信息检索—Web 搜索引擎实现文档

1511426 黄静平

December 13, 2017

1 前言

一般的搜索引擎架构如 1 所示。所以为了实现一个简易的搜索引擎，我们需要实现爬虫，链接分析，文本构建索引，Page rank，查询接口五部分。我们先通过爬虫爬取网页，然后通过链接分析部分分析处理爬下来的网页内容，同时对文本内容构建索引和 Page rank。前四个部分完成后，我们实现查询接口，处理用户的查询请求。下面我对这五个部分进行一一介绍。

2 模块介绍

2.1 网页爬虫

爬虫部分使用 Python 实现。具体步骤是我们先给一个种子 url，这里我选择的是南开大学官网首页。然后我们爬取这个网页，分析其内容，提取出页面包含的链接，将它加入到待爬取队列中。这里我设置了一些数据结构，比如待爬取队列，黑名单，白名单，已爬列表等。用来提高爬虫效率，防止爬取重复网页和无效网页。同时我还对重定向之类的问题进行了处理，防止保存下来的数据和实际情况不符。为了减小爬虫过程中对学校网站造成的负担，我的爬虫在爬取一定数量网页后会休息 30 秒。同时为了防止爬到一半因为各种原因程序崩了而导致前功尽弃的情况，我也设置了在爬取一定数目的页面后及时的将数据从内存保存到磁盘中。具体保存的数据包括：已爬取 url 列表，待爬取页面队列，黑名单，页面信息，爬下来的 html 等文件等等。这里我为了程序的移植性更好，使用的数据库是 Python 自带的 Sqlite。这里程序有两个入口，一个是对整个爬虫程序重新开始，另一个是读取之前保存的数据，然后从上次停止的地方开始。这个可以由用户自己设置。2是爬虫过程的截图。

2.2 链接分析

链接分析是和爬虫一起进行的。爬虫将爬下来的数据交给链接分析模块，链接分析模块使用正则表达式加上 Python 的 BeautifulSoup 包提取出链接和对应的锚文本，然后将链接关系保存到数据库中。为了实现网页快照功能，我们需要将此时爬下来的网页内容进行存储。由于一些网页是通过 javascript 生成的，所以这里我们需要先执行 Javascript 代码才能获取真实页面。这里用到了 BeautifulSoup 中的 extract 函数。前面保存了链接关系，接着我们要保存当前页面的信息，包括当前页面类型 (比如 pdf,word,html 等等)，页面 title，页面 id 等等。然后将解析出来的 url 返回给爬虫模块，爬虫模块将新的 url 添加到待爬队列中。

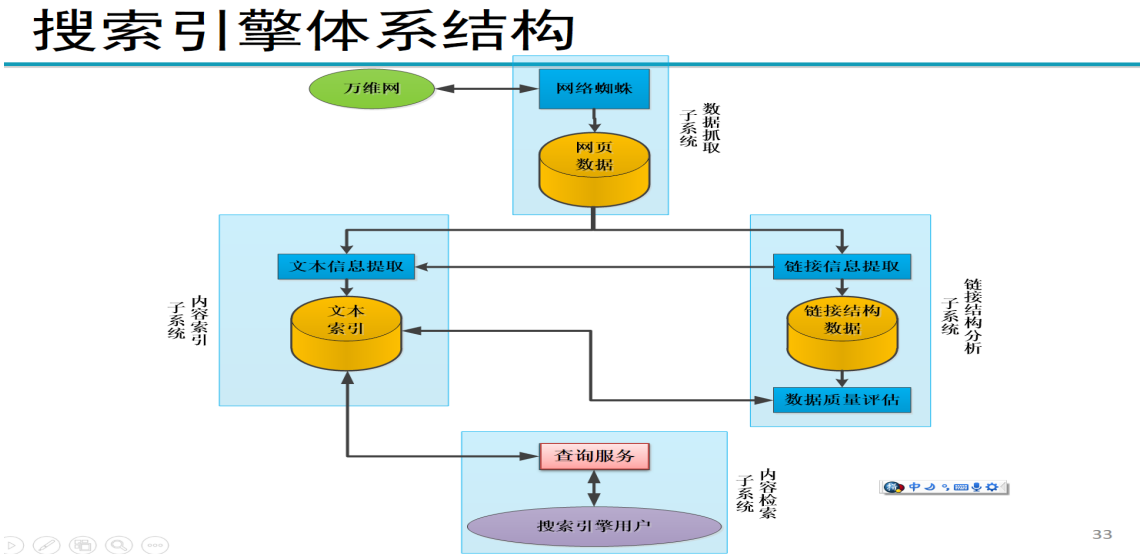

Figure 1: 搜索引擎体系结构.

