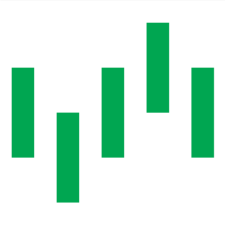
**北京理工大学研究生课程**

**创新实践报告**

****

厚德 明理 慎独 求是

|  |  |
| --- | --- |
| 实验名称： | **网络静态数据获取与解析系统研制** |
| 学生姓名： | 马越 |
| 学生学号： | 3220231244 |
| 学生学科： | 计算机科学和技术 |
| 学生学院： | 信息与电子学院 |
| 课程名称： | 0500070-信息系统及其安全对抗 |
| 任课教师： | 王越、罗森林 |
| 开课单位： | 信息系统及安全对抗实验中心（工信部） |

目 录

[1 实验目的 3](#_Toc526356525)

[2 实验题目及解析 3](#_Toc526356526)

[3 实验环境和条件 3](#_Toc526356527)

[4 实验效果的评价 3](#_Toc526356528)

[5 实验过程 3](#_Toc526356529)

[6 实验结果及分析 3](#_Toc526356530)

[7 问题与讨论 3](#_Toc526356531)

[8 附录 4](#_Toc526356532)

# 实验目的

基于Python语言设计网络爬虫系统,实现对网页内容的爬取,并下载至本地。

# 实验题目及解析

【题目名称，实验重点和难点分析，主要功能和性能】

**实验内容与功能**：

* 基于windows系统使用python设计能够抓取百度新闻网页实时信息的爬虫系统，爬取新闻的词条信息和链接
* 初步实现了网页排重
* 针对词条信息的关键词抓取
* 2级网页链接以上的抓取
* 将所有抓取到的网页进行结构化数据的存储（词条信息+url+爬取日期+网页存储位置）

**整体功能介绍**：

输入关键词（或不输入）后对当前百度新闻进行基于关键词的2级网页爬取，并将爬取到的结果（词条信息+url+爬取日期+网页存储位置）保存到本地。

# 实验环境和条件

【网络拓扑结构图、逻辑关系图，实验所需软件硬设备，实验所需数据资源及其特征说明等】

联网计算机1台，内存大于2G，windows操作系统

编程语言：Python

开发软件：Vscode

# 实验效果的评价

【实验效果评价方法及详细说明】

无评价方法（仅包含爬取数据的输出，没法进行定量或者定性的评价）

# 实验过程

【实验流程图、实验涉及参数说明等】



**涉及参数及说明**：

# 设置种子URL和相关参数

seed\_urls = ["https://news.baidu.com/"]

由于任务是对百度新闻进行爬取，所以这个seed\_urls直接固定设置成这一个就行

MAX\_DEPTH = 2  # 设置最大爬取深度

代表从seed\_urls开始继续能够探索的最大深度，也就是从seed\_urls向下广度优先探索2层就会终止

MY\_SEARCH\_KEYWORD = "游戏"  #设置关键词，如果不需要关键词则设置为None

这里可以设置自己想要检索的关键词，如果不想使用关键词过滤则写None，否则则会对所有超链接tag的文本进行分词并判断是否包含关键词，不包含关键词的将不会被保存

FILE\_SAVE\_DIRECTORY = "./"  # 设置最终文件保存的路径

这里可以设置最终数据保存的路径，默认设置为当前文件夹

**一些功能的具体实现细节**：

为了记录url的深度，这里使用了两个队列

# 创建待抓取URL队列和初始深度队列

url\_queue = Queue()

depth\_queue = Queue()

在add url的时候同时add它的深度，在取出url的同时取出它的深度。

为了记录访问过的url和url对应html\_content的hash值，使用集合进行存储和对比

visited\_urls = set()  # 存储已访问的URL

visited\_urls\_hash = set()  # 存储已访问的URL的html\_content的hash值，进行双重判断去重

为了保证爬取速度不会太慢，在requests请求3s拿不到response的话就直接跳过

response = requests.get(url, timeout=3)

解析的时候仅仅提取了所有的超链接

links = soup.find\_all('a', href=True)

并且如果某个链接的url是无效的话（比如 href="javascript:void(0);" )就跳过

    # 遍历每个找到的链接

    for link in links:

        # 判断链接是否是无效url，如果是则跳过

        if link['href'].startswith('javascript:') or link['href'].startswith('Javascript:') or link['href'].startswith('javascri6pt:') or link['href'].startswith('mail'):

            continue

同时还会对url补全后使用urllib.parse.urlparse进行有效性的检查

        # 构建绝对URL，使用urljoin确保URL的正确性

        url = urljoin(seed\_urls[0], link['href'])

        # 使用 urlparse 检查链接的格式

        parsed\_url = urlparse(url)

关键词提取使用的是jieba分词库，使用全模式（因为能有冗余输出方便匹配关键词）分词并剔除了中文英文的标点符号

def chinese\_segmentation(text):

    # 使用全模式进行分词，可能会输出冗余词汇，但是根据冗余词汇检索比骄方便

    # 如果是空的字符串则会返回[]

    seg\_list = jieba.cut(text, cut\_all=True)

    chinese\_punctuation = set("，。！？&#8203;``【oaicite:0】``&#8203;（）《》")  # 中文标点符号

    return [word for word in seg\_list if word not in string.punctuation and word not in chinese\_punctuation]

结构化存贮这里一共只保存了url地址，分词后的关键词，标题信息，此时此刻的时间，和网页存储位置

downloaded\_pages[entry['url']] = {

                            'url':entry['url'],

                            'keywords':keywords,

                            'title':entry['title'],

                            'crawl\_datetime':crawl\_datetime,

                            # 'raw\_content':download\_page(entry['url']),

                            'storage\_location':storage\_location}

为了保证稳定性，每次爬完以后需要等待一段时间

# 添加延迟，以减轻对服务器的访问压力

time.sleep(0.2)

