分类号 学号M20178262

学校代码 10487 密级



**硕士学位论文**

**面向招聘领域的垂直搜索引擎系统的设计与实现**

|  |  |
| --- | --- |
| 学位申请人： | 方聪 |
| 学科专业： | 软件工程 |
| 指导教师： | 卢力副教授 |
| 答辩日期： | 2019.12.18 |

**A Dissertation Submitted in Partial Fulfillment of the Requirements**

**for the Degree of Master of Engineering**

**Design and Implementation of Vertical Search Engine System for Recruitment**

|  |  |
| --- | --- |
| **Candidate :** | **FangCong** |
| **Major :** | **SoftwareEngeering** |
| **Supervisor :** | **Prof. LuLi** |
|  |  |

**Huazhong University of Science & Technology**

**Wuhan 430074, P. R. China**

**December, 2019**

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 2019年12月18日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在年解密后适用本授权书。

本论文属于

不保密□√。

（请在以上方框内打“√”）

学位论文作者签名： 指导教师签名：

日期：年月日 日期：年月日

# 摘要

随着互联网的飞速发展以及大数据的到来，各个领域充斥着大量的信息。如何检索出有用的信息成为了各领域的首要任务。在招聘领域，种类繁多的招聘网站给应聘者提供了大量的招聘信息的同时也给其带来了一些困扰。由于各个招聘网站的数据相互封闭，应聘者想要全面的搜集相关的招聘信息就需要浏览多达数十个招聘网站，这给应聘者带来了极大的不便。因此，结合垂直搜索引擎技术开发出招聘信息的搜索引擎平台成为了该领域的亟需解决的问题。

面向招聘领域的垂直搜索引擎系统是一个由网页爬虫模块、文档解析模块、索引模块和检索模块等主要模块构成的。它是按照软件工程的流程进行开发的。首先是对面向招聘领域的垂直搜索引擎系统进行了详细的需求分析，接着依据需求分析进行系统设计，包括系统的整体架构设计和各个功能模块的设计。系统的后端部分是采用java语言开发，整体上采用了SpringBoot作为开发框架。数据库使用HBase作为分布式非关系型数据库，用来存储网页信息和经过文档解析模块处理后生成的结构化数据，为索引模块的构建提供了数据支持。其中网页爬虫模块是基于一个轻量级的分布式爬虫框架JLiteSpider实现的，JLiteSpider是一个由java语言开发的分布式爬虫框架，可以与面向招聘领域的垂直搜索引擎的其他模块进行无缝整合。检索模块采用的是当前主流的全文检索框架Lucene，它的倒排索引架构在处理海量数据查询的时候表现优良。面向招聘领域的垂直搜索引擎爬取的是全网数十个招聘网站的招聘信息，因此存储的数据量十分庞大。采用Lucene作为检索引擎可以缩短系统的检索响应时间，给用户更加良好的体验。

面向招聘领域的垂直搜索引擎对全网的招聘信息进行了整合，并且提供了检索功能，这对应聘者获取招聘信息带来了极大的便利，让应聘者能够更加专注于专业内容的复习和面试准备。最后对招聘领域的垂直搜索引擎中进行系统测试，并与通用搜索引擎的检索效果进行对比实验，系统的检索结果具有很高的准确率。在对检索结果和原始数据进行对比分析，发现系统的检索结果具有很好的完整性。

**关键词：**垂直搜索引擎 主题网络爬虫 Lucene HBase

# ABSTRACT

With the rapid development of the Internet and the arrival of big data, various fields are full of a lot of information. How to retrieve useful information has become the primary task in various fields. In the field of recruitment, a variety of recruitment websites provide a large number of recruitment information to the candidates, but also bring them some problems. Because the data of various recruitment websites are closed to each other, candidates need to browse dozens of recruitment websites to collect relevant recruitment information comprehensively, which brings great inconvenience to candidates. Therefore, it is an urgent problem to develop a search engine platform for recruitment information with vertical search engine technology.

The vertical search engine system for recruitment is composed of web crawler module, document analysis module, index module and retrieval module. It is developed according to the process of software engineering. First of all, the vertical search engine system in the field of recruitment is analyzed in detail, then the system is designed according to the demand analysis, including the overall architecture design and the design of each functional module. The back-end part of the system is developed in Java language, and spring boot is used as the development framework as a whole. The database uses HBase as a distributed non relational database, which is used to store web page information and structured data generated after processing by the document parsing module, providing data support for the construction of index module. Among them, the web crawler module is based on a lightweight distributed crawler framework, JLiteSpider, which is a distributed crawler framework developed by java language, and can be seamlessly integrated with other modules of vertical search engine in recruitment field. The retrieval module adopts Lucene, the current mainstream full-text retrieval framework. Its inverted index architecture performs well when dealing with massive data queries. The vertical search engine for recruitment field crawls the recruitment information of dozens of recruitment websites, so the amount of data stored is very large. Using Lucene as the search engine can shorten the response time of the system and give users a better experience.

The vertical search engine for the recruitment field integrates the recruitment information of the whole network and provides the retrieval function, which brings great convenience for the candidates to obtain the recruitment information, and enables the candidates to focus more on the review of professional content and interview preparation. Finally, the vertical search engine in the recruitment field is tested, and compared with the general search engine, the retrieval results of the system have a high accuracy. After comparing the retrieval results with the original data, it is found that the retrieval results of the system have good integrity.

**Key words:** vertical search engine , topic web crawler, Lucene, HBase

目录

[摘要 4](#_Toc26977782)

[ABSTRACT 5](#_Toc26977783)

[1绪论 9](#_Toc26977784)

[1.1研究的背景及意义 9](#_Toc26977785)

[1.2国内外研究现状 10](#_Toc26977786)

[1.3主要内容及结构安排 11](#_Toc26977787)

[2关键技术研究 13](#_Toc26977788)

[2.1垂直搜索引擎技术特点 13](#_Toc26977789)

[2.2网络爬虫技术 14](#_Toc26977790)

[2.3中文分词技术 16](#_Toc26977791)

[2.4主题页面抽取技术 18](#_Toc26977792)

[2.5 非关系型数据库 HBase 20](#_Toc26977793)

[2.6本章小结 23](#_Toc26977794)

[3面向招聘领域的垂直搜索引擎系统分析与设计 24](#_Toc26977795)

[3.1面向招聘领域的垂直搜索引擎系统需求分析 24](#_Toc26977796)

[3.2整体设计 25](#_Toc26977797)

[3.3网页爬虫模块的设计 28](#_Toc26977798)

[3.4 索引模块的设计 32](#_Toc26977799)

[3.5 检索模块的设计 35](#_Toc26977800)

[3.6基于LUCENE检索模型的查询优化设计 37](#_Toc26977801)

[3.7分布式数据库HBase架构以及表设计 41](#_Toc26977802)

[3.8本章小结 44](#_Toc26977803)

[4面向招聘领域的垂直搜索引擎系统实现 45](#_Toc26977804)

[4.1HBase集群搭建 45](#_Toc26977805)

[4.2网页爬虫模块的实现 47](#_Toc26977806)

[4.3索引模块的实现 49](#_Toc26977807)

[4.4检索模块的实现 51](#_Toc26977808)

[4.5本章小结 52](#_Toc26977809)

[5面向招聘领域的垂直搜索引擎系统测试 53](#_Toc26977810)

[5.1功能测试 53](#_Toc26977811)

[5.2性能测试 54](#_Toc26977812)

[5.3测试结论 56](#_Toc26977813)

[6总结与展望 57](#_Toc26977814)

[6.1总结 57](#_Toc26977815)

[6.2展望 58](#_Toc26977816)

[参考文献 59](#_Toc26977817)

# 1绪论

## 1.1研究的背景及意义

随着互联网的飞速发展，人们的生活中出现了越来越多的智能化产品，而这些产品产生了不同类型的信息，同时也在不同程度上改变人们的生活方式。根据中国互联网络信息中心最新报告表明[1]：截至2018年12月，我国网民规模达8.29亿，普及率达59.6%，较2017年底提升3.8个百分点，全年新增网民5653万。我国手机网民规模达8.17亿，网民通过手机接入互联网的比例高达98.6%。

面对日益增长的信息，如何能快速的检索到所关注的信息成为了关键。搜索引擎的出现在很大程度上解决了该问题，它为互联网用户提供了一个快速检索信息的平台[2]。搜索引擎旨在利用从互联网上采集到信息资源，进而提供信息查询的服务。目前市场上出现了很多搜索引擎，比如谷歌、百度，涵盖了多样化的信息，并能够根据用户意图检索出相关的信息。目前，搜索引擎成为了人们用于信息检索的最为常用的工具。

当前“互联网+”时代下，人才招聘的渠道越来越广泛。在我国改革开放初期，企业招聘大多采用报纸招聘、人才市场定点招聘和电视广告招聘形式。随着科技的发展，陆续出现了许多大型的招聘平台，企业和人才通过平台实现信息的交互，扩大了企业招聘渠道。近年来，“互联网+”时代的到来，招聘渠道实现了进一步扩展，通过网络能够实现人才与企业之间的面对面交流；通过互联网与手机终端的结合，实现企业信息的广泛发布。

通用搜索引擎可以为互联网用户提供查询服务，例如百度、谷歌、雅虎等[3]，但近几年通用搜索引擎面临着来自领域大数据带来了巨大的挑战[4,5]，因为通用搜索引擎旨在为所有用户提供服务，而面对特定领域的信息检索，可能造成搜索结果不匹配，因此通用搜索引擎不能满足用户对特定领域信息的搜索需求。针对这类问题，各个网站利用自己的信息源来提供信息的检索功能，在某种方面上改善了这类问题。例如在企业招聘领域当中，各大招聘网站都有提供招聘信息的检索功能，但这些招聘网站的信息并不做到数据共享，使得应聘者仍然需要在各大招聘网站反复检索，才能检索到满足自己需求的招聘信息。所以，对于校招以及社招的应聘者来说，一个亟待解决的问题也随之而来，即如何在这海量的招聘信息中快速并准确的定位到自己所需要的招聘信息。

垂直搜索引擎能够针对性的处理以上问题。因为垂直搜索引擎可以针对特定的领域或者该领域中某一模块进行专业和深入的分析、挖掘以及定位，更为精准的提供有一定价值的信息和相关服务，有效地弥补了通用搜索引擎对特定领域中信息检索不准确的缺陷。因而，如何利用垂直搜索引擎的相关技术，为招聘领域构建一套招聘信息的垂直搜索引擎成为本文所要解决的问题。

## 1.2国内外研究现状

在这个大数据的时代，对于垂直搜索引擎等相关技术的研究仍然是社会各界的热点方向，越来越多研究者们结合人工智能的知识，进而对垂直搜索引擎进行改进。大数据的到来同样使得人们对快速准确地检索所需信息的要求越来越高，因此各种搜索引擎不断更新换代。正是由于这两个方面的目标与需求，社会各界人士都参与了垂直搜索引擎的研究中。如 Yue针对数据急速增长的情况[6]，提出了金融领域的垂直搜索引擎的设计和实现，该搜索引擎能很好的满足金融领域的实际需求。Xu等人设计了基于垂直搜索引擎的论文提交推荐系统[7]，通过改进基于网页链接的过滤算法和改进基于内容的过滤算法提高了搜索的准确性。WangWei旨在对海量的网络数据进行提取和分类，建立了有效的垂直搜索引擎[8]。Cheng等人设计和开发了一个基于 Hadoop 的垂直搜索引擎 HVSE[9]，虽然该引擎是基于传统搜索引擎的基本原理，但其对目前面向主题的爬虫算法进行了改进，并结合 MapReduce 编程模型进行数据处理。Konstantin描述了一种自动集成Web 查询接口的方法[10]。该方法允许用户通过统一的 Web 查询接口以透明方式查询特定域中的多个网页数据库。Sejal通过将垂直搜索引擎应用到图像推荐领域[11]，基于余弦相似图像推荐框架以及文本搜索，对原始查询进行扩展，进而返回结果集。Hong通过对专业搜索引擎的研究[12]，并结合改进的网页排名算法，进而设计并实现了一个专业的垂直搜索引擎。Nikulin公开了一种生成搜索引擎结果页面的方法[13]，融合垂直搜索引擎以及历史交互数据，使得搜索结果页面的排序效果更好。English有效的设计了一种旅行预订搜索引擎[14]，该系统根据一个或多个约束构造双重查询，搜索出双重结果集，并且第一个结果集具有很高的购买率。

同样在工业界也相继出现了很多垂直搜索引擎的产品，例如常见的国外垂直搜索引擎有航班搜索引擎 SkyScanner、博客搜索引擎 Ice Rocket、图片搜索引擎TinEye、音乐搜索引擎 MixTurtle、域名搜索引擎 Panabee 等。其中，SkyScanner是面向航班领域的垂直搜索引擎，旨在提供对航班的信息的检索服务[15]，IceRocket 旨在针对 Twitter、博客、Facebook 等站点进行专业化的搜索[16]，MixTurtle 作为国外最好的音乐搜索引擎。同样科研机构也开发了一系列的学术领域的垂直搜索引擎系统，例如，citesee 是计算机科学领域的一个非常著名的检索系统，由NEC 研究院研发而成，其核心是自动索引与分类电子文件[17]。IRISWeb系统是由计算机科学系和北卡罗莱纳大学法学院合作开发，可以检索互联网上使用自然语言全文法律信息，使检索效率大大提高[18]。国家科学数字图书馆是一套集科学、数学、工程和技术的大型在线数字图书检索系统[19]。

国内早在 2005 年也加入了垂直搜索引擎领域的研究中，比较有名的垂直搜索引擎网站有百度学术、亨者搜索、美团、去哪儿、奇虎、搜狗、职友集、咕嘟妈咪等。其中，百度学术是一个专业的面向学术搜索引擎，美团是一个集娱乐、美食于一体的搜索引擎系统，去哪儿是目前较为流行的面向旅游搜索引擎之一，而房天下是一个房产领域的垂直搜索引擎。同样在招聘领域也有相应的垂直搜索引擎产品，如智联招聘、前程无忧、猎聘网、BOOS直聘、拉勾网、58同城等专业招聘搜索引擎。

如今，大量的招聘搜索引擎提供相同或相似的功能，即一个招聘领域搜索引擎根据用户指定的一个地点，行业，职能，薪资等，搜索出相似的招聘信息。类似前程无忧、拉勾网、智联招聘等招聘网站搜索平台，由于单个招聘平台的信息不足以及招聘平台的多样性，从而应聘者需要到各个招聘网站平台上搜索满足自己需求的职位，这将直接导致应聘者时间的浪费，同时也影响了应聘者的体验。所以对于应聘者来说，对现有招聘资源进行数据整合成为了关键。为有效利用已有的招聘信息资源来发挥其价值，能否利用垂直搜索引擎相关技术，如网页爬虫、网页分类、分词算法等，设计一个高覆盖率的招聘信息垂直搜索引擎平台，从而使得招聘垂直搜索引擎更为智能化与人性化，也方便应聘者找到自己满意的工作。本文针对以上搜索引擎和招聘信息资源存在的问题，展开了招聘领域的垂直搜索引擎平台的设计与实现。

