Computer Science and Application 计算机科学与应用, 2019, 9(1), 96-105
Published Online January 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.91012

Hans汉斯

Design and Implementation of Graphene Chinese Literature Search Engine Based on Lucene

Xiandong Xiao[1], Qinsheng Wang[2], Yongqiang Yang[2*], Guobao Zhang[1]

[1]School of Automation, Southeast University, Nanjing Jiangsu
[2]National Graphene Products Quality Supervision and Inspection Center (Jiangsu), Jiangsu Province Special Equipment Safety Supervision Inspection Institute, Wuxi Jiangsu
Email: *yqyang@wxtjy.com

Received: Dec. 28th, 2018; accepted: Jan. 11th, 2019; published: Jan. 18th, 2019

Abstract

In recent years, the exploration of graphene in the field of materials has become more and more in-depth, and at the same time, a large number of professional literatures have been published in various journals and conferences. Finding the required literature in a large body of literature has also become increasingly difficult. Traditional database searches are inefficient and search results are not as satisfactory. Therefore, this paper designs a local full-text retrieval system based on Lucene toolkit, Python crawler technology and so on. Firstly, the crawler technology is used to obtain relevant documents on the Internet, and then saved to the MySQL database to create a complete set of search engine system services by customizing Lucene to create an index and search service. A large number of tests have shown that the accuracy of the search results of the system is much higher than that of the database search, and the search speed is much faster than the traditional database query. Therefore, the system can be used for searching and querying graphene Chinese literature, thereby improving the efficiency of researchers in accessing documents.

Keywords

Lucene Spider, Search-Engine, MySQL Graphene

基于Lucene的石墨烯中文文献搜索引擎设计与实现

肖显东[1]，王勤生[2]，杨永强[2*]，章国宝[1]

[1]东南大学自动化学院，江苏 南京
[*]通讯作者。

文章引用: 肖显东, 王勤生, 杨永强, 章国宝. 基于 Lucene 的石墨烯中文文献搜索引擎设计与实现[J]. 计算机科学与应用, 2019, 9(1): 96-105. DOI: 10.12677/csa.2019.91012

[2]江苏省特种设备安全监督检验研究院/国家石墨烯产品质量监督检验中心(江苏)，江苏 无锡
Email: *yqyang@wxtjy.com

收稿日期：2018年12月28日；录用日期：2019年1月11日；发布日期：2019年1月18日

摘 要

近年来，材料领域对于石墨烯的探索越来越深入，与此同时，大量的专业文献被发表在各类期刊、会议中。在大量文献中查找所需要的文献也变得越来越困难。传统数据库搜索效率低下且搜索结果并不尽如人意。因此，本文设计了一种基于Lucene工具包，Python爬虫技术等开发的一套本地全文检索系统。首先利用爬虫技术获取互联网上相关文献资料，接着保存到MySQL数据库中，通过自定义开发Lucene创建索引和搜索服务，从而完成一整套搜索引擎系统服务。经大量测试表明，该系统搜索结果准确率接近数据库搜索，搜索速度也远快于传统的数据库查询。因此，该系统可以用于进行石墨烯中文文献的搜索查询，从而提高科研人员查阅文献的效率。

关键词

Lucene工具包爬虫，搜索引擎，MySQL石墨烯

1. 引言

石墨烯是进入二十一世纪以来材料领域研究的热点，被称为“黑金”，因为它出色的光学、电学、力学等特性，使得石墨烯在各个领域都发挥了巨大的作用。与此同时，大量的石墨烯制备方法和相关改进的论文被发表，却没有相应的数据检索系统管理，以致每次搜索相关文献时需要花费大量的精力。现有的百度学术、谷歌学术等搜索引擎提供的搜索结果并没有分类、相关度等功能；万方、知网等知识库提供的搜索结果又不太准确。中文文献一方面发布在对应出版社的官方网站上，另一方面主要是被知网、万方、维普等网站收录并发布。因此，为了提高石墨烯领域科研人员查找文献的效率，让其更专注于实验，在此提出一种利用 Python 爬虫 + 基于 Lucene 的搜索引擎设计方案[1]，利用爬虫不断从互联网爬取数据保证文献的时效性，利用搜索引擎搜索文献保证文献的准确性，解决现有的搜索结果不准确、没有相关度分析等问题。

