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**

**Research on optimization method of click rate estimation in recommendation system**

**——？**

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

**Huazhong University of Science & Technology**

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

**May, 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。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。

主题网络爬虫策略融合了网页采集、网页分块、主题相关度判定以及信息提取等技术。该策略在领域网页采集方面有着明显的优势，如具有很高的网络资源利用率以及信息采集的准确率。本文在考虑网页的相关度计算中，融合了链接所在的内容块与主题的相似度，通过对链接的相似度与网页内容的相似度的权重组合，来过滤与主题无关的网页，同时链接与主题相似度也能够降低爬虫服务对无关链接的访问量。

本文结合全文检索框架 Lucene、分布式爬虫框架以及 HBase 集群开发了一套招聘信息的垂直搜索引擎。该爬虫框架实时抓取国内多个招聘网站数据，目前已经抓取的网页库达到千万级的招聘信息数据，并存储到 HBase 集群，可用于数据分析以及挖掘等。该搜索引擎融合了同义词以及Rocchio相关反馈算法对原始查询的扩展，进而优化 Lucene 的默认查询，并提供了域查询以及模糊查询等多样化的功能。

最后对招聘领域的垂直搜索引擎中的网络爬虫以及检索模块进行了功能测试，并与通用搜索引擎的检索效果进行对比实验，本系统的检索结果具有很高的准确率。同时也对扩展查询与域查询进行了测试，结果表明扩展查询具有更高的召回率。

——一般分3段来写：第1段描述研究意义，即为什么要研究；第2段描述论文研究了的主要内容（系统采用什么工具按照什么方法步骤开发的以及实现的主要功能等），即研究了什么，不要记流水帐；第3段描述对论文的研究工作的评价（如系统测试及运行的状况、评价等），即研究得咋样。其中第2段是重点，第2、3段的文字量合起来约为第1段文字量的2倍。一般600～800字左右。

——摘要以第3人称形式写，不要出现“本文”、“我们”、……。

**关键词：垂直 搜索引擎主题网络爬虫Lucene相关反馈算法**

# 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. Therefore, it is an urgent problem to develop a search engine platform for recruitment information with vertical search engine technology.

The theme web crawler strategy integrates the technologies of web page collection, web page segmentation, theme relevance determination and information extraction. This strategy has obvious advantages in the field of web page collection, such as high utilization rate of network resources and accuracy of information collection. In this paper, the similarity between the content block and the topic of the link is integrated in the calculation of the correlation degree of the web page. Through the weight combination of the similarity between the link and the content of the web page, the irrelevant web page is filtered. At the same time, the similarity between the link and the topic can also reduce the visit amount of the crawler service to the irrelevant link.

This paper develops a vertical search engine for recruitment information based on Lucene, distributed crawler and HBase cluster. The crawler framework grabs real-time data of many recruitment websites in China. At present, the web page database that has been grabbed has reached tens of millions of recruitment information data and stored in HBase cluster, which can be used for data analysis and mining. The search engine combines synonyms and Rocchio feedback algorithm to expand the original query, and then optimizes Lucene's default query, and provides a variety of functions such as domain query and fuzzy query.

Finally, we test the function of the web crawler and the search module in the vertical search engine in the recruitment field, and compare the results with the general search engine. The search results of this system have a high accuracy. At the same time, extended query and domain query are tested. The results show that extended query has higher recall rate.

**Key words:**Vertical search engine Topic web crawler LuceneRelated feedback algorithm

目录

[摘要 4](#_Toc25618597)

[ABSTRACT 5](#_Toc25618598)

[1 绪论 8](#_Toc25618599)

[1.1研究的背景及意义 8](#_Toc25618600)

[1.2国内外研究现状 9](#_Toc25618601)

[1.3主要内容及结构安排 10](#_Toc25618602)

[2 关键技术研究 12](#_Toc25618603)

[2.1 垂直搜索引擎技术特点 12](#_Toc25618604)

[2.2网络爬虫技术 13](#_Toc25618605)

[2.2.1主题爬虫架构 13](#_Toc25618606)

[2.2.2网络爬虫策略 13](#_Toc25618607)

[2.2.3分布式爬虫框架JLiteSpider 14](#_Toc25618608)

[2.3 中文分词技术 14](#_Toc25618609)

[2.3.1 中文分词概述 14](#_Toc25618610)

[2.3.2 中文分词基本算法 15](#_Toc25618611)

[2.4主题页面抽取技术 17](#_Toc25618612)

[2.4.1 DOM 树 17](#_Toc25618613)

[2.4.2 HTMLParser技术 18](#_Toc25618614)

[2.5 非关系型数据库 HBase 19](#_Toc25618615)

[2.5.1 HBase 概述 19](#_Toc25618616)

[2.5.3 HBase 集群架构 22](#_Toc25618617)

[2.6 本章小结 23](#_Toc25618618)

[3 基于Lucene检索模型的查询优化 24](#_Toc25618619)

[3.1 问题描述 24](#_Toc25618620)

[3.2 Lucene 检索模型及评分算法 24](#_Toc25618621)

[3.2.1 布尔检索模型 24](#_Toc25618622)

[3.2.2 向量空间检索模型 24](#_Toc25618623)

[3.4 融合同义词与 Rocchio 算法的查询优化 27](#_Toc25618624)

[3.4.1 流程描述 27](#_Toc25618625)

[3.4.2 评价原则与标准 28](#_Toc25618626)

[3.4.3 实验环境及数据集 28](#_Toc25618627)

[3.4.4 实验设计及分析 28](#_Toc25618628)

[3.5 本章小结 29](#_Toc25618629)

[4 系统的需求分析与设计 31](#_Toc25618630)

[4.1 需求分析 31](#_Toc25618631)

[4.1.1功能需求分析 31](#_Toc25618632)

[4.1.2性能需求分析 33](#_Toc25618633)

[4.2系统的整体设计 33](#_Toc25618634)

[4.2.1系统架构设计 34](#_Toc25618635)

[4.2.2系统数据流设计 35](#_Toc25618636)

[4.3网页爬虫模块的设计 36](#_Toc25618637)

[4.3.1 爬虫服务架构设计 36](#_Toc25618638)

[4.3.2 网页抓取 38](#_Toc25618639)

[4.3.3文档解析与处理模块 38](#_Toc25618640)

[4.4 索引模块的设计 40](#_Toc25618641)

[4.4.1 数据预处理 40](#_Toc25618642)

[4.4.2 IKAnalyzer 中文分词器 41](#_Toc25618643)

[4.4.3 索引构建 42](#_Toc25618644)

[4.5 检索模块的设计 43](#_Toc25618645)

[4.5.1 检索过程 43](#_Toc25618646)

[4.6 分布式数据库HBase架构以及表设计 45](#_Toc25618647)

[4.7 本章小结 48](#_Toc25618648)

[5 系统实现与测试 49](#_Toc25618649)

[5.1系统开发环境 49](#_Toc25618650)

[5.2系统实现 49](#_Toc25618651)

[5.2.1前台部分实现 49](#_Toc25618652)

[5.2.2后台部分实现 51](#_Toc25618653)

[5.3系统测试 54](#_Toc25618654)

[5.3.1网络爬虫测试 54](#_Toc25618655)

[5.3.2 查询模块测试 55](#_Toc25618656)

[6 总结与展望 57](#_Toc25618657)

[参考文献 58](#_Toc25618658)

# 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%。

——注意参考文献标注的位置：摘要、正文的各级标题、每章开头的一段话、“本章小结”、图表的编号与说明文字等处不宜标注参考文献的编号。

——**文献编号标在正文中所引文字的后面及标点符号的前面处，而不是人名处。全文检查。**

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[3]。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[4]、雅虎等，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[5,6]，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[7,8]。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[9]针对数据急速增长的情况，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[10]等人设计了基于垂直搜索引擎的论文提交推荐系统，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[11]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。Cheng[12]等人设计和开发了一个基于 Hadoop 的垂直搜索引擎 HVSE，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[13]描述了一种自动集成Web 查询接口的方法。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[14]通过将垂直搜索引擎应用到图像推荐领域，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[15]通过对专业搜索引擎的研究，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[16]公开了一种生成搜索引擎结果页面的方法，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[17]有效的设计了一种旅行预订搜索引擎，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[18]，IceRocket 旨在针对 Twitter、博客、Facebook 等站点进行专业化的搜索[19]，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[20]。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[21]。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[22,23]。

