文章编号:2096-1472(2024)02-0044-05
DOI:10.19644/j.cnki.issn2096-1472.2024.002.009

一种新型高效全文检索引擎的设计

董宗然[1,2]，闻柏智[3]，朱 毅[1,2]

[1.大连外国语大学软件学院，辽宁 大连 116044；
2.大连外国语大学大数据图书情报研究中心，辽宁 大连 116044；
3.联通(辽宁)产业互联网有限公司，辽宁 沈阳 110041]
✉ dongzongran@163.com; wbz1234569@126.com; zhuyidl@163.com

摘 要:为了改善常规存储方式模糊查询性能较低的问题,提出一种针对大文本文档数据的高效模糊查询方法。通过对文档建立倒排索引,将索引以及部分文档信息提取到内存中以降低磁盘输入和输出(Input/Output, I/O)。根据内存中的倒排索引和数据库中主键形成的映射查询数据,然后通过相关度算法对这些数据进行排序,并以字典树作为搜索提示,实现高效的全文检索。实验结果表明:与 ElasticSearch 使用相同词集时,随着测试数据量的变化,所设计的全文检索引擎的查询效率是 ElasticSearch 效率的 80～1 200 倍,其效率优势随着数据量增加呈现反比例关系变化,并且在 17 919 条文档数据下,其内存占用不超过 2.5 GB,适合用于海量文档数据检索。

关键词:倒排索引;全文检索;检索引擎;模糊查询;字典树
中图分类号:TP391.3 **文献标志码**:A

Design of a New Efficient Full-text Search Engine

DONG Zongran[1,2], WEN Baizhi[3], ZHU Yi[1,2]

[1. *School of Software, Dalian University of Foreign Languages, Dalian* 116044, *China*;
2.*Big Data Library and Information Research Center, Dalian University of Foreign Languages, Dalian* 116044, *China*;
3.*China Unicom (Liaoning) Industrial Internet Co., Ltd., Shenyang* 110041, *China*]
✉ dongzongran@163.com; wbz1234569@126.com; zhuyidl@163.com

Abstract: In order to improve the low performance issue of fuzzy query in conventional storage, an effective fuzzy query method for large-text document data is proposed. By establishing inverted indexes on documents, the indexes and some document information are extracted into memory to reduce disk I/O. The data is queried based on the maps formed by inverted indexes in memory and the primary keys in database, and then these data is sorted by the relevance algorithm, and the Tire tree is used as the search prompt to achieve an efficient full-text search. The experimental results show that when using the same word set as ElasticSearch, the efficiency of the designed full-text search engine is 80 to 1 200 times that of ElasticSearch, depending on the amount of test data. With 17 919 document data, the memory size does not exceed 2.5 GB, making it suitable for massive document data retrieval.

Key words: inverted index; full-text search; search engine; fuzzy query; tire tree

0 引言(Introduction)

全文检索是指根据字符串在大量文档中找到与该字符串匹配的文档集合,常规文档管理手段基于文档遍历,查询效率极低,因此全文检索引擎研究对构建高效、智能的信息检索系

收稿日期:2023-05-30
基金项目:2022 年度辽宁省高等学校基本科研项目(LJKMZ20221547)

统,提高信息处理能力具有重要的意义。

文档管理系统通常使用关系型数据库,模糊查询使用"like"配合"%"关键字,索引类型为"B-树(Oracle)""B+树(MySQL)"等[1],前者会进行大量磁盘 I/O [2],后者也会进行4次磁盘I/O,并且高并发时会占用大量内存[3]。MySQL 内置了 N-gram 全文检索插件,用于文本查错、校对等[4],但对中文支持较差。目前,主流的全文检索技术如 Lucene、ElasticSearch 会先构造词集,遍历文档数据后建立"词-文档"数据,该数据称为"倒排索引"[5-6],倒排索引常存储在磁盘中,读取过程会产生大量磁盘 I/O。聂玉峰等[7]将倒排索引存储介质换成固态硬盘,提高了时间效率,但成本较高。袁琴琴等[8]使用 Oracle 组件搭建全文检索系统,但 Oracle 是收费数据库,并且搭配组件使用会对服务器运行造成负担。葛云生等[9]基于 Zookeeper 分布式资源管理引擎与 Lucene 搜索引擎搭建分布式检索系统,由于该系统需要部署集群,所以成本较高。