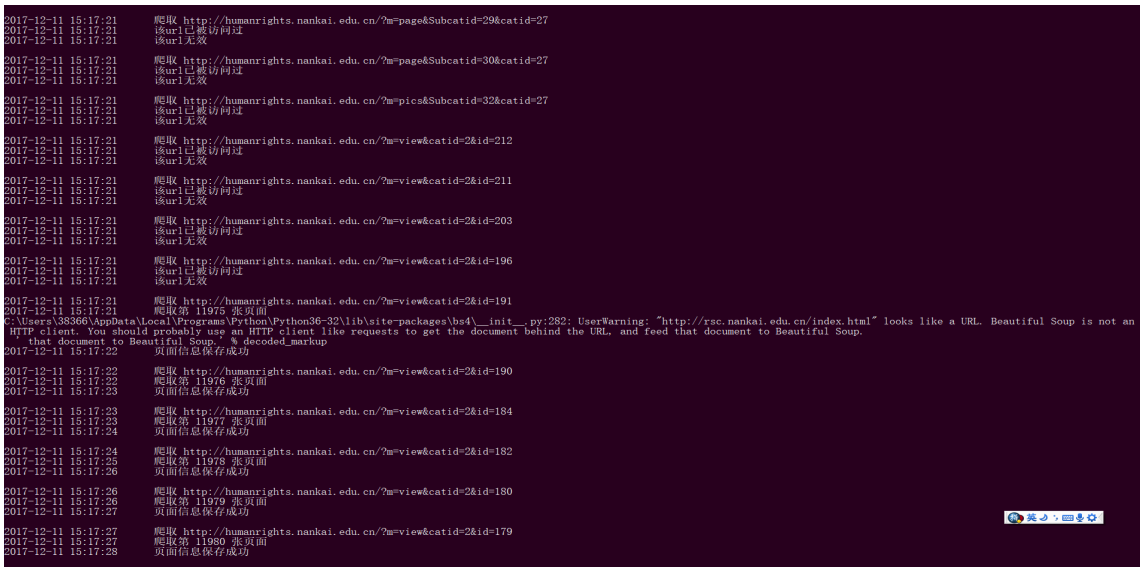

Figure 2: 爬虫截图

2.3 文本索引构建

文本索引构建用的是 Python 的 Pylucene 包。通过这个包，我们爬好的页面就可以构建出索引了，以此提高我们搜索时的效率。

2.4 Page rank

Page rank 是对页面重要性的排序。这里我们主要通过链接关系来决定一个页面的重要程度。由于我们之前通过链接分析得到链接关系，将之保存在数据库中，现在我们从数据库中将这些关系读取到内存中，假设有 count 个页面，那么我们就构建一个 count×count 的二维数组 link，其中 link[i][j] 表示页面 j 指向页面 i。但是不同页面的链接关系的价值是不同的，所以我们需要对每个链接关系给出其价值。这里我们使用的方法很简单，遵从一个原则就是“沉默是金”。即如果一个页面只有 1 个链出的链接，那么这个链接关系价值很大；如果一个页面有几千个链出，那么其每个链接关系价值就相对比较小。

2.5 查询接口

提供了多种类型的查询服务，包括短语检索、通配符、site、filetype、查询日志等等。所以我们要先通过分析传过来的查询字符串确定用户想要的查询类型，然后根据不同的查询类型获取对应的页面，这里会用之前构建是索引和数据库。然后我们获取一个结果集。我们根据之前通过 Page rank 得到的结果对这个结果集进行排序，然后将排好序的结果返回给前端。前端会采用分页机制展示搜索结果，即我们并不会一次性的将数据全部返回，而是返回部分数据给前端，等用户点击了切换页码时我们再在后台重新搜索。这样就能大大降低后台压力了。当然，因为这里我们只有一万多张页面，所以我直接是一次性搜索出来的。对于用户的每次搜索我们都会将他保存在查询日志中，包括他点击链接。这样可以帮助我们后期对页面重要性进行更新。

3 实验成果展示

该实验在本地完成，具体环境是 Windows10 64 位，Java1.8，Python3.6。

4 实验复现

对这次实验的复现比较麻烦，因为可能涉及到安装 pyLucene 比较麻烦，同时爬虫和索引构建、链接分析等等都需要耗费较长时间。如果你准备复现的话, 请在前面提到过的环境下车时，然后点击根目录下的 run.bat 即可，不过需要等待 2 ～ 3 小时，之后浏览器中打开 http://localhost:8000 即可使用。