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。

本文有五个组成部分，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章基于 Lucene 检索模型的查询优化，主要对基于 Lucene 检索模型的检索问题、Lucene 检索模型、Rocchio 相关反馈算法做了详细的介绍。最后，本文通过结合同义词以及 Rocchio 算法，在 Lucene 检索模型的基础上扩展查询向量进而优化查询功能。

第4章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5章招聘领域的垂直搜索引擎的实现与系统测试，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。然后是对网络爬虫以及检索模型进行功能测试。首先对国内多个招聘网站中的招聘模块进行爬虫测试，并生成测试报告。最后，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。

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。

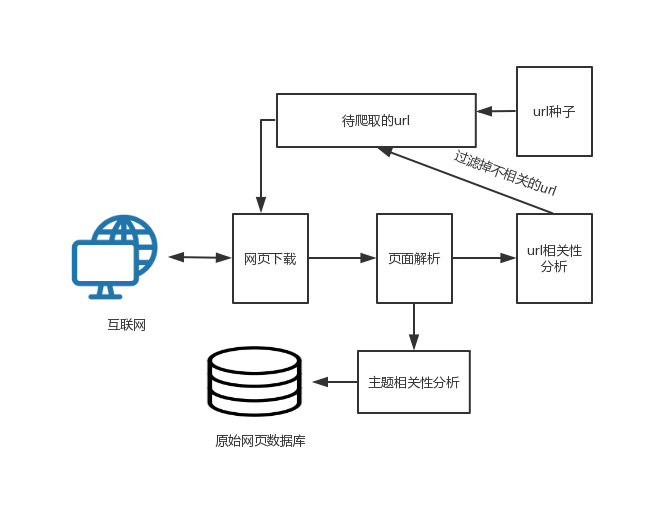
（4）个性化的复合检索以及排序方式

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[32]，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[33]。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[34]。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[35]。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[36]。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[37]。

（1）正向最大匹配算法

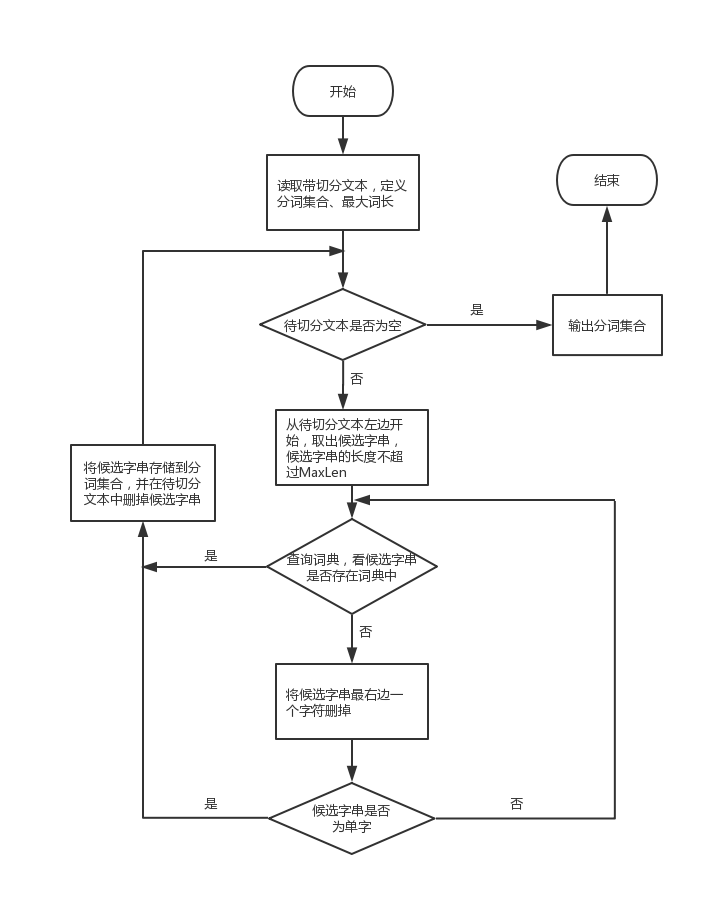
正向最大匹配算法，简称 MM 分词法。其主要思想就是从左到右将待分词文本中的 MaxLen个连续字符与词表匹配，其中 MaxLen 为词典中汉字最多的词条长度，如果待分字符与词表匹配上，则将该连续字符切分出一个词语。如果没有匹配上，则删掉该连续字符的最后一个字符，进而生成一个新的词重新再与词典中的词条进行匹配，依次重复以上操作，直到剩下最后一个字符，将其切分出一个单字。接着剩余的字符按照上述操作依次执行下去，直到整个待分词文本切分完成[38]。正向最大匹配法流程图如图 2-3所示

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[39]。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[40]。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[48]，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。

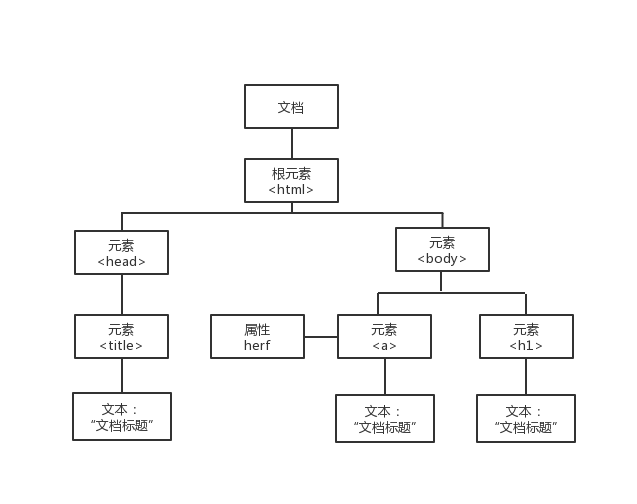


图2-4HTMLDOM树实例图

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[78]，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[45]基础上实现的一款非关系型数据库，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[46]。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[47]。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[48]。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[49]。

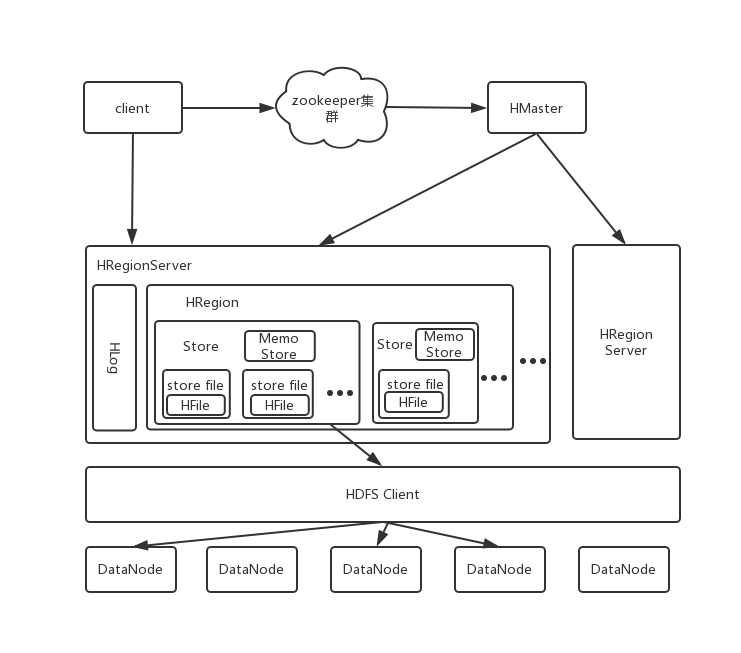
（4）高并发

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[50]。总体结构如图 2-5 所示。

2-5HBase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基于Lucene检索模型的查询优化

查询优化一直是信息检索领域的一个重要话题[67]。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[68]。本章主要介绍在基于 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。

