基于 Heritrix 的视频垂直搜索引擎①

张 林

(商洛学院 数学与计算机应用学院, 商洛 726000)

摘 要: 针对目前精品课程网站视频信息多、分布散的特点, 通过 lucene 与 heritrix, 设计了专门针对视频信息的垂直搜索引擎, 使用 BKDRHash 算法, 实现了对视频信息的针对性搜索. 搜索时将网页抓取、内容筛选和建立索引的过程结合在一起, 大大减少了系统的开销, 改变了以往传统搜索引擎网页全文收录的做法, 筛选了无用信息, 对页面信息的提取, 以及播放链接的提取过程具有一定的创新性.

关键词: 垂直搜索引擎; 视频; lucene; heritrix

Video Vertical Search Engine Based on Heritrix

ZHANG Lin

(College of Mathematics and Computer Application, Shangluo University, Shangluo 726000, China)

Abstract: As to the vast video information and its loose distribution, a vertical search engine is especially designed for video information through lucene and heritrix. With BKDRHash algorithm, a pointed video research is implemented. Web pages capture, content filtering and index set-up are integrated to greatly decrease the consumption the system, changes the traditional method to capture the whole page by the search engine, filter the useless information. It is creative in picking up page information and picking process of playing linking.

Key words: vertical search engine; video; lucene; Heritrix

从上世纪 80 年代开始, 国家在教育信息化建设方面投入大量资金, 推出了基础教育资源库、高等教育精品课程资源库等许多优秀资源共享系统, 力图通过这些手段来缩小各地区之前的教育资源差距, 平衡教育发展. 有数据显示, 国家投入了大量的资金建立的精品课程网, 却远没有达到期望的利用率, 造成了教育资源的浪费, 如何对这些资源重新开发利用, 则是目前迫切需要解决的问题.

精品课程网上有着丰富的视频资源, 如何才能让学习者快速方便的获得自己所需要的资源则显得十分重要. 本文主要针对视频资源, 设计一个搜索引擎, 通过对已有教学资源网资源的收集和利用, 可提供数量更多、质量更优的符合学习者要求的学习资源.

为了取得尽量丰富的视频资源, 系统将使用网络爬虫抓取从教育网上的精品课程网收集和整理的视频资源. 通过使用合理的分词工具, 使用索引等技术而接受使用者的文本查询, 返回精确匹配的视频结果[1].

由于视频资源在网络上的集中性, 本系统采用垂直搜索引擎技术, 将搜索范围限定在精品课程网站内部.

1 相关知识

1.1 垂直搜索引擎

搜索引擎目前在互联网领域有着举足轻重的作用, 搜索引擎的出现使得人们在海量的数据面前, 可以迅速快捷地发现和定位自己所需的信息. 搜索引擎可以根据一定的策略、运用特定的程序从互联网上搜集信息, 对信息进行组织和处理, 它会对与用户查询条件相匹配的记录在已建立好的数据库中进行检索, 然后返回其结果[2].

① 基金项目:陕西省自然科学基金(2014JM2-6122)

收稿时间:2015-12-25;收到修改稿时间:2016-01-22 [doi:10.15888/j.cnki.csa.005299]

随着信息量的快速增长, 通用搜索引擎的一些缺点开始越来越明显, 例如搜索的精度问题, 在处理垃圾信息方面的问题, 还有搜素精度过低等问题. 2006 年以后, 垂直搜索引擎开始出现, 不同于通用的搜索引擎, 垂直搜索关注于特定的搜索领域和搜索需求, 在其特定的搜索范围内有更强大的功能[3,4].

1.2 开源搜索引擎框架 Lucene[5]

1.2.1 概述

Lucene 是一个开放源代码的全文检索引擎工具包, 它是一个全文检索引擎的架构, 为软件开发人员提供一个简单易用的全文检索工具包. 这些工具包包括: 分析器 analysis, 用来进行文本分析, 例如分词等; 文档对象 document, 代表一个抓取结果; 索引器 index, 提供了建立索引的功能; 查询分析器 queryParser, 用来对用户的检索请求进行分析, 通过调用分析器来实现分析功能; 检索 search, 执行查询索引的功能; 存储 store, 用来执行索引的 IO, 写入或读取索引; 常用工具 util 提供了一般索引等操作常用的功能.

