

法律搜索引擎索引系统同步模块的设计与实现

方跃胜^{1,2}, 姚宏亮¹

(1. 合肥工业大学 计算机与信息学院, 安徽 合肥 230009;

2. 安徽水利水电职业技术学院, 安徽 合肥 231603)

摘 要:随着互联网的迅猛发展,信息爆炸出现在人们面前,如何快速、方便、有效地从信息的海洋中找到需要的信息逐渐成为越来越重要的问题,搜索引擎也应需而生。按照搜索引擎的“全、准、快、好”评测标准,目前的许多搜索引擎已经不能满足人们的需求,存在很多问题。针对这一问题,设计了一个应用于中英文法律信息检索的搜索引擎系统。搜索引擎系统的设计涉及多种关键技术,而索引是搜索引擎的核心,搜索引擎后台工作的目的就是要建立关键词和文档的索引。建立高效、快速、优化的索引结构会对搜索引擎的性能产生决定性的影响。通过对索引系统功能模块的分析,引入了索引系统的设计问题。围绕索引系统关键词同步和文档同步,提出了一种适用于中英文法律信息检索的搜索引擎索引系统同步模块的设计方法与 PHP 程序实现,并通过实验数据分析其优劣。

关键词:中文法律;搜索引擎;索引系统;同步模块

中图分类号:TP391

文献标识码:A

文章编号:1673-629X(2011)03-0137-05

Design and Implementation of Law Search Engine Index System's Synchronization Module

FANG Yue-sheng^{1,2}, YAO Hong-liang¹

(1. School of Computer and Information, Hefei University of Technology, Hefei 230009, China;

2. Anhui Water Conservancy Technical College, Hefei 231603, China)

Abstract: Along with the rapid development of the Internet, information explosion exists in front of people, the problem that how to find required information from the ocean of information by fast, convenient and effective method is becoming more and more important, search engine generates according to people's needs. In accordance with search engine's "full, accurate, fast and good" evaluation standards, lots of search engines can not meet people's needs currently, there are many problems needed to solve. To solve this problem, have designed an information retrieval system that applied to Chinese and English law. The design of search engine involves a lot of key technologies, but the index is the coral technology, the purpose of search engine at the background work is to establish the index of keywords and documents. Establishment of efficient, fast, optimized search engine index structure will have a decisive impact performance. By function module's analysis of the index system, this article has inducted the index system's design question. Regarding the index system key word synchronization and the documents synchronization, has proposed a kind of design method that is suitable in Chinese and English law about information retrieval search engine index system synchronization module, and realized in PHP language, and analyzed its fit or unfit quality through the empirical datum.

Key words: Chinese law; search engine; index system; synchronization module

0 引言

按照搜索引擎的“全、准、快、好”评测标准^[1],目前的搜索引擎仍不能很好地满足用户需求。存在的主要问题是:得到的搜索结果精确度明显不够,存在大量

的重复信息和垃圾信息;对于返回结果的重要性判断不恰当;智能化程度较低,搜索引擎的查询形式单调^[2]。

因此,开发出性能优良、界面友好的搜索引擎是一项具有重要意义的工作。搜索引擎系统的设计涉及到数据收集、中文分词、索引组织、检索结果排序、用户行为分析等关键技术,而索引是搜索引擎的核心,建立高效、快速的索引结构会对搜索引擎的性能产生决定性的影响^[3]。

针对这一问题,我们开发了一个中英文法律案例

收稿日期:2010-07-03;修回日期:2010-10-10

基金项目:国家重点基础研究发展计划资助项目(2009CB326203, 2009CB219708);国家自然科学基金资助项目(60705015)

作者简介:方跃胜(1975-),男,安徽舒城人,硕士,讲师,研究方向为人工智能和图形学;姚宏亮,博士,副教授,主要研究方向为人工智能和数据挖掘。

检索的搜索引擎系统。文中仅通过对索引系统功能模块的分析,围绕索引系统关键词同步和文档同步,提出了一种适用于该搜索引擎的索引系统同步处理模块的设计方法与 PHP 实现,并通过实验数据分析其优劣。

1 索引核心模块分析

索引管理模块是索引数据库中的主控程序,在搜索引擎中一直运行,它不停的监控文档元数据库和关键词元数据库,当发现有未同步的关键词或文档时,调用同步模块对关键词和文档进行同步。它包括数据库监控模块和同步处理调度模块^[4]。