## 1.3主要内容及结构安排

论文首先概述了垂直搜索引擎的研究背景和意义、垂直搜索引擎以及招聘领域的特点以及相关技术。其次，设计了网络爬虫模块中的网页抓取策略，利用网页分块技术提取链接所在的内容块，并结合链接及网页与主题相似度来过滤无关链接以及页面，进而实现招聘主题爬虫系统。然后运用分布式系统基础架构Hadoop以及分布式数据库 HBase 建立分布式存储平台，结合HBase分布式数据库的特性，设计表结构，从而实现海量数据存储。最后结合全文检索框架 Lucene以及中文分词器IKAnalyzer建立招聘信息索引库，同时融合同义词以及Rocchio算法对全文检索框架Lucene进行查询优化，并实现了一个招聘领域的垂直搜索引擎。

本文有六个组成部分，各个部分如下：

第1章 绪论，对招聘信息的垂直搜索引擎的背景及意义进行详细的阐述，接着对国内外在垂直搜索引擎领域的研究与工作情况进行阐述与分析，最后对整篇文章的结构与编排进行描述。

第2章 垂直搜索引擎相关技术，首先对搜索引擎的工作原理进行了介绍，同时结合招聘领域的专业知识，分析招聘领域的垂直搜索引擎的特性。接着对网络爬虫技术中的主题爬虫框架与爬虫策略进行介绍。再对目前中文分词的主流算法进行介绍，同时详细阐述了非关系型数据库 HBase 涉及到的相关知识，如基本架构、特性等。

第3章 面向招聘领域的垂直搜索引擎的需求分析与设计，首先对系统的需求进行了阐述，详细地介绍了系统的功能性需求和非功能性需求。接着对系统的整体架构设计和各个模块的详细设计进行了说明，主要是包括网页采集模块的设计、索引构建模块的设计以及检索模块的设计。

第4章 面向招聘领域的垂直搜索引擎的实现，首先介绍了系统的开发环境，主要是系统对软硬件的基本要求。接着对网页采集模块、索引构建模块以及检索模块的具体实现进行了展示和说明。

第5章 面向招聘领域的垂直搜索引擎的测试，首先网络爬虫以及检索模型进行功能测试。然后，对检索模块中的默认查询、扩展查询以及域查询等功能进行测试。最后对系统性能进行压测。

第6章 总结与展望，主要是对全文的研究成果进行一个归纳，以及待进一步研究的内容进行了分析和考察。

# 2关键技术研究

## 2.1垂直搜索引擎技术特点

相比于通用搜索引擎，垂直搜索引擎侧重于某一领域的信息检索，它的数据量以及信息种类多样性都相差甚多，正是由于这个原因，垂直搜索引擎更为侧重的是信息的准确性以及完整性。如果垂直搜索引擎检索出来的大部分信息都是与领域无关的，那么会降低整个垂直搜索引擎的用户体验。根据领域相关性的特性，总结出垂直搜索引擎的特点如下所示。

（1）面向领域的网页信息

对于垂直搜索引擎而言，只需要抓取特定领域的网页数据，所以在网页的抓取策略中，需要融合对链接以及网页的相关性计算，进而过滤无关的网页。抓取的网页与主题相关程度越高，则检索效果也就越高。除了加入链接与网页的主题相关性的判断之外，设置与主题相关的初始地址同样也助于主题爬虫的准确率，而这些初始地址可以通过人工选择来实现。

（2）领域化的中文分词

由于垂直搜索引擎的目的就是对某个领域的信息进行采集、处理、检索等操作，而这些操作都涉及到中文分词[31]，其中，中文分词技术仍然还是垂直搜索引擎技术中的一大难题。如何对专业词汇的切分处理成为中文分词的关键。因此垂直搜索引擎涉及到的中文分词模块需要该领域的专业词汇，并不是通用的词库。所以构建出完善的领域词库有利于改善垂直搜索引擎的中文分词效果。

（3）结构化信息的提取

由于垂直搜索引擎能够提供个性化的查询功能，如域检索、全文检索、域排序等功能，所以垂直搜索引擎需要对各个关键域信息的解析，如招聘领域的“行业”、“职能”、“薪资”、“地点”等信息。

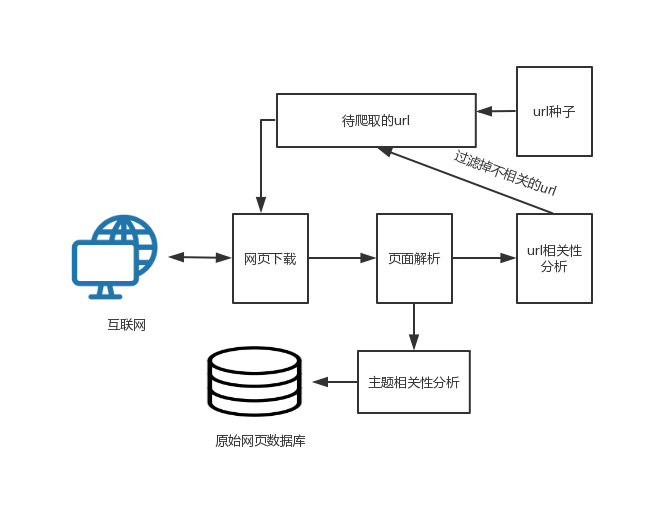
（4）个性化的复合检索以及排序方式

垂直搜索引擎在检索功能上需要提供丰富的个性化功能，同时在排序方面也同样需要更为多样化的方式，不再仅仅依靠文档的评分，可以依靠域属性进行排序。这些个性化功能使垂直搜索引擎更能满足用户的查询需求。

## 2.2网络爬虫技术

2.2.1主题爬虫架构

主题爬虫架构融入了在网页抓取过程中过滤与主题无关网页的判定流程[20]，其他的流程与通用的爬虫框架构大体一致。对于通用的爬虫架构而言，它将存储所有的原始网页信息。主题爬虫中的过滤流程主要分为两部分，其一是通过事先设定好的 URL 访问规则，将符合特定的 URL 访问规则的 URL 存放到待爬取URL 队列中，反之，则直接过滤掉。其二是将原始网页解析出特定区域的页面内容，并结合主题相关性分析计算出该网页与主题的相关度，同时与某一阈值进行比较，确定是否存储该网页。主题爬虫架构如图 2-2 所示。

图2-1主图爬虫架构图

2.2.2网络爬虫策略

垂直搜索引擎的网络爬虫策略旨在抓取具有较高的主题相关度的网页，所以它可以加入不同的限定规则来优化网络爬虫策略[21]。下面介绍常用的策略主要有以下两种策略。

（1）域名预选策略

域名预选策略，是通过限制与主题相关的域名来抓取网页信息，这样的策略可以有效的保证网页与主题的相关度，但是它的缺点就是抓取的网页信息不够全面。由于域名预选策略所抓取的网页有着网页结构单一、明确等特点，所以有利于后续的网页解析。

（2）基于相关度的策略

基于相关度的策略，是在抓取网页的过程中，通过计算链接的相似度进而确定该链接是否存储到待访问的地址队列中，同时也会计算网页的相似度来选择是否保存网页信息。这种策略不仅能加快网络爬虫速度以及提升抓取网页的纯度，而且能在一定程度上降低无关网页的访问量以及存储量[22]。由于加入网页相关度的计算，所以相比于域名预选策略而言，复杂度较高，并且一定时间内抓取的网页数量会减少很多。

2.2.3分布式爬虫框架JLiteSpider

该分布式爬虫框架JLiteSpider是基于java实现的，它是一个强大，但又轻量级的分布式爬虫框架。该爬虫框架分布式的实现方式是基于任务以及消息队列来实现的，各个任务之间通过一个或者多个消息队列来链接，这里消息队列采用Rabbitmq来实现的；任务与消息之间可以是一对一、一对多、多对一以及多对多的关系。消息队列中存储的消息分为四种：URL、页面源码、解析后的结果以及自定义的消息。同样的，任务的工作也分为四部分：下载页面、解析页面、数据持久化以及自定义的操作。 用户只需要在配置文件中，规定好的任务和消息队列之间的关系。接着在代码中，定义好任务的四部分工作机制。即可完成分布式爬虫的任务。该分布式爬虫框架 JLiteSpider 的使用流程如下：

（1）启动Rabbitmq服务。

（2）在配置文件中定义任务以及消息队列的关联

（3）编写相应任务的业务逻辑

（4）最后，启动爬虫服务

## 2.3中文分词技术

2.3.1 中文分词概述

中文分词作为自然语言处理的一个基础模块，分好的词集可以应用到自然语言处理、机器学习等多个领域中。相比于英文以及其他语言的分词难度而言，中文分词显得更为棘手。原因在于英文中每个词之间是通过空格来区别开的，而在中文词语之间是紧密相连的，除了标点符号之外；同时在中文里大多数词语在不同语境下可能表达着不同的意思。正是出于这样的原因，目前大多数应用都是根据不同需求制定不同语法规则以及中文词库等限制来保证满意的分词效果。目前，由于机器学习、深度学习的技术日渐成熟，开始引入了人工智能算法来提高中文分词的准确率。如隐马尔科夫模型、条件随机场、支持向量机、深度学习等算法。不少的研究机构以及高校都在基于机器学习算法来实现中文分词。例如，哈工大的“语言云”，以语音技术平台为基础，提供中文分词、词性标注等接口，斯坦福的基于条件随机场算法的分词器等。下面对中文分词算法的种类进行详细介绍。

2.3.2 中文分词基本算法

现有的中文分词算法都是基于不同分词规则以及语料库。不同的分词算法具有不同的应用场景。基于词典的匹配算法仍然是目前主流的中文分词方法。词典是汉语自动分词的基础，它的结构直接关系到分词的速度和效率[23]。自动分词是中文信息处理系统的基础，根据分词结果进而对汉语文本进行句法分析和语义分析。词典会直接影响切分速度。字典的数据结构主要是通过索引方法，包括索引表、倒排表、哈希表和搜索树[24]。该算法的思想是将提取出的连续字符与词典进行匹配，如果能够匹配到词语，则将其作为一个词语切分出来，而提取连续字符的规则存在多种方式。下面介绍几种基于词典的分词算法[25]。

（1）正向最大匹配算法

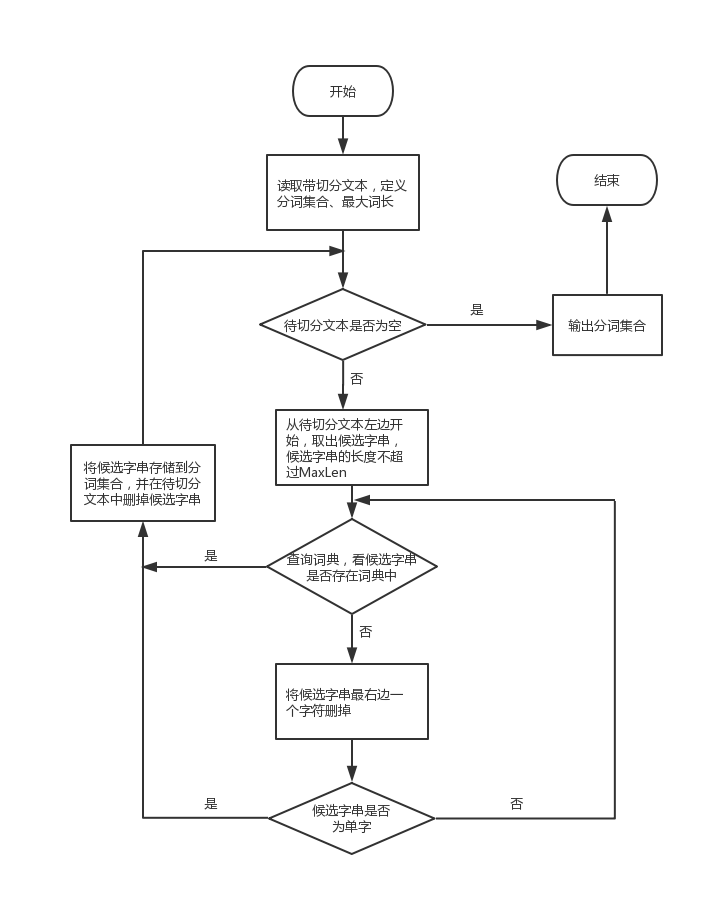
正向最大匹配算法，简称 MM 分词法。其主要思想就是从左到右将待分词文本中的 MaxLen个连续字符与词表匹配，其中 MaxLen 为词典中汉字最多的词条长度，如果待分字符与词表匹配上，则将该连续字符切分出一个词语。如果没有匹配上，则删掉该连续字符的最后一个字符，进而生成一个新的词重新再与词典中的词条进行匹配，依次重复以上操作，直到剩下最后一个字符，将其切分出一个单字。接着剩余的字符按照上述操作依次执行下去，直到整个待分词文本切分完成[26]。正向最大匹配法流程图如图 2-3所示

图2-3最大正向匹配算法

（2）逆向最大匹配算法

逆向最大匹配算法，简称 RMM 分词法。其分词原理是每次从待分词文本的尾部开始选取 MaxLen 长度的连续字符与词典进行匹配，如果匹配上，则将该连续字符切分成词语，如果匹配失败，则去掉该连续字符的首个字符重新与词典匹配，直到待分词文本切分完为止[27]。其思想与正向最大匹配算法正好相反。

（3）最佳匹配分词法

最佳匹配分词法，简称 OM 分词法，就是结合正向最大匹配法和逆向最大匹配法的分词效果进行比较，进而确定分词结果[28]。如果两者分词结果重合时，则该分词结果表示为正确的，如果是两者分词结果不一致时，则根据一些启发式规则确定分词结果，如取分词数量较少的结果作为分词结果等。 根据统计信息可得，逆向匹配的切分精度要好于正向匹配，处理歧义的地方不太多。由于基于词典的中文分词算法具有复杂度较低、实现简单等特性，所以它被广泛应用在实际项目中。当然需要结合领域的语言信息，进而将其精度提高到项目所需的要求。

## 2.4主题页面抽取技术

通过主题爬虫从互联网上采集的网页信息是以 HTML 标签集合为主要标志文件格式，里面包含了大量的格式信息和标记信息，呈现非结构化特性。搜索引擎对文本信息进行分词索引前，必须预先对主题爬虫采集到的网页信息进行结构化处理和主题相关内容的提取，这就是信息抽取技术。