**——本章的定位要弄清楚，如果是系统的一部分，就不能独立于系统之外。如果仅仅是介绍他人的工作就放到第二章中去；如果是自己的成果就作为后面系统设计和实现中的节。**

## 3.1 问题描述

在搜索过程中，存在很多因素会影响搜索结果的准确性以及排序效果。例如在查询词的选择方面，如果查询词并不能很好的表达用户的搜索意图时，则会直接导致查询结果的不准确[69]。导致出现这种情况的因素有：一方面是用户对于查询词表示不清楚，例如，用户想查找成都的“沁园新村”楼盘，但由于用户记不住了，只输入了“沁村”；另一方面是用户输入的查询词有多个同义词之类的词语，如用户想找成都西南交通大学旁的房源信息，输入“西南交通大学”，但由于该存在着“交大”、“西南交大”等同义词。对于这些因素导致的查询结果不理想，本章利用同义词词典来优化查询词，进而改善这类问题。除了这类问题，基于Lucene的默认检索模型仅仅是根据查询词与文档向量空间模型并结合余弦夹角来确定查询结果集并排序，并没有融合相关反馈信息来优化排序功能，所以本章在Lucene的检索模型的基础上融合相关反馈算法进而达到优化查询结果集。

## 3.2 Lucene检索模型及评分算法

3.2.1 布尔检索模型

布尔检索模型是最为常用的检索方式[70]，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[71]。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。

3.2.2 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[72,73]。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，接着通过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。

3.3 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[74]。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[75]。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34-4 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 l 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[76]。目前这三个参数的最优设置分别是α=1、β=0.75、γ=0.15。

## 3.4 融合同义词与Rocchio算法的查询优化

本文针对上述的问题，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。

3.4.1 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。

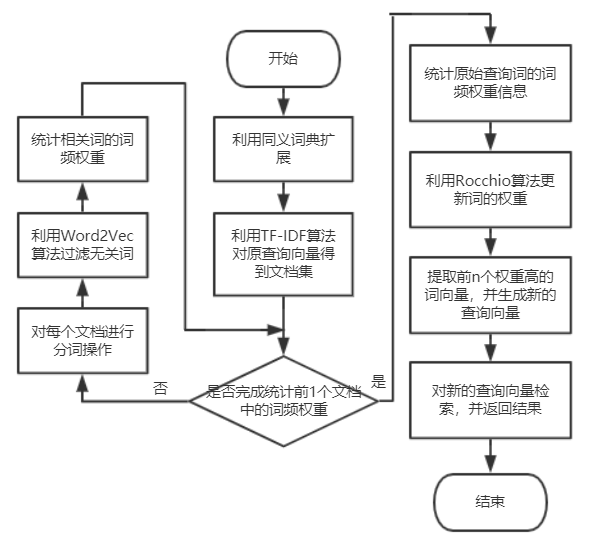


图3-1检索算法流程图

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，同义词典主要是二手房中的各个属性的同义词集合，如“装修”属性存在着“简单装修”、“简装”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 l 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 l 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。

3.4.2 评价原则与标准

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3-6所示。

(3-6)

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3-7所示。

(3-7)

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。

3.4.3 实验环境及数据集

本实验采用 Eclipse 开发工具以及 Lucene4.8 工具包来实现的，项目采用Maven 管理工具来实现项目管理以及项目架构的搭建，同时结合 Junit 单元测试工具来实现对接口以及类进行单元测试。

实验数据集来自于分布式网络爬虫系统采集到的二手房数据集，数据集达到千万级，本次实验采用部分数据集进行实验。通过对数据集构建索引库，并存储到本地索引库中。

3.4.4 实验设计及分析

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3-2所示。

表3-2参数表

|  |  |
| --- | --- |
| 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3-3、3-4所示。

表3-3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3-4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，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，扩展查询方式能够在一定程度上提升查询的召回率。

## 3.5 本章小结

本章首先对基于 Lucene 默认检索模型的查询问题进行了描述，指出了基于Lucene 默认检索方式存在查询词的出错、排序不合理等缺陷。其次，本章还对全文检索框架 Lucene 中检索模型进行了阐述，主要包括布尔模型以及向量空间模型进行详细介绍。接着，对同义词库的概念以及构建进行了介绍，其中，同义词的构建的好坏对于检索结果集会有很大的影响，一个完善、准确的二手房领域的同义词库将会有效的提高检索结果的准确率。同时，结合搜索引擎中相关反馈的机制理论知识，对Rocchio相关反馈算法作了详细介绍，阐述Rocchio算法如何将原始查询向量转换为最优查询向量，进而优化查询结果集的排序功能。最后，本章结合同义词优化以及Rocchio相关反馈算法，优化查询向量，进而优化查询结果集以及改变排序功能。

# 4系统的~~需求~~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。

——标题中的系统用全称。

## 4.1系统的需求分析

4.1.1功能需求分析

招聘领域的垂直搜索引擎系统的目标旨在实时搜集全国在互联网上发布的每日的招聘信息，同时提供用户接口，以供用户根据自身的购房需求检索出满足需求的招聘信息。从系统目标以及垂直搜索引擎的基础需求可以总结出，该系统主要包括网页采集模块、中文分词模块、索引模块以及检索模块等四个主要模块，而网页采集模块中包含主题网页爬虫、结构化数据解析、文本信息分析等模块，而中文分词模块同样包括构建招聘领域的词库等模块，检索模块中也包括对全文检索框架 Lucene 默认检索的查询优化等模块。详细功能模块结构图如图4-1 所示。

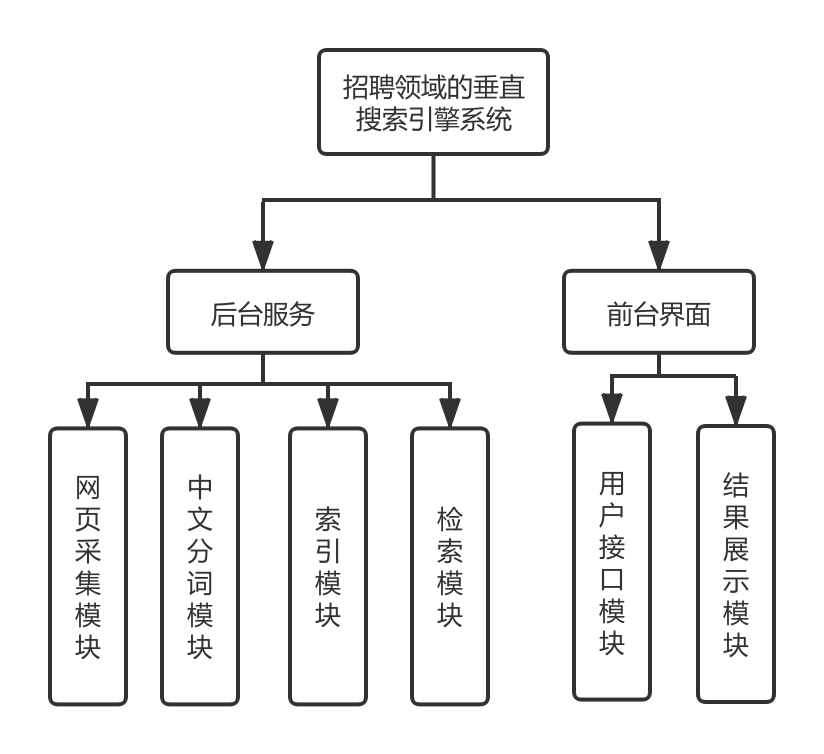


图4-1系统功能结构图——属于设计部分的内容，位置不正确

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二手房领域的垂直搜索引擎的研究，本文二手房的垂直

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二手房的垂直搜索引擎的文献进行了学习以及研究发现：需要二手房数据具有完整性，需要搜集好国内所有的二手房网站平台，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）丰富而又准确的二手房信息的检索功能。针对于国内的二手房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。

4.1.2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［43］。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。