Lucene 是通过索引检索来完成的, 通过索引文件, 来提高用户检索时的相应速度. Lucene 中提高的是搜索内核, 对于任何文档, 只要将其转换成文本格式就可以被索引检索.

1.2.2 Lucene 的结构

Lucene 索引逻辑结构主要由四部分组成:段(segment), 文档(Document), 域(Field)和项(Term).

(1) 段(Segment) 通常在一个索引中, 包含三个段, 段与段之间相互独立, 每个 Segment 都能独立成为一个完整的索引段, 供 Lucene 查询. 为了提高在大数据量时索引建立的效率, Lucene 在建立索引时使用增量索引的方式.

(2) 文档(Document) Lucene 不能直接以物理文件作为数据源进行索引的建立, 要将文件转换成为 Lucene 自定义的文件格式才能完成索引的操作. 转换映射过程中可以有两种映射方式: 一是单文件映射, 即一个物理文件映射成为一个 Document 文档; 第二种是多文件映射, 即可以将多个物理文件的单一字段映射成为一个 Document 文档.

(3) 域(Field) Field 类似于 Java 中的 Map 结构, 由 key 和 Value 两部分组成, 在查询和创建时是一一对应的, 根据 key 值来对 value 字段的内容进行操作. 不同的域需求可能不一样, 比如有的需要被索引, 有些只需要保存不用索引, 有些需要先分词再索引等.

(4) 项(Term) 类似于 Field, 主要应用在信息查询时, 指定查询域的 key, 同时给出查询的内容 value, 构成一个最小的查询单元.

1.2.3 Lucene 的分词工具

Lucene 为了进行建立倒排索引前的准备工作, 要将原始数据进行分割, 就需要使用分词工具, 这些工具在 Analysis 包中. Lucene3.6 的版本提供了 StandardAnalyzer, SimpleAnalyzer 等文本分析工具进行分词工作, 其中, StandardAnalyzer 可以提供对中文的解析, 但是基本是按照单字划分的方式进行. 所以对于中文网页的解析, 需要借助其他专门的中文分词工具. 著名的分词工具有 ICTCLAS, Paoding, MMSEG4J, IKAnalyzer 等. 其中, MMSEG4J 和 IKAnalyzer 对 Lucene 都提供了接口, 可以在 Lucene 项目中方便地使用它们.

IKAnalyze 在 2006 年出现 1.0 版本, 现在已经推出了 3 个大版本. 它是基于 java 语言的轻量级的中文分词工具包, 1.0 版本是以开源项目 Luence 为应用主体的, 结合词典分词和文法分析算法的中文分词组件. IKAnalyzer3.0 则成为面向 Java 的公用分词组件, 独立于 Lucene 项目.

1.3 网络爬虫 Heritrix[6]

1.3.1 概述

Heritrix 是一个可以添加多种可交互组件的爬虫框架. 它是开源的, 采用模板方法构造了一个链式处理模式, 每个环节上的处理器都是可替换的, 从而降低了软件内部耦合度, 大大提高了可扩展性. 定制爬虫程序, 就是通过对基础框架进行扩展, 满足自己的需要.

如图 1 所示, Heritrix 使用了 5 个处理器链, 每个处理器链上可以增加任意多个处理器. 5 个处理器链构成了一个处理器链列表, 开始时从候选链接队列里取出未抓取的链接, 进行抓取, 并分析页面, 提取新链接, 最后将新的链接加入候选链接队列[7,8].

每个处理器链列表的整个流程工作是由一个线程 ToeThread 来执行, Heritrix 是一个多线程的爬虫, 允许开启多个线程执行抓取任务, 任务的开始和结束都是由一个 CrawlController 类来实现的, 它是 Heritrix 的中枢.