2.4.1DOM 树

HTML 页面是一个节点文档，每个 HTML 标签如<html>、<head>、<title>、<p>、<div>等都是一个节点。DOM 根据这种节点的层次结构将 HTML 页面解析成一种树形结构，这个树形结构就是 DOM 树[29]，每个 HTML 页面都对应着一个DOM树，DOM全称是 Document Object Model，即文档对象模型，它本身和里面的所有东西比如 <div></div> 这些标签都看做一个对象，每个对象都叫做一个节点（Node），DOM 的内部逻辑通过节点的树状层次来表示，这种树状层次结构有五种最重要的节点类型，分别是：文档节点、元素节点、注释节点、属性节点文本节点。其中文档节点是一个 DOM 树的根节点，由此开始并继续向下延伸出枝条，直到处于这棵树最低级别的所有文本节点为止。一个 HTML DOM 树的实例如图 2-4 所示。

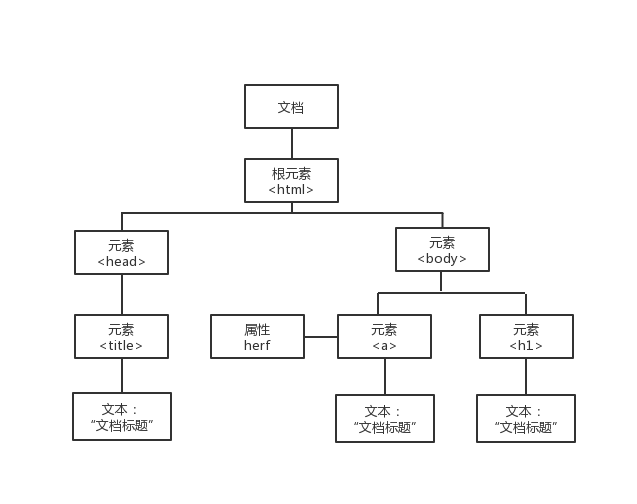


图2-4HTMLDOM树实例图

DOM树的不同节点对HTM页面内容进行标记，定义所有HTML元素的对象和属性，以及访问它们的方法（接口），通过操作 DOM 树节点就可以快速修改或者提取 HTML 页面指定的内容。

2.4.2HTMLParser技术

本系统利用 HTMLparser 对招聘信息网页进行结构化数据的提取、预处理以及分析等操作，HTMLparser 提供了一系列的网页解析方法。HTMLparser是纯 Java编写的开源轻量级网页解析工具库[31]，通过将网页解析成文档树，进而对网页各个节点进行遍历、查找、删除等操作。它可以快速的提取各个标签元素的文本以及属性信息。HTMLparser在解析网页方面有着速度快的特点。HTMLparser主要提供对文档中关键信息的提取以及对提取之后的信息变换等功能。

（1）信息提取主要是对网页中的文本信息、链接以及图片等信息的提取，当然除了信息的提取，HTMLparser还提供辅助功能来补充信息提取功能。例如对图片、声音等资源信息得检查以及分析，还有对网页中的链接进行检查，确定该链接是否为有效的网页链接等。

（2）为了对提取好的信息进行分析、检查等处理，HTMLparser 提供了一系列的信息转换的接口，主要包括对链接的重写、网页内容检验等方法。同时还可以利用正则表达式对内容信息的过滤，例如垃圾信息、广告等噪声信息。

## 2.5 非关系型数据库 HBase

2.5.1 HBase 概述

非关系型数据库HBase，是在分布式文件系统 Hadoop基础上实现的一款非关系型数据库[32]，并且它摒弃了关系型数据库的事务等特性，正是由于这种设计方式，它具有高可靠性、面向列、高性能、可伸缩等特性[33]。在面向处理海量数据业务时，HBase 集群可以通过 MapReduce 来实现并行化，进而提高数据处理速度。不同于传统的关系型数据库的表结构，HBase 数据库中的表结构是由行键、列族、列名以及时间戳构成，而对于每个存储数据由这四种类型的信息组成[34]。同时 HBase 数据库并没有限制存储数据的种类，允许动态存储不同类型的数据。由于 HBase 数据库是分布式的，则它需要 ZooKeeper 来调节各个节点服务器的工作情况。除了数据表结构中的扩展，HBase 数据库集群同时也可以支持节点服务器以及数据存储的横向扩展。

2.5.2 HBase 特性

相比于传统的关系型数据库，在处理海量数据的业务时，HBase 在很多方面都表现的很好，如海量存储、并高发等，而这些优点归功于它的几个特性，如图2-4 所示。

图2-4HBase特性

（1）海量存储

HBase 能够做到对 PB 量级的数据进行存储，而且对于 PB 量级的数据进行查询，能在几十到几百毫秒内返回该数据信息[35]。而拥有这么好的性能归功于HBase 的极易扩展性。仅仅通过添加从节点服务器就可以扩展 HBase 的存储性能，同时也可以提升数据处理性能。

（2）列式存储

HBase 采用的是列族存储，HBase 表中包含多个列族，而列族又包含多个列，每个列存储着详细数据，在创建表的时候就必须将列族确定好。为了与关系型数据库的表结构进行比较，看出 HBase 的表结构的不同之处，下面对一个关系型数据库的表和 HBase 数据库的表进行描述，如表 2-1、2-2 所示。

表2-1关系型数据库表结构

|  |  |  |
| --- | --- | --- |
| PrimaryKey | 列1 | 列2 |
| Row1 | Xxx | Xxx |
| Row2 | Xxx | Xxx |

表2-2非关系型数据库表结

|  |  |  |
| --- | --- | --- |
| RowKey | 列族1 | 列族2 |
| Row1 | 列1、列2、列n | 列1、列2、列n |
| Row2 | 列1、列2、列3 | 列1 |

（3）极易扩展

HBase 的扩展性分为两个方面，一个是基于节点服务器上的扩展，一个是基于存储的扩展。通过横向添加节点服务器进行水平扩展，提升集群的数据处理能力，同时提升集群处理分区的能力。节点服务器的作用就是管理分区、承接业务的访问，这个后面会做详细的介绍。其次通过横向添加数据节点，进而对存储方面进行扩容，提升集群中数据存储能力以及后端存储的读写能力[36]。

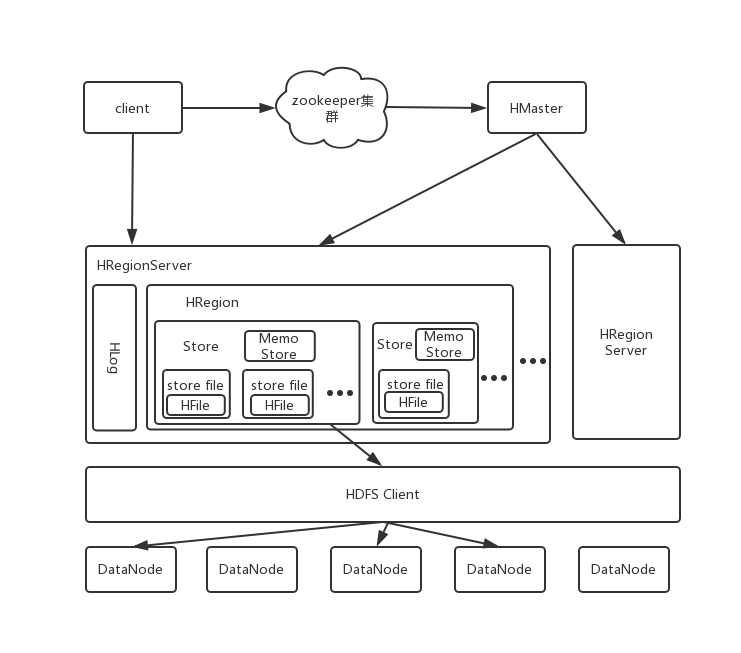
（4）高并发

由于目前大部分使用 HBase 集群架构，都是采用廉价服务器来搭建的，因此单个节点服务器读写的延迟其实并不小，一般在几十到上百毫秒之间。但是在并发非常高的时候，HBase的单个节点服务器上读写延迟却下降的很少。因此HBase集群能维持一个低延迟、高并发的状态。

（5）稀疏

稀疏主要是针对 HBase 列的灵活性，在列族中，你可以指定任意多的列，在列数据为空的情况下，是不会占用存储空间的。

2.5.3 HBase 集群架构

HBase 集群采用主从服务器架构搭建的，并且需要由分布式文件系统 Hadoop集群来支撑，同时该集群还需要 ZooKeeper 来负责各个服务器相互协同工作[37]。总体结构如图 2-5 所示。

2-5 HBase系统架构图

从上面的架构图可以清晰地看出 HBase中的各个组件的相互关系，下面详细介绍各个组件的功能，如下所示。

其中主节点的详细功能如下所示。

1）管理从节点服务器，实现其负载均衡。

2）管理和分配分区，主要是处理在分区合并、分裂以及分区所在的服务器宕机时，完成对分区数据的转移。

3）实现数据库定义语言操作。

4）管理命名空间和表的元数据，实现在数据查询过程中，快速定义到数据所在的从节点服务器以及分区。

5）权限控制。

从节点的详细功能如下所示。

1）对本地分区的读写与管理。

2）读写分布式存储文件并管理表中的数据。

3）接受从客户端提交过来的查询请求，直接通过从节点中读写数据。ZooKeeper集群主要是用作调节各个节点服务器的任务分配、状态信息等。功能如下所示。

1）维护着元数据信息以及协调元数据的更新。

2）实现在主服务器宕机时从服务器代替为主服务器提供服务，进而做到容灾机制。

## 2.6本章小结

本章首先对垂直搜索引擎的基础框架以及技术特点进行了描述，接着对网络爬虫技术、中文分词技术、非关系型数据库 HBase分别进行了详细阐述。其中网络爬虫技术主要详细介绍了主题爬虫架构以及三种网络爬虫策略，并分析两种爬虫策略的优缺点。在中文分词技术中详细介绍了三类分词算法的实现机制。而在非关系型数据库 HBase 中对关系型数据库与非关系型数据库的表结构进行了对比，并描述了两者之间的优缺点。

# 3面向招聘领域的垂直搜索引擎系统分析与设计

本章旨在阐述了面向招聘领域的垂直搜索引擎的需求分析及设计。首先介绍了面向招聘领域垂直搜索引擎的总体需求分析，主要包括功能需求分析和性能需求分析。紧接着对招聘领域的垂直搜索引擎的整体设计进行描述，其中主要包括架构设计、系统的数据流、工作流程以及数据库的设计等；然后详细介绍垂直搜索引擎各个功能模块的层次关系以及详细设计，主要包括网页爬虫模块、中文分词模块、索引模块以及检索模块的设计；最后对搜索引擎的性能设计进行介绍。

## 3.1面向招聘领域的垂直搜索引擎系统需求分析

3.1.1功能需求分析

面向招聘领域的垂直搜索引擎系统为了实现简洁舒适、响应快速的用户体验，对系统的功能需求和性能需求做出了如下说明：

1）前台功能需求

前台的检索功能需要实现对关键字的检索，在检索结果中应该能够查看到标题、内容预览和网站链接，点击网站链接应该能够进入到该网页。每条检索功能应该能够清晰地分隔开，当检索结果较多的时候应该能够对检索结果进行分页。

2）后台功能需求

（1）可以实时而又精确的采集全国每天的互联网上的招聘领域数据信息的分

布式主题爬虫系统。

（2）融合招聘领域的专业词库信息的中文分词器

（3）结合同义词以及Rocchio相关反馈算法对全文检索框架Lucene默认检索的查询优化，同时加入更为丰富的排序方式，使其更好的符合招聘领域的排序需求。

（4）友好而简洁的后台管理界面。

对比国内已有的相关招聘领域的垂直搜索引擎的研究，本文招聘领域的垂直搜索引擎主要有以下优势：

（1）数据完整性及可靠性。本文在对国内已有的招聘领域的垂直搜索引擎的文献进行了学习以及研究发现：需要招聘数据具有完整性，需要搜集好国内所有的招聘网站平台，对各网站平台进行分布式网页爬虫，从而达到数据的完整性，同时需要过滤其他与主题无关的网页，如楼盘页面、房产资讯页面等，结合网页相关性分析实现一套分布式主题爬虫，这样在保证数据完整性的同时保证了数据的可靠性。

（2）丰富而又准确的招聘信息的检索功能。针对于国内的招聘领域的垂直搜索引擎的文献中，本文在全文检索框架 Lucene 的检索模型的基础上结合了Rocchio 算法来实现优化查询功能，同时也加入了自定义字段排序，从而丰富了排序功能，达到满足购房者的排序需求。

3.1.2性能需求分析

（1）对用户搜索操作进行快速响应。用户提交检索后得到结果返回的等待时间，系统响应时间越短，对用户来说体验性更好，其不仅取决于搜索引擎的性能同时与网络带宽也有关系，一般响应时间要低于1s。

（2）支持高并发访问，当存在大量用户并发操作时，搜索响应不能延迟。

（3）搜索结果需具有较高的召回率与查准率［47］。召回率是指搜索引擎所返回的检索结果占数据库中全部相关结果的比率。查准率是指符合用户需求的结果占所有查询结果的比例，取值范围在０-１之间。理想情况是两者越接近１越好，但是实际情况是两者成反比。

（4）系统可靠性。应该采用成熟、稳定、通用的技术，保证系统的可靠性，数据具备自备份能力，能够保证系统长期稳定运行，有较强的容错和系统恢复能力，并为原系统提供调用接口。

（5）具有系统运行日志、数据更新日志等日志。

## 3.2整体设计

根据2.1节中提出的总体需求，本节对招聘领域的垂直搜索引擎系统进行了整体设计，主要包括架构设计、数据流设计、工作流程设计和HBase数据库设计。

3.2.1系统架构设计

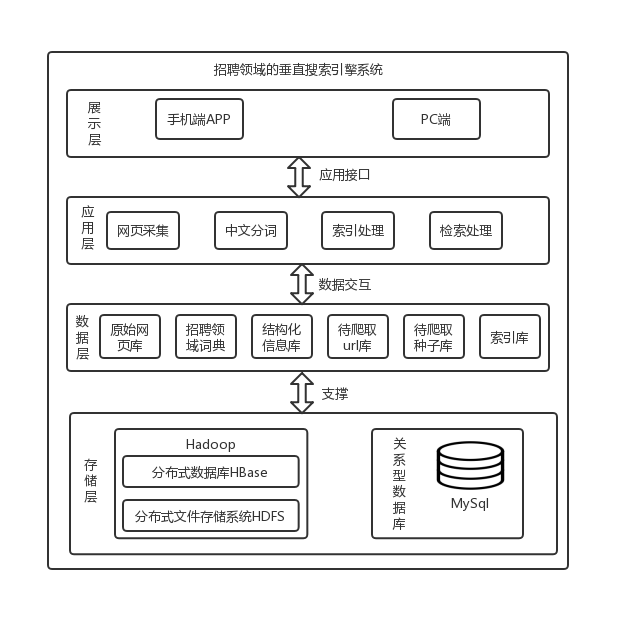
招聘领域的垂直搜索引擎整体架构如图3-1所示。

图3-1系统整体架构图

招聘领域的垂直搜索引擎由展示层、应用层、数据层以及存储层组成。展示层即本系统提供给用户的查询界面。应用层则是垂直搜索引擎中最重要的部分，它负责所有数据的处理以及业务逻辑操作，它包含很多核心模块，如网页采集及信息抽取、中文分词、索引和检索等。而数据层则是招聘领域的垂直搜索引擎的数据中心，它负责系统中数据的读写操作，它直接影响着本系统的数据读写性能。存储层主要负责数据的存储方式，并提供数据存储的服务。