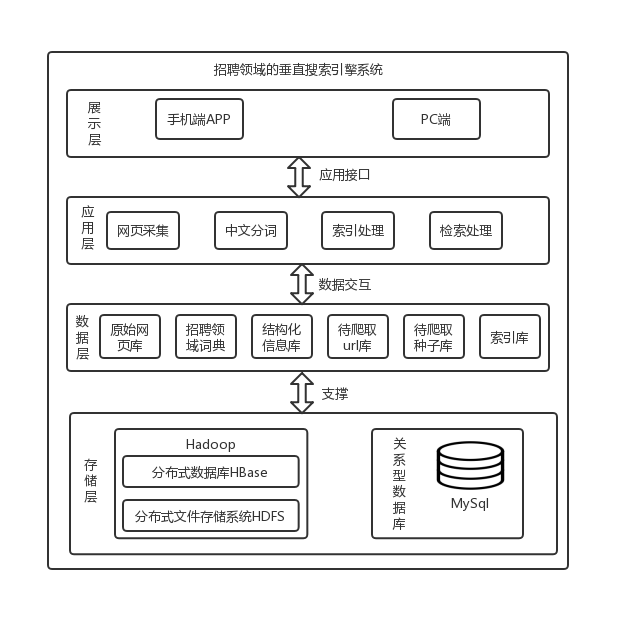
（5）具有系统运行日志、数据更新日志等日志。

## 4.2系统的整体设计

根据3.1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。

4.2.1系统架构设计

招聘领域的垂直搜索引擎整体架构如图4-2所示。

图4-2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。

4.2.2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4-3所示。

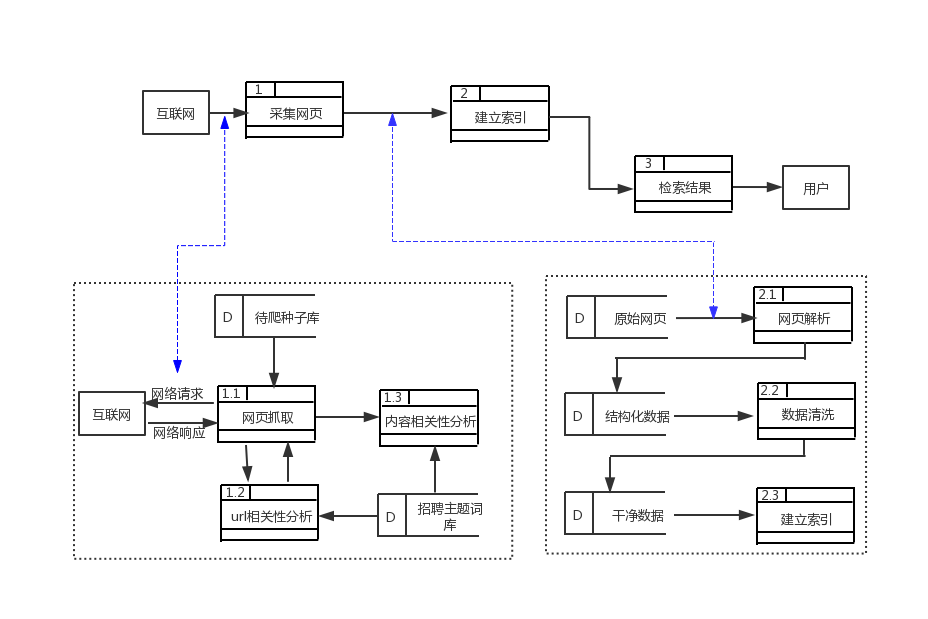


图4-3系统数据流图

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。

## 4.3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。

4.3.1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4-5所示。

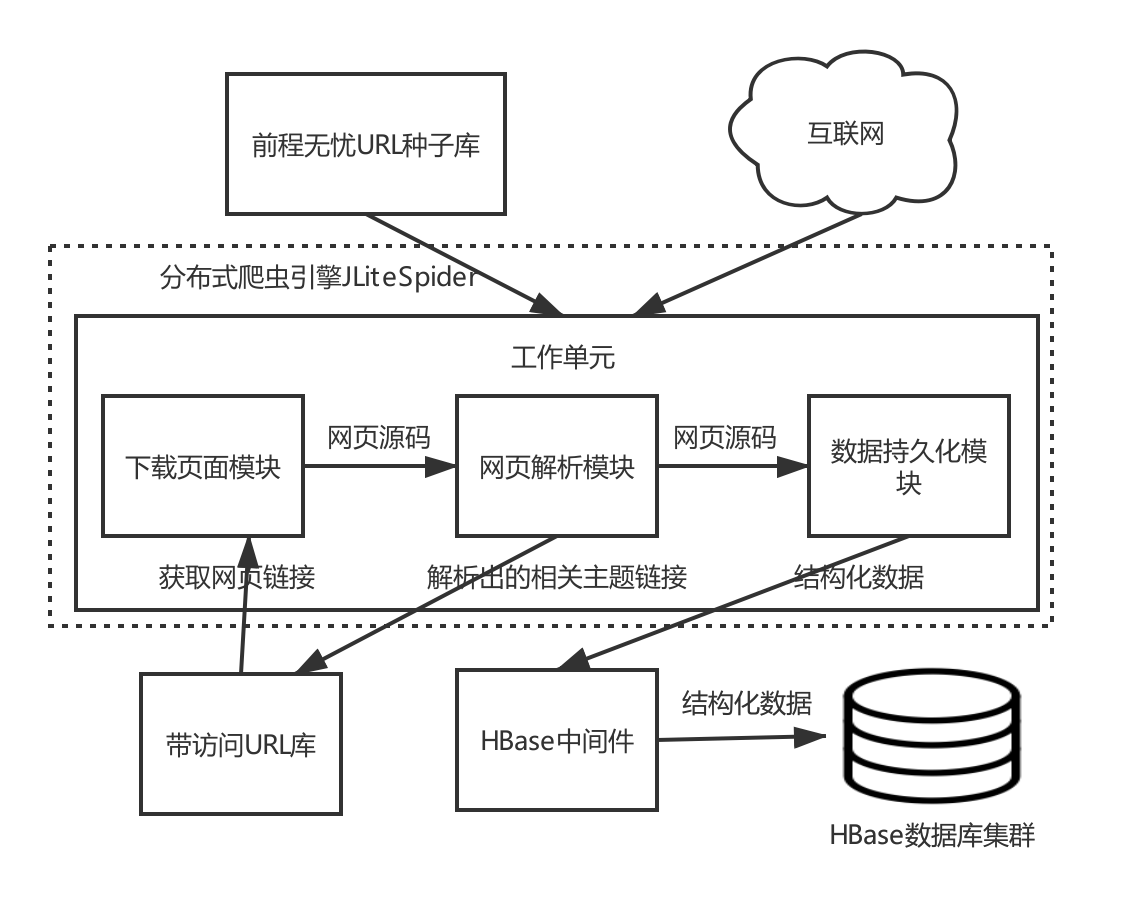


图4-5分布式爬虫服务架构

如图 4-5 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。

4.3.2网页抓取

本文将抓取数个网站中招聘信息页面，根据前面所提到的设计，每个分布式爬虫服务引擎针对一个网站所有城市的招聘信息页面进行抓取。这里以“前程无忧”网站为例。首先将网站的每个城市招聘信息模块链接地址添加到 URL 种子库中，启动分布式爬虫服务引擎，至于分布式爬虫服务中任务数可以根据网站的URL种子数来定，爬虫服务引擎启动之后，下载页面模块会请求访问地址，将网页下载下来，然后根据链接相似度计算模块并结合文档解析模块将与招聘主题相关的链接提取出来，并加入待访问URL队列中，同时也会根据网页相似度计算模块来保存原始网页，并解析出结构化数据与其一同存储到 HBase 数据库集群中。这里设置起始URL为https://www.51job.com。