2. 技术选型

2.1. Lucene 工具包

Lucene 是 Apache 软件基金会的一个子项目，是一个开放源码的全文检索引擎工具包，并不是个完整的全文搜索引擎。因此我们可以利用 Lucene 源代码的基础上加入自己的算法。目前比较流行的 ElasticSearch 和 Solr 都是基于 Lucene 二次开发的搜索引擎框架。本文采用最新的 lucene-7.5.0 工具包。

本文中，主要使用 analysis 分词包、index 索引包、document 索引存储包、search 检索包、store 存储包和 queryparser 查询解析包等，以及第三方分词库等。

利用 Lucene 开发的流程主要分为两大部分：创建索引和进行检索。创建索引部分需要先获取原始内容(本研究中爬虫获取的数据) > 创建文档 > 分析文档 > 索引文档；进行检索主要就是创建查询和执行搜索两部分。

2.2. 第三方分词库

本文主要研究针对中文文献进行处理，而 Lucene 提供的中文分词器效果较差，所以需要利用第三方中文分词器。这里简单评测一下不同分词器的分词效果。如表 1 所示。

Table 1. Word segmenter comparison
表 1. 分词器效果比较

分词器种类	分解后的文本	原理
StandaAnalyzer	石|墨|烯|具|有|非|常|良|好|的|光|学|特|性|在|较|宽|波|长|范|围|内|吸|收|率|约|为|2.3|看|上|去|几|乎|是|透|明|的|在|几|层|石|墨|烯|厚|度|范|围|内|厚|度|每|增|加|一|层|吸|收|率|增|加|2.3|	一元分词，很明显使用这种分词器会破坏原有汉语词汇的含义，不适合用来做中文分词
SmartChineseAnalyzer	石墨|烯|具有|非常|良好|的|光学|特性|在|较|宽|波长|范围|内|吸收率|约|为|2|3|看上去|几乎|是|透明|的|在|几|层|石墨|烯|厚度|范围|内|厚度|每|增加|一|层|吸收率|增加|2|3|	Lucene 中的 smartcn 模块之中提供的分词器，实际使用效果并不好
CJKAnalyzer	石墨|墨烯|烯具|具有|有非|非常|常良|良好|好的|的光|光学|学特|特性|在较|较宽|宽波|波长|长范|范围|围内|内吸|吸收|收率|率约|约为|2.3|看上|上去|去几|几乎|乎是|是透|透明|明的|在几|几层|层石|石墨|墨烯|烯厚|厚度|度范|范围|围内|厚度|度每|每增|增加|加一|一层|吸收|收率|率增|增加|2.3|	Lucene 的 common 包中提供的分词器，通过源码可以发现需要提供停止字符，从而提高分词正确率。基本原理是使用二元分词效果。
IKAnalyzer	石墨|烯|具有|非常|良好|的|光学|特性|在|较宽|波长|范围内|吸收率|约为|2.3|看上去|几乎是|透明|的|在|几层|石墨|烯|厚度|范围内|厚度|每|增加|一层|吸收率|增加|2.3|	基于词典分词，这里开启了智能模式，看起来分词效果很好，因为没有使用特定字典，石墨烯被分开为两部分。
ChineseAnalyzer	无	该分词器在 lucene5 版本时被移除，官网已经将此类设定为废弃。建议使用 StandardAnalyzer 代替

还有很多其他的分词器诸如 PaodingAnalyzer 等，由于 2008 年就已经停止维护，相对于 2012 年才停止维护的 IKAnalyzer 来讲，隐藏的问题可能会更多。所以本设计采用 IKAnalyzer2012FF_u1 版本的分词器。虽然最后一个版本的 IKAnalyzer 声称支持 3.3 以上版本的 Lucene，但是实际使用过程中，由于 Lucene 的接口变化等原因，需要重写 IKAnalyzer。

