搜索引擎爬虫与预处理

The duration: 4 weeks

Submission date	2022.04.26
Student's name	陈浩楠
Student's ID	2019112256
From	计算机2019-01班

project location: `https://github.com/bianyuanop/search_engine_course_design`

Contents

1 前言

爬虫是搜索引擎的前置程序，用于收集网络中的网页数据，之后存储到本地做预处理。最后通过一系列算法的处理，可通过关键词使用户的到相关的页面。预处理作为搜索引擎的一个必要组成部分，在搜索结果的内容准确度上有着非常重要的作用。这样的预处理技术同样被应用于NLP领域，本项目采用Python作为主要开发语言，一方面考虑到Python作为一种胶水语言可以很好的组合不同的模块，加快开发速度；另一方面，因为Python在深度学习、人工智能领域的发展，其在相关领域的库和包非常丰富，减少了程序编写者的代码编写时间。例如在本项目中所使用的nltk(natural language toolkit)，jieba(结巴)中的peddle-cut功能都收益于其发展。

2 项目架构

本项目采用自下而上的构架方法，逐步实现了爬虫、分词、数据库等模块，之后通过对单个模块进行测试，在保证一定程度上的可用性后，将各个模块组合起来，最后得到了一个自动的爬取-处理程序。在对整体性能、错误等进行一些修正后，最终得到最后的实际生产模块。

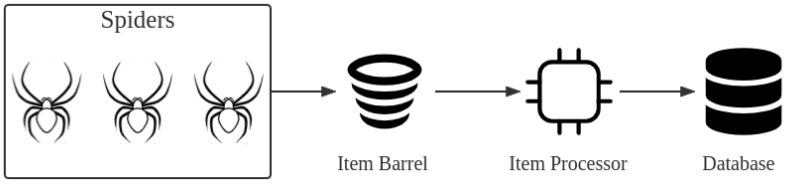

Figure 1: 流水线

2.1 Item Definition

本项目使用了Scrapy作为基础Item的爬去器，在爬取到对应页面后，再通过流水线内的HTTP请求器请求对应页面获取HTML文本后再做处理。Item定义如下，

```
class SpiderItem(scrapy.Item):
    base_site = scrapy.Field()
    title = scrapy.Field()
    href = scrapy.Field()
    url = scrapy.Field()
    filename = scrapy.Field()
    words_file = scrapy.Field()
```

2.2 Processor

2.2.1 规范化

Processor中则做两个操作，一个是规范化爬取后得到的链接，一般href中可以包含以下几类

- //bilibili.com/av10492
- http://bilibili.com/cv1000
- https://bilibili.com/cv1

故需要将上述链接规范化为http开头或https开头的链接。如将$//bilibili.com/av10492$转换为$https://bilibili.com/av10492$

2.2.2 文档处理

对于不同的爬虫，采用不同的处理方法。如对于中文站，我们采取分词，再删除其停用词后再保存。而对于英文站，我们则需先获取词根，再删除停用词。最后再储存至文件，存储完成后在数据库中写入记录(网页文件名, url, 关键词文件名)。

2.3 数据库

按存储需要，数据库中只存储一张表，共四个字段。

```
CREATE TABLE IF NOT EXISTS data
(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    filename TEXT,
    url TEXT,
    words_file TEXT
);
```

按操作需要，这里需要两条操作语句:

- 查询某url是否已经被爬取过
- 插入记录

即:

```
self.cursor.execute('''
INSERT INTO data (filename, url, words_file)
VALUES (?, ?, ?)
''', (item['filename'],item['url'], item['words_file']))
```

和

```
cur = self.cursor.execute("SELECT * FROM data WHERE url = ?", (url,))
return cur.fetchone() is not None
```