4.3.3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。

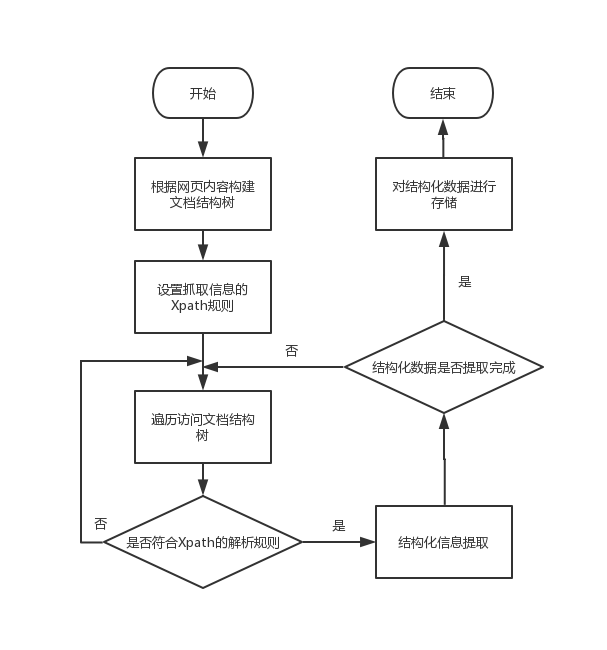
网页文档解析是从网页下载器中获得网页源代码，然后结合 HTMLparser 组件对网页文档构建文档结构树，为要抓取的招聘信息的属性信息设置不同的Xpath解析规则，然后结合文档结构树解析出招聘信息的各个属性信息，即完成了信息提取过程。HTMLparser组件对信息提取有着很高的效率，并且它的容错率也很高，同时也是目前最好的解析网页的工具库。本文采用HTMLparser进行网页解析与信息抽取，流程图如图 4-6所示。

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、价格、户型、楼盘、楼层等相关属性信息。

（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4-6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4-4所示，以及解析出的数据存储到 HBase 集群中的数据格式如表3-5 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4-4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() |

## 4.4 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。

4.4.1 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。

4.4.2 IKAnalyzer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，本文采用 IKAnalyzer 分词工具作为本系统实现中文分词模块的基础，IKAnalyzer是一个开源的，基于java语言开发的轻量级的中文分词工具包。本文整合IKAnalyzer工具类到搜索引擎系统的流程如图4-7所示。

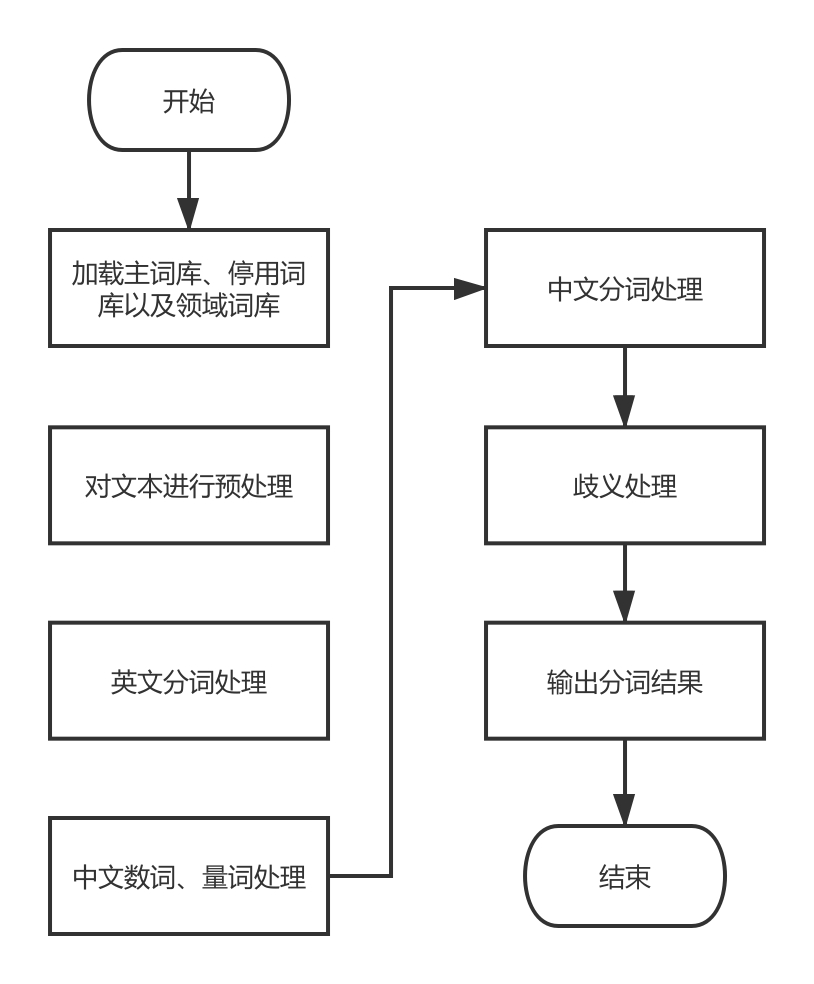


图4-7IKAnalyzer分词流程图

本系统根据爬虫服务搜集招聘领域专业词库以及各个城市楼盘名等数据，构建招聘领域词库用作扩展词库，同时结合基础词典，在全文检索框架Lucene基础上整合 IKAnzlyzer 中文分词器实现中文分词模块。

4.4.3 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4-8所示。

（1）配置招聘领域的词典并创建中文分词器 IKAnalyzer 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 2.4.1 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4-9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。

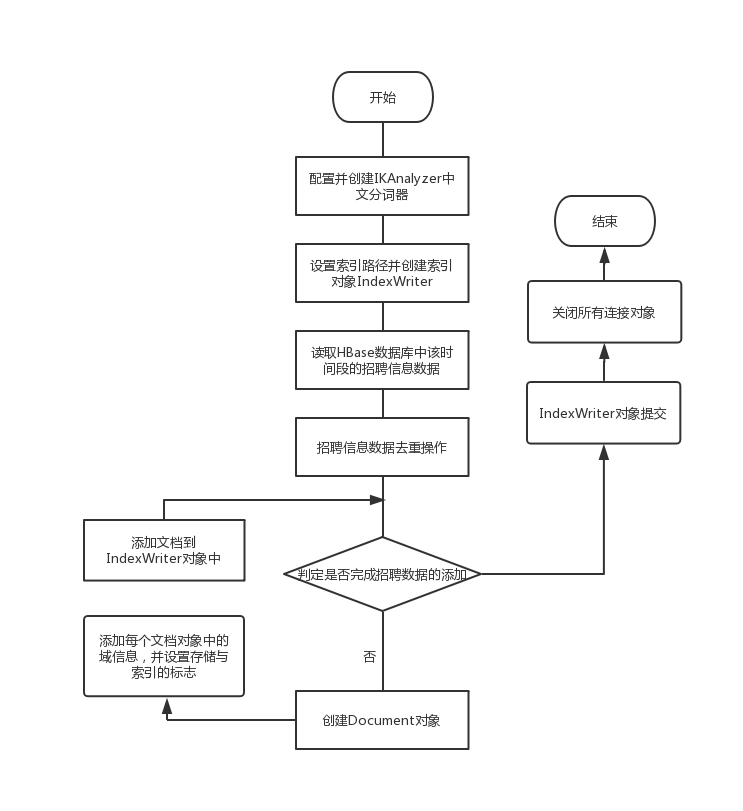


图4-8索引构建流程图——掉了“是”

## 4.5 检索模块的设计

4.5.1 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四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4-9所示。