下面主要对应用层模块进行阐述。

（1）网页采集模块

本文采用域名预选策略并结合链接与主题相似度以及网页与主题相似度的判定，可以很好的保证主题相关性。同时，为了能够不断的丰富可采集的资源，加入了对各大招聘网站平台的相关页面进行更新检测，如果检测到存在更新的招聘信息，则会通知爬虫服务进行抓取，同时，原本的爬虫服务会每隔一段时间循环抓取，从而实现对原网页库的更新，这样就实现了数据的全面性。 该设计方案可以保证数据的准确性和完整性，又因为主题判定时是基于文档结构树以及向量空间模型来实现的，复杂度不高，保证了爬虫的高效性。

（2）中文分词模块

中文分词模块主要是对网页采集模块提取的结构化招聘信息数据进行分词操作，本文采用目前比较流行的中文分词器IKAnalyzer来实现的，其中该分词器是基于正向迭代最细度切分算法实现的，并结合招聘领域专业词汇，使其能够有效的对招聘领域的文本进行分词。最后将其与全文检索框架 Lucene 进行整合应用到该系统中。

（3）索引模块

索引模块是基于全文检索框架Lucene来实现的。定时读取HBase数据库中招聘信息数据，为每个字段设置不同索引、分词等标志并存储到不同域中，追加更新到索引库中，以供后续检索模块的查询操作。

（4）检索模块

检索模块同样也是基于全文检索框架Lucene来实现的。利用全文检索框架Lucene 的特定项查询、布尔逻辑表达、模糊匹配、高亮查询等功能，实现了招聘信息特定字段查询、招聘信息全文信息检索等查询方式，同时整合全文检索框架Lucene中默认检索模型与 Rocchio算法来优化查询功能，使其检索结果更符合用户的需求，而且本系统还加入了特定字段的排序功能，从而延伸出更为个性化的排序功能，使应聘者更为方便地找到满意的招聘信息。

3.2.2系统数据流设计

本文招聘领域的垂直搜索引擎主要功能部分数据流图如图3-2所示。

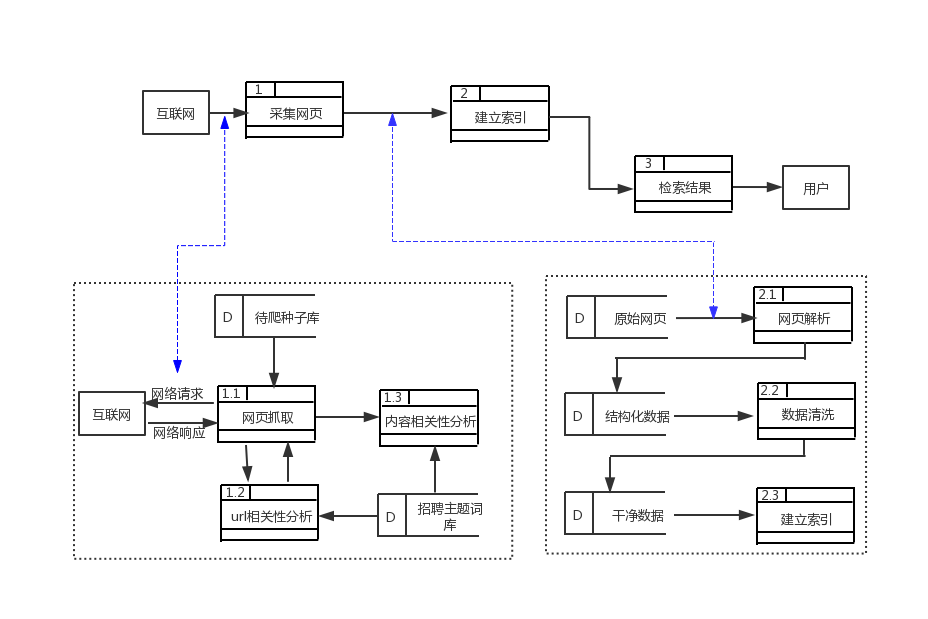


图3-2系统数据流图

如图 3-3 所示，数据流主要分为网页采集和信息检索两大部分，首先网页数据通过网页爬虫服务抓取下来，经过一系列的处理以及分析，将与主题相关的网页存储到原始网页数据库中，其中，中间处理操作包括 URL 相关性分析、页面内容相关性分析等，然后通过网页解析器，将非结构化数据解析成结构化数据，之后进行数据清洗操作，将重复信息以及异常数据剔除掉，得到清洗后的结构化数据，再对其建立索引库，最后实现好的检索器将根据用户输入的查询关键字，查询出所需的结果集返回给用户，这样就完成了数据在整个系统的处理以及变换流程。

## 3.3网页爬虫模块的设计

招聘领域的垂直搜索引擎的网页爬虫模型是基于一个轻量级的分布式爬虫框架 JLiteSpider 以及HBase数据库集群来实现的，轻量级的分布式爬虫框架JLiteSpider 主要负责业务逻辑层的实现，而HBase数据库集群负责数据库的存储工作。

3.3.1爬虫服务架构设计

本文针对网站的招聘信息发布情况、各个招聘网站的网页布局、更新情况以及详情页面的结构等因素，同时结合对国内网站的招聘信息真实性的调研，选取出符合招聘信息的抓取的网站，如智联招聘、前程无忧、拉勾网、BOOS直聘等网站。本文通过对这些网站中的招聘数据信息进行抓取，从而构建招聘网页数据源。由于所抓取的网站数量较多，本文对不同的网站构建相对应的分布式网页爬虫服务引擎，这样可以更好的抓取到准确以及全面的招聘数据信息。每个分布式网页爬虫服务引擎负责该网站的招聘信息抓取工作，接下来对“前程无忧”网站的分布式爬虫服务引擎架构图进行详细介绍。具体架构图如图3-4所示。

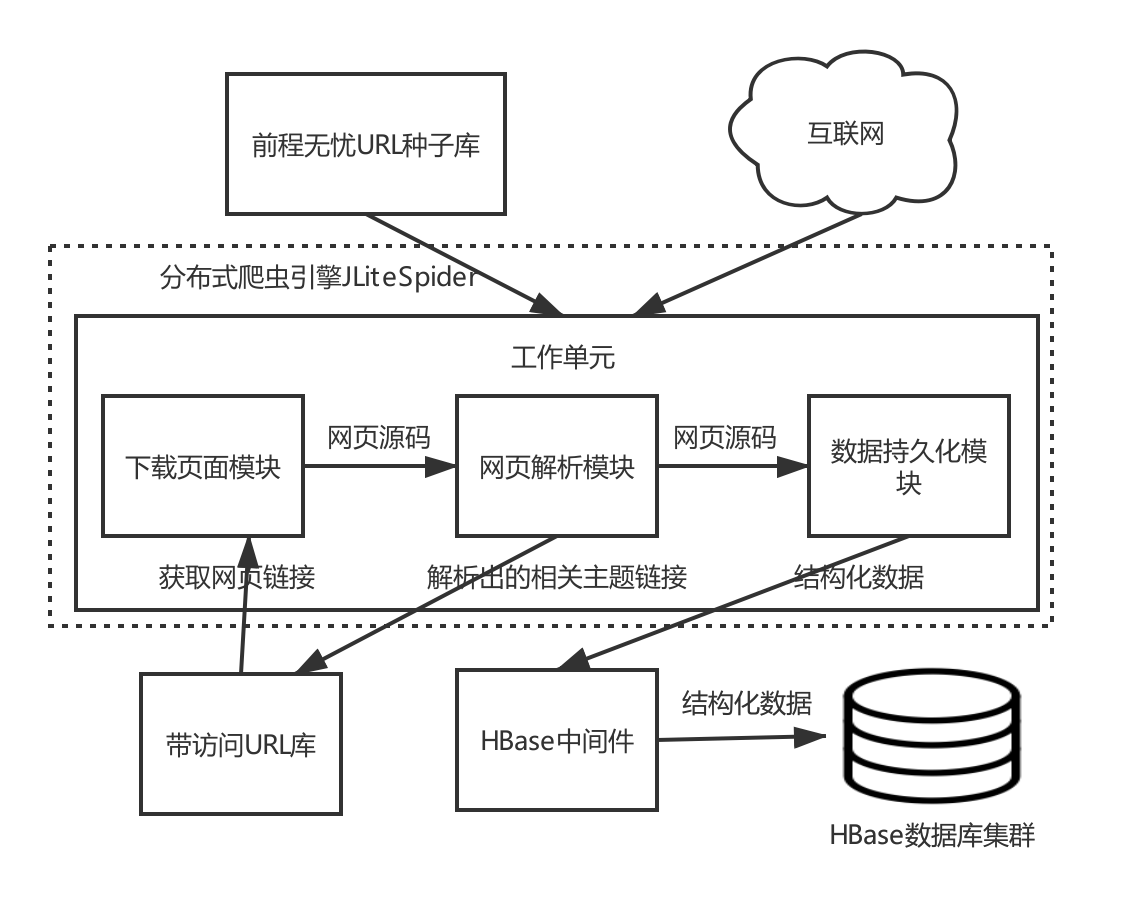


图3-4分布式爬虫服务架构

如图 3-4 所示，分布式爬虫引擎 JLiteSpider 从前程无忧URL种子库中，获取到初始 URL 链接，进而开始网页爬虫服务，分布式爬虫引擎 JLiteSpider 包含多个工作单元协同工作，每个工作单元中包含下载页面模块、解析网页模块以及数据持久化模型等。下载页面模块从待访问 URL 库中获取与主题相关网页链接，通过 Http 请求将网页源码下载下来，利用 Rabbitmq 消息队列引擎将网页源码传递给解析网页模块，解析网页模块分析出与主题相关的网页链接存放到待访问URL库，同时也会将源码传递给数据持久化模块，进而数据持久化模块通过调用HBase数据层中间件，将网页数据信息存储到 HBase 数据库集群中，进而完成对“前程无忧”网站中招聘信息的分布式爬虫服务。

3.3.2文档解析与处理模块

网页文档是由html标签组织成的非结构化的文本信息，对于垂直搜索引擎而言，查询的结果是一个结构化数据信息，所以如何将招聘信息中的关键属性信息提取出来成为了本节所要解决的问题。

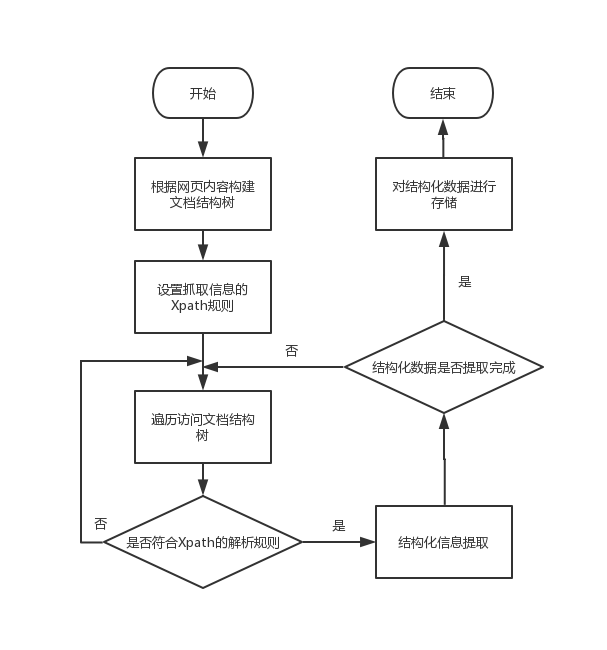
网页文档解析是从网页下载器中获得网页源代码，然后结合 HTMLparser 组件对网页文档构建文档结构树，为要抓取的招聘信息的属性信息设置不同的Xpath解析规则，然后结合文档结构树解析出招聘信息的各个属性信息，即完成了信息提取过程。HTMLparser组件对信息提取有着很高的效率，并且它的容错率也很高，同时也是目前最好的解析网页的工具库。本文采用HTMLparser进行网页解析与信息抽取，流程图如图 3-5所示。

网页解析与信息抽取过程：

（1）对整个网页源码进行文档结构树构建，为不同属性定制不同的 Xpath解析规则，根据此规则可以提取出包含标题、薪水、职位、工作地点、企业性质等相关属性信息。

（2）遍历文档结构树中的节点对象，提取符合各个 Xpath 解析规则的结构化数据。

（3）对解析好的结构化数据存储到 HBase 数据库中。

图3-5结构化数据提取流程图

采用HTMLparser对抓下来的HTM网页文档进行网页解析与信息抽取后，将结构化的数据保存到HBase数据库集群中。各个属性的解析 Xpath 的格式如表3-1所示，这里并没有将所有列名的数据表示出来，仅仅列出了“标题”、“薪水”、“职位”、“公司名称、工作地点”等列名数据。

表3-1属性解析Xpath格式表

|  |  |
| --- | --- |
| 属性 | Xpath值 |
| 标题 | //div[@id=lpn name]/div[1]/text() |
| 薪水 | //div[@class=tsl\_item\_top]/div[1]/i/text() |
| 职位 | //div[@class=tab\_cont\_left]/div[@class=tr\_line  clearfix]/text() |
| 公司名称 | //div[@class=tab\_cont\_left]/span[@class=rcont]/text() |
| 工作地点 | //div[@class=item clearfix][4]/span[@class=rcont]/text() |

## 3.4 索引模块的设计

在搜索引擎中，索引设计的好坏会直接影响搜索结果的准确性，如果对那些无关的数据进行索引，会导致用户搜索不到相关的数据信息，同时还会导致索引文件膨胀等问题，同样分词的好坏也会影响搜索引擎的检索效果，索引模块的设计包括对数据预处理、中文分词器、索引构建等。