3 爬虫

网络爬虫（英语：web crawler），也叫网络蜘蛛（spider），是一种用来自动浏览万维网的网络机器人。其目的一般为编纂网络索引。网络搜索引擎等站点通过爬虫软件更新自身的网站内容或其对其他网站的索引。网络爬虫可以将自己所访问的页面保存下来，以便搜索引擎事后生成索引供用户搜索。爬虫访问网站的过程会消耗目标系统资源。不少网络系统并不默许爬虫工作。因此在访问大量页面时，爬虫需要考虑到规划、负载，还需要讲“礼貌”。不愿意被爬虫访问、被爬虫主人知晓的公开站点可以使用robots.txt文件之类的方法避免访问。这个文件可以要求机器人只对网站的一部分进行索引，或完全不作处理。互联网上的页面极多，即使是最大的爬虫系统也无法做出完整的索引。因此在公元2000年之前的万维网出现初期，搜索引擎经常找不到多少相关结果。现在的搜索引擎在这方面已经进步很多，能够即刻给出高质量结果。[1]

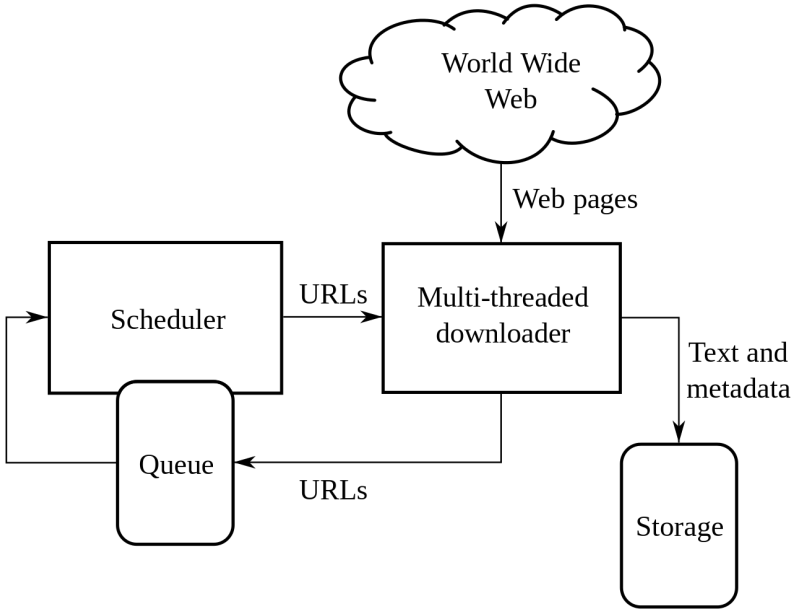

Figure 2: 一般网络爬虫架构[1]