1.3.2 抓取任务的配置

Heritrix 实现了一个简易的 Web 服务器, 可以在

浏览器上可视化配置抓取任务. 实际上任务的配置信息最后都保存在一个.XML 文件中.

当以 Web UI 方式使用 Heritrix 时, 点击任务开始(start)按钮, Heritrix 将首先执行 CrawlJobHandler 类的 startCrawler()方法, 其流程如图 2.

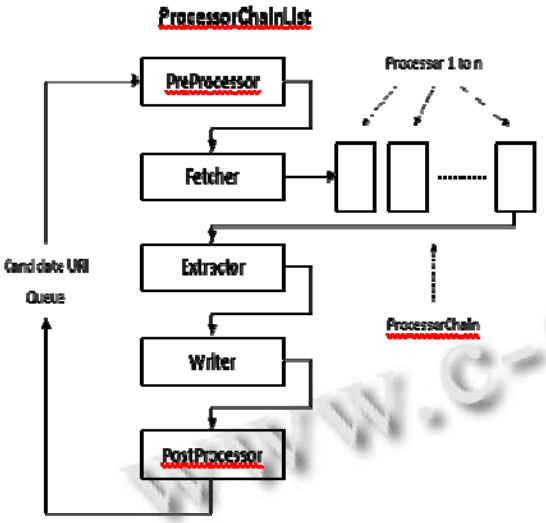

图 1　Heritrix 处理器链结构示意图

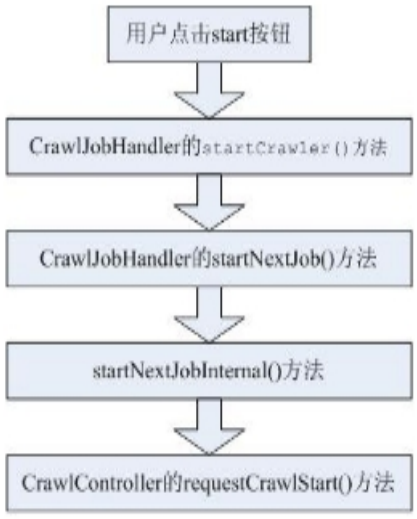

图 2　Heritrix 启动流程图

可以看出, 最后将启动 CrawlController 的 requestCrawlStart()方法. CrawlController 类中也实现了暂停方法 requestCrawlPause(), 恢复方法 requestCrawlResume()和停止方法 requestCrawlStop(), 可以看出所有与抓取控制有关的动作都是由 CrawlController 实现的, 它可以说是 Heritrix 的控制中枢. 因此可以参照 CrawlJobHandler 的初始化过程, 根据实际需要使用 CrawlController 中的控制方法来定制抓取流程, 通过读取编辑好的抓取任务(JOB)的.XML 配置文件, 来进行互联网的抓取.

2　视频搜索引擎的设计

2.1　总体设计

视频搜索引擎主要有两大部分的功能需求: 第一部分是抓取、索引和存储视频资源; 第二部分是对接受用户的查询请求, 返回搜索结果.

在系统的第一部分中, 输入精品课程资源库的视频网页(具体为精品课程网站内视频播放页面), 经过一系列处理(包括分析页面, 剔除噪声信息, 对有效信息切割关键字, 以及建立索引等), 输出建立了索引后的视频资源的链地址(而非视频文件本身)的文档.

在系统的第二部分中, 使用一个在线服务等待系统接收用户在搜索框输入的查询字符串. 服务会调用执行检索程序, 首先对查询字符串进行切分关键字, 然后进入索引库进行匹配, 取出命中的文档, 返回给网站文档中存储的视频链接地址.

通常情况下, 本服务在逻辑上独立运行于 一台服务器上, 其结构如图 3 所示.

