取 title

get_keywords(soup,num):从 BeautifulSoup 对象中提取 num 个关键词，默认 60

sim(kw_list1,kw_list2):计算两组关键词相似度

search_href(soup):从 BeautifulSoup 某节点提取当前节点中包含的超链接 url（不包含子节点）

url_spider(url):对一个 url 进行爬取和处理

wl_deal():对 waiting_list 进行处理（即不断取出 url 爬取）

wl_add_first(url):将 url 手动加入 waiting_list

run(urls):以一组 urls 作为起点，运行爬虫