在本项目中，则采用了Scrapy框架作为Spider的reactor，简单的编辑起始站点、解析函数则可得到一个同要架构的高性能爬虫。

3.1 抓取

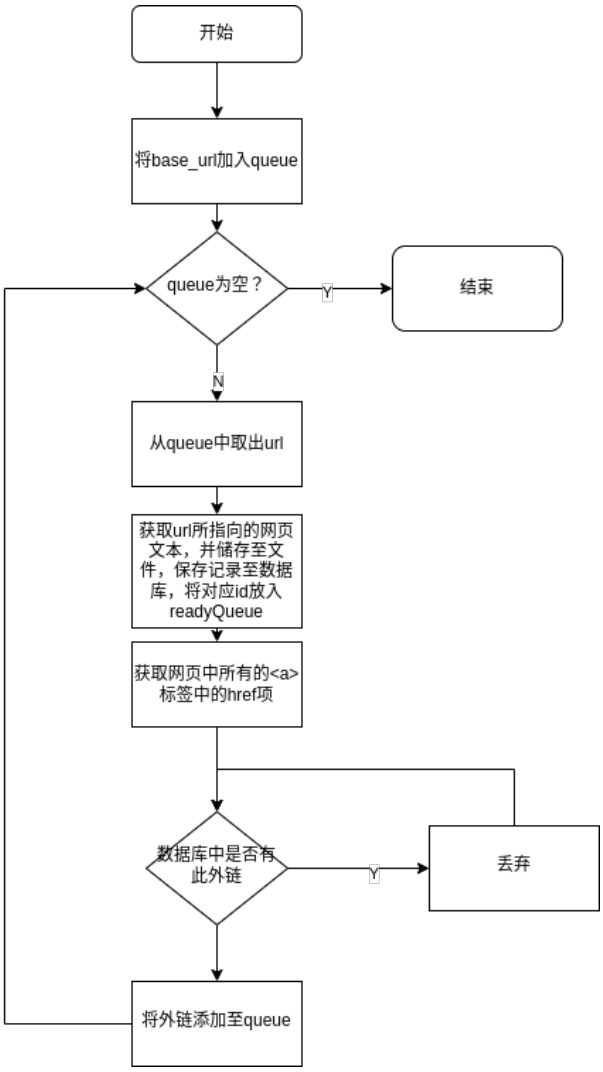

Figure 3: 爬取流程

在上述流程的每一步，Scrapy都会生成一个Item作为输入投入流水管线。在流水管线中采用了requests作为单个网页的抓取，简单的几行代码即可得到网页的信息，如:

```
import requests
url = "someurl"
response = requests.get(url)
```

```
if response.ok:
        content = response.content
```

3.2 Scrapy

Scrapy是一个开源、社区协作完成的爬虫框架，用于提取用户需要的特定数据。

```
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['https://www.zyte.com/blog/']

    def parse(self, response):
        for title in response.css('.oxy-post-title'):
            yield {'title': title.css('::text').get()}

        for next_page in response.css('a.next'):
            yield response.follow(next_page, self.parse)
EOF
$ scrapy runspider myspider.py
```

Figure 4: 示例Spider[6]

```
$ pip install shub
$ shub login
Insert your Zyte Scrapy Cloud API Key: <API_KEY>

# Deploy the spider to Zyte Scrapy Cloud
$ shub deploy

# Schedule the spider for execution
$ shub schedule blogspider
Spider blogspider scheduled, watch it running here:
https://app.zyte.com/p/26731/job/1/8

# Retrieve the scraped data
$ shub items 26731/1/8
{"title": "Improved Frontera: Web Crawling at Scale with Python 3 Support"}
{"title": "How to Crawl the Web Politely with Scrapy"}
...
```

Figure 5: 运行[6]

3.2.1 bilibili spider

Bilibili，又称b站。本项目爬取的内容为b站在专栏推送的文章，在它的主页分析了一下xhr请求发现，它有着一个固定的api调用，且此api返回的结果每一次

都是随机的。故使用此api返回的结果作为起始页开始爬取。

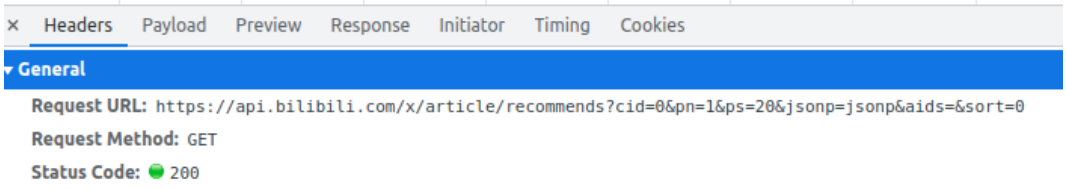

Figure 6: Api请求

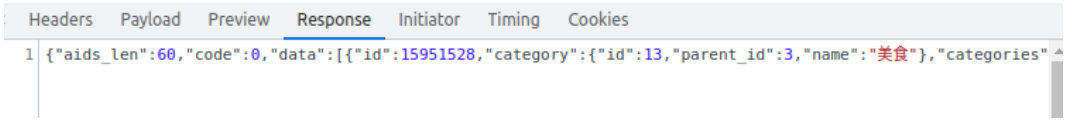

Figure 7: JSON返回值

启动函数*start_requests*:

```
start_api = 'https://api.bilibili.com/x/article/recommends\?cid\=0\&pn\=1\&ps\=2
base_url = 'https://www.bilibili.com/read/cv'

stream = os.popen("curl https://api.bilibili.com/x/article/recommends\?cid\=0\&p
jsonStr = stream.read()
jsonObj = json.loads(jsonStr)
pagesInfos = jsonObj['data']
for pageInfo in pagesInfos:
    link = base_url + str(pageInfo['id'])
    yield scrapy.Request(url=link, callback=self.parse)
```