2.3. Python 爬虫

Lucene 工具包并不提供抓取数据的功能，所以需要自行获取数据。一小部分数据来自于无锡特检院的数据库数据，大部分使用爬虫技术从互联网上获取。基于 Python3 语言开发的爬虫软件系统，主要用于爬取知网、万方等中文期刊收录网站收录文章的标题和摘要，并保存在数据库中。配合 Python3 使用的开发工具有方便开发的 IDE：PyCharm，浏览器 Chrome 和相关驱动 ChromeDriver 用于自动化控制软件驱动，以及 selenium 自动化测试工具驱动 Chrome 浏览器解析网页内容。由于现代网站开发大多使用了 ajax 方式进行数据传输，通过模拟浏览器发出单纯的网页请求获取返回数据的开发方式只能获取到原始 HTML 信息，而没办法获取到异步请求返回的信息和 JS 渲染的信息。根据爬虫的所见即所得原理，利用 selenium 调动浏览器获取的浏览器渲染结果就是我们最终需要的数据。

2.4. MySQL 数据库

当爬虫在网络上爬取数据后，可以通过文本存储或者数据库存储等方式将数据持久化存储到硬盘中。

通过文本存储的方式在数据量较小时文件体积小，但是文件体积会随着存储数据量增大而增大、搜索时会因为使用线性搜索方式，时间复杂度 O(n)而拖延搜索速度；使用关系型数据库存储数据，主要是针对在大量数据时有较好的处理性能，利用 b/b + 树等数据结构作为索引，提高搜索速率。其中特检院部分的数据也是保存在 MySQL 数据库中，在创建索引的时候需要导出。同时也会使用数据库和搜索引擎进行性能等方面的比较。

3. 系统设计与实现

3.1. 系统总体架构

搜索的基础是数据。整个系统先利用 Python 从相关网站爬取数据，保存到数据库中；由于 Lucene 是 Java 语言实现的工具包，接下来使用 Java 连接到数据库；在获取到数据库连接后，利用 Lucene 开始创建索引并保存到本地磁盘或服务器中；最后用户输入关键字进行检索，并按照一定的顺序将检索结果输出。整体流程如图 1 所示。

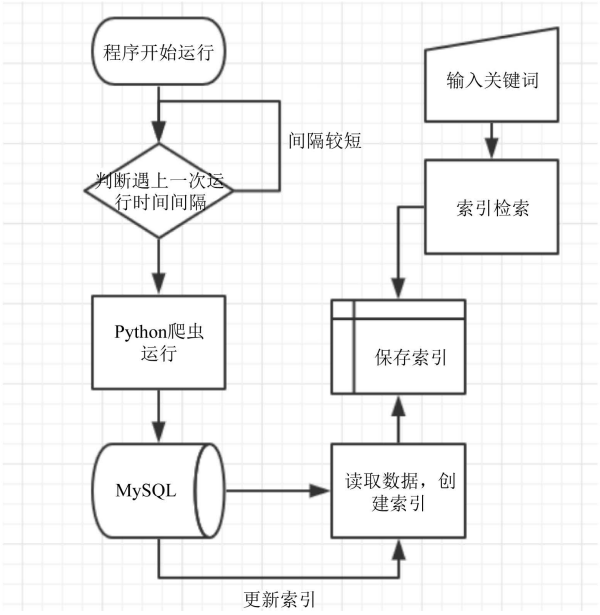

Figure 1. Flow chart
图 1. 流程图

3.2. 爬虫的设计与实现

因为 Lucene 并没有提供数据的收集功能，在不借用 solr 这类二次封装的框架或者 scrapy 爬虫框架的情况下，需要使用 Python 编写的爬虫来从互联网获取想要的数据。

1) 确定数据来源

本文使用万方数据库、知网数据库等作为数据源，通过检索关键词石墨烯查询到全部文献信息并爬取。经过统计，十万以上数量级的数据可以作为搜索引擎检索和数据库检索速度比较的数据源。

2) 网站分析

以万方数据库为例，分析万方数据库的网络请求过程：新版网站在搜索框输入“石墨烯”并搜索后，就会展示相关检索结果，通过点击按钮实现数据翻页。进入浏览器 F12 开发者模式，分析网络请求，发现请求是 XHR 类 GET 请求，也就是异步 GET 请求。所以接下来只要分析异步请求的参数，并利用模拟