图 3　视频搜索引擎简易拓扑图

2.2　内部流程设计

视频搜索引擎的主要作用是对视频资源的抓取、存储、维护以及查询. 其中, 视频的抓取需要使用网络爬虫Heritrix从精品课程网站的网页抓取, 为了使用户以关键字作为匹配文本查询相关视频, 搜索引擎可以在抓取网页的同时, 使用 HTML Parser 对与视频相关的网页信息进行提取作为被匹配文本. 同时, 为了对这些抓取到的视频资源提供高效快速的查询, 必须对这些描述拆分关键字并进行索引. 这里使用 Lucene 的索引框架对抓取到的资源建立索引, 并进行存储. 最后, 视频的查询也是通过 Lucene 的索引搜索程序, 打开索引库, 对索引进行搜索, 由索引搜索结果找到索

引所指向的代表视频的内容(即视频链接地址).

搜索引擎内部处理结构如图 4 所示.

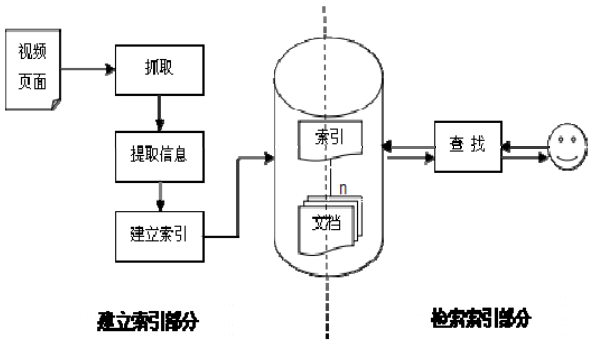

图 4 搜索引擎内部处理结构图

3 建立资源索引的实现

3.1 整体设计

网页抓取、内容筛选和索引建立这三个功能具有不同的时序, 可以将其按以下几种组合方式进行搭配设计:

(1) 第一种方式将三个过程完全独立开, 全部抓取完毕后再对抓取内容进行分析, 全部分析完毕后再调用基于 Lucene 开发的索引建立程序对所有文档进行索引工作. 这样的好处是降低了软件了耦合性, 缺点却也很明显: IO 操作需要消耗大量的资源, 特别是在数据量非常大的情况下, 资源和时间的消耗将会很大.

(2) 第二种方式是将前两个步骤, 即网页抓取和内容筛选同时进行, 即每抓取一个网页就对网页内容进行分析筛选, 仅保存所需的信息, 待所有抓取和分析完成后, 再集中对抓取的信息进行索引的建立. 这种方式大大减小了 IO 开销, 但是仍然存在着一次 IO 读取的过程.

(3) 第三种方式是将后两个步骤合并, 即在爬虫完成所有抓取工作后, 对保存的文档批处理, 将内容筛选与索引建立同时进行. 这种方式与第二种方式相似, 优缺点也相同, 但实际效率不如第二种方式, 因为精品课程网站的网页中存在着不含视频的页面, 对于这些页面, 使用第二种方式时通过判断页面类型, 可以无需保存这些网页, 而使用这种方式将会先将网页保存下来. 当不含视频的网页占整个网站的比重较大时, 这种方式的执行效率远低于第二种方式.

(4) 第四种方式是将三个过程同时进行, 即对一个网页抓取完成后, 立即对于内容进行提取和筛选, 将其加入索引之中. 这种方式较前几种方式开销最小, 由于 Lucene 建立索引的过程是在内存中进行, 除了最后将索引文件写入磁盘需要消耗 IO 资源以外, 其余时候几乎不需要 IO 资源. 而且由于 Lucene 的压缩存储技术, 对磁盘空间的占用也很小.

本文采用第四种方式进行, 在一般情况下能收到最佳的抓取和索引效果. 流程如图 5 所示.

