中文分词技术在搜索引擎中的研究与应用

学位论文完成日期：_____

指导教师签字：_____

答辩委员会成员签字：_____

_____

_____

_____

_____

_____

_____

中文分词技术在搜索引擎中的研究与应用

摘 要

中文分词是计算机进行汉语文本分析的关键技术，分词算法的好坏直接影响中文分析系统的实用性，搜索引擎是中文分词技术的重要应用之一。如何用更短的时间得到更高的分词精确度是目前的研究重点和热点。基于字符串匹配的分词算法是当前使用最为广泛的中文分词算法，而最大匹配算法是最常用的基于字符串匹配的分词算法。本文通过分析最大匹配算法的不足，结合高效的双字哈希词典机制，提出基于双字哈希词长分组词典结构的正向最大匹配改进算法，分词性能明显提高；然后借用匹配过程进行歧义处理，减少错误切分；之后使用改进算法思想重新设计 Lucene 中的中文文本分析模块，优化搜索引擎系统。实验表明本文提出的基于双字哈希词长分组词典结构的正向最大匹配改进算法比最大匹配算法有较大性能提升。本文所做工作总结如下：

1.通过对最大匹配算法的研究，分析最大匹配算法存在的 3 个问题，并针对每个问题提出解决办法。

2.根据最大匹配算法不足改进算法流程，提高分词性能，并针对改进算法的需求设计双字哈希词长分组的词典机制，提出基于双字哈希词长分组词典结构的正向最大匹配改进算法。该算法对于每一次匹配都能动态选择合适匹配初始位置和匹配长度，并能快速的对词典进行查找，减少不必要的匹配消耗，无论从分词速度还是精确度上都较传统算法有了一定提升。

3.根据改进算法的匹配过程，结合最大匹配算法+回退一字法算法思想，有效消除部分交集型歧义，使分词结果更加准确。

4.通过对搜索引擎知识和 Lucene 开发包的学习，基于 Lucene 搭建简单的搜索引擎系统。根据改进算法重新设计 Lucene 中文分析模块，提高了基于 Lucene 的搜索引擎系统应用性能。

5.对基于双字哈希词长分组词典结构的正向最大匹配改进算法进行实验评估。首先使用不同词典机制对相同语料进行分词，验证本文选取的双字哈希词典性能；然后通过使用本文改进算法及正向最大匹配算法对相同语料分别进行分词，比较结果。实验结果表明，本文提出算法在分词速度及分词精度上都优于正向最大匹配算法，达到了改进目的。

关键词：中文分词 搜索引擎 Lucene 最大匹配算法 双字哈希 歧义处理

THE RESERRCH AND APPLICATION OF CHINESE WORD SEGMENTATION TECHNOLOGY IN SEARCH ENGINE

ABSTRACT

Chinese words segmentation is the key technology for computers to carry out Chinese text analysis. Therefore, whether Chinese word segmentation algorithm is good or bad has a direct impact on the practicability of Chinese analysis system. Search engine is one of the important applications of Chinese word segmentation technology. How to get higher precision in shorter time is the focus and hot spot of current relevant research. The maximum matching method is the most commonly-used word segmentation algorithm based on string matching, which is the most widely-used Chinese word segmentation algorithm. By analyzing the disadvantages of maximum matching algorithm, combining with efficient dictionary mechanism of double character hash, this paper proposed improved forward maximum matching method based on double character hash words length grouped dictionary structure, and the performance of word segmentation has been improved obviously. Then matching process was handily used for disambiguation, which was to reduce the wrong word segmentation. Later according to the thoughts of improved algorithm, Lucene Chinese text analysis modules of were designed anew to optimize the search engine system. Tests showed that the improved forward maximum matching method based on double character hash words length grouped dictionary structure proposed in this paper had greater performance than maximum matching method. Work summaries in this paper are as follows:

1.Based on the research of the maximum matching method, this paper analyzed three problems of the maximum matching method, and proposed corresponding solutions for each problem.

2.According to the disadvantages of maximum matching algorithm, performance

of word segmentation has been improved. And the author also designed dictionary mechanism of double character hash words length grouped and proposed the improved forward maximum matching method based on double character hash words length grouped dictionary structure, to meet the demand of improved algorithm. This algorithm can dynamically select appropriate initial matching position and length for each matching, and search dictionary quickly, which can reduce unnecessary matching consumption. Therefore, both segmentation rate and accuracy have greater improvement than traditional algorithm.

3.On the basis of matching process of improved algorithm, combined with the algorithm thought-maximum matching algorithm and back word method, some crossing ambiguities were effectively eliminated. That made the results of word segmentation more accurate.

4.By studying the knowledge of search engine and Lucene development kit, simple search engine system based on Lucene was established. According to the thoughts of improved algorithm, Lucene Chinese text analysis modules of were designed anew, which optimized application performance of search engine system based on Lucene.