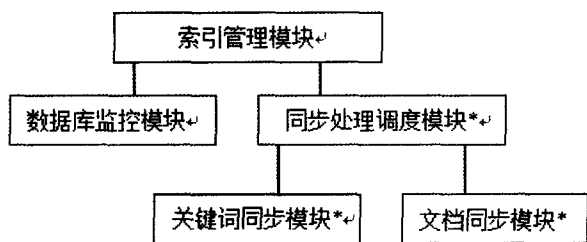


图1 索引核心模块

各个模块的功能:

(1) 数据库监控模块通过对数据库的监控,发现未同步的文档或关键词,再将未同步的文档或关键词传入同步处理调度模块。

(2) 同步处理调度模块的主要目的是对线程池中的多个同步处理线程进行调度,保证系统同步处理的并发性。同步处理模块将未同步关键词和文档进行同步处理,生成索引,存入索引库,建立和维护可供检索的文档索引数据库,也叫索引生成程序。同步处理模块又分为关键词同步处理模块和文档同步处理模块。

(3) 关键词同步处理模块用于同步关键词元数据库中未同步的关键词,将这些关键词与所有已同步文档建立索引,矢量化后存储在索引数据库中。

(4) 文档同步处理模块则用于同步文档元数据库中未同步的文档,将这些文档所有已同步关键词建立索引,矢量化后存储在索引数据库,并通过中文分词程序自动发现新词,更新关键词元数据库。

关键词和文档同步定义^[5]:

1) 关键词同步: 如果一个关键词 Keyword, 对所有文档的都进行了匹配操作, 并将匹配结果存储在索引数据库中, 称为该关键词已同步。把该关键词与所有文档的匹配过程叫做关键词同步过程。

2) 文档同步: 如果一个文档 Document, 系统中所有关键词都对该文档进行了匹配操作, 并将匹配结果存储在索引数据库中, 称为该文档已同步。把该文档与所有关键词的匹配过程叫做文档同步过程。

2 索引设计

2.1 索引构筑

1. 数据的索引。

数据索引分为两个步骤: 词的识别和索引库的建立^[6]。

(1) 词的识别。

就是通过词典库对法律案例文档中的词进行识别。识别出文档的每个词, 并分配唯一的 wordID 号, 用于为数据索引的索引模块服务。

(2) 索引库的建立。

需要建立两种标引: 文档标引和关键词标引。

1) 文档标引分配每个文档页一个唯一的 docID 号, 根据 docID 标引出在这个文档中出现过多少个 wordID, 每个 wordID 出现的次数、位置、大小写格式等, 形成 docID 对应 wordID 的数据列表;

2) 关键词标引是对文档标引的逆标引, 根据 wordID 标引出这个词出现在哪些文档(用 docID 表示), 出现在每个文档的次数、位置、大小写格式等, 形成 wordID 对应 docID 的列表。

2. 使用混合技术来构建倒排索引。

倒排表的存储是整个算法的主要部分, 本系统提出了三种方案: 直接将倒排表放入内存; 将倒排表存入硬盘上的文件; 将倒排表及必要的文件存入数据库(此处使用 MySQL)。

混合索引是在建立倒排索引过程中的一种索引词选择方法与技术, 构建方法如下^[7,8]:

(1) 处理所有的文档。

1) 初始化内存索引, 为词汇表使用动态结构, 而倒排列表使用静态编码; 把列表要么存在可动态调整大小的数组中, 或者一个链式块中;

2) 读取文档和倒排<d, fd, t>指针, 放到内存索引中, 一直这么做直到用完内存;

3) 把这个临时索引保存到硬盘上, 包括其词汇表。

(2) 将这些零碎的部分索引合并成一个索引文件。

3. 索引检索模型。

在本搜索引擎的实现中, 考虑到用户提交查询后不可能长时间等待检索结果, 项目中索引技术将按照总体设计方案中的设计思想采用布尔逻辑模型和向量空间模型组合建立一个索引检索模型^[9]。

2.2 索引同步设计

索引的创建一般来说可以分为两个步骤: 关键词同步和文档同步。

(1) 关键词同步。

在程序的实际运行中加入了添加、删除等选项。

同时对原来的算法进行了一些改进,开始建立的是一个运用散列表^[10]的方法进行关键词同步。对所有的关键词用 ELFHash 函数运算,计算出每个关键词的散列码。将相同散列码的关键词存在散列表的同一个位置,并记录其所在文档信息。存储结构如图 2 所示。