本文将介绍一种使用网络小说文本作为处理对象,基于开源框架和技术搭建的高效全文检索引擎系统的设计与实现,在保持较低内存消耗的情况下实现对文本的高效检索。

1　问题模型(Problem model)

1.1　传统文档数据存储方式

对于一般的软件系统,常将文档本身存储在磁盘中,将文档的路径以及关联信息存储到关系型数据库中,在数据量较小或不需要提供全文检索的情况下,这种方式只需存储必要的信息,节省了磁盘空间。但是,当需使用文档的一部分搜索出文档信息时,需要遍历全部的文档以及每篇文档中存储的所有信息,这种搜索算法的时间复杂度为 $O(MN)$,其中 M 表示文件总数,N 表示每个文件的平均字符数,通常无法满足用户性能需求。传统索引的存储结果如表1所示,传统存储方式查询流程如图1所示。

表1　传统索引的存储结果

Tab.1 The storage results of conventional indexes

文档编号	文档内容
1	我喜欢大连外国语大学
2	小明吃了兰州拉面

1.2　系统改进模型

系统会对每篇文章分词,建立词-文档在数据库中的主键集合形式的倒排索引,通过倒排索引搜索文档的效率极高,为节省提取倒排索引所需的时间,系统将倒排索引数据存放在内存中,最终将主键集合作为查询条件从数据库中提取数据,当数据库索引树为"B+树"时,查询时间复杂度为 $M\times\log(N)$,其中,M 为文档主键个数,N 为数据库索引树(B+树)中页结点的元素数量[10]。倒排索引的存储方式如表2所示。系统改进后的查询流程如图2所示。

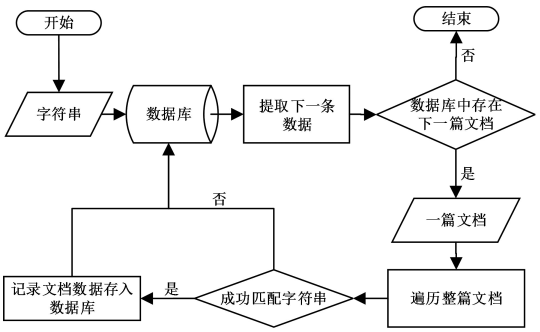

图1　传统存储方式查询流程

Fig. 1 The query process of the traditional storage method

表2　倒排索引的存储方式

Tab.2 The storage method of inverted index

索引词	文档编号
我	1
喜欢	1
大连外国语大学	1
小明	2
吃了	2
兰州拉面	2

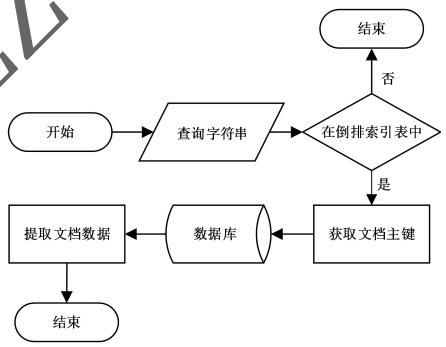

图2　系统改进后的查询流程

Fig. 2 The query process of the improved model

2　系统和算法设计(System and algorithm design)