最后保存文件是写入了json文件中，并且这部分的逻辑放入了finally:中，保证在时间紧急的情况下终端程序也会保存目前爬取到的相关的数据。（实际上2级广度优先要爬非常长的时间，同时其中还有部分url会没法get到数据）

finally:

    # 保存为 JSON 文件

    with open(FILE\_SAVE\_DIRECTORY + 'data.json', 'w') as json\_file:

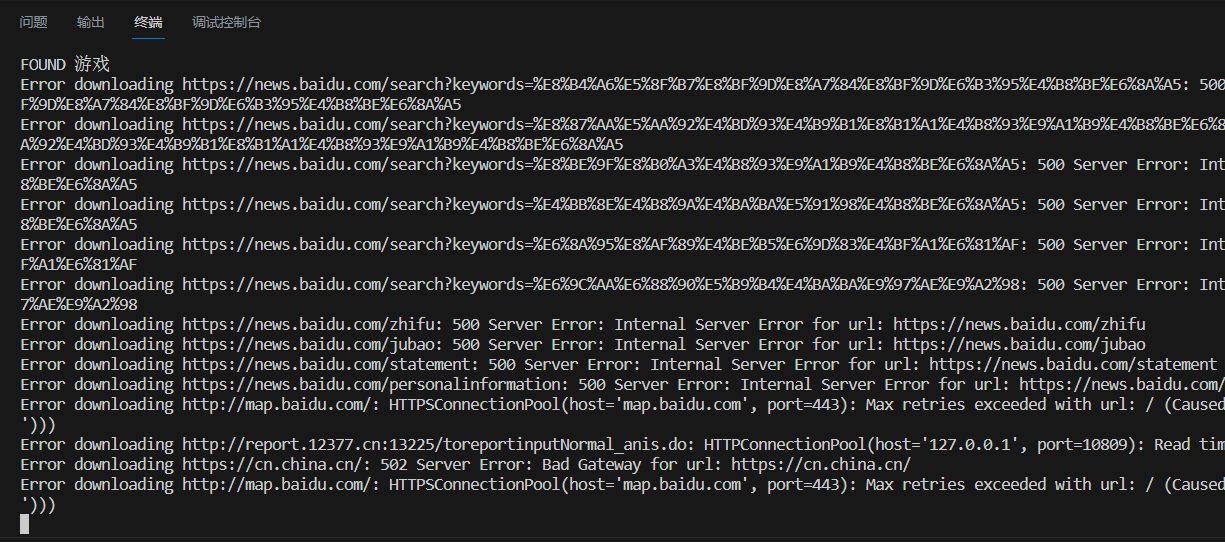
        json.dump(downloaded\_pages, json\_file)

    print(f"结果已保存到{FILE\_SAVE\_DIRECTORY + 'data.json'}")

# 实验结果及分析

【基于图表的定量实验结果、定性实验结果，实验结果的分析讨论】

在找到带有关键词的url时会在终端输出，同时对某url没法得到response时也会在终端输出信息。



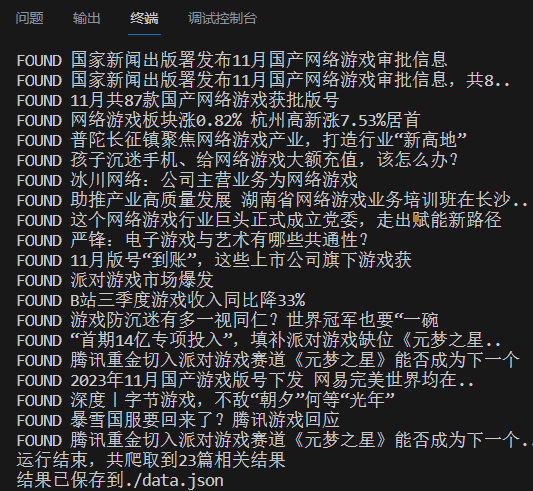
绝大部分的url都没法访问，并且复制粘贴到浏览器内也没法访问，当然也有抓取到的信息，在运行了接近30分钟后直接中断了程序并给出了如下输出：



这样来看效率是非常低下的。

当然还有一种极大地提高程序运行效率的改变是：不做完全的2级广度搜索，而是只对和关键词相关的url展开搜索，这种做法是因为某个已经具有关键词的url的下一级的url更有可能包含关键词。

因此在做出改动之后，不进行完整的2级广度搜索而是只对囊括关键字的url展开搜索，程序在约10s内就完成了搜索，输出了如下结果：



效率非常的高，但是搜索到的结果数量也相对少了很多。

不过在给出的代码中还是按照完全广度搜索2级做的，因为程序可以随时中断并保存结果。

# 问题与讨论

【实验中遇到的问题及解决方案，实验效果提升的建议等】

基本上最开始各种概念都不懂全是问题，但是能解决的都解决了，没解决的写在下方：

问题1：

读取信息有时候会读到乱码（尝试读取了网站中所有<p>的内容并且拼接作为结构化信息的一部分，有些可以正常读取，有些则是乱码没法处理，大部分则是根本什么都读不到），没有解决因此最终的结构化信息中不包含具体网站内的正文信息。

问题2：

关键词搜索目前只是根据<a>的文本内容来检索的，而且非常暴力，如果没有包含关键词就直接抛弃。这里应该有一个和关键词的信息相关度和时效性强弱来排序所有检索到的结果

# 附录

【实验所需其他需要说明的内容，不包括代码】

无