3.4.1 数据预处理

数据预处理主要是对结构化的招聘信息数据去重等操作，因为同一个招聘信息有可能在不同的网站上发布过，这样导致招聘信息存在重复问题。本文通过短文本相似度来计算招聘信息的相似度，进而实现招聘信息的去重功能。由于招聘信息的数据量很大，原始数据库中存放在上千万条数据，如果全表扫描数据库的招聘信息数据，然后依次进行相似度计算，这样的设计复杂度会很高，同时结合根据网站信息的招聘信息发布规律，本文设定一个月的招聘数据来进行相似度计算，从而降低相似度的复杂度，同时也能够降低招聘信息的重复率。 短文本相似度算法有很多种，如欧式距离、余弦距离、编辑距离、Jaccard 相似度、Shingling 相似度等，本文采用结合 Shingling 与 Jaccard 相似度来实现短文本相似度的计算。

（1）Jaccard 相似度

Jaccard 相似度主要是用在比较两个集合的相似度，将两个集合的交集与并集的比例作为两个集合的相似度。

（2）Shingling算法

Shingling算法可以整合Jaccard相似度算法来实现文档之间的相似度计算，通过将文档转换为连续k字符的集合，然后结合Jaccard相似度来求出文档之间的相似度。本文利用对文档中每2个连续的字符进行散列，得到一组散列值组，用该散列值集合表示文档更为有效。

（3）编辑距离算法

该算法是通过统计两个字符串之间转换成对方的步骤，将转换的次数作为两个字符串之间的相似度，其中转换的操作包含插入、删除两个操作。

3.4.2 Word中文分词器

中文分词算法在第2章已经介绍了，目前国内有很多的中文分词工具都是基于中文分词算法来实现的，word分词是一个Java实现的分布式的中文分词组件，提供了多种基于词典的分词算法，并利用ngram模型来消除歧义。能准确识别英文、数字，以及日期、时间等数量词，能识别人名、地名、组织机构名等未登录词。能通过自定义配置文件来改变组件行为，能自定义用户词库、自动检测词库变化、支持大规模分布式环境，能灵活指定多种分词算法，提供了10种分词算法，还提供了10种文本相似度算法，同时还无缝和Lucene、Solr、ElasticSearch、Luke集成。本文整合word工具类到搜索引擎系统的流程如图3-6所示。

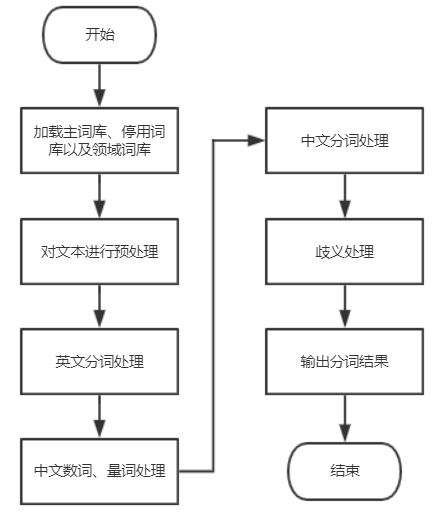


图3-6Word分词流程图

本系统根据爬虫服务搜集招聘领域专业词库数据，构建招聘领域词库用作扩展词库，同时结合基础词典来实现中文分词模块。

3.4.3 索引构建

本文对招聘领域结构化数据进行索引构建，整体流程如下所示，其中，索引构建流程如图3-7所示。

（1）配置招聘领域的词典并创建中文分词器word对象。

（2）设置索引创建目录与分词器并创建写索引对象 IndexWriter。

（3）创建HBase数据库读取对象，并读取整点时间段的招聘数据信息。

（4）对整点时间段的招聘数据信息去重，采用 3.4.1 所提到的短文本相似度计算，将重复的招聘信息数据删除。

（5）对检索出来的招聘数据进行添加到索引库中，如果没有添加完成，则依次继续完成每条招聘数据的索引添加操作。如图3-7所示。

（6）如果整点时间段的招聘数据完成添加，则IndexWriter进行提交。

（7）关闭所有连接对象。

在索引构建过程中，还有对索引的更新操作，本系统通过设置定时任务，判定HBase 数据库中是否存在新的招聘信息数据，存在则会进行索引的追加操作，操作流程与该流程一致。

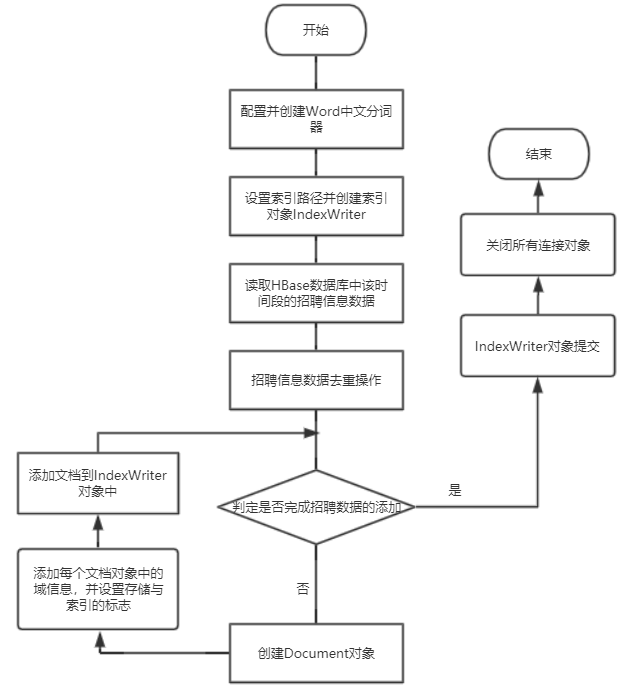


图3-7索引构建流程图

## 3.5 检索模块的设计

3.5.1 检索过程

信息检索的流程是通过给定一个用户输入的查询词，对查询词进行中文分词以及查询扩展，接着对索引库中词项进行检索，得到所有具有评分的文档集合并结合Top-k排序算法将前k个评分最大的结果集返回给用户，而其中具体评分方式由排序方式来决定，例如相关性排序、特定域排序等。本文设计的招聘领域垂直搜索引擎对检索模块中两个检索方式都进行了实现，即默认查询与扩展查询，默认查询采用的是通用全文检索框架 Lucene 的检索方式，而扩展查询采用的是第2章提出的融合同义词和 Rocchio 的相关反馈算法来优化查询。检索模块的流程如图3-8所示。

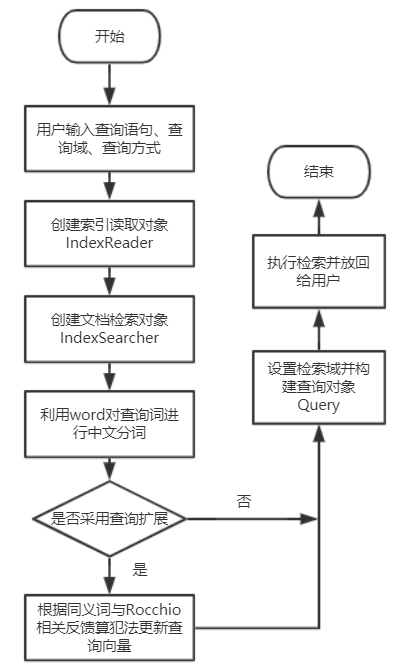


图3-8检索模块流程图

具体检索过程如下所示。

（1）用户输入查询词、查询方式、查询域。

（2）指定索引目录并打开索引读取对象 IndexReader。

（3）创建文档检索实例 IndexSearcher 并设置相似度计算类型。

（4）利用word分词器对查询词进行分词操作。

（5）判断是否对查询向量进行优化，如果查询方式为“扩展查询”，则根

据同义词优化与 Rocchio 相关反馈算法，优化查询向量。

（6）通过设置域信息以查询向量构建查询对象 Query。

（7）根据查询对象 Query 进行检索，将检索出的文档集合按照指定排序算

法以及 k 的值，将 k个评分的查询结果集返回浏览器。

## 3.6基于LUCENE检索模型的查询优化设计

查询优化一直是信息检索领域的一个重要话题[38]。特别对于专业领域的垂直搜索引擎而言，更好的表示出用户所表达的意图是搜索出最理想的结果的关键。当下的搜索引擎都需要面对用户的错误输入以及不完整的输入等问题，也能给出尽量相关的查询结果，这些都需要用到查询扩展[39]。本节主要介绍在基于 Lucene全文检索引擎下通过融合同义词优化以及相关反馈的策略，从而达到优化查询结果以及排序功能。

3.6.1 Lucene检索模型及评分算法

1）布尔模型算法

布尔检索模型是最为常用的检索方式[41]，在关系型数据库中采用的就是类似布尔模型的检索方式。在该模型中只存在两个状态，即匹配与不匹配，完全匹配的结果才会被检索出来。在该模型中包含 AND、OR 以及 NOT 三种关系符号，这些关系符号用于组合各个单个查询项之间的逻辑关系，使得整个待查询式TRUE 的结果才算匹配成功，否则表示匹配失败[42]。这种检索模型非常直观，类似查询匹配，将包含所有查询词中的关键词的文档作为搜索结果。这种检索模型会存在搜索的文档数过少，而对于查询表达式的编写也较为复杂，而且搜索结果并没有排序功能。

2）向量空间检索模型

在向量空间模型中，利用向量来表示一个文本，并结合向量的数学运算操作来实现文本的运算，而文本挖掘、信息检索以及自然语言处理等领域都涉及到文本的运算。全文检索框架 Lucene 默认检索方式采用基于布尔模型与向量空间模型来实现的，基本思路是利用空间向量来表示文档以及查询词，并结合向量距离公式来计算文档与查询词的相似度，进而确定文档的排序。

（1）文挡表示

文档通过分词可以得到特征词集合，并为每个特征词赋不同权重，进而得到一个特征词向量，将其作为文本的表示。假设文档 d 包含一组特征词(t1,t2,…,tn)，而每个特征词ti的权重为wi，那么该文档 d 可以表示成：d=(w1,w2,…,wi,…,wn)。

（2）特征权重计算

在全文检索框架 Lucene 中，特征词的权重采用 TF-IDF 公式来计算的，其中，TF-IDF 公式其实是词频 TF 与逆文档频率 IDF 的乘积而得到的[43]。TF称为词项频率，指的是词项在文档中出现的次数。DF 称为文档集频率，指的是包含该词的文档数量。IDF 称为逆文档频率，通常IDF= log N/DF，其中，N 表示总文档的个数，该值是为了降低每篇文档都出现的词项的权重信息。TF-IDF 计算公式如公式3-1 所示。

(3-1)

如公式3-1 所示，表示特征词t在文档d的权重；t表示词项频率，是特征词 t 在文档 d 出现的次数；表示文档集频率，是包含特征词t的文档的数量；N 为文档的总个数。

（3）文档相似度计算

根据上述的文本表示以及权重计算方式来得到查询词 q 以及文档 d 的特征向量，接着利用向量空间的余弦夹角来表示两个特征向量的相似度。可见，余弦夹角越大，那么两个文本相似度越低。文本的相似度计算方法如公式3-2 所示。

(3-2)

例如现在查询词为“北京AND java”，文档与词项对应关系如表 4-1所示，现在分别计算查询语句 q 与各个文档 di之间的相似度。

|  |  |
| --- | --- |
| 文档ID | （词：TF） |
| 查询语句q | （北京：1）（java：1） |
| 文档d1 | （北京：1），（Java：1），（月薪1.5万：1） |
| 文档d2 | （北京：1），（pyhon：1），（月薪2万：1） |
| 文档d3 | （武汉：1），（c++：1） |

查询语句 q、文档 d1、文档 d2、文档 d3分别用向量表示为：q=(1,0,1,0,0,0,0,0)，d1=(1,0,1,0,1,0,0,0)，d2=(1,0,0,1,0,0,1,1)，d3=(0,1,0,0,0,1,0,0)。

根据公式 4-2，查询语句 q 与文档 d1的相似度：cos (q,d1)=2/√2\*√3≈0.8164；查询语句 q 与文档 d2的相似度：cos (q,d2) =1/√2≈0.7071；查询语句 q 与文档 d3的相似度：cos (q,d3)=0。因此，查询语句 q 与文档 d1、d2、d3的相似度从大到小依次为 d1、d2、d3。

3）Rocchio算法模型

相关反馈算法在不同的检索模型中已经被证明是有效的[44]。在向量空间模型中，通常采用Rocchio相关反馈算法，该算法旨在通过构建新的查询向量，而该查询向量可以做到最大限度地提高相关文档的排名，同时也降低不相关文档的排名[45]。该算法的前提是对于原始查询文档集中相关文档集合与不相关文档集是确定的，然后通过结合相关文档集与不相关文件集两类的特征向量对原始查询进行修改以获取最优查询向量，进而达到优化查询结果集。现在假设需要得到一个最优查询向量 q，该查询向量与相关文档之间的相似度最大且同时又和不相关文档之间的相似度最小。若 Dr表示相关文档集，Dnr表示不相关文档集，那么希望找到的最优的查询 q 如公式3-3所示。

(3-3)

其中，sim 函数用于表示查询向量与文档向量的相似性。这里采用余弦相似度作为相似度计算公式，则对最优查询向量进行变换，得到新的公式如公式3-4所示

(3-4)

从公式3-4 所示，最优的查询向量等于相关文档的质心向量与不相关文的质心向量的之差。由于查询文档集中相关文档以及不相关文档事先都是未知的，所以设定排名前 t个数据作为相关数据，而剩余的数据作为不相关数据。假设能够找到一个距离相关文档数据最近同时又要保证距离不相关文档数据的最远的数据点，接着将该点作为查询向量，那么在检索并排序的结果集中，相关文档的排名将会得到提升，而对于不相关文档的排名将会下降。根据该思想进而对公式4-7 进行改造，得到近似最优查询向量 q 公式，如公式3-5所示。

(3-5)

其中，qinit表示原始查询向量，Dr是设定原始查询结果集中排名靠前的文档集，即相关文档集，Dnr为不相关文档集合，即查询结果集去掉 Dr的集合。α、β及γ表示各项的权重。这些权重用于控制优化查询向量与原始查询向量之间的权重关系[46]。目前这三个参数的最优设置分别是α=1、β=0.75、γ=0.15。