2.1　总体设计

系统分为9个后端模块和2个前端模块,系统结构如图3所示。

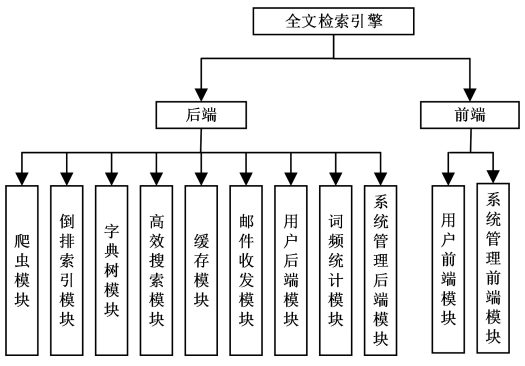

图3　系统结构

Fig. 3 System architecture

其中，爬虫模块、倒排索引模块必须在服务运行前运行一次。本系统使用的文档数据、用户、管理员信息存储在 MySQL 中，索引信息存储在 MongoDB 中，缓存信息存储在 Redis 中，数据库表及其关系如图 4 所示。

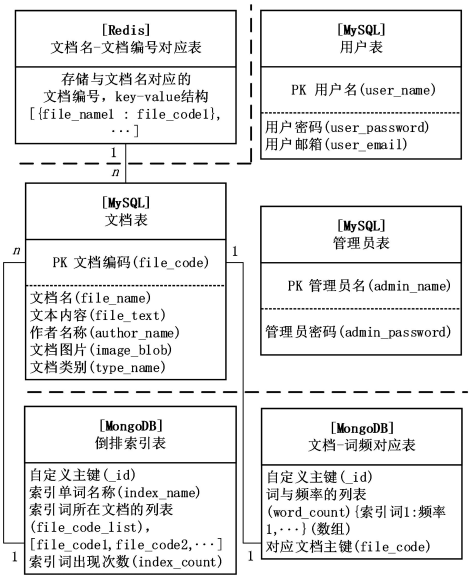

图 4 数据库表及其关系

Fig. 4 Database tables and their relationships

2.2 核心功能设计

2.2.1 构建倒排索引与文档词频表

遍历文档表，依次获取文档的 file_code（主键）与文本内容，使用 jieba 库的 lcut_for_search()方法对文本内容进行分词，得到列表形式的词集，记为 word_list，使用 collections 库中 Counter()方法统计 word_list 中的元素出现次数，即词频，生成字典，记为 word_dict，删除出现频率低于 5 的词，借助哈工大停用词表[11]，删除停用词。将 word_dict 转化成 json 格式存储到 MySQL 中，在 MongoDB 中建立集合，表结构参见图 4 中的倒排索引表。遍历上述 word_dict 的 key，即单词，若倒排索引集合存在该词，则将文档主键插入 file_code_list，若集合中不存在该词，则新建以 key 值为 index_name，file_code_list 为空列表的文档，构建倒排索引和词频表流程如图 5 所示。项目运行时，高效搜索模块使用字典对象将倒排索引加载到内存中，词频统计模块使用 pandas.DataFrame 对象将文档词频数据加载到内存中。

2.2.2 构建字典树内存数据库

字典树又称前缀树、Trie 树，是一种高效字符串查询树，相同的前缀字符共享同一节点，在节省大量存储空间的同时，提高查询效率[12]，如存储词集[沈阳，沈阳人，沈老师，沈阳房地产，沈阳房屋]，字典树存储结构如图 6 所示。项目使用字典树完成文档名称和索引词的搜索提示。项目使用字典类型嵌套实现字典树，字符串中前一字符作为后一字符的 key，并将其封装进类中，提供插入、删除、查询方法。