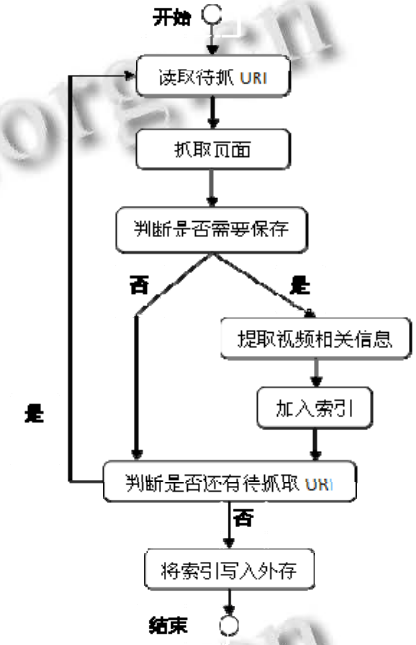

图 5 系统抓取处理流程图

从图中可以看到, 前两个行为“读取待抓取 URI”和“抓取页面”的工作都可由 Heritrix 完成, 因此, 可以改写定制 Heritrix 的抓取流程中某个环节实现剩余的部分. 分析上文中 Heritrix 的处理器链列表可知, 抓取页面的工作是由处理器链 Fecher 实现的, 因此可以定制 Fecher 之后的处理器链来完成上述工作. Fecher 之后有三个处理器链, 分别为: Extractor, Writer 和 PostProcessor, 由于建立索引的工作与 Writer 在语义逻辑上更加接近, 因此定制一个 IndexWriterProcessor 类来完成图中的工作.

3.2 抓取任务中定制 Queue-assignment-policy

Heritirx 的 URI 的存储采用了轻量级数据库 BerkelyDB, 以队列形式存储. URI 分配到这些队列中的方式类似于哈希表, 是以 Key-Value 对存在的. 在默认情况下的 Heritrix 使用 HostnameAssignmentPolicy,

根据域名为 Key 产生 Value 值，这样就会使相同 Hostname 下的 URL 放置在同一个队列中. 对于垂直搜索来说，例如本系统对精品课程网一个系统进行抓取，所有的网址的域名相同，就会造成 URI 分布在一个队列的情况，当一个线程从这个队列中取出一个 URI 进行处理时，其他线程被阻塞，从而造成 Heritrix 的抓取效率低下，会在某个时间段造成系统假死的情况.

为了解决这个问题，可以通过改进 Heritrix 的 queue-assignment-policy，实现一个继承自 Queue AssignmentPolicy 的类，覆盖其的 getClassKey( )值，改变 Key 值的生成方式，这样就使 URL 比较平均地分布到不同的队列中. 本系统采用散列程度高且简单易行的 BK-DRHash 算法生成 Key 值[9].

结果证明，使用 BKDRHash 算法可以使 Heritrix 的抓取效率大大提高.

3.3 页面内容分析、筛选和提取的实现

3.3.1 概述

页面的分析需要完成以下两个工作:

(1) 判断网页是否需要保存:

网络爬虫的对象是网络上的文件，包括 HTML 文件，JS 文件，CSS 文件，以及 FTP 资源和 DNS 信息等等. 对于 HTML 的文件，可以简单地通过判断 URI 的协议名(头部)和后缀名来进行过滤. 但是即使是 HTML 页面，并不是所有的都需要进行记录——由于搜索引擎需要最终保存的信息是与视频资源相关的内容，所以只有包含视频播放的页面，才需要进行后续的处理，包括提取信息和保存等.

(2) 提取网页中需要保存的信息:

视频页面中含有大量的噪声信息，精品课程网的视频播放页面存在着大量与视频内容无关的信息，需要将这部分噪声信息过滤掉，提取建立索引时必须的信息.

由于所有的页面分析和提取工作都会用到 HTML Parser 这个工具包，通过 HTML Parser 提取非噪声区域，舍弃噪声区域，为后续建立索引做好准备. 可以将这部分工作设计成一个类 WriteFilter.

3.3.2 WriteFilter 类的设计

为了使系统具有可扩展性，以便今后对更多的精品课程资源库进行分析抓取，使用工厂模式，让系统在运行的时候动态的使用合适的 Filter 筛选器.