器发送请求并获取服务器响应即可。如图 2 所示。

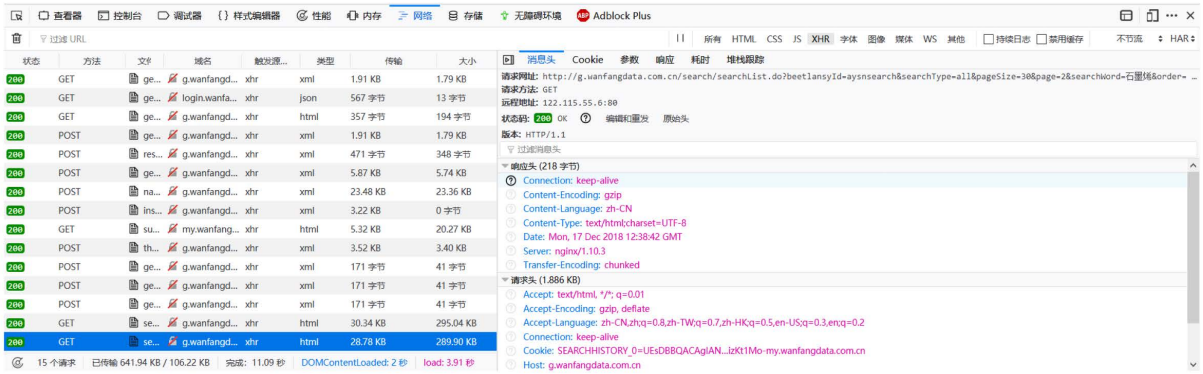

Figure 2. New version of Wanfang database asynchronous request analysis
图 2. 新版万方数据库异步请求分析

经过实际运行发现，新版网站暴露的接口最多只能爬取 100 * 50 条即 5000 数据，数量级太低，没有办法作为搜索引擎数据集，也没有办法作为数据库和搜索引擎比较的样本。旧版网站的搜索接口比较简单，只需要构建 *http://s.wanfangdata.com.cn/Paper.aspx?q=石墨烯&f=top&p=2* 这样的 URL 就可以获取到全部数据。如图 3 所示。

Figure 3. Old Wanfang database request analysis
图 3. 旧版万方数据库请求分析

3) 参数分析并编写代码

通过分析 GET 请求 URL 可以分析出关键参数，通过动态赋值既可以实现翻页等功能。利用 selenium 自动化测试工具调动 Chrome 浏览器访问构建好的 URL 并获取到服务器响应结果，利用 BeautifulSoup 开源库进行解析，并获取到需要的相关数据。再利用 pymysql 连接到本地 MySQL 数据库并保存。关键 Python 代码如下。

```
browser.get(get_url)
wait = WebDriverWait(browser, 5)
soup = BeautifulSoup(browser.page_source, "lxml")
```

4) 数据处理

虽然网络提供了大量数据，但是不免其中有大量的重复数据和无关数据。在爬虫的实际运行中，会发现通过在网站上检索出的数据会出现不间断的重复、以及文献摘要部分的缺失。因此，从两个方面对数据进行处理。一方面是在爬虫运行过程中，如果解析出的数据格式不正确或者为空，则跳过这条数据的写入；另一个方面，是创建数据库时，设立唯一索引从源头保证不写入重复数据，同时在写入数据库时，通过唯一索引保证重复数据替换而不是新增。数据表创建语句如下：

```
CREATE TABLE `spiders`.`wanfang_spiders`  (
`doc_id` int(0) NOT NULL AUTO_INCREMENT,
`doc_title` varchar(255) NOT NULL,
`doc_summary` varchar(2000) NULL,
`doc_date` datetime(0) NOT NULL,
`doc_papers` varchar(255) NOT NULL,
`doc_readtime` bigint(0) NOT NULL,
`doc_clicktime` bigint(0) NOT NULL,
PRIMARY KEY (`doc_id`),
UNIQUE INDEX(`doc_title`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8;
```