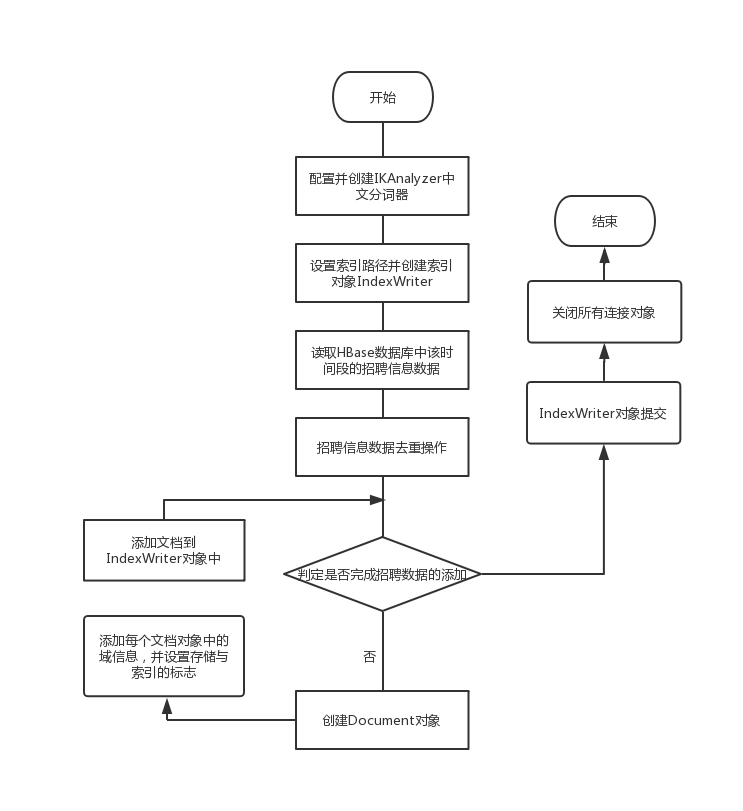


图4-9检索模块流程图——掉了“是”

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 IKAnalyzer 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。

4.6分布式数据库HBase架构以及表设计

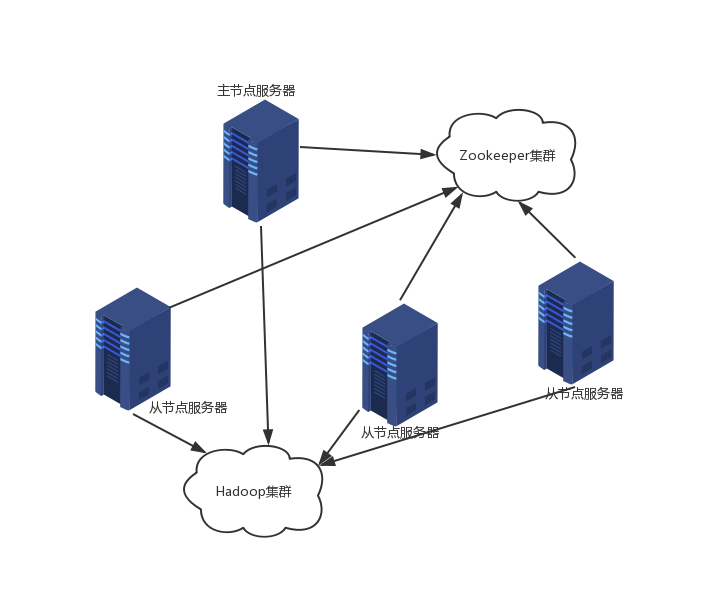
本文招聘领域的垂直搜索引擎采用 1 台主服务器与 3 台从服务器的设计方案来搭建分布式数据库 HBase 集群，而该集群又是基于 Hadoop 集群之上的，集群架构如图4-10所示。

图4-10HBase集群架构图

如图 3-4 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4-5所示。

表4-5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 |
| CompanyCategory | 公司类别 |
| StaffNamber | 公司人数 |
| Business | 公司业务 |
| CompanyIntroduction | 公司简介 |
| 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4-5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4-6所示。

表4-6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 3-2 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4-7下。

表4-7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4-7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。

## 4.7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、中文分块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。

# 5系统实现与测试

本章基于在线租房系统的系统设计，给出系统的开发环境配置，然后分别对系统的前台搜索和后台管理两部分的实现进行介绍。最后对系统的各个功能模块进行测试。

——实现部分应重点描述如何利用开发工具来实现功能的，不要写成操作手册，或者截图的堆砌。

——章末补充“本章小结”。

## 5.1系统开发环境

该系统的开发环境以及工具如表5-1所示。

表5-1系统开发环境

|  |  |
| --- | --- |
| 名称 | 描述 |
| HBase数据库集群 | 4台1T+8G服务器 Hadoop2.6.5、HBase1.2.5、Centos7.0 |
| 爬虫服务器 | 三台服务器 |
| 开发工具 | IDEA |
| 应用服务器 | Tomcat8 |
| 开发语言 | Java、Python |

## 5.2系统实现——本章的重点之一

在对系统的模块以及模块功能和系统的数据库进行了设计之后，通过代码的编写完成了对面向搜索领域的垂直搜索引擎的业务逻辑和系统功能的实现。下面将对面向招聘领域的垂直搜索引擎的前台搜索和后台管理两部分的实现进行介绍。

5.2.1前台部分实现

在前台实现了面向美食的信息检索的功能。在前端页面方面，使用了JQueryEasyUI框架，并将界面主色调设置为了浅蓝色。在页面最上方显示的是面向招聘领域的垂直搜索引擎的标题。然后下面是搜索功能框。从左到右分别是栏目选择下拉框（选择城市，默认是全部城市）、关键词输入框和搜索按钮。中间部分分为4个模块，从左到右分别是原文、预处理、查询分词和分词权重。前台的搜索界面如图5-1所示。

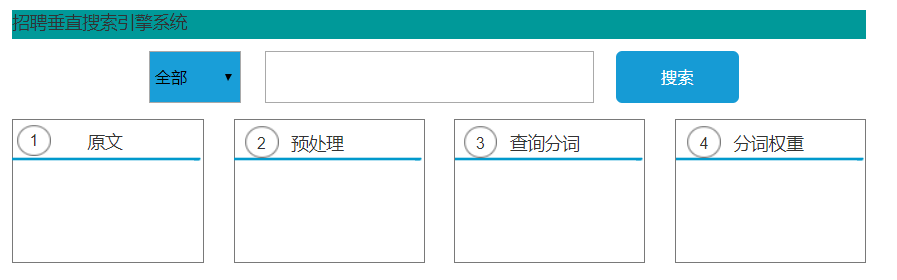


图5-1前台搜索界面图

选择好栏目，输入关键词，点击搜索按钮，就可以看到如图5-2所示的搜索结果界面了。



图5-2搜索结果界面图

从图中可以看出，第一部分展示了分词的情况，并且可以调整分词后每个词的权重。第二部分是检索的结果，每条内容中都包含职位、公司名称、工作地点、月薪和网页url链接地址。点击地址链接，则可看到如图5-3所示的网页详细内容。