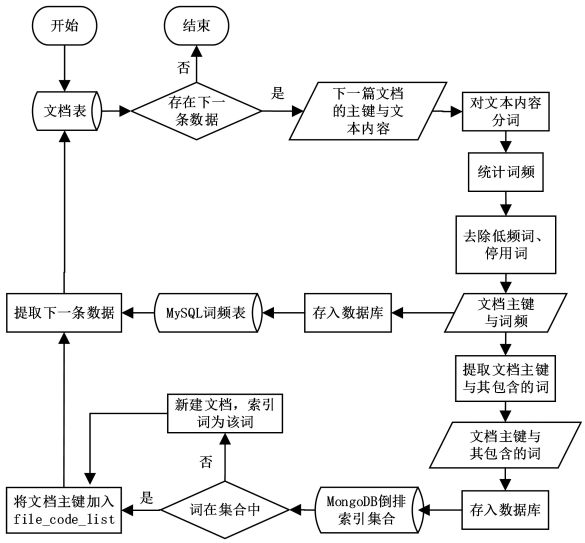

图 5 构建倒排索引和词频表流程

Fig. 5 The process of building inverted index and word frequency table

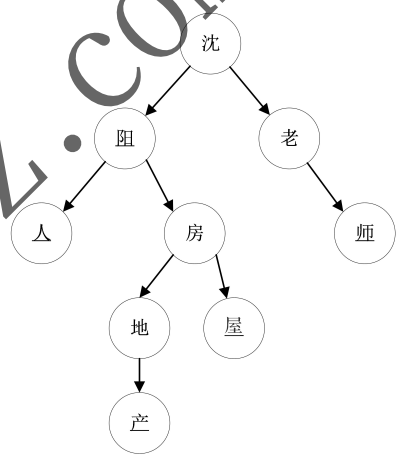

图 6 字典树存储结构

Fig. 6 Dictionary tree storage structure

2.2.3 搜索功能设计

搜索为项目的核心功能，即通过用户输入的信息，查询出匹配的文档集合。用户在前端的搜索框中输入字符串，该字符串不一定是完整的索引词或文档名称，此时前端会调用后端字典树模块的接口，输出以该字符串为前缀的文档名称和索引词集合（图 7），当用户点击目标文档名称时，调用后端 Redis 模块的接口，获取该文档的主键，在 MySQL 数据库中查询该文档的信息输出到前端页面；当用户点击目标索引词时，调用后端高效搜索模块的接口，获取包含该词的文档的主键集合，构造 SQL 语句，将主键集合作为 where 条件从 MySQL 数据库中查询数据并输出到前端；当用户点击回车键时，判定该字符串为句子，调用后端高效搜索模块的接口，对该句子进行分词生成词集，对该词集的每个词对应的文档主键集合进行合并、去重，生成新的主键集合，后续操作依次按照搜索词的方式进行。搜索功能流程如图 8 所示。

图 7　前端搜索提示

Fig. 7 Front side search tips

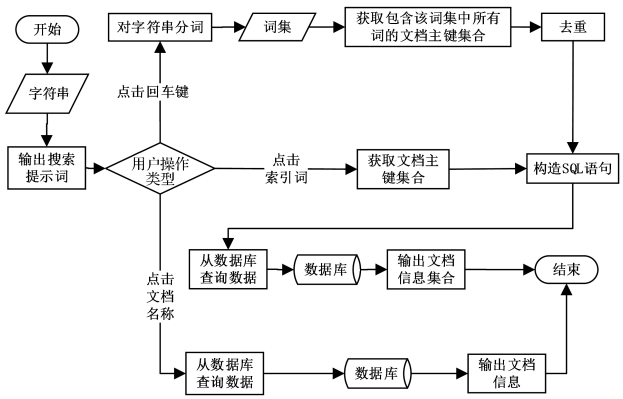

图 8　搜索功能流程

Fig. 8 The process of search function

2.2.4　排序功能设计

当系统的文本量足够大时,同一 query 通常会查询出大量的文档,用户很难从众多的文档中筛选出匹配度最高的文档,因此需要一种帮助用户筛选高匹配度文档的方法.系统使用 TF-IDF (Term Frequency-Inverse Document Frequency)算法[13]计算 query 与文档的相关度,依照相关度对文档排序并逆序输出。用户点击搜索结构页面的排序按钮,前端将用户输入的 query 和所有查询结果的文档主键提交给高效搜索模块,高效搜索模块不断地向词频统计模块发送请求,依次得到每篇文档的总词数,记为 word_count,每篇文档中 query 词出现次数占该篇文档中所有词数的比重,记为 *TF*,随后计算出包含 query 词的文档数占总文档数的比重,记为 *IDF*,生成哈希表(字典)数据,其中 key 为文档主键,value 为该篇文档按照 *TF* 与 *IDF* 的乘积逆序排序后的位置,将其以 json 格式返回给前端。前端收到 json 数据后,将其解析为 JavaScript 中对象类型,依照该对象对文档信息进行排序。使用这样的排序算法,前端只需要向后端返回文档主键的集合,不需要将文档信息返回,减轻了系统的压力。排序算法流程如图 9 所示。