3.6.2融合同义词与Rocchio算法的查询优化

本节针对上述的问题，在 Lucene 的默认检索模型基础上融入了同义词优化以及Rocchio反馈算法，进而优化基于 Lucene 的默认检索模型的查询结果。

1. 流程描述

融合同义词优化和Rocchio算法的查询优化包括扩展查询词的同义词以及结合Rocchio相关反馈算法两个步骤，具体算法流程如图3-1 所示。

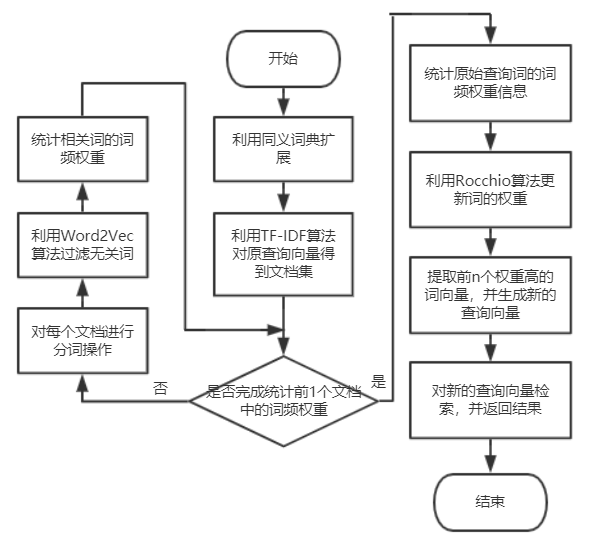


图3-1检索算法流程图

如图3-1 所示，融合同义词优化和Rocchio算法的 Lucene 检索模型主要包含三个步骤，这三个步骤分别是：第一步将用户提交的查询词通过分词器进行分词操作，得到查询词集合，然后将其与同义词典进行扫描匹配，其中，同义词典主要是招聘领域中的各个属性的同义词集合，如“JavaScript”属性存在着“js”、“jquery”等同义词，将与查询词相关的同义词筛选出来，组成同义词集合；第二步利用Lucene的默认排序算法对同义词集合进行搜索，得到搜索结果集，根据事先设定好的前t个搜索结果为相关文档集合，剩下的全为不相关集合，然后对前t个相关文件集进行分词操作，得到词集合，并利用 Word2Vec 算法计算每个词与查询词的相似度，然后将与查询词相关的词合并到原始查询词中，并统计词的词频权重。利用公式 4-5 更新每个词的权重。其中，α=1、β=0.75、γ=0；第三步提取前 n 个权重高的词向量生成新的查询向量，对新的查询向量进行搜索，并利用 Top-k 算法将前 k 个搜索结果集返回给用户，完成整个搜索过程。

## 3.7分布式数据库HBase架构以及表设计

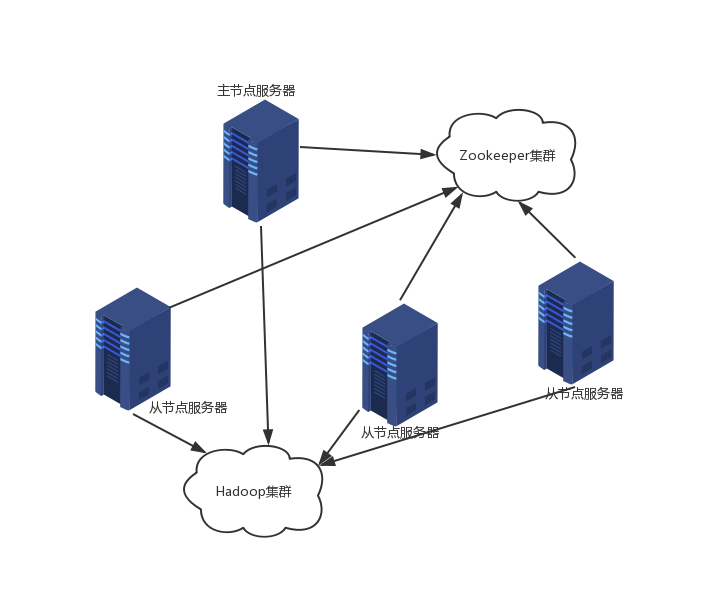
本文招聘领域的垂直搜索引擎采用1台主服务器与3台从服务器的设计方案来搭建分布式数据库 HBase集群，而该集群又是基于Hadoop集群之上的，集群架构如图3-9所示。

图3-9HBase集群架构图

如图 3-9 所示，分布式数据库HBase集群建立在Hadoop集群之上，通过ZooKeeper 集群服务来协调 1 台主服务器以及 3 台从服务器的之间的通信，从而完成对海量数据进行查询和分析等操作。招聘领域的垂直搜索引擎的分布式数据库HBase的表设计，包括行键设计以及列族的设计，这两者都会影响分布式数据库HBase的存储以及查询性能[77]，在本文的招聘领域的垂直搜索引擎中，每天又存在海量的招聘数据存储以及查询操作，所以这两者的设计尤为重要，尤其是行键的设计，因为分布式数据库HBase是根据行键来进行检索的，系统通过某个行键（或者某个行键范围）所在的分区，然后将查询数据的请求路由到该分区获取数据。而分布式数据库HBase的行键以及列族的设计都有需要遵循的一定的规则，行键的设计规范如下。

（1）结合需求以及硬件等条件进行行键的设置。尽量让行键的长度保证在字节的整数倍，例如 8 字节大小。

（2）行键的设计不宜过长，过长的行键会导致内存的存储增大，并会影响检索操作的效率。

（3）行键的设计不宜采用连续增长的数字，因为这样的设计会导致数据存储到同一台节点服务器中，不利于节点服务器的负载均衡。

而同样在列族的设计规则中，合理设计规范如下。

（1）表中列族的个数不宜超过 3 个。一个分区中存储着多个列族，而当一个列族中操作大量数据的时候，会引发文件写入操作，而这个操作会影响到其他的列族，同样文件的合并操作也会影响其他列族，所以这会引起多余的文件读写等步骤。

（2）由于每个数据文件都会存储列族以及列名等信息，所以必须保证名字简短些。并且，需要操作频繁的列名与不频繁的列名分开存储。 根据上述所说的规则，对招聘数据进行行键及列族设计，并结合分布式数据库HBase 的特性，确定出招聘数据的表结构的设计。 招聘数据是通过网页采集模块采集得到的原始网页数据库，接着经过网页解析器得到的结构化数据信息，其中结构化数据各属性描述如表3-2所示。

表3-2招聘信息结构化数据描述

|  |  |
| --- | --- |
| 字段 | 描述 |
| Title | 标题 |
| Salary | 薪水 |
| CompanyName | 公司名称 |
| JobCategory | 职能类别 |
| WorkPlace | 工作地点 |
| Education | 学历 |
| WorkExperience | 工作经验 |
| ReleaseTime | 发布时间 |
| Email | 联系邮箱 |

如表3-2所示，这里标识了招聘信息结构化数据各个属性以及描述，接下来详细介绍该结构化数据表的设计。根据结构化数据信息、系统的功能需求以及性能需求分析、分布式数据库HBase 的行键设计规则等综合因素，对其进行行键的设计如表3-3所示。

表3-3行键的设计

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0字节 | 1字节 | 2字节 | 3字节 | 4字节 | 5字节 | 6字节 | 7字节 |
| 散列字段  当前整点时间的散列值  0-(0x00000000-OxFFFFFFFF) | | | | 扩展字段  唯一标识  0-(0x00000000-OxFFFFFFFF) | | | |

如表 3-3所示，行键分为 8 个字节，刚好对齐计算机 64 位操作系统，内存8 字节，这样利于操作系统的存取数据的速度，在第 0 到 3 个字节存放当前整点时间格式的散列值，其中当前整点时间指的是当前日期加上当前小时的组合，它保证每个小时爬取的数据存放在不同的节点服务器中，这样可以在每个节点服务器实现负载均衡，同时也是为了提高后续的时间段的查询效率，让其追加到索引库中，从而实现实时检索招聘信息。由于该段数据是一段连续时间的数据，所以这样的设计是有利于提高查询效率的。其余 4个字节用于唯一标识，可以解决行键重复的问题。在完成行键的设计之后，紧接着列族的设计，同样按照列族的遵循的设计规则，需要将字段分配到不同的列族当中，不可能将所有的字段信息都设计到一个列族里，所以列族的设计如表3-4下。

表3-4列族的设计

|  |  |  |
| --- | --- | --- |
| 行键：RowKey | 列族：MInfo | 列族EInfo |
| 整点时间散列+  唯一标识符 | MInfo:title | EInfo:Salary |
| MInfo:companyName | EInfo:staffNamber |
| MInfo:companyCategory | EInfo:companyIntroduction |
| MInfo:workPlace | EInfo:releaseTime |
| MInfo:education | EInfo:email |
| MInfo:workExperience | EInfo:business |
| MInfo:jobCategory |  |

如表3-4所示，表中包含两个列族，即 MInfo 和 EInfo，这样的设计是出于列族MInfo 主要是存放用于构建索引的字段，这些字段会在后续的操作中被查询，所以才放置到同一个列族当中，而对于列族EInfo主要存储暂时不需要被查询的字段，这样的设计即不影响后续的索引构建，同样还有提供很好的查询效率。另外设计成两个列族也满足不超过 3 个列族的规则。

以上就是分布式数据库HBase中表的设计，在本文招聘领域的垂直搜索引擎系统中通过采用以上的设计实现了数据在每个节点服务器上的负载均衡，同时也实现了系统对查询性能的要求。

## 3.8本章小结

本章旨在对招聘领域的垂直搜索引擎的需求分析与设计进行详细的介绍。首先是对招聘领域的垂直搜索引擎的需求进行分析，确定该系统的功能以及目标。接着对垂直搜索引擎中的数据库架构设计以及表的设计，主要包括 HBase 数据库集群的设计以及招聘信息表的设计，其中招聘信息表的设计涉及到行键的设计以及列族的设计。接着对网页爬虫模块的设计进行了介绍，其中包括基于分布式爬虫框架JLiteSpider实现一套抓取招聘网站中招聘信息的爬虫框架服务程序以及网页解析模块。然后对索引模块的设计进行了介绍，主要包括数据预处理、中文分词以及索引构建等部分。最后对检索模块进行详细的介绍，主要包括检索过程以及查询效果等部分。

# 4面向招聘领域的垂直搜索引擎系统实现

根据第3章的系统整体架构设计与相关功能模块设计，本节将介绍面向招聘领域垂直搜索引擎的具体实现，主要包括系统平台的搭建，各个模块的具体实现细节以及垂直搜索引擎页面效果的展示。

## 4.1HBase集群搭建

根据前面几节对系统的设计与功能的介绍，本文搭建了HBase集群来对海量数据进行存储以及提供高效快速的查询服务。本文中HBase集群部署在Linux系统集群中，本文以１个Master节点和２个Slave节点进行部署，并在其上完成基础步骤的安装，主要包括JDK安装、配置SSH免密码登陆、Hadoop集群安装配置、Zookeeper集群安装配置、HBase集群安装配置。本集群采用的版本是Hadoop2.7.3、HBase1.2.5和Zookeeper3.4.6。

HBase集权搭建主要步骤如下：

Step1：准备3台物理机或者虚拟机，安装centos7.0操作系统，保证3个节点之间可以相互ping通。

Setp2：下载软件包hadoop-2.7.3.tar.gz、hbase-1.2.5-bin.tar.gz、zookeeper-3.4.6.tar.gz和jdk-8u111-linux-x64.rpm。

Setp3：安装jdk，并进行jdk的环境配置。使用rpm -ivh jdk-8u111-linux-x64.rpm直接安装，vim打开/etc/profile，在文件的末尾追加以下内容：

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_111

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=.:$JAVA\_HOME/lib

export PATH=${JAVA\_HOME}/bin:${PATH}

Step4：配置集群间SSH无密码登录。执行ssh-keygen -t rsa（生成公钥和私钥）、

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys（导入公钥到认证文件）、chmod 700 ~/.ssh && chmod 600 ~/.ssh/\*（更改权限）。第一次登录需要yes确认，之后就可以直接登录了。

Setp5：将hadoop、hbase和zookeeper的安装包都解压到/data/yunva/文件夹下并重命名，安装目录如下：

/data/yunva/hadoop-2.7.3

/data/yunva/hbase-1.2.5  
/data/yunva/zookeeper-3.4.6

使用vim打开/data/yunva/hadoop-2.7.3/etc/hadoop/Hadoop-env.sh在文件末尾追加export JAVA\_HOME=/usr/java/jdk1.8.0\_111。三台机器上都是相同的配置，使用scp命令进行其他两个节点的配置：

scp -r /data/yunva/hadoop-2.7.3/     slave1:/data/yunva  
scp -r /data/yunva/hadoop-2.7.3/     slave2:/data/yunva

Step6：配置zookeeper集群环境，使用vim打开/data/yunva/zookeeper-3.4.6目录下的zoo\_sample.cfg，进行如下配置：

server.1=10.10.0.20:2888:3888  
server.2=10.10.0.21:2888:3888  
server.3=10.10.0.22:2888:3888

最后将该文件改为zoo.cfg。

Step7：配置HBase集群环境，vim打开/data/yunva/hbase-1.2.5/conf/hbase-evn.sh,在文末追加：

export JAVA\_HOME=/usr/java/jdk1.8.0\_111

export HBASE\_CLASSPATH=/data/yunva/hadoop-2.7.3/etc/hadoop/

export HBASE\_MANAGES\_ZK=false

export HBASE\_SSH\_OPTS="-p 48490"  # ssh端口非默认22需要修改

使用scp将整个hbase安装目录都拷贝到slave服务器上：

scp -P 48490 -r /data/yunva/hbase-1.2.5  slave1:/data/yunva/  
scp -P 48490 -r /data/yunva/hbase-1.2.5  slave2:/data/yunva/

至此HBase集群基本搭建完成。接下来启动HBase集群，查看集群是否安装成功。

1. 启动zookeeper，进入//data/yunva/zookeeper-3.4.6/bin/目录，执行./Server.sh statrt。使用jps检查是否有ZookeeperMain进程。
2. 启动hadoop，进入/data/yunva/hadoop-2.7.3/sbin/目录，执行./start-all.sh。使用jps检查是否有ResourceManager进程。
3. 启动hbase，进入/data/yunva/hbase-1.2.5/bin/目录，执行./start-hbase.sh。使用jps检查是否有HMaster进程。
4. 查看Hbase集群监控，在浏览器上输入<http://10.10.0.20/16010/master-status,master>节点显示如下图4-1所示：

//

图片

图4-1HBase集群监控管理图

## 4.2网页爬虫模块的实现

网络爬虫模块的主要功能是集成了前文所提JLiteSpider网络爬虫框架后，根据管理员设定好的招聘网站种子站点和爬取规则，从互联网上源源不断的爬取招聘信息网页存，然后经过HTMLparser处理网页信息，生成结构化的数据存入到HBase集群中。为之后的文本索引、索引创建等工作做准备。

4.2.1爬虫业务逻辑

该模块的主要工作是完成JLiteSpider提供的worker接口对应的业务逻辑的编写。JLiteSpider的worker接口由四个部分组成，分别是downloader、processor、saver和freeeman。Worker的四个接口的功能、接口中的主要函数进行详细说明：