5 总结

这次实验难度略大，除了需要掌握如爬虫、正则匹配等知识，更需要好的架构设计，因为多个模块之间关系较为复杂，容易搞混。虽然这次花费了很长时间来做这个搜索引擎，但是其功能包括准确性等都还不够，如果想要实现一个真正的好用的搜索引擎，仍旧是任重而道远。

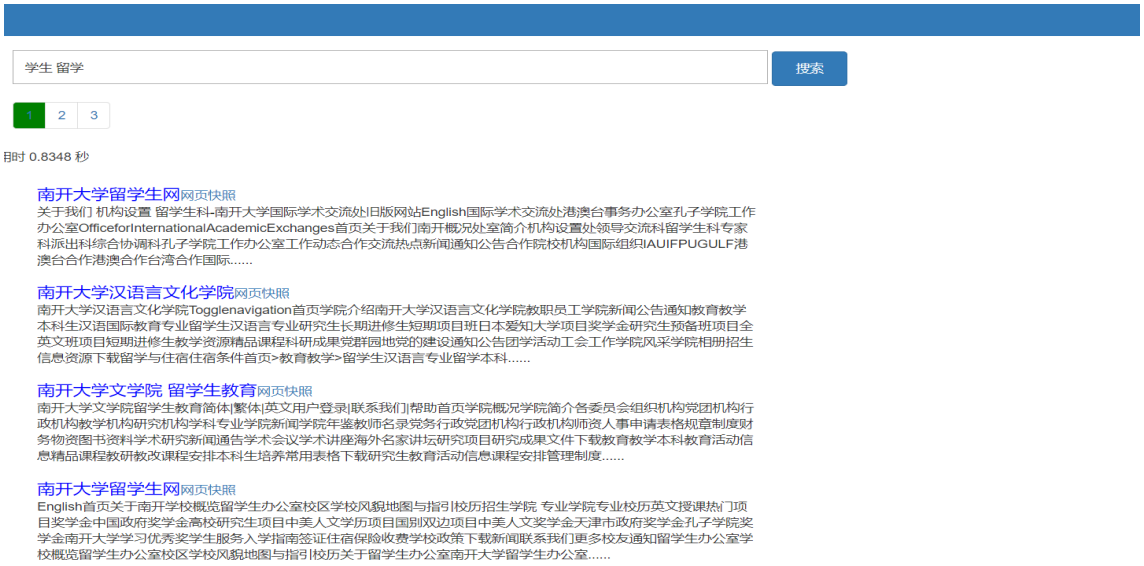

Figure 3: 普通短语查询.

Figure 4: 支持快照.

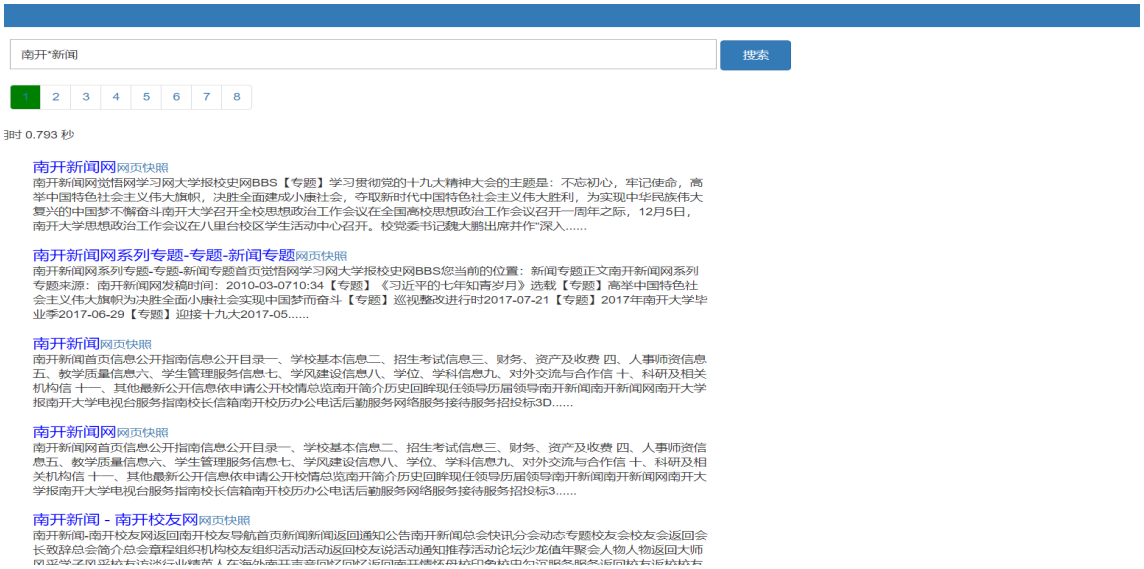

Figure 5: 支持通配符.