具体实现做法是，将 WriteFilter 定义为抽象类，通过继承它来实现对不同网站的分析筛选. 对于精品课程网，可以设计一个 WriteFilterJPK 类，继承自类 WriteFilter:

public WriteFilterJPK extends WriteFilter {}

经过分析，精品课程网含有视频的网页有两种:一种网址以 resourse.jingpinke.com 开头，一种以 video.jingpinke.com 开头，两种网页的页面布局不一样，因此需要采用不同的处理方式，分别设计两个类 WriteFilterJPKResource 和 WriteFilterJPKVideo，继承类 WriteFilterJPK.

动态绑定 WriteFilter 子类的方法是设计一个 WriteFilterFactory 类，使用一个静态方法通过判断 URI 属于哪个网站，来返回给调用者匹配的 WriteFilter 子类.

设计完成后类的关系如图 6 所示.

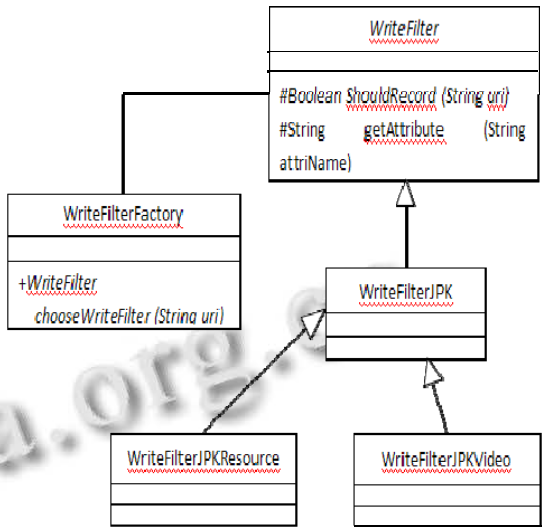

图 6 页面处理类继承关系设计图

3.3.3 页面提取内容与 Lucene 文档字段设计

对页面内容的筛选，最终都将作为 Lucene 文档(document)的内容进行保存. 可以将 Lucene 的段(segment)看做数据库的表(table)，而把文档看做数据库表的一条记录，这样 Lucene 的字段(field)就是数据库表记录的字段. 因此，需要针对所提取的内容设计 Lucene 文档字段. Lucene 的字段类似于哈希表表示形式，一个字段由一个键名和一个键值组成.

经过分析，Lucene 的文档中应当保存一个网页的如下内容:

(1) description: 视频内容的文字描述. 这个将作为首要进行查询匹配的内容, 因此需要对此文本进行分词, 然后执行建立索引的操作;

(2) share_link: 这个是该视频资源的播放地址, 将作为针对查询操作的最终返回结果, 不需要对此项内容进行查询匹配操作, 因此不需要被索引;

(3) category: 这个是该视频资源的分类名称, 如果系统提供给用户按照分类查询的功能, 则需要记录此项并对其进行索引. 如果学科分类的名称不可划分, 可不对其进行分词操作, 也可对较长的交叉学科的名字进行分词, 例如“生物化学”可切分为“生物”和“化学”.

(4) university: 提供视频的高校单位. 此项可提供通过学校搜索视频的功能.

(5) web_url: 视频网页的地址. 此项主要用来与该网页抓取时间 crawl_time 配合, 在下次抓取的时候判断此网页是否已经记录, 以及网页是否产生变化需要重新抓取.

(6) crawl_time: 该网页的抓取时间. 主要用来与该网页的网址 web_url 配合, 在下次抓取的时候判断此网页是否已经记录, 以及网页是否产生变化需要重新抓取.

其中, 第 3 项和第 4 项仅在需要对用户提供此参考信息或者通过此项进行查询时才需要记录, 同样, 如果需要记录其他信息, 或者通过其他信息进行视频的搜索, 则需额外记录其他字段.

在系统中, 可以使用一个 getAttribute 方法, 通过传入的参数来表示想取得的字段名, 在方法内部通过判断字段名来执行不同的操作, 筛选提取所需的字段值并返回.

3.3.4 HTML Parser 构造解析器 parser

HTML Parser 对 HTML 文档有两种访问方式, 分别是 visitor 和 filter. visitor 方式是一种遍历的方式, 而 filter 方式是筛选的方式. 由于对页面的处理都是以提取内容为目的, 所以系统采用 filter 方式进行.