1）downloader接口

该部分实现的是页面下载的任务，将想要抓取的url链表，转化为相应的页面数据链表。接口中主要函数的入参和返回值的说明如表4-1所示：

表4-1downloader函数

|  |  |
| --- | --- |
| 函数 | public void download(Object url, Map<String, MessageQueue> mQueue) throws IOException |
| 入参 | @param url：下载url所指定的页面  @param mQueue：提供把消息发送到各个消息队列的方法  @throws IOException 异常处理 |
| 返回值 | @return：void |

2）processor接口

processor接口时解析器的接口，这里会从网页的原始页面中提取出结构化的数据。在该接口中整合第三章中提到的HTMLparser，并且根据爬取的网站不同，设置对应的Xpath规则。接口中主要函数的入参和 返回值的说明如下表4-2所示：

表4-2processor函数

|  |  |
| --- | --- |
| 函数 | public void process(Object page, Map<String, MessageQueue> mQueue) throws IOException |
| 入参 | @param page：处理下载下来的页面源代码  @param mQueue：提供把消息发送到各个消息队列的方法  @throws IOException： 异常处理 |
| 返回值 | @return：void |

3）saver接口

save接口实现的时对解析得到结果的处理，这里时将解析得到的结构化数据存入HBase数据库、并且把与主题相关的url放入到队列中进行迭代抓取。接口中的主要函数的入参和返回值的说明如下表4-3所示：

表4-3saver函数

|  |  |
| --- | --- |
| 函数 | public void save(Object result, Map<String, MessageQueue> mQueue) throws IOException |
| 入参 | @param result：消息队列推送过去的结果消息  @param mQueue：提供把消息发送到各个消息队列的方法 |
| 返回值 | @return：void |

4）freeman接口

freeman接口时一个处理自定义消息格式的接口，本文利用该接口进行url的主题相关性分析和去重。接口中的抓哟函数的入参和返回hi之的说明如下表4-4所示：

表4-4freeman函数

|  |  |
| --- | --- |
| 函数 | public void doSomeThing(String key, Object msg, Map<String, MessageQueue> mQueue) throws IOException |
| 入参 | @param key：自定义的消息标记  @param msg：消息队列推送的消息  @param mQueue：提供把消息发送到各个消息队列的方法 |
| 返回值 | @return：void |

根据上面的各个work中的函数可以发现，每个work中都有一个入参Map<String, MessageQueue> mQueue，它是消息队列的一个对象，用于各个work之间的数据流转。worker和消息队列之间的关系可以是一对一，多对一，一对多，多对多，因此mQueue的数据类型采用java.util.Map。

4.2.2网页爬虫后台管理程序

网页爬虫后台管理程序可以对网络爬虫的爬取任务进行具体的设置、对运行的爬虫的状态进行监控同时可以暂停和终止爬取过程。后台管理界面如下图4-2所示：

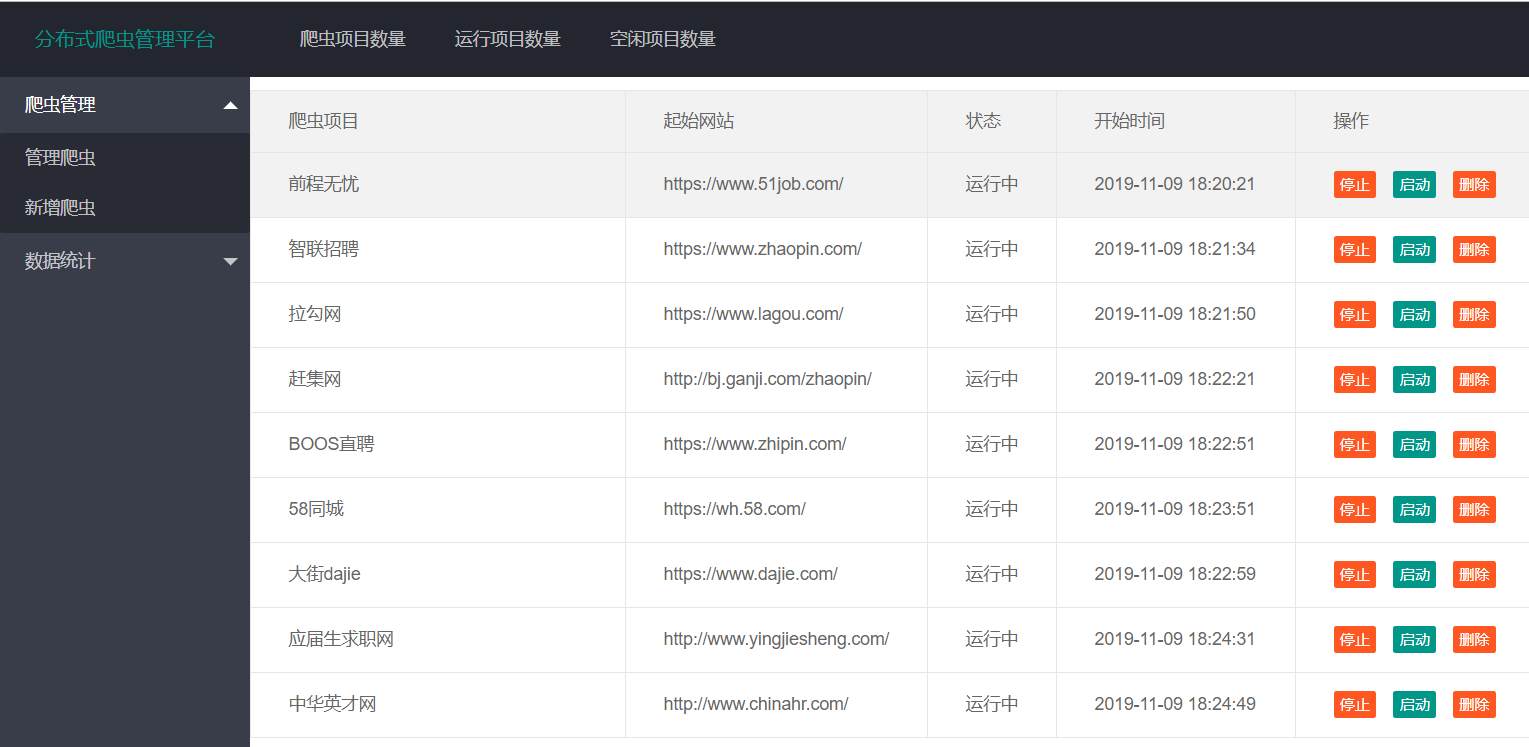


图4-2后台爬虫系统图

## 4.3索引模块的实现

索引模块主要包括两个部分，分别是使用word分词器对结构化数据进行分词，以及使用lucene创建索引库。

4.3.1中文分词模块的实现

在系统给中引入word分词器的maven依赖，就可以使用word分词器提供的分词功能。word分词器中核心函数如下表4-5所示：

表4-5word分词器核心函数seg

|  |  |
| --- | --- |
| 函数 | public static List<Word> seg(String text, SegmentationAlgorithm segmentationAlgorithm) |
| 入参 | @param text:待分词的文本  @param segmentationAlgorithm：指定分词算法 |
| 返回值 | @List<Word>:分词结果 |

这里的入参text可以是利用HBase数据库中的招聘信息生成的lucene文档，或者是用户输入的查询语句。然后选择合适的分词算法进行分词。Word分词器提供十种分词算法，具体的分词算法如下表4-6所示：

表4-6word分词的十种分词算法

|  |  |
| --- | --- |
| 算法 | 说明 |
| MaximumMatching | 正向最大匹配算法 |
| ReverseMaximumMatching | 逆向最大匹配算法 |
| MinimumMatching | 正向最小匹配算法 |
| ReverseMaximumMatching | 逆向最小匹配算法 |
| BidirectionalMaximumMatching | 双向最大匹配算法 |
| BidirectionalMinimumMatching | 双向最小匹配算法 |
| BidirectionalMaximumMinimumMatching | 双向最大最小匹配算法 |
| FullSegmentation | 全切分算法 |
| MinimalWordCount | 最少词数算法 |
| MaxNgramScore | 最大Ngram分值算法 |

默认使用的是BidirectionalMaximumMatching算法。分词结果如下表4-7所示：

表4-7分词结果

|  |  |
| --- | --- |
| 文本 | 分词结果 |
| 贝壳招聘高级资深java开发工程师25k-40k北京工作经验3-5年本科及以上25k-40k全职后端  快手招聘java开发工程师AI服务应用30k-50k北京工作经验3-5年本科及以上全职后端  建信金科基础技术中心招聘区块链技术专家50k-80k北京工作经验10年以上硕士及以上全职Golang后端  人人贷金融机构事业部招聘产品经理15k-25k北京工作经验2-5年本科及以上全职金融  拉勾网销售招聘电话销售7k-14k北京经验1-3年大专及以上全职 | 贝壳 招聘 高级 资深 java 开发 工程师 25k 40k 北京 工作 经验 3 5年 本科 及 以上 25k 40k 全职 后端  快手 招聘 java 开发 工程师 AI 服务 应用 30k 50k 北京 工作 经验 3 5年 本科 及 以上 全职 后端  建信 金科 基础 技术 中心 招聘 区块 链 技术 专家 50k 80k 北京 工作 经验 10年 以上 硕士 及 以上 全职 Golang 后端  人人贷 金融 机构 事业部 招聘 产品 经理 15k 25k 北京 工作 经验 2 5年 本科 及 以上 全职 金融  拉勾 网 销售 招聘 电话 销售 7k 14k 北京 经验 1 3年 大专 及 以上 全职 |

4.3.2构建索引库

创建索引库分为两个部分，将数据转化成lucene文档以及将文档写入索引库。通过IndexReader对象读取HBase数据库中的数据，构建lucene文档。创建IndexWriter对象，在创建该对象时需要指定分词器和存储索引的目录。调用IndexWriter对象的add()方法添加lucene文档，最后调用commit()方法生成索引库。执行上述操作后，结果如下图4-2所示：

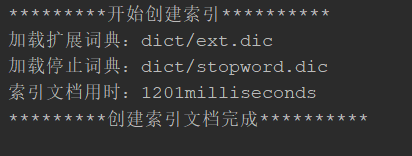


图4-2生成索引库图

## 4.4检索模块的实现

处理用户输入的查询语句其实就是对查询语句进行分词并且构建 Query 对象的过程。本文使用word分词器对用户输入的查询语句进行分词，并且将得到的分词结果容入Rocchio同义词生成新的查询关键词。接着使用Lucene 搜索文档，首先需要先读入索引文件，实例化一个 IndexReader 对象，然后实例化出 IndexSearch 对象，IndexSearch 对象的 search() 方法完成搜索过程，Query 对象作为 search() 方法的对象。搜索结果保存在一个 TopDocs 类型的文档集合中，然后利用Rocchio算法优化查询结果集，遍历集合输出文档信息。搜索结果如下图4-3所示：



图4-3检索结果

## 4.5本章小结

本章首先介绍了如何搭建分布式数据库HBase集群，HBase是由java语言开发的一个分布式数据库，并且HBase集群依赖于Hadoop集群。因此同时配置了java环境依赖和搭建了Hadoop集群依赖。然后是介绍各个模块的具体实现，分别是网页爬虫模块、索引模块和检索模块。

# 5面向招聘领域的垂直搜索引擎系统测试

本节主要是对招聘领域的垂直搜索引擎系统进行功能测试和性能测试。功能测试主要对网络爬虫模块以及检索模块进行测试。性能测试主要分为并发测试、搜索结果质量测试。

## 5.1功能测试

5.1.1网络爬虫测试

对网络爬虫的测试主要分为抓取的相关链接以及保存的相关网页数量两个部分，本文通过对前程无忧、智联招聘、拉勾网的招聘模块进行网络爬虫，抓取时间为一小时。详细的测试结果如表5-1、5-2、5-3所示。

表5-1前程无忧测试结果

|  |  |
| --- | --- |
| 指标 | 值 |
| 爬虫时间 | 60分钟 |
| 本次测试的所有链接 | 381972 |
| 相关链接数 | 3100 |
| 过滤的链接数 | 378872 |
| 抓取网页数 | 2865 |
| 存储网页数 | 2812 |

表5-2智联招聘测试结果

|  |  |
| --- | --- |
| 指标 | 值 |
| 爬虫时间 | 60分钟 |
| 本次测试的所有链接 | 321972 |
| 相关链接数 | 3522 |
| 过滤的链接数 | 318450 |
| 抓取网页数 | 2965 |
| 存储网页数 | 2509 |

表5-3拉勾网测试结果

|  |  |
| --- | --- |
| 指标 | 值 |
| 爬虫时间 | 60分钟 |
| 本次测试的所有链接 | 263972 |
| 相关链接数 | 4208 |
| 过滤的链接数 | 259464 |
| 抓取网页数 | 3221 |
| 存储网页数 | 2603 |

如表5-1、5-2、5-3所示，从三种测试中的网页所有链接数以及过滤的链接数可以得出，相似度计算过滤了大量与主题不相关的链接，进而降低了处理器、内存、带宽等网络资源的浪费，使得爬虫服务尽量只访问与主题相关的网页链接。再者根据三种测试中的抓取网页数以及存储网页数可以得出，网页与主题相似度的判定过滤了一些网页。然后根据浏览前程无忧、智联招聘、拉勾网的招聘模块的信息，可以发现对应的招聘模块与主题相关的链接数分别为 3100、3522、4208，而三种测试中的存储网页数分别为2812、2509、2603，其中智联招聘中的存储网页数较小，经过辅助测试，发现拉勾网中有重复的招聘信息，爬虫测试中过滤了重复的房源信息，所以导致只存储2603个网页信息。综上所述，可得该爬虫服务抓取了大部分的招聘网页信息，即证明该分布式爬虫服务的有效性。

5.1.2检索模块测试

通过对全文检索、不同域检索都进行了测试，全文检索的查询效果非常好。如输入“北京java开发”，该系统检索出的结果集都是与“北京”以及“java”关键词相关的数据，准确率高达 90%以上。而同样在百度输入该关键词，检索出的结果集包含一些无关的信息以及广告，准确率不高，证明了本文所设计的招聘垂直搜索引擎的有效性。

## 5.2性能测试

5.2.1压力测试

本文主要采用逐步加压的方式，测试搜索系统负载承受能力。采用Webbench软件来做压力测试。具体方案是使用一台具有４Ｇ内存、双核ＣＰＵ的64位服务器作为压力机，系统并发测试时长300秒，设置不同数量的并发用户进行测试。测试结果如表5-4所示

表5-4压力测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用户数量（人） | 响应（次） | 未响应（次） | 响应率（%） |
| 100 | 24321 | 21 | 99.91 |
| 200 | 26536 | 13 | 99.95 |
| 300 | 23965 | 26 | 99.89 |
| 400 | 24986 | 32 | 99.87 |