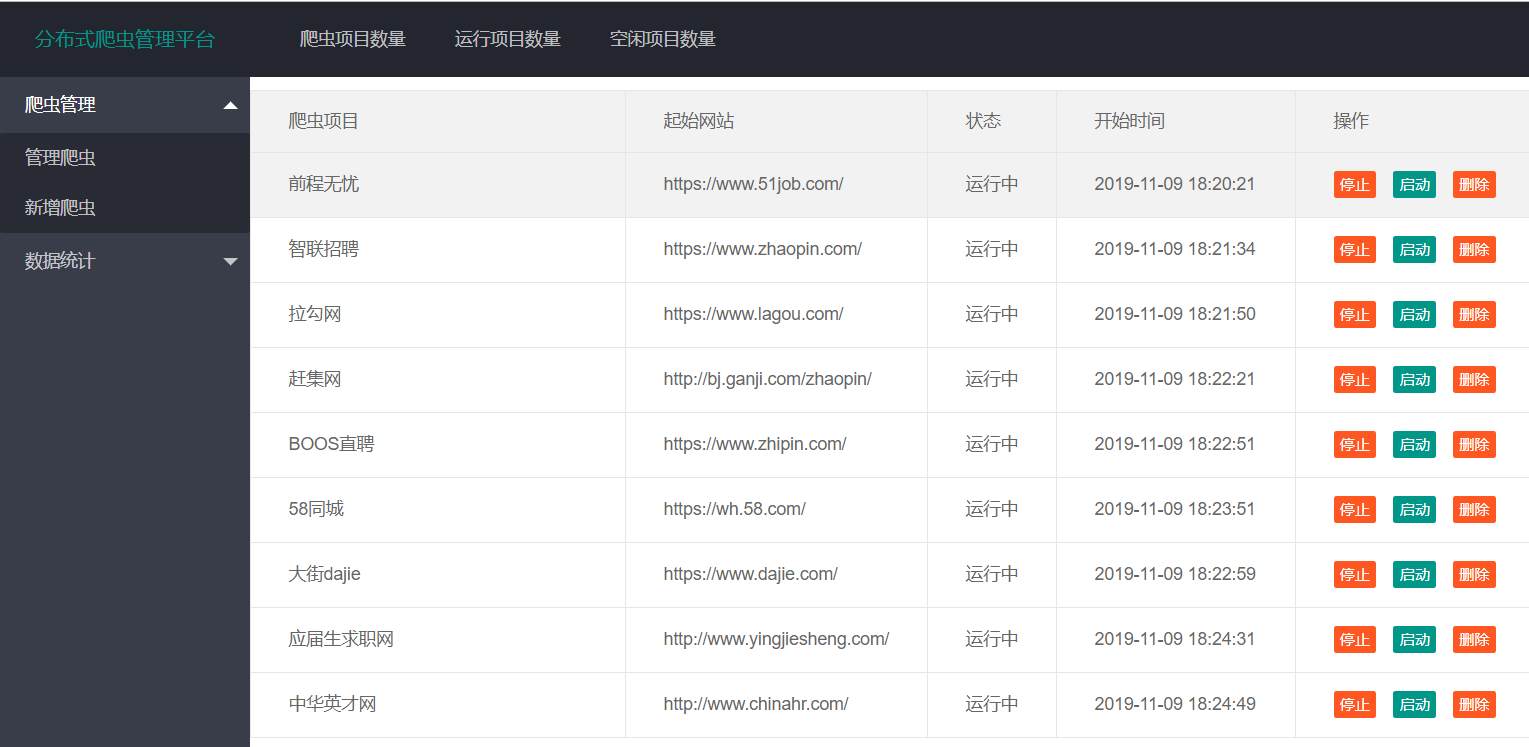
图5-3网页详情内容界面图

5.2.2后台部分实现

后台管理部分的界面，为了与前台搜索一致，也选择使用JQueryEasyUI框架。各功能模块以目录的形式显示在系统界面左侧。在后台管理部分，本文中户主要介绍一下栏目管理模块、本地数据管理模块和网络数据管理模块的实现。

1）爬虫管理模块

在栏目管理模块目录下，有管理爬虫和新增爬虫两个单元。爬虫管理界面如图5-4所示。在该界面的功能中，可以显示出各个爬虫的运行状态、开始时间和对应的操作。对爬虫可以进行停止、启动和删除操作。另外还有一个新增爬虫的模块，可以添加新的爬虫。



5-4爬虫管理界面图

2）本地数据管理模块

在本地数据管理模块功能目录下共有配置索引和创建索引两个小单元。其中，配置索引界面如图5-5所示。在页面中添加完数据后，点击“提交”按钮，页面中的数据将会被插入到数据库的本地索引数据表中。业务层的该单元的部分代码如下所示。

public String addIndex(){

this.index.setAction("add");

this.index.setCreateTime(DateUtils.getCurrentYMDHMS());

this.indexService.addObject(this.index);

return "addIndex";

}



图5-5配置索引界面图

3）数据统计模块

数据统计模块目录下面共分为网页数据和爬虫数据管理两个单元。其中，网页数据界面如图5-6所示。这里，系统需要获得数据库中网页表中的网页的数量其在业务层中的主要代码：this.columnList=(List)this.columnService.getAllObject(Column.class, "columnId", false)，然后在其对应的页面中获取columnList中的 id和name。主要

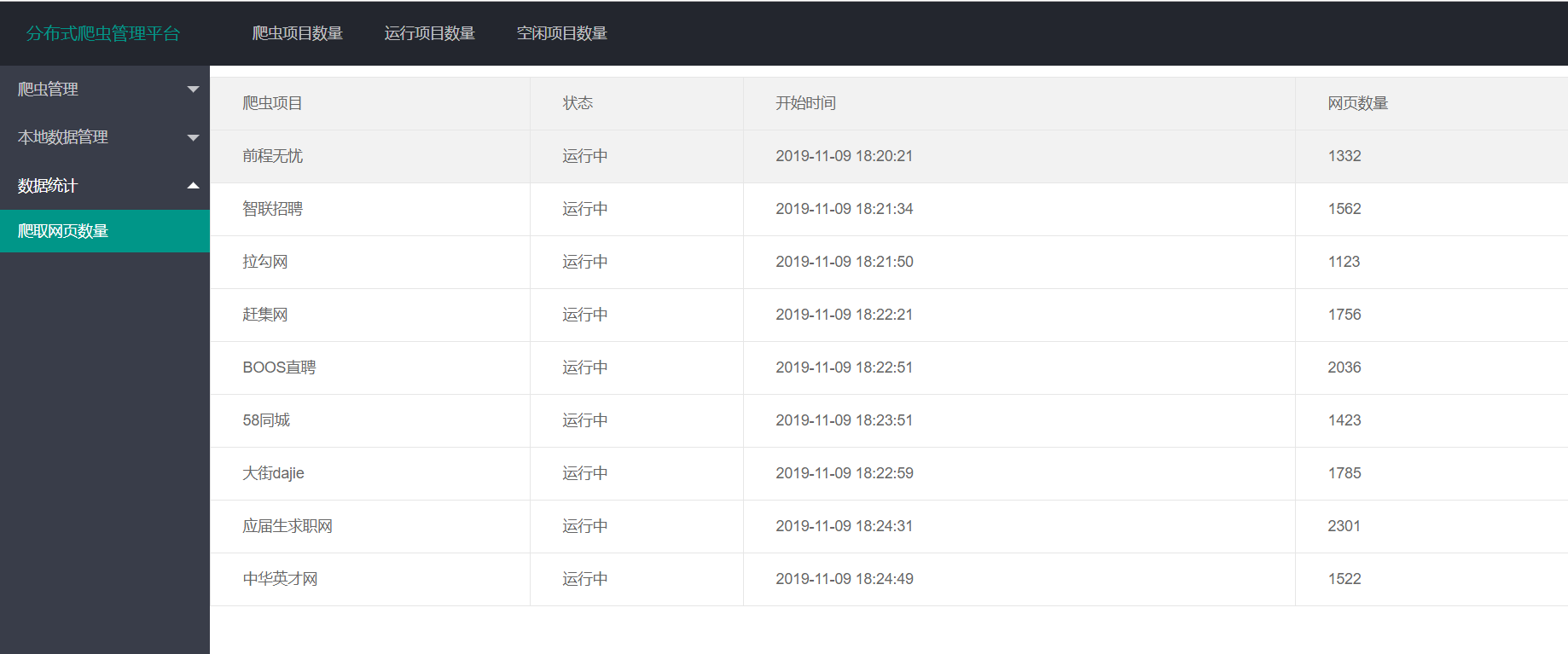


图5-6网页数据图

## 5.3系统测试——建议分功能测试与性能测试来写，最后增加“测试结论”

本节主要是对二手房的垂直搜索引擎中的网络爬虫模块以及查询模块进行功能测试，其中，网络爬虫模块的功能测试涉及到的网站数存在多个，这里仅仅对房天下网站上的二手房模块进行网络爬虫服务的测试，测试时间定为半个小时。而查询模块中功能测试包括全文查询以及特定域查询的测试。

5.3.1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 |
| 相关连接数 | 3100 |
| 过滤的连接数 | 378972 |
| 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 |
| 相关连接数 | 3522 |
| 过滤的连接数 | 325972 |
| 抓取网页数 | 2665 |
| 存储网页数 | 2409 |

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 |
| 相关连接数 | 4200 |
| 过滤的连接数 | 364978 |
| 抓取网页数 | 2821 |
| 存储网页数 | 2756 |

如表4-1、4-2、4-3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网页数分别在 3100、3500、4200左右，而三种测试中的存储网页数分别为2812、2409、2756，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，所以导致只存储了2756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。