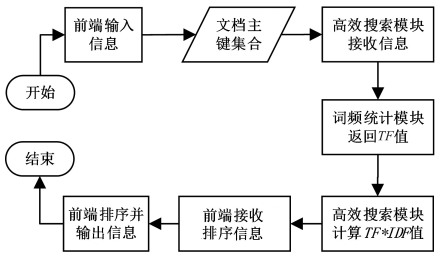

图 9　排序算法流程

Fig. 9 The process of sorting algorithm

3　实验及结果分析(Experiment and results analysis)

为了分析系统效率,梯度增加测试的词集长度,并与主流的同类产品 ElasticSearch(7. 17. 4 版本)对比,开展系统压力测试和性能测试。

开发环境:PyCharm 2021,WebStorm 2021,Python 3,Vue 3,Flask 1. 1. 2。

运行环境:Inter i7-7700HQ CPU,RAM 16. 0 GB,Windows 10 x64。

3.1　系统性能测试

使用爬虫模块抓取“爱九九小说网”的 17 919 篇文档,将其作为测试文档,将测试文档信息存储到 MySQL 中,从倒排索引的 key 中随机提取 10 000 个索引词,依次截取前 10、100、1 000、5 000、10 000 个词作为输入数据,词集的形式如[‘风有’,‘回母’,‘合个’,‘小梨窝’,‘那脸’,……],向系统输入词集,返回包含该词集中任意一词的文档的名称与主键。本文设计的系统性能测试结果如表 3 所示。从表 3 可以看出,测试数据量与总耗时成正比,符合对数函数曲线,平均每次搜索耗时随着数据量的增长而降低,这是因为一个索引词往往对应多篇文档,系统在内存中已经对重复的文档进行去重,保证在数据库中不会有重复的磁盘 I/O 操作。

表3　本文设计的系统性能测试结果

Tab.3 Performance testing results of the proposed system

总耗时/s	平均每次搜索耗时/s	测试数据量/个
0. 100 309	0. 010 030 900	10
0. 166 666	0. 001 666 660	100
0. 570 043	0. 000 570 043	1 000
1. 316 560	0. 000 263 312	5 000
2. 397 620	0. 000 239 762	10 000

3.2　与 ElasticSearch 的对比测试

在 ElasticSearch 中使用同样的词集进行测试,ElasticSearch 性能测试如表 4 所示。对比表 3 和表 4 可以看到,ElasticSearch 的各项数据之间的关系与本文设计的系统相同,但效率远远低于本系统,随着测试数据量的变化,即数据量

从10～10 000增加时，本文系统效率优势相对ElasticSearch逐渐降低，呈现反比例变化关系。但优势仍较明显，两者效率差为80～1 200倍，原因主要为ElasticSearch是基于磁盘的存储软件且索引结构为字典树，设字符串的长度为n，单次查询需要n次磁盘I/O，当数据量过大时，需要的磁盘I/O次数也会非常多[14]，而本文设计的系统将索引文件提取到内存中，只有向数据库中提取数据时才会进行磁盘I/O，在内存中的数据以哈希表形式存储，时间复杂度为O(1)，选用MySQL数据库，其索引结构为B+树，设一页节点的长度为n，改善查询语句以充分发挥MySQL性能，查找时间复杂度为O(log(n))[15]。