解析函数*parse*:

```
        items = SpiderItem()
        items['base_site'] = 'bilibili.com'

        for q in response.css('.article-item'):
            items['title'] = q.css('.article-title::text').get()
            items['href'] = q.css('a::attr("href")').get()

            yield items

        next_page = items.get('href')
        if next_page is not None and self.count < 10:
            yield response.follow(next_page, callback=self.parse)
```

3.2.2 stackoverflow spider

至于StackOverflow，则使用$https://stackoverflow.com/questions$此链接下的文章，以及使用最下的页脚作跳转。启动链接$start_urls$:

```
start_urls = [
        'https://stackoverflow.com/questions'
    ]
```

解析函数$parse$:

```
items = SpiderItem()

for q in response.css('.s-post-summary--content-title'):
    items['base_site'] = 'stackoverflow.com'
    items['title'] = q.css('a::text').get()
    items['href'] = q.css('a::attr("href")').get()

    yield items

next_page = response.css('.s-pagination--item.js-pagination-item[rel="next"] \
        ::attr("href")').get()
if next_page is not None and self.count < 10:
    yield response.follow(next_page, callback=self.parse)
```

3.3 网页解析

网页解析方面，使用了一个著名的Html解析库- BeautifulSoup4，这个库提供了基本的html tag解析功能，在html文本输入后，可以生成一个soup对象，对于这个对象，我们可以对它进行遍历以获取所有标签链接，以及每个链接中的文本。对于外链的获取，一般而言，html中的外链一般使用tag名为a的标签包含外链。一般格式为$< a\ href = "somelink" > text\ description < /a >$，用css选择器的语法则可以这样提取:

```
tags = document.querySelectorAll('a[href]')
// iterate tags and further disposing
...
```

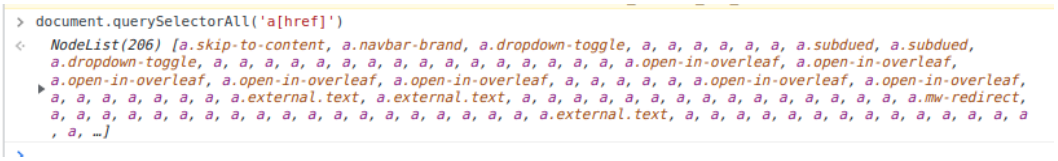

Figure 8: css选择题样例

而对于文本提取，则有以下两种情况:

情况一：文本全位于叶节点

```
<html>
<body>
<div>
  <a href="hello">hello</a>
  <a href="hello">another</a>
<div>
</body>
</html>
```

情况二：部分文本位于父节点

```
<html>
<body>
<div>
  <a href="hello">hello</a>
  <a href="hello">another</a>
  <!-- parent text -->
  parent text
<div>
</body>
</html>
```

在BeautifulSoup[3]中，一个节点的文本信息可以用*.string* 获取，但是这种情况只在叶节点有效，如果节点为父节点，那么这个方法将会将字节点的raw content作为输出。所以通常我们只在叶节点调用这个方法。但这样的方法在第二种情况会导致信息丢失。故我们需要在遍历时，在取出叶节点文本后，将叶节点删除，最后使它的父节点退化成叶节点已进行操作。

两种方法分别对情况二中的文本进行处理后的结果分别为:

BFS	BFS with deletion
hello another	hello another parent text

伪代码如下:

```
FUNC GET_TEXT(node)
        IF node is leaf:
                APPEND node.content
        ELSE:
                FOR child in node:
                        GET_TEXT(child)
                        DELETE child
                APPEND node.content
ENDFUNC
```

4 预处理

预处理的流程为

Step	
1	分词
2	删除重复词
3	删除停用词
4	生成词根

4.1 停用词

Stopwords ISO 中的英文和中文听用词表:

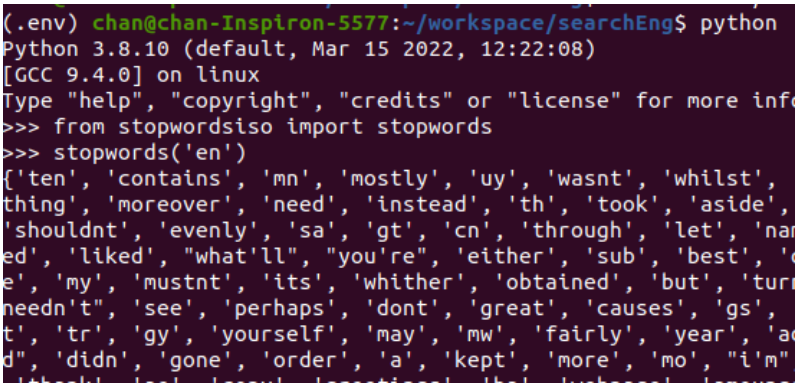

Figure 9: en stopwords

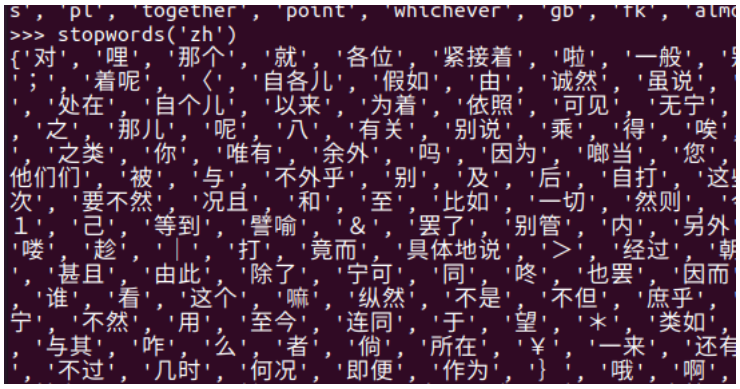

Figure 10: zh stopwords

4.2 分词

4.2.1 中文分词

中文分词使用了Python中的jieba[2]库，jieba支持一下几种模式

- 精确模式，试图将句子最精确地切开，适合文本分析。
- 全模式，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义。
- 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- paddle模式，利用PaddlePaddle深度学习框架，训练序列标注（双向GRU）网络模型实现分词。同时支持词性标注。

一个样例输入：
维基百科最早是在吉米·威尔士与拉里·桑格两人的合作下于2001年1月13日在互联网上推出的网站服务，并于1月15日正式展开网络百科全书项目。其中桑格结合了维基百科网站合作核心之“Wiki”以及具有百科全书之意的“encyclopedia”创造出新混成词“Wikipedia”。在创立之初，维基百科的目标是向全人类提供自由的百科全书，并希望各地民众能够使用自己选择的语言来参与编辑条目。其他书面印刷的百科全书多是由专家主导编辑，之后再由出版商印刷并加以销售。维基百科在性质上一如其号称般属于可自由访问和编辑的全球知识体，这也意味着除传统百科全书所收录的信息外，维基百科也能够收录非学术但仍具有一定媒体关注度的动态事件。2006年《时代》杂志所评选的时代年度风云人物“你”中，便提到了全球上百万人于线上以协作方式促进了维基百科的快速成长，同年提及的其他重要网站还有YouTube、MySpace和Facebook。来自`https://zh.wikipedia.org/wiki/%E7%BB%B4%E5%9F%BA%E7%99%BE%E7%A7%91`
分词输出：