对于一个页面, 由上文可知需要提取多种信息, 为了减少每次提取信息都要对页面进行解析所花费的开销, 可以使用一个 init 方法, 在其中实例化一个 Parser 对象, 将解析后的页面保存在一个 HtmlPage 对象中, 这个对象即以树状结构存储了整个页面的标签节点的内容, 在此后的操作中, 只需使用 extractAllNodesThatMatch 方法取出所需的标签节点. 构造 Parser 对象和 HtmlPage 对象的过程如下:

```
Parser parser = Parser.createParser(content, charset);
    HtmlPage visitor = new HtmlPage(parser);
    parser.visitAllNodesWith(visitor);
```

3.4 索引建立的实现

3.4.1 写索引的创建和关闭

为了提高软件的执行效率, 减少 IO 读写, 可以将索引的建立放在分析页面同时进行. 因此在工程中导入 org.apache.lucene 包, 本项目使用 lucene3.6.0.

Lucene 使用 IndexWriter 类进行索引的写操作, IndexWriter 对象在为一批文档建立索引前需要进行初始化, 最后需要进行关闭操作. 初始化时可以设置一些控制行为, 例如打开磁盘存储地址, 内存缓冲大小分配, 段合并因子, 分词器选择等等, 操作完成后需要对索引进行优化, 对 IndexWriter 对象进行关闭, 将索引写入指定位置.

由于抓取过程是个循环过程, 为了减少系统每次初始化和关闭 IndexWriter 对象所需要的系统开销, 将初始化和关闭工作放在循环体外是个比较好的选择, 即在整个抓取工作之前初始化 IndexWriter 对象, 每次抓取到页面时, 使用 IndexWriter 对象加入索引, 直到全部抓取结束.

IndexWriterProcessor 继承的 Processor 类中有两个未实现功能的方法, 其源代码书写如下:

```
public class Processor {
/**
Classes subclassing this one should override this method to perform
processor specific actions.
*
This method is garanteed to be called after the crawl is set up, but
before any URI-processing has occured.
*/
protected void initialTasks () {
    . by default do nothing
}

/**
Classes subclassing this one should override this
```

```
method to perform
    processor specific actions.
    *
    */
    protected void finalTasks () {
        . by default do nothing
    }
    }
```

根据其中的注释可以知道, 这两个方法分别在所有 URI 处理过程开始之前和抓取结束的时候被调用, 实际上是由 Heritrix 的抓取行为控制中枢 CrawlController 调用. 因此在 IndexWriterProcessor 中重载这两个方法, 分别进行 IndexWriter 的初始化和关闭动作.

初始化时选择 IKAnalyzer 作为分词工具. 为了减少磁盘读写, 将内存缓冲设置稍大. 由于要进行大规模的索引建立, 将合并因子设置较大, 以减少在合并上消耗的资源.

3.4.2 将视频信息加入索引文件

为了获取视频内容描述关键字、播放链接、学科分类、制作单位等信息, Heritrix 在抓取到一个页面后, 会调用 IndexWriterProcessor 的 innerProcess()方法, 通过 WriteFilterJPK 类的 getAttribute 方法获得通过页面解析得到的视频信息, 将其加入 Document 对象中, 最后使用 IndexWriter 执行 addDoc()方法, 将这个包含了资源信息的文档对象加入到索引中去.

4 检索索引的实现

4.1 索引查询对象的初始化

对于 Lucene 格式的索引, 使用 Lucene 的 IndexReader 进行读取, 由类 IndexSearcher 执行查询工作. IndexSearcher 初始化的过程需要传入一个 IndexReader 对象作为参数.