Figure 6: 支持分页查询.

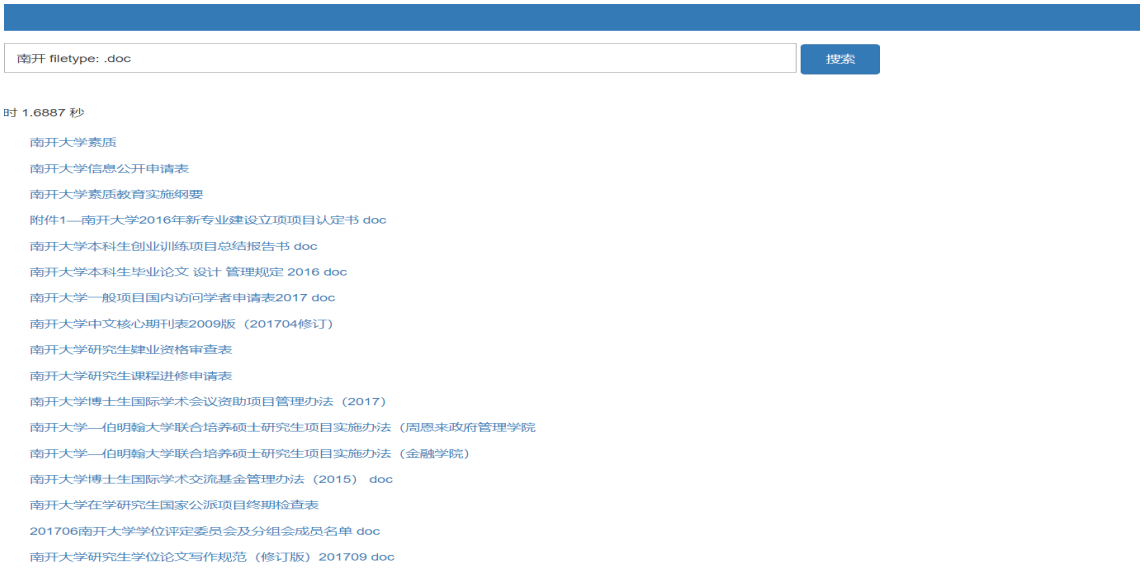

Figure 7: 支持按文件类型查找.

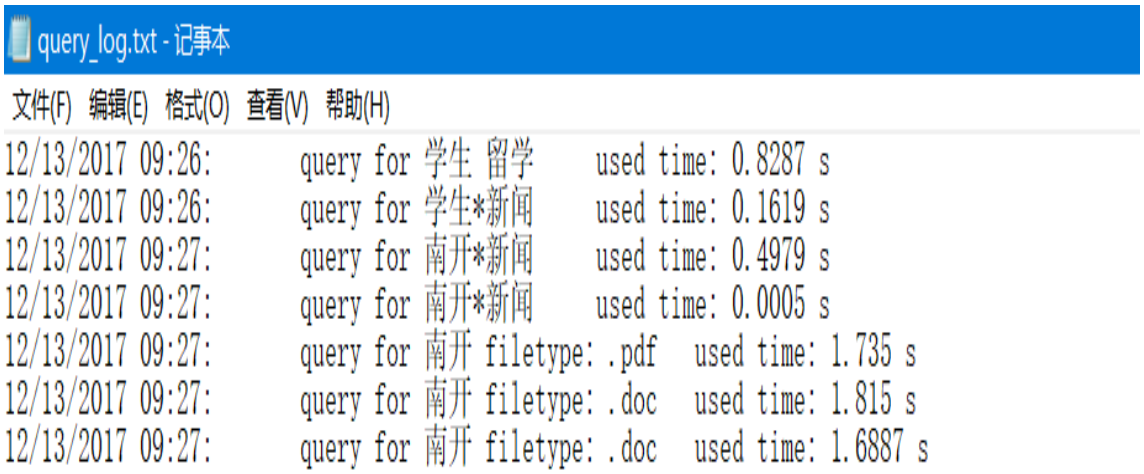

Figure 8: 查询日志.

6 额外说明

因为不是很懂 latex 的排版原理，所以这次文档的排版中的图片位置比较奇怪，望谅解。