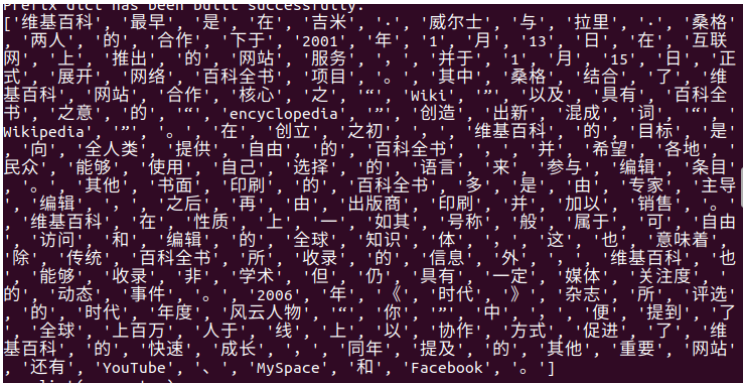

Figure 11: 中文分词样例

4.2.2 英文词根生成

样例输入：
NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical

resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.
来自`https://www.nltk.org/`
样例输出:

```
(.env) chan@chan-Inspiron-5577:~/workspace/searchEng$ python project/preproces
sing/stemming.py
['suit', 'wordnet,', 'industrial-strength', 'interfac', 'process', 'it', 'huma
n', 'languag', 'activ', 'discuss', '50', 'nlp', 'build', 'corpora', 'libraries
,', 'platform', 'lexic', 'forum.', 'reasoning,', 'nltk', 'lead', 'wrapper', 'p
rogram', 'resourc', 'tagging,', 'easy-to-us', 'parsing,', 'data.', 'classifica
tion,', 'semant', 'stemming,', 'tokenization,', 'python', 'librari']
```

Figure 12: 词根生成

5 结果

```
2022-04-17 20:18:15 [scrapy.core.scraper] DEBUG: Scraped from <200 https://stackoverflow.com/questions?tab=newest&page=10>
{'base_site': 'stackoverflow.com',
 'filename': '_questions_71900880_new-struct-table-in-swifui-for-macos-how-to-use-it-with-navigationlink.html',
 'href': '/questions/71900880/new-struct-table-in-swifui-for-macos-how-to-use-it-with-navigationlink',
 'title': 'New struct Table in SwifUI for MacOS. How to use it with '
          'NavigationLink?',
 'url': 'https://stackoverflow.com/questions/71900880/new-struct-table-in-swifui-for-macos-how-to-use-it-with-navigationlink',
 'words_file': '_questions_71900880_new-struct-table-in-swifui-for-macos-how-to-use-it-with-navigationlink.txt'}
2022-04-17 20:18:15 [scrapy.core.engine] INFO: Closing spider (finished)
2022-04-17 20:18:15 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 3696,
```

Figure 13: 主程序运行时

SQL ▾ 4 / 17 151 - 200 of 848

id	filename	url
151	_questions_71856134_how-to-connect-wcf-service-to-a-uwp-appl-in-another-computer.html	https://stackoverflow.com/questions/71856134/how-to-connect-wcf-service-to-a-uwp-appl
152	_questions_71856133_how-to-center-navbar-items-exactly-to-the-center.html	https://stackoverflow.com/questions/71856133/how-to-center-navbar-items-exactly-to-th
153	_questions_71856132_disable-payment-gateway-minimum-maximum.html	https://stackoverflow.com/questions/71856132/disable-payment-gateway-minimum-maximum
154	_questions_71856131_how-to-use-this-sql-statement-in-django-orm.html	https://stackoverflow.com/questions/71856131/how-to-use-this-sql-statement-in-django-
155	_questions_71856130_any-way-to-get-data-from-console-log.html	https://stackoverflow.com/questions/71856130/any-way-to-get-data-from-console-log
156	_questions_71856129_powershell-compare-two-csv.html	https://stackoverflow.com/questions/71856129/powershell-compare-two-csv
157	_questions_71856128_unable-to-update-jenkins-using-yum-in-rhel8.html	https://stackoverflow.com/questions/71856128/unable-to-update-jenkins-using-yum-in-rh
158	_questions_71856125_how-to-take-positions-from-np-where.html	https://stackoverflow.com/questions/71856125/how-to-take-positions-from-np-where
159	_questions_71856124_connecting-aws-rds-retun-is-the-server-running-on-that-host-and-accepting-tcp-ip.html	https://stackoverflow.com/questions/71856124/connecting-aws-rds-retun-is-the-server-r
160	_questions_71856123_is-there-a-type-of-encoding-that-can-transform-this-kind-of-categorical-values.html	https://stackoverflow.com/questions/71856123/is-there-a-type-of-encoding-that-can-tra
161	_questions_71856122_instanciation-of-ocx-object-with-qt.html	https://stackoverflow.com/questions/71856122/instanciation-of-ocx-object-with-qt
162	_questions_71856121_accessing-compiled-file-name-when-building-ts.html	https://stackoverflow.com/questions/71856121/accessing-compiled-file-name-when-buildi
163	_questions_71856120_get-childs-height-to-carousel-sliders-height-flutter.html	https://stackoverflow.com/questions/71856120/get-childs-height-to-carousel-sliders-he
164	_questions_71856114_cant-set-liststring-value-from-uirepeat-var-status.html	https://stackoverflow.com/questions/71856114/cant-set-liststring-value-from-uirepeat-
165	_questions_71856112_asp-net-core-web-api-system-aggregateexception-unable-to-cast-object-of-type.html	https://stackoverflow.com/questions/71856112/asp-net-core-web-api-system-aggregateexc
166	_questions_71856111_what-is-the-right-way-to-generated-optimized-object-from-a-llvm-ir-file.html	https://stackoverflow.com/questions/71856111/what-is-the-right-way-to-generated-optim
167	_questions_71856109_how-to-add-remove-line-in-xml-txt-file-within-logic-that-navigates-through-a-map.html	https://stackoverflow.com/questions/71856109/how-to-add-remove-line-in-xml-txt-file-w
168	_questions_71856107_routing-intranet-with-network.html	https://stackoverflow.com/questions/71856107/routing-intranet-with-network
169	_questions_71856105_how-to-compare-items-among-3-lists.html	https://stackoverflow.com/questions/71856105/how-to-compare-items-among-3-lists
170	_questions_71856103_nightwatchjs-is-ignoring-chrome-options-when-trying-to-launch-no-sandbox-and-hea.html	https://stackoverflow.com/questions/71856103/nightwatchjs-is-ignoring-chrome-options-