表4 ElasticSearch性能测试

Tab.4 ElasticSearch performance testing

总耗时/s	平均每次搜索耗时/s	测试数据量/个	与本文设计系统的耗时比值
105.058	10.505 800 0	10	1 047.343 7
198.628	1.986 280 0	100	1 191.772 7
201.328	0.201 328 0	1 000	353.180 3
203.626	0.040 725 3	5 000	154.665 1
202.440	0.020 244 0	10 000	84.433 7

4 结论(Conclusion)

本文设计了一种高效的全文检索引擎系统，该系统的核心设计思想为对文本建立倒排索引，将索引数据提取到内存中，以少部分内存空间为代价大大降低了磁盘I/O量，系统的搜索效率提升十分可观，单次搜索的平均时间远小于ElasticSearch，在17 919条文档的情况下，内存占用也不超过2.5 GB，适合大型企业作为对海量文档的搜索引擎，并且系统内置了对文档的增、删、改、查功能，因此也适合个人用户作为对文档的管理系统。由于系统依赖的模块较多且没有做分布式的优化，因此后期工作将围绕减少模块依赖，增强系统的可移植性，去中心化以便分布式搭建，减少重复数据以减轻内存占用等方面做进一步优化。

参考文献(References)

[1] 吕盛泽. 基于数据挖掘的数据库索引优化方法研究[D]. 杭州：浙江理工大学，2019.

[2] 陆家俊，顾梅. Oracle查询优化的研究与应用[J]. 信息技术与信息化，2022(1)：57-60.

[3] 林荣杭，刘小英. MySQL索引改进的B+树的研究[J]. 电脑知识与技术，2022，18(16)：12-13，18.

[4] 王琼，旷文珍，许丽. 基于改进的N-gram模型和知识库的文本查错算法[J]. 计算机应用与软件，2021，38(10)：310-315，320.

[5] GIRIDHARAN J，VAIRAVAN S V. Inverted index and interval lists for keyword search[C]//IEEE. Proceedings of the IEEE：2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE). Piscataway：IEEE，2014：1-4.

[6] 刘炜，白晓丹，佘维，等. 基于倒排索引的可搜索加密数据共享方案[J]. 计算机工程与应用，2023，59(10)：270-279.

[7] 聂玉峰，陈雪帆. 基于固态硬盘的在线全文索引管理策略研究[J]. 计算机工程与设计，2013，34(2)：539-544.

[8] 袁琴琴，李志勋，吕林涛. 基于Oracle组件的数据采集与全文检索系统设计与优化[J]. 现代电子技术，2016，39(8)：37-40，44.

[9] 葛云生，孔杰. 分布式全文检索技术的研究及应用[J]. 计算机工程与设计，2018，39(9)：2997-3001.

[10] ADMIN. B+树搜索时间复杂度到底是什么：mlogmN/logN[EB/OL]. (2021-08-24)[2023-03-22]. https://zhuanlan.zhihu.com/p/402951795.

[11] goto456. 中文常用停用词表(哈工大停用词表、百度停用词表等)[EB/OL]. (2020-03-04)[2023-03-29]. https://github.com/goto456/stopwords.

[12] 黄诚，赵倩锐. 基于语言模型词嵌入和注意力机制的敏感信息检测方法[J]. 计算机应用，2022，42(7)：2009-2014.

[13] 杜永兴，牛丽静，秦岭，等. 基于改进TF-IDF算法的牛疾病智能诊断系统[J]. 计算机应用与软件，2021，38(2)：50-53，57.

[14] MEENA S M，VALA V，TIWARI H，et al. File based Trie for embedded devices[C]//IEEE. Proceedings of the IEEE：2019 IEEE 9th International Conference on Advanced Computing(IACC). Piscataway：IEEE，2019：163-170.

[15] 王淼. MySQL查询优化与研究[J]. 信息记录材料，2022，23(5)：227-229.

作者简介：

董宗然(1981-)，男，博士，副教授。研究领域：人工智能技术，文献推荐。

闻柏智(1998-)，男，本科，初级工程师。研究领域：全文检索算法。

朱 毅(1979-)，男，硕士，副教授。研究领域：人工智能和大数据。