IndexSearcher 的实例化过程如下:

```
        IndexReader ireader = null;
        Directory directory;
        try {
            directory = FSDirectory.open(new File(indexPath));
            ireader = IndexReader.open(directory); . read-only=true
        } catch (IOException e) {
            System.err.println(e);
            e.printStackTrace();
            throw e;
        }
    isearcher = new IndexSearcher(ireader);
```

4.2 执行检索工作

对于用户的查询语句, 首先要使用类 QueryParser 进行解析. QueryParser 类在实例化的时候需要传入一个 Analyzer 类的对象, 用来对进行匹配的文字进行分析, 即分词等工作. 由于 Lucene 项目实现的分析器没有针对中文分词做出优化, 因此, 为了提高查准率, 在初始化 IndexReader 的时候使用 IKAnalyzer 作为分析器.

检索方法如下:

```
        QueryParser parser = new QueryParser(Version.LUCENE_36,
                "content", analyzer);
        Query query = parser.parse(searchString);
        ScoreDoc[] hits = isearcher.search(query, null, 1000).scoreDocs;
        Document[] hitDocs = new Document[hits.length];
        for (int i = 0; i < hits.length; i++) {
          hitDocs[i] = isearcher.doc(hits[i].doc);
        }
```

得到的 Document 对象数组 hitDocs 即代表了之前抓取到的资源的数组, 可以使用 hitDocs[i].getFieldable(fieldName)函数得到相应的字段的值.

5 系统测试

本文采用的实验环境是 Windows 操作系统, 1G 内存, Intel(R) Core(TM)2 CPU 6320 @1.86GHz. 设置线程数为 20, 使原始的 Heritrix 与改进后的 Heritrix 在相同的时间内分别在两台配置相同的计算机上同时运行. 图 7 为原始 Heritrix 与改进后的 Heritrix 在平均抓取速度方面的对比.

从图 7 中可以看出, 在 Heritrix 运行过程中嵌入索引建立操作, 对运行效率影响不大, 能够满足实际需要, 达到了预期要求.

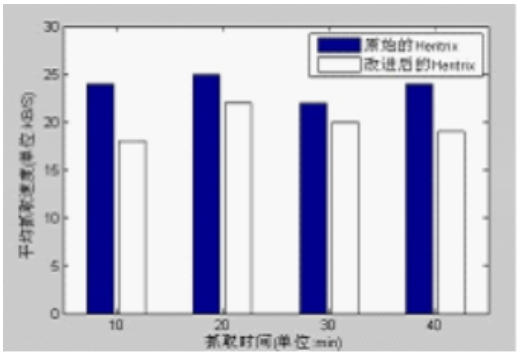

图 7 平均抓取速度对比图

6 总结

本系统为了减少系统开销, 将网页抓取、内容筛选和索引建立三个过程结合在一起, 对 I/O 资源的开销几乎没有. 对整体运行效率没有多大的影响, 能满足实际的需要, 特别是对于视频资源的抓取改变了以往传统搜索引擎网页全文收录的行为, 对页面信息的提取, 以及播放链接的提取过程具有一定的创新性.

本系统也有考虑不周的地方, 在通用性和规模性上面还欠考虑, 这也是下一步我们要重点考虑的方向.

参考文献

1 严莉莉,王倩倩,孟杰.基于聚类的个性化元搜索引擎设计.计算机技术与发展,2007,17(4):186–188.

2 沈贺丹,潘亚楠,邵良彬.关于搜索引擎的研究综述.计算机技术与发展,2006,16(4):147–149.

3 于娟,刘强.主题网络爬虫研究综述.计算机工程与科学,2015,37(2):231–232.

4 白坤,耿国华.基于 Lucene/Heritrix 的垂直搜索引擎的研究与应用.计算机应用与软件,2009,26(1):212–213.

5 Lucene Open Source Material. http://lucene.apache.org/java/doca/index.html.

6 http://crawler.archive.org. [2011-04].

7 王帅,周国民.主题爬虫相关算法综述.计算机与现代化,2013,(4):27–30.

8 白玉韶,梁久祯.基于概率模型的主题爬虫的研究和实现.计算机工程与科学,2013,35(1):160–165.

9 朱敏,罗省贤.基于 Heritrix 的面向特征主题的聚焦爬虫研究.计算机技术与发展,2012,22(2):67.