Figure 14: 数据记录

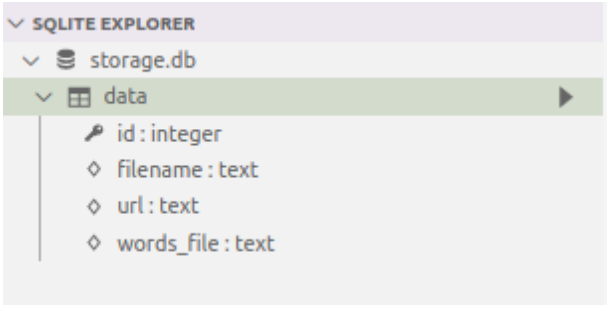

Figure 15: 数据表

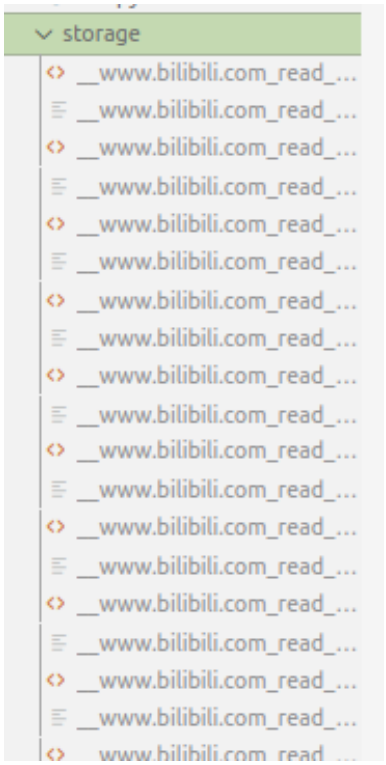

Figure 16: 储存文件夹

References

[1] 网络爬虫. (2022). In 维基百科，自由的百科全书. https://zh.wikipedia.org/w/index.php?title=网络爬虫

[2] Junyi, S. (2022). Jieba [Python]. https://github.com/fxsjy/jieba (Original work published 2012)

[3] Beautiful Soup Documentation—Beautiful Soup 4.4.0 documentation. (n.d.). Retrieved March 30, 2022, from https://beautiful-soup-4.readthedocs.io/en/latest/

[4] NLTK:: Natural Language Toolkit. (n.d.). Retrieved March 30, 2022, from https://www.nltk.org/

[5] Stopwords ISO. (2022). [JavaScript]. Stopwords ISO. https://github.com/stopwords-iso/stopwords-iso (Original work published 2016)

[6] Scrapy — A Fast and Powerful Scraping and Web Crawling Framework. (不详). 取读于2022年4月17日, 从https://scrapy.org/