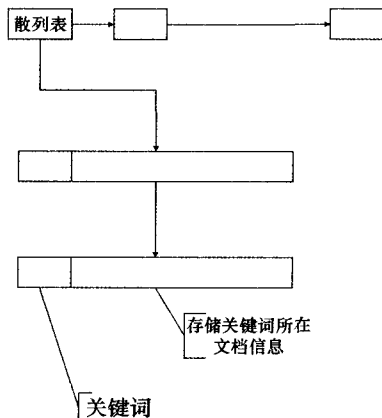


图2 关键词同步散列表式存储结构

但是在实际操作中用了很多的类以及静态数组的数组,在使用七万个关键词的时候会占用很大的空间,另外,在使用很多类的时候造成程序的可读性和效率方面有很大的障碍^[11]。PHP 自带的字符串查找函数已经运用到散列表的方法,查找效率本身已经很快。因此对关键词同步重新设计了存储结构。在新的存储结构中,建立了两个数组:一个用来存储关键词、另一个用来存储与关键词索引值相同的文档编号,使其一一对应。在程序运行的过程可以直接通过 `array_search` 函数来查找到相应的关键词及其对应的文档信息^[12]。存储结构如图 3 所示。

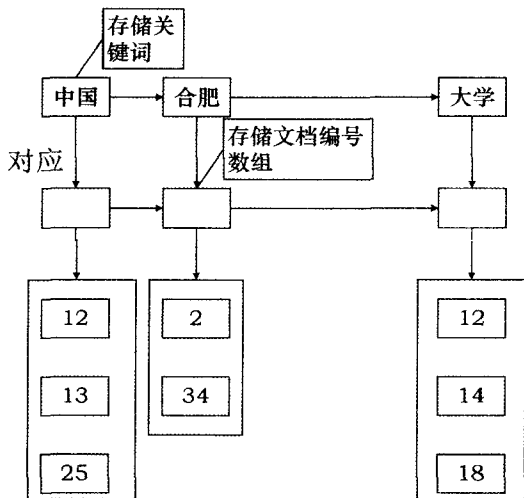


图3 关键词同步的两个一维数组存储结构

在程序运行的过程中,首先初始化关键词数组并建立文档类数组并使其一一对应,然后遍历所有的文档并将关键词所在文档添加进入文档类数组。在遍历文档时,得到该文档分词后的某一个关键词,判断关键

词所在文档是否在文档数组中的对应位置。若在其对应的位置则证明该文档已经添加进入其对应位置,不做任何处理;若不在其对应位置,则将其信息添加进文档数组。同步流程图如图 4 所示。