在插入数据时，利用唯一索引保证数据不重复，插入的 SQL 语句如下：INSERT INTO wanfang_spiders (***) VALUES (***) ON DUPLICATE KEY UPDATE doc_title=doc_title，这样就会保证当数据重复时，新插入的数据替代旧数据的内容，同时自增 id 不会变化。

3.3. 搜索引擎设计与实现

Lucene 工具包提供了创建索引、保存索引、检索索引等功能。

3.3.1. 索引种类

在搜索引擎中，目前主要使用的是正排索引(forward index)和倒排索引(inverted index)这两种索引[2]。

正排索引：以本文研究为例，使用文档名作为 key，将文档中出现的词作为对应的 value 保存下来。如此一来，如果两篇不同的文章中某一个词都出现多次，就会造成数据的冗余。优点是易于维护，缺点是搜索时间过长。

倒排索引：与正排索引相反，存在一个词典和倒排列表。所有出现的词保存在词典中，倒排列表中保存的是每一个词所在的文档和位置。与正排索引相反，优点是搜索时间短，但是不易维护。

综上所述，针对需要大量数据的搜索引擎设计，需要使用倒排索引。另一个原因是，索引的更新需要配合文献库的更新，而且提供增量更新功能，维护问题并不明显[3]。

3.3.2. 数据导入

利用 Java 连接 MySQL 并读取数据，并没有采用业界流行的 hibernate 或者 mybatis 等框架，而是使用原生方法。一是考虑到开源框架更新较快，某些 API 可能会经常变动导致旧系统崩溃；二是由于网络政策，连接到远程仓库可能会出现问题导致 jar 包无法引用从而导致旧系统崩溃；三是 bug 修复更新发布时，如果将 jar 包保存在服务器上，则需要更新服务端资源；如果使用 maven 构建，则会遇到第二点涉及的问题。因此选用 Java 内嵌提供的 sql 包实现。关键代码如下所示。

```
conn = DriverManager.getConnection(DATABASE_NAME, username, password);
stat = conn.createStatement();
set = stat.executeQuery(sql);
```

3.3.3. 分词器设计

如前文所述，使用 IKAnalyzer，但由于版本不再维护，针对 Lucen7.5.0 版本，需要重写分词器。主要涉及到 Analyzer 类(分析器)和 Tokenizer 类(分词器)。分析器关键代码如下。

```
@Override
```

```
protected TokenStreamComponents createComponents(String s) {
  IKTokenizer4lucene750 ik = new IKTokenizer4lucene750(this.useSmart);
  return new TokenStreamComponents(ik);
      }
```

重写 IKAnalyzer 的关键在于分词器，其基于 IKSegmenter 类(具体分词实现类)，其内部会加载自定义词典和停止词典，并据此进行分词功能。关键代码如下。

```
Lexeme l = segmenter.next();
if (l != null){
xxxTermAttribute.xxx();
return true;
}
```

3.3.4. 创建索引

主要涉及到 Lucene 的 IndexWriter 用于写入索引和 Document 用于保存索引这两个类。Document 可以类比于 MySQL 中数据表的行，TextField 可以类比于数据表的字段。最后再将索引文件保存到本地磁盘或者服务器上[4]。关键代码如下。

```
luceneDoc.add(titleFiled);
docList.add(luceneDoc);
IndexWriter writer = new IndexWriter(directory, iwConfig);
writer.addDocument(docList.get(i));
```

3.3.5. 创建查询

索引保存完毕之后，根据用户输入字符串进行检索。主要涉及到 QueryParser 和 Query 类，用于解析和查询功能[5] [6]。关键代码如下。

```
Query query = new QueryParser("title", analyzer).parse(queryWord);
IndexSearcher s = new IndexSearcher(DirectoryReader.open(directory));
TopDocs topDocs = s.search(query, 10);//至多搜索 10 条
```