5.3.2查询模块测试

本系统对爬虫抓取的招聘数据进行了索引库的构建，其中招聘数据总共千万条数据。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，同时检索速度也在毫秒级别。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。而在全文检索中，加入了查询扩展模块，用作扩展查询以及排序；例如同样输入“北京Java开发”查询词，选择扩展查询选项进行检索，发现与之前的没有做查询扩展的检索结果有所不同，并且文档排序也发生了改变。扩展查询能够扩展出更多的关键词，而这些关键词有可能会影响原始查询，所以将不相关的关键词过滤掉，留下相关的关键词用作扩展原始查询；同时，扩展查询会更新查询词的权重，进而达到改变查询以及排序效果。扩展查询通过在排名靠前的招聘信息中提取与原查询词相关的关键词，进而达到优化查询以及排序功能。由于搜索引擎的排序特点，即搜索结果集中排在前面的数据一般能满足用户的需求，所以扩展查询模块能够优化查询结果集使其排序的方式更适应的用户需求，进而提升用户的搜索体验。而且从用户使用搜索引擎的习惯来说，大部分用户只对检索结果中前面几页的数据感兴趣。除此之外，本系统也同样加入了不同域的检索，进而提供更为多样化的检索方式。比如用户可以按照“城市”、“职位”、“月薪”等域进行检索信息。这里以“城市”、“职位”两个不同域进行检索测试，检索效果同样也是很精确的，结果集都是与查询关键词相关的招聘信息。同时对于每个域实现了模糊查询。在面对用户输入错别字的时候，可以起到一定的优化查询的作用。全文检索展示如图5-7所示。



图5-7全文检索结果图

# 6总结与展望

——分“总结”与“展望”两节写，一般2页。

——总结的内容应比展望的多。

随着垂直搜索引擎相关技术的迅猛发展，该技术几乎被应用到了各行各业，但是，在招聘领域中，还并没有出现类似的产品，所以本文针对招聘领域开发一套实时抓取招聘信息的垂直搜索引擎。本文的主要的工作及贡献如下：

1）本文分析本课题的研究背景以及意义。

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、非关系数据库 HBase 等，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）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，改善了用户输错查询词问题以及优化查询的排序效果。

5）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。

针对垂直搜索引擎领域的现有问题，今后的工作主要从以下几个方面进行深入研究与分析。

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。由于招聘模块只是各大房产网站中一个子模块，加入特定网站的规则判定，所以它具有很大的局限性。

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。

# 参考文献

[1] 中国互联网络信息中心(CNNIC). 第41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, 2018.

[2] 严磊,马勇男,丁宾,郑涛.垂直搜索引擎之主题网络爬虫[J].福建电脑,2013,03:83-85.

[3] 王文钧,李巍.垂直搜索引擎的现状与发展探究[J].情报科学,2010,03:477-480.

[4] 刘畅.综合搜索引擎与垂直搜索引擎的比较研究[J].情报科学,2007,01:97-102.

[5] 周德懋,李舟军.高性能网络爬虫:研究综述[J].计算机科学,2009,08:26-29+53.

[6] 罗刚,自己动手写网络爬虫[M].清华大学出版社,2016.09:12-67.

[7] 邱哲，符滔滔.开发自己的搜索引擎[M].人民邮电出版社,2008.08:3-208.

[8] MichealMcCandless,ErikHatcher,OtisGospodnetic著.牛长流,肖宇译. Lucene实战[M].人民邮电出版社,2011.12,:.

[9] Trey Grainger,Timothy Potter 著.范炜等译. Solr实战[M]. 电子工业出版社,2017.05:8-438:.

[10] RajashreeShettar,RahulBhuptani. A Vertical Search Engine – Based On Domain Classifier[J]. International Journal of Computer Science and Security,2008,24:.

[11] RajashreeShettar,RahulBhuptani. A Vertical Search Engine – Based On Domain Classifier[J]. International Journal of Computer Science and Security,2008,24:.

[12] 李晓明.搜索引擎：原理、技术与系统[M].北京：科学出版社.2005.

[13] 陈英.基于语义的美食垂直搜索研究[D].武汉科技大学,2013.

[14] 杜雷.垂直搜索引擎网络爬虫的研究与设计[D].北京邮电大学,2015.

[15] 张环.垂直搜索引擎中主题网络爬虫算法研究[D].山东师范大学,2015.

[16] 罗路天.垂直搜索引擎中主题网络爬虫算法的设计与研究[D].广东工业大学,2016.

[17] 罗立宏 , 陈志 . 基于语义分析的垂直搜索网络蜘蛛 [J]. 计算机工程与设计,2008,18:4662-4665+4812.

[18] 郑如滨,撒力,谢婷.基于Heritrix与 Lucene 的垂直搜索引擎研究[J].电脑知识与技术,2008,29:350-352.

[19] 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.

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

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

[22] MichaelKohlhase, IoanSucan.A Search Engine for Mathematical Formulae. Berlin:Springer Berlin Heidelberg, 2006,pp.241-253.

[23] Albert Bifet, Carlos Castillo, Paul-alexandruChirita An Analysis of Factors Used in Search Engine Ranking. Adversarial Information Retrieval on the Web - AIRWEB, 2005, pp. 48-57.

[24] Jansen B J, Spink A. How are we searching the World Wide Web? A comparison of nine search engine transaction logs[M]. Pergamon Press, Inc. 2006.

[25] Jansen, B.J, Spink, A. &Saracevic, T. (2000). Real life, real users, and real needs: A study and analysis of user queries on the Web. Inf. Proc. Manage. 36, 2, 207-227.

[26] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine [C]// International Conference on World Wide Web. Elsevier Science Publishers B. V. 1998:107-117.

[27] Parul Gupta, Dr. A.K. Sharma, A framework for multilevel Indexing in Search Engines, accepted in International Journal of Applied Engineering Research.

[28] Richard Zanibbi and Dorothea Blostein. Recognitionand retrieval of mathematical expressions. International Journal on Document Analysis and Recognition (IJDAR),15(4):331–357, 2012.

[29] Xu X C, Xu X S, Wang Y, et al. A heterogenous automatic feedback semi-supervised method for image reranking[J]. 2013:999-1008.

[30] 张正龙. 基于 LUCENE的主题搜索引擎研究与实现[D]. 重庆:重庆大学, 2008, 9-10.

[31] 曹勇刚, 曹羽中, 金茂忠, 等. 面向信息检索的自适应中文分词系统[J]. 软件学报, 2006, 17(3):356-363.

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

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

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

[35] 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.

[36] 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.

[37] 任书琴. 健康领域的垂直搜索引擎的研究与实现[D]. 电子科技大学, 2016.

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

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

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

[41] Jiang J H, Zhao S Z, Luo M. Analysis and application of Chinese word segmentation model which consist of dictionary and statistics method[J]. Computer Engineering & Design, 2012, 33(1):387-391.

[42] 张旭. 一个基于词典与统计的中文分词算法[D]. 电子科技大学, 2006.

[43] Peng H C, Tong M W, Zou J H, et al. Rule-based Preprocessing Algorithm for Web Page Segmentation[J]. Computer Science, 2013.

[44] Greibus M, Telksnys L. Speech Keyword Spotting with Rule Based Segmentation[J]. Communications in Computer & Information Science, 2013, 403:186-197.

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

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

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

[48]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.

[49] 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.

[50] Husain M F, Doshi P, Khan L, et al. Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce[C] Cloud Computing, First InternationalConference,CloudCom2009,Beijing, China, December 1-4, 2009. Proceedings. DBLP, 2009:680-686.

[51] Gospodnetić O, Hatcher E, Cutting D. Lucene in action[M]. Manning, 2005.

[52] 胡博. 基于Lucene 的垂直搜索引擎研究与实现[D]. 北京工业大学, 2016.

[53] 江宇, 姜琨, 宋省身,等. 倒排链表多层自索引结构的分析与设计[J]. 计算机应用研究, 2017, 34(2):473-476.

[54] 王甲坤. 基于互联网的话题分类及敏感话题发现技术研究与实现[D]. 电子科技大学, 2011.

——所有参考文献按在正文中被引用的先后顺序进行编号，并在正文中被引用的地方进行标注！

——先分清所引文献的类型，然后按照该类文献的格式要求进行修改；所引参考书应标明引用的范围（页数不要多）；删除文献最后面的句点。

——参考文献按照格式进行补充和修改。注意所有标点符号均为英文标点符号，英文标点符号后面留一个空格。