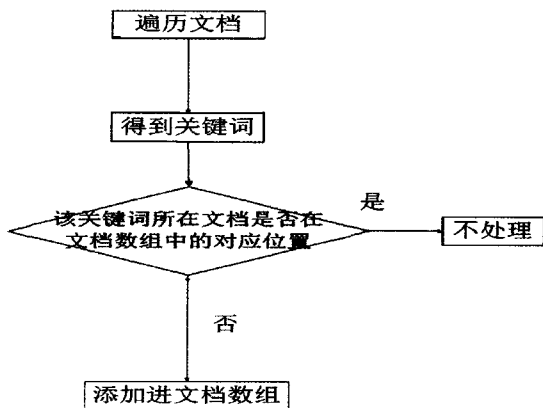


图4 关键词同步流程图

(2) 文档同步。

对于文档同步,依旧使用原有的算法,建立一个关键词类的数组,用数组的下标作为文档的编号,存储关键词在此文档中出现的次数、位置等信息。其结构如图 5 所示。

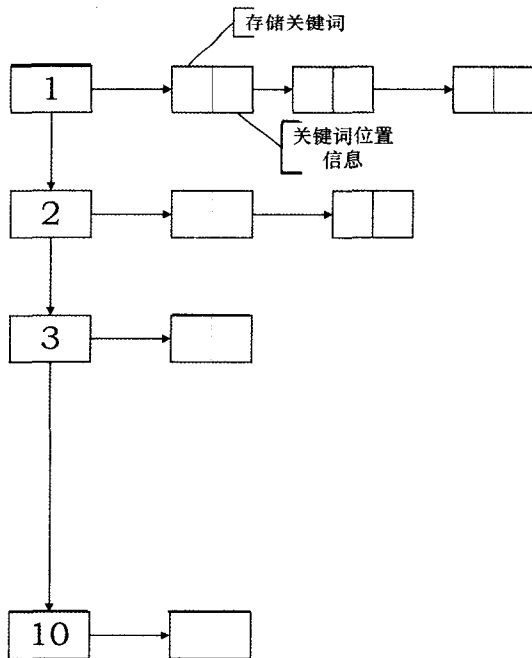


图5 文档同步的两个一维数组存储结构

在程序运行的过程中,对分词后的关键词数组进行遍历得到某一个关键词,然后判断该关键词是否在该篇文档中的关键词数组中,如果在数组中就添加其位置信息,如果不在则添加关键词及其位置信息进入关键词数组中。具体流程如图6所示。

在程序的实际运行中,对文档的遍历以及位置信息不断地记录下来,并且在程序的实际运行中,并且在

对文档进行编号时,用文档进入程序的顺序作为文档的编号。

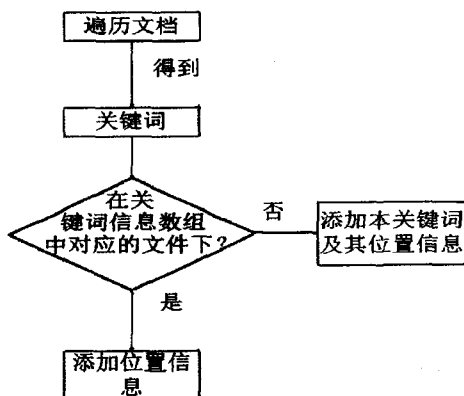


图 6 文档同步流程图

3 索引同步 PHP 实现

3.1 关键词同步 PHP 程序

```

<? php
function GetNum() {
    $var = 6000;
    return $var;
}
//关键词同步[12,13]
// $time_start = microtime(true);
echo '<br>';
//建立包含文档信息的类
$Word_num = GetNum();
function Insert_num($var) {
    $key = array();
    $file_name = "D:\\Data\\wor.txt";
    $file_stream = fopen($file_name, 'r');
    for($i=0; $i<=$var; $i++) {
        $line = fgets($file_stream);
        array_push($key, $line);
    }
    return $key;
}
class ArticleArray {
    public $article;
    //构造函数
    function ArticleArray() { $this->article = array(); }
    //插入一篇文档
    function InsertArticle($var) { if(array_search($var, $this->article) == FALSE) { array_push($this->article, $var); } }
}

```

```

//显示文档信息
function Show() {
    print "包含的文档有:"; for($i=0; $i<count($this->article); $i++) { print $this->article[$i]. " "; }
    echo '<br>';
}
//删除一篇文档
function Del($var) { if(array_search($var, $this->article) != false) { array_splice($this->article, array_search($var, $this->article), 1); } }
//文档与关键词一一对应
class WordToArticle { public $wordarray; public $articlearray;
    //构造函数将关键词读入内存,并建立与其相对应文档信息类
    function WordToArticle() { $this->wordarray = array();
        // $key = array("中国","合肥","法院","著作权","专用权","不正当","竞争","纠纷","五金","电影");
        $key = Insert_num(GetNum()); for($i=0; $i<count($key); $i++) {
            $line = $key[$i]; array_push($this->wordarray, $line);
            $this->articlearray[$i] = new ArticleArray();
        }
        //将某篇文档及其分词后的数组插入文档信息类
        function Insert($art, $var) { for($i=0; $i<count($var); $i++) {
            $loc = array_search($var[$i], $this->wordarray); $this->articlearray[$loc] -> InsertArticle($art);
        } }
        //删除某个词
        function DelWord($word) { $loc = array_search($word, $this->wordarray); array_splice($this->wordarray, $loc, 1); array_splice($this->articlearray, $loc, 1); }
        //删除某篇文档
        function DelArticle($art) { for($i=0; $i<count($this->articlearray); $i++) { $this->articlearray[$i] -> Del($art); } }
    }
}