至此，所有检索出来的数据都保存到数组中。操作数据就可以获取到相应数据。

3.3.6. 评分机制

Lucene 自身内置的评分算法是 TF-IDF 算法，也就是词频算法，TF 指词频(*Term Frequency*)，某一个词在所有分词中出现的频率；IDF 指逆文本频率指数(*Inverse Document Frequency*)，即所有文本数除以包含特定词的文本数的以 10 为底的对数。最后针对每个文档，将 TF-IDF 计算的结果相加，就是该文档最后的评分。

针对文献的评分，利用默认的评分机制效果并不理想。因为一是发表在不同等级期刊上的文献无法区别，所以需要期刊种类相关；二是一般新发表的文献的需求更大，所以需要发表时间相关。所以需要自定义实现 Lucene 的评分机制。先了解一下 Lucene 自身的评分机制，如式-1 所示[7] [8]。

$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{t\ in\ q}\left(tf(t\ in\ d) * idf(t)^2 * t.getBoost(\ ) * norm(t,d)\right) \quad (式\text{-}1)$$

$coord(q,d)$ 评分因子，用于计算检索命中的比例。诸如检索“石墨烯的氧化还原法”，经过分词器之后变成“石墨烯”和“氧化还原法”。另有一个文档“在石墨烯制备中的化学气相沉积法的改进”，

那么就只有“石墨烯”命中，而检索条件一共有 2 个元素，所以评分因子为 1/2，即 0.5；$queryNorm(q)$ 查询的标准查询，对所有的文档都是一致的，可以认为是一个常数值；$f(t\ in\ d)$ 指项 t 在文档 d 中出现的次数。就如 a 的例子中描述，石墨烯在文档中出现的次数为 1，那么最后计算结果就是 $\sqrt{1}$；$idf(t)$ 反转文档频率，最后的结果为 $t\lg\left(\frac{all\_doc}{index\_doc}\right)+1$，其中 all_doc 为所有文档，index_doc 为检索命中的文档数；$t.getBoost$ 某一个查询项的加权系数，针对某一类文献，可以把这一类文献关键词的系数设置得更大一点；$norm(t,d)$ 长度相关的加权因子，目的是为了将匹配结果相等的文档中，保证更短的在前面，这样更符合“完全匹配”原则[9]。

接下来是将额外的时间、期刊等参数添加到分数计算之中。因为时间、期刊等元素并不在搜索关键词中出现，所以并不能通过更改上述公式来达到调参的目的，需要重写 Lucene 的 Custom Score Query 和 Custom Score Provider 两类来将非检索参数转化为分数添加进最终计算分数中。针对期刊的系数调整，可以简单地根据 SCI/EI/中文核心/CSCD 等条件进行常数操作，根据 Lucene 计算出的原始分数动态调整；针对时间的系数调整，距离搜索日期越近，相应的系数影响越大，即基于时间衰减。同时又需要考虑到时间与文献质量并不具有强关联性，尤其是一些经典文献发表时间久远。因而基于时间的系数调整需要具备如下特征：既能保证最新的相关文献排在推荐列表的前部，又需要保证经典文献不会因为时间因素而排在推荐列表的后部。董立岩等人的研究中[10]，通过融合艾宾浩斯曲线的方式，将时间因素添加到协同过滤推荐算法中。本研究借用这种思路，将艾宾浩斯曲线的计算结果作为系数，还需要获取到每个文献的阅读次数和停留时间以筛选出经典文献。最终的计算公式如式-2 所示。

$$\begin{aligned} final\_score &= lucene\_score * papers\_score * date\_score \\ papers\_score &= e^{\frac{papers\_index}{papers\_base}*(lucene\_score+1)} \\ date\_score &= \ln\left(Ebbinghaus\left(papers\_date, now\_date\right) * papers\_hot\right) \end{aligned} \quad (式\text{-}2)$$

4. 性能测试

性能测试主要是从时间和空间上，比较数据库检索和搜索引擎检索的差异。如表 2 所示为性能比较。

Table 2. Database retrieval and search engine retrieval comparison
表 2. 数据库检索和搜索引擎检索性能比较