从表5-4可以看出，当系统并发用户为200时，系统依然可以达到很好的响应效率，同时达到用户的访问量峰值。

5.2.2搜索结果质量评估

搜索引擎的评价标准包括准确率、召回率等，本文采用准确率以及召回率来对实验结果进行评价。

（1）准确率

准确率是搜索引擎检索出的实际跟检索关键字有关的文档数与搜索引擎检索得到的文档总数的比率，如公式5-1所示。

(5-1)

其中，Precision 表示为准确率，tp表示搜索结果中相关文档数，fp表示搜索结果中不相关文档数。

（2）召回率

召回率是指搜索引擎检索出的跟检索关键字有关的文档数与实际跟该关键字有关的总文档数的比率，如公式5-2所示。

(5-2)

其中，Recall 表示为召回率，tp表示搜索结果中相关文档数，fn表示搜索结果中为搜索出的相关文档数。

本次实验通过设计两组实验，并对实验结果进行分析。两组实验分为 Lucene原始查询方式以及融合了Rocchio算法的扩展查询方式，通过对 5 组查询词分别进行两组查询操作，根据查询结果计算每组查询词的准确率以及召回率，进而对实验结果进行分析得出结论。本次实验涉及到一系列的参数的设置，如表5-4所示。

表5-5参数表

|  |  |  |
| --- | --- | --- |
| 参数名 |  | 值 |
|  |  | 1 |
|  |  | 0.75 |
|  |  | 0 |
|  |  | 30 |
| 最大查询个数 |  | 10 |

通过统计每组查询词的两组查询方式下查询结果中的相关文档数以及总相关文档数，得到每组查询词对应的准确率以及召回率。实验结果如表5-5、5-6所示。

表5-5原始查询结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 查询词 | 相关文档数 | 总文档数 | 索引库总相关文档数 |
| 北京 | 7889 | 7889 | 8210 |
| Java | 12115 | 12115 | 15233 |
| 全职 | 23331 | 23331 | 27322 |
| 高级工程师 | 2553 | 2553 | 2996 |

表5-6扩展查询结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 查询词 | 相关文档数 | 总文档数 | 索引库总相关文档数 |
| 北京 | 7989 | 7989 | 8210 |
| Java | 12915 | 12915 | 15233 |
| 全职 | 24331 | 24331 | 27322 |
| 高级工程师 | 2853 | 2853 | 2996 |

如表3-2、3-3 所示，通过对4组查询词的准确率以及召回率对比，融合Rocchio算法的扩展查询方式下的召回率要高于原始查询方式下的召回率，而这一优势则会使得扩展查询方式下的准确率低于原始查询方式的准确率。根据“北京”、“java”、“全职”三个查询词在两种查询方式下的准确率以及召回率的对比，可以证实召回率与准确率之间存在互逆关系。综上分析可得，融合Roochio算法的扩展查询方式能够在一定程度上提升查询的召回率。

## 5.3测试结论

通过上述测试结果表明，在网页爬虫模块中，主题网页爬虫具有较好的性能，通过url以及内容相关性分析过滤掉了大量与主题不相关的网页。在压力测试中，系统的表现也是良好。最后通过对搜索结果的准确率和召回率进行测试，发现融合了Rocchio算法的扩展查询拥有更好的效果。

# 6总结与展望

## 6.1总结

本论文详细介绍了面向招聘领域的垂直搜搜引擎系统的设计与实现的全过程，现在就整篇论文的工作进行总结。

1)本文首先介绍了面向招聘领域的垂直搜索引擎的研究背景与意义。随着互联网的飞速发展，网络上面的数据越来越多，面对日益增长的信息，如何能快速的检索到所关注的信息成为了关键。垂直搜索引擎了实现了只针对某一领域的搜索，大大提高了搜索效率。

2）本文介绍垂直搜索引擎的相关技术，首先分析了垂直搜索引擎的工作原理、招聘领域的垂直搜索引擎的技术特点以及该领域的信息特点。接着对垂直搜索引擎的开发步骤以及所涉及到的技术进行了详细的介绍，如网页采集技术、中文分词技术、主题页面抽取技术以及非关系数据库 HBase。这些技术都为本文后续的系统设计以及开发提供了有力的支撑。

3）本文对招聘的主题网络爬虫策略进行了设计，与传统的搜索引擎中的网络爬虫策略不同，招聘的主题网络爬虫策略在保证抓取的网页数量同时保证了抓取的网页具有很高的主题相关性，本策略结合网页分块的技术将网页链接的内容块考虑到链接与主题相似度的计算中，并同时将链接的相似度以及网页正文内容块的相似度的权重组合作为网页的相似度，进而保证主题网络爬虫策略的准确率。

4）索引是垂直搜索引擎中很重要的一部分，索引的建立可以大大提高检索的速度。在索引构建模块的设计中，利用了word中文分词工具对文本数据进行分词，为lucene建立文本的倒排索引提供了有力的支持。Lucene为文本建立了索引库，分为索引区和文档区，系统可以利用索引区来快速的对文档进行全文搜索。

5）本文针对基于 Lucene 检索模型的查询优化，通过对基于 Lucene 检索模块的查询问题进行描述，结合全文检索框架 Lucene 以及Rocchio相关反馈算法进行介绍，利用同义词优化以及Rocchio相关反馈算法对查询进行优化，通过与lucene原始的查询方式进行对比，融合了Rocchio算法的扩展查询有更好的准去率和召回率。

6）根据上述所提到的技术以及方法，对招聘的垂直搜索引擎进行实现，并对实现成果进行了介绍和展示。同时，对实现的主要功能模块进行了测试，以确保整个系统运行通畅、功能正确实现。最后对系统进行了性能测试，确保系统有一定能力经受多用户并发访问，并且用户搜索的结果有较好的查准率和召回率。

## 6.2展望

本文的面向招聘领域的垂直搜索引擎在一定程度上解决了招聘信息的搜索问题，但仍然存在一些不足，如网页爬取准确性和完成性问题、系统的搜索结果可以进一步的优化、某些网站设置了反爬虫限制以至于无法爬取的问题等。为了给用户提供更好的招聘信息查询服务，可以在以下几个方面深入研究与改进：

1）主题网络爬虫的性能直接影响着垂直搜索引擎的查询效果。本文所提及的招聘的主题网络爬虫的策略在抓取数据方面还是存在缺陷的，比如信息不全、准确率不高等问题。如何提高主题爬虫的抓取的准确性以及完整性是未来一个重点研究方向。

2）本文在垂直搜索引擎中查询优化方面还有待改进，本文所引入的相关反馈是属于伪相关反馈，并没有加入用户与垂直搜索引擎交互方面的信息，后继工作可以通过收集用户的交互数据，进而做到真实的相关反馈算法，使其查询效果更能满足用户的实际需求。

3）如何绕过网站的反爬策略，进而爬取更加完整的信息是今后需要改进的地方。

# 参考文献

[1] 中国互联网络信息中心(CNNIC). 第43次中国互联网络发展状况统计报告[R]. 北京:中华人民共和国互联网信息办公室, 2019

[2]Y.Yao,"Library Resource Vertical Search Engine Based on Ontology, " 2017 International Conference on Smart Grid and Electrical Automation (ICSGEA).Changsha, 2017, pp. 672-675

[3] W. Bruce Croft, Donald Metzler, Trevor Strohman. Search Engines, Information retrieval in practice. 1st edition, 2016

[4] Enjun Ding, Design and implementation of agricultural information resources vertical search engine based on nutch, Chemical Engineering Transactions, 201651: 619-624

[5]Yue H G, Ling Z, Meng F J, etal. Research and Implementation of a Vertical Search Engine in the Financial Domain[J]. International Journal of u-and e- Service, Science and Technology, 2014, 7

[6]Yue H G, Ling Z, Meng F J, etal. Research and Implementation of a Vertical Search Engine in the Financial Domain[J]. International Journal of u-and e-Service, Science and Technology, 2014, 7

[7] Xu Z, Yang Y, Wang F, etal. A Recommendation System for Paper Submission Based on Vertical Search Engine[J]. Lecture Notes in Electrical Engineering, 2014, 277:201-208

[8] WangWei. The Design and Implementation of Computer Vertical Search Engine[C] Seventh International Conference on Measuring Technology and Mechatronics Automation. IEEE, 2015:1168-1171

[9] Cheng L, Ma Y. Design and Implementation of Vertical Search Engine Based onHadoop[C] Eighth International Conference on Measuring Technology and Mechatronics Automation. IEEE, 2016:199-205

[10] Konstantin E. Avrachenkov and Vivek S. Borkar.Automatic construction of vertical search tools for the Deep Web[J].IEEE Journals & Magazines,2018:574-584

[11] Sejal D, Ganeshsingh T, Venugopal K R, etal. ACSIR: ANOVA Cosine Similarity Image Recommendation in vertical search[J]. International Journal of Multimedia Information Retrieval, 2017:1-12

[12] Hong Y, Lv C. Research of a professional search engine system based on lucene and heritrix[J]. Journal of Chemical & Pharmaceutical Research, 2014.

[13] Nikulin V A. System and method for ranking search results based on usefulness parameter: US20170075897[P]. 2017

[14] English P M, Gebhardt T M, Harkness K P, etal. Multi-phase search and presentation for vertical search websites[J]. 2017

[15] Example F, Skyscanner. The Impact of Price Comparison Service on Pricing Strategy in a Dual-Channel Supply Chain[J]. Mathematical Problems in Engineering,2013,(2013-10-29), 2013, 2013(6):206-226

[16] Diego Oswaldo Camacho Vega. Twitter and Blogs in Social Movements: Ayotzinapa around the World[M] In book: Global Perspectives on Media Events in Contemporary Society, Edition: First edition. 2016

[17] Linna Li, Zhifeng Li. The research of vertical search engine for recruitment, International Conference on Mechatronics and Information Technology, 2017, pp. 221-223

[18] Yuangui Lei, Victoria Uren, Enrico Motta. SemSearch: A Search Engine for the Semantic Web. Berlin:Springer Berlin Heidelberg, 2006,pp.208-245

[19] Michael Kohlhase, Ioan Sucan.A Search Engine for Mathematical Formulae. Berlin:Springer Berlin Heidelberg, 2006,pp.241-253

[20] Croft B, Metzler D, Strohman T. Search Engines - Information Retrieval in Practice[J]. Computer Journal, 2010, 54(5):831-832

[21] Johnson J, Tsioutsiouliklis K, Giles C L. Evolving Strategies for Focused Web Crawling[C] Machine Learning, Proceedings of the Twentieth International Conference. DBLP, 2003:298-305

[22] Kumar M, Vig R. Learnable Focused Meta Crawling Through Web [J]. Procedia Technology, 2012, 6(4):606-611

[23] Yang W, Liu J, Yu M. Research of an Improved Algorithm for Chinese Word Segmentation Dictionary Based on Double-Array Trie Tree[J]. Computer Engineering & Science, 2013, 205(9):1038-1042

[24] Karpierz K, Wolfman S A. Misconceptions and concept inventory questions for binary search trees and hash tables[C]// ACM Technical Symposium on Computer Science Education. ACM, 2014:109-114.

[25]杜明卉. 面向美食的垂直搜索引擎的设计与实现[D].吉林大学,2018

[26] Dai S, Shi C, Wu G. Study on forward increasing maximum matching algorithm for Chinese word segmentation[J]. Microcomputer & Its Applications, 2014

[27] Li C, Wang Y. Elimination of Chinese overlapping ambiguity based on dictionary and morpheme[J]. Microcomputer & Its Applications, 2013.

[28] Chen Z Y, Xiao-Jie L I, Zhu S H, et al. Bi-direction Maximum Matching Method Based on Hash Structural Dictionary[J]. Computer Science, 2015

[29]丁宝琼,谢远平,吴琼.基于改进DOM树的网页去噪声方法[J].计算机应用,2009,29(S1):175-177

[30] Yeh Chin gong, Jian Lee, Rule based word indentification for Mandarin Chinese sentences- a unification approach,Computer Processing of Chinese and Oriental Languages, 5(2), 1991,97-118

[31] Lang F. HTMLParser Extract Web Page Hyperlink Research[J]. Computer Programming Skills & Maintenance, 2010

[32] White T, Cutting D. Hadoop : the definitive guide[J]. O’reilly Media Inc Gravenstein Highway North, 2009, 215(11):1 - 4

[33] Vora M N. Hadoop-HBase for Large-Scale Data[C]International Conference on Computer Science and Network Technology. IEEE, 2012:601-605

[34] Carstoiu D, Lepadatu E, Gaspar M. Hbase - non SQL Database, Performances Evaluation[J]. International Journal of Advancements in Computing Technology, 2010, 2(5):42-52

[35] Sun J, Jin Q. Scalable RDF store based on HBase and MapReduce[C]International Conference on Advanced Computer Theory and Engineering. IEEE, 2010:V1-633 - V1-636

[36] Ding H, Jin Y, Cui Y, et al. Distributed storage of network measurement data on HBase[C] IEEE, International Conference on Cloud Computing and Intelligent Systems. IEEE, 2012:716-720

[37] Husain M F, Doshi P, Khan L, et al. Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce[C] Cloud Computing, First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings. DBLP, 2009:680-686

[38] Qiu Y, Frei H P. Concept based query expansion[C] 1993:160-169

[39] Cao G, Nie J Y, Gao J, et al. Selecting good expansion terms for pseudo-relevance feedback[C] International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2008:243-250

[40] H. Yu, Y. Liu, M. ZHANG, L. RU, and S. Ma, “Research in search engine user behavior based on log analysis,” JOUR-NAL OF CHINESE INFORMATION PROCESSING, vol. 21,no. 1, pp. 109–114, January 2007

[41] Yan T W. Index structures for selective dissemination of information under the Boolean model[J]. Acm Transactions on Database Systems, 1994, 19(2):332-364

[42] Lashkari A H, Mahdavi F, Ghomi V. A Boolean Model in Information Retrieval for Search Engines (PDF)[M]. IEEE, 2009

[43] Salton G, McGill M J. Introduction to modern information retrieval[M]. New York: McGraw-Hill, 1983

[44] Lv Y, Zhai C X. Adaptive relevance feedback in information retrieval[C] ACM Conference on Information and Knowledge Management. ACM, 2009:255-264

[45] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, The SMART retrieval system:Experiments in automatic document, pages 313–323,1971

[46] Jordan C, Watters C. Extending the Rocchio Relevance Feedback Algorithm to Provide Contextual Retrieval[C] Advances in Web Intelligence, Second International Atlantic Web Intelligence Conference, AWIC 2004, Cancun, Mexico, May 16-19, 2004. Proceedings. DBLP, 2004:135-144

[47]刘奕群.搜索引擎用户满意度评估[J].计算机研究与发展,2017,54(06):1133-1143