```

```

//插入一个词
function InsertWord( $ var ) { array_push( $ this->
wordarray, $ var ); $ this->articlarray[ count( $ this->
wordarray ) ] = new ArticleArray();
}
//显示信息
function Show() { for( $ i = 0; $ i < count( $ this->
wordarray ); $ i++ ) { print " 词语: ". $ this->wordarray
[ $ i ]. "    "; $ this->articlarray[ $ i ]->Show();
}
}
//主函数开始(对以上各个类及其函数进行测试)
/*
$ W = new WordToArticle();
print_r( $ W->wordarray );
echo '<br>';
print array_search( " 竞争 ", $ W->wordarray );
echo '<br>';
$ article_1 = array( " 中国 ", " 五金 ", " 专用权 ", "
五金 ", " 竞争 ", " 纠纷 ", " 著作权 ", " 专用权 ", " 纠
纷 " );
.....
$ article_10 = array( " 中国 ", " 竞争 ", " 合肥 ", "
竞争 ", " 竞争 ", " 不正当 ", " 竞争 ", " 纠纷 " );
for( $ i = 1; $ i <= 10; $ i++ ) { $ file_name = " ar-
ticle_". " $ i ";
$ W->Insert( $ i, $ file_name );
}
$ W->Show();
time_end = microtime(true);
$ time_title = $ time_end - $ time_start;
echo " ##### ". $ time_title; */
? >

```

3.2 文档同步 PHP 程序

鉴于篇幅限制,代码及同步实验结果图此处从略。

4 结束语

用两种方法实现文章同步和关键词同步:(1)二维数组方法;(2)两个一维数组方法。二维数组法结构简单、清楚,但占用内存空间较大,并且在检索时会浪费很多的时间,因为它要遍历所有的案例,不管该案

例包不包含这个关键词。两个一维数组法在检索时就比二维数组方法要快,且可从数据上和理论上进行验证。通过 PHP 实验结果与数据分析,在相同的条件下,运用此方法和二维数组方法相比,在数据较小的条件下两者没有太大的区别,但在数据量比较大的时候,可以看出此种方法的优点。从理论上说,运用二维数组法在做关键词同步和文档同步必须对所有的文档或关键词进行一次遍历,在很多情况下就会浪费很多时间。但在运用一维数组法时,只需要对含有此关键词的文档进行遍历。在实际使用中认为应该使用此方法。

该搜索引擎索引系统在 Windows 环境下采用 Apache+PHP+MySQL 黄金搭档来实现,运行环境为双核 2.0GHz CPU,内存是 2GB。通过简单的测试了 70000 个关键词,100 篇文章,每篇文章 2000 个词语,一共用了 15 分钟。以后同步的时候,可以将其分开同步,比如有七千篇文章,可以用 10 * 700 这样的方案来解决。

参考文献:

- [1] Tisseghem P, Fastrup L. 深入索引和搜索引擎[M]. 北京:清华大学出版社,2009.
- [2] 李晓明,闫宏飞,王继民. 搜索引擎:原理、技术与系统[M]. 北京:科学出版社,2005.
- [3] 郑榕增,林世平. 基于 Lucene 的中文倒排索引技术的研究[J]. 计算机技术与发展,2010,20(3):55-56.
- [4] 陈海波. 基于自动分词的企业文档搜索引擎设计与实现[D]. 西安:西北工业大学,2007:68-70.
- [5] 陈海波,张新家. 企业文档服务器中英文搜索引擎的设计与实现[J]. 微处理机,2009(2):122-125.
- [6] 卢 亮,张博文. 搜索引擎原理、实践与应用[M]. 北京:电子工业出版社,2007.
- [7] 彭 波. 搜索引擎检索系统的效率优化[D]. 北京:北京大学,2004:12-15.
- [8] 杨安生. 基于倒排表的中文全文检索研究[J]. 情报探索,2009(7):77-80.
- [9] Witten L H, Moffat A. 深入搜索引擎——海量信息的压缩、索引和查询[M]. 北京:电子工业出版社,2009.
- [10] 肖 丽. 哈希查找中散列函数的运用[J]. 技术与市场,2009(8):18-19.
- [11] 格拉夫,库辛. 搜索引擎优化[M]. 杨明军 译. 北京:清华大学出版社,2007.
- [12] 四衍思维,曹衍龙,赵斯思. PHP 网络编程技术与实例[M]. 北京:人民邮电出版社,2006.
- [13] 三扬科技. 大道 PHP: LAMP+Zend+开源框架整合开发与实战[M]. 北京:电子工业出版社,2009.