数据库	针对万方数据库中，初步 21,000+条数据，利用 MySQL 的 InnoDB 存储引擎，共占据空间 17,417 KB。其中元数据信息占据 9 KB (frm 文件)，数据信息占据 17,408 KB (ibd 文件)	利用 SQL 语句，SELECT * FROM table WHERE title LIKE “%关键字%”的方式，可以检索到相关信息。比如搜索光子晶体，最终查询出 39 条信息，耗时约 1 秒，如图 4 所示
搜索引擎	针对数据库存储内容，转换为搜索引擎的倒排索引结构保存，最终占据约 4719 KB 大小	在搜索界面输入关键词进行搜索，同样搜索光子晶体，搜索出 37 条信息。耗时约 0.02 秒，如图 5 所示

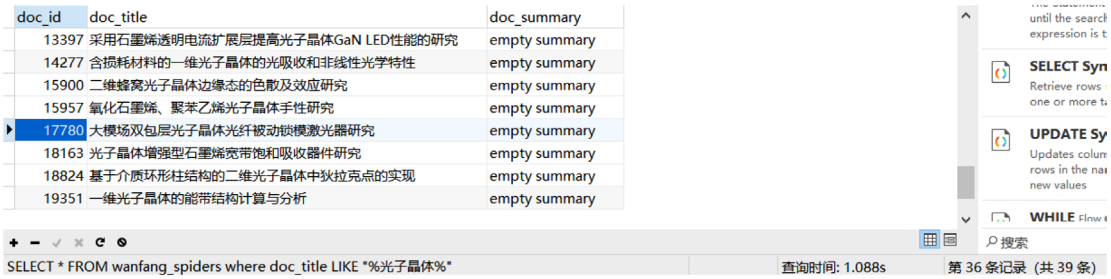

Figure 4. Database search results
图 4. 数据库检索结果

加载扩展词典：ext. dic
加载扩展停止词典：stopword. dic
cost time: 20
==========search results total :37

Figure 5. Lucene search results
图 5. Lucene 检索结果

从表 1 可以看出，在空间上数据库存储所需要的空间远大于搜索引擎所需要的空间。主要的原因在于文献中存在大量的重复词汇，数据库保存的是全文，数据表中的每一行都是完整的数据，当数据越来越多时，相应的冗余数据也会越来越多，就导致了所需的物理空间增大；而搜索引擎由于使用了倒排索引的结构，每个分词只会出现一次，分词出现的位置保存到对应的倒排列表中，当保存新的数据时，只是在倒排列表中添加新的位置而不是全部数据，因此占用的空间变化较小，但是更新列表花费的时间要比数据库直接添加一条数据要久。

同样的，在时间上搜索引擎花费的时间要远小于数据库搜索所需的时间。主要原因在于数据库的搜索条件并没有创建索引，因此就按顺序检索，时间复杂度为 O(n)。如果为每一个搜索条件创建索引，那么所需物理空间会迅速膨胀；搜索引擎由于倒排索引结构的存在，只需要在词典中查找到对应的分词后，查找倒排列表中的信息并按照分数排序即可，时间复杂度为 O(1)。

在准确度上，数据库精确搜索因为是一条一条的比对，所以准确率是 100%；但是在搜索文献时并不知道文献的具体名称，只能通过搜索关键词，即模糊搜索“LIKE”，因此会搜索出不同的数据。例如要搜索“氧化石墨”的文章，使用数据库模糊搜索会同时搜索“氧化石墨烯”的文章；然而使用 Lucene 全文检索，就像词典一样只会搜索“氧化石墨”的文章，“氧化石墨烯”的不会进行检索。因此，相对而言，Lucene 检索准确度要略高于数据库模糊搜索。

5. 展望与思考

本系统与现有的搜索引擎和数据库检索进行比对的结果如表 3 所示。

Table 3. Comparison between this system and existing systems
表 3. 本系统和现有系统的比较

	时效性	准确度	适用度	特点
百度学术	需要等待百度自身的爬虫更新网页索引才会搜索到相应文献	针对中文文献的准确度较高，但是由于是新服务，导致可检索到的数据较少	比较适合国内网络情况，但是内容相对比较贫乏	方便，大部分国人习惯使用百度搜索进行初次检索；对中文的支持比较好
谷歌学术	同样需要自身爬虫更新索引，无法第一时间获得最新文献	针对英文文献的准确度较高，但是对中文不友好	不适合国内大部分网络，需要搭载部分校园网和 VPN 才可以访问	网页排序使用 PageRank 算法而不是 TD-IDF 算法，相对推荐更准确
万方等数据库	会第一时间发布新的文献并可以本站搜索	模糊搜索会搜索更多的结果，不够精确；同时又提供按文献、作者分类等功能	速度较慢，没有权限则无法查看、下载文献，并且不同期刊被收录在不同的数据库	搜索引擎的数据来源，但是文献分散在不同的数据库，需要跨数据库搜索
本搜索系统	爬虫会随着数据库的更新而自动开始爬虫，保证时效性	针对中文文献实现精确查找，未来可以添加英文文献的搜索功能	无网络访问限制，搜索较快	整合各大数据库的文献数据，同时拥有自定义评分机制，更符合科研人员搜索需求

本系统很好地实现了石墨烯中文文献的检索功能，但是还有以下几点需要改进：

1) 可以从石墨烯出发，针对不同类别的文献创建相应的搜索引擎，所以需要针对不同类别的文献创立不同的专业词汇文档，保证中文分词的准确性；而针对英文文献，可以直接使用 Lucene 自带的 Standard Analyzer 进行分词的工作。

2) 本套系统的爬虫和搜索引擎都是建立在本机上，只有简单的人机交互界面。后续可以通过 Spring 相关框架、Tomcat 等中间件提供的技术部署在远程服务器上，同时制作相应的网络搜索界面[11]。

3) 由于文献是在不断更新的，爬虫需要不断的运行并更新数据库，搜索引擎需要根据新的数据库内容更新索引。未来需要优化爬虫和索引算法，从而保证可以及时更新最新文献和索引。

在网络检索系统中，可以根据用户文献查阅内容，设立推荐算法，推荐用户经常浏览的某一专题的最新文献。

基金项目

本工作得到了江苏省特种设备安全监督检验研究院科研项目基金(KJ(Y)2015012)支持。

参考文献

[1] 韩云凤. 基于 Lucene 的期刊论文库的检索技术研究[D]: [硕士学位论文]. 北京: 北方工业大学, 2018.

[2] 崔庆才. Python3 网络爬虫开发实战[M]. 北京: 人民邮电出版社, 2018.

[3] [日]山田浩之, 末永匡. 自制搜索引擎[M]. 北京: 人民邮电出版社, 2016.

[4] McCandless, M., Hatcher, E. and Gospodnetic, O. Lucene 实战[M]. 北京: 人民邮电出版社, 2011.

[5] 罗刚. 解密搜索引擎技术实战——Lucene & Java 精华版[M]. 第 3 版. 北京: 电子工业出版社, 2016.

[6] Bruce Croft, W., Metzler, D. and Strohman, T. 搜索引擎: 信息检索实战[M]. 北京: 机械工业出版社, 2010.

[7] 焦洋, 王纯, 韩静茹. 基于 Lucene 的科研查新系统构建[J]. 计算机技术与发展, 2018, 28(5): 193-196. https://doi.org/10.3969/j.issn.1673-629X.2018.05.043

[8] 邱敏明, 任洪敏, 顾利军. 基于 Lucene 的多源数据全文检索的研究与实现[J]. 现代计算机, 2018(22): 88-92. https://doi.org/10.3969/j.issn.1007-1423.2018.22.020

[9] 秦杰, 宋金玉, 张广星. 基于 Lucene 的本地搜索引擎研究与实现[J]. 计算机科学, 2014, 41(z2): 368-370.

[10] 董立岩, 王越群, 贺嘉楠, 等. 基于时间衰减的协同过滤推荐算法[J]. 吉林大学学报(工学版), 2017, 47(4): 1268-1272. https://doi.org/10.13229/j.cnki.jdxbgxb201704036

[11] 张全明, 曹开江, 罗毅, 等. 基于 Lucene 的全文检索技术在电力项目计划审核中的应用[J]. 科学技术与工程, 2018, 18(18): 188-191.