5.Experimental evaluations was conducted for the improved forward maximum matching method based on double character hash words length grouped dictionary structure. Firstly, the same corpus was segmented with different dictionary mechanisms, to test the performance of double character dictionary selected by this paper. Then the same corpus was segmented on the basis of improved algorithm and forward maximum matching method, later the results were compared. Experiments showed that word segmentation rate and precision of algorithm propose by this paper were better than that of forward maximum matching method. In conclusion, the improvement is achieved.

KEY WORDS: Chinese words segmentation; Search engine; Lucene; Forward maximum matching algorithm; Double hash structure; Ambiguity processing

目　录

1 绪论

1.1 研究背景与意义

互联网是当前最重要的信息聚集地，聚集了海量的数据信息，而搜索引擎作为一种重要的信息提取平台，在用户的信息获取过程中发挥了重要作用。在这种大背景下，如何提高搜索引擎的检索精度和速度，使用户能够快速高效的获取自己想得到的信息，是需要持续思考和解决的问题。鉴于汉语词汇语义的复杂性，搜索引擎中对中文的处理技术成为尤其关键的问题。

中文分词（或中文切词）是指将连续的字序列按照一定的规范重新组合成词序列。以“我们马上要毕业了”为例，中文分词的过程就是如何将例句切分为“我们”、“马上”、“要”、“毕业”、“了”五个词语单元。中文分词是中文信息的关键技术之一[1]，是中文网页分析的前提。分词的精准度会直接影响搜索引擎对关键字的处理，关系到是否能够提取用户真正需要的信息；分词算法的时间复杂度也会影响搜索引擎的检索速度。中文分词的详细框架，如图 1-1 所示。

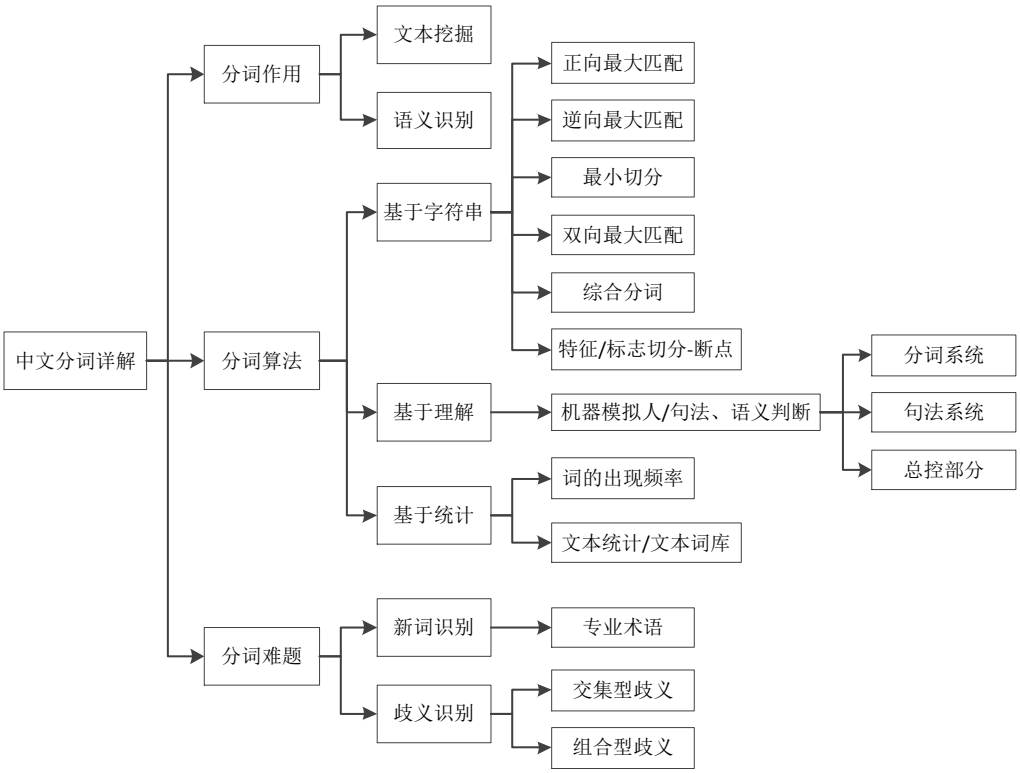

图 1-1 中文分词详解

Fig.1-1 The introduction of Chinese words segmentation

网页分析是将一个文档表示为特征项的过程，在对中文文本进行文本分析

前，需要先将完整的句子分割成小的词语单位，即中文分词。在特征提取时，汉语又面临了和英语处理方式不同的问题。中文信息与英文信息有一个明显的差别：英语单词之间用空格分隔；而在汉字文本中，词与词之间没有自然的间隔符，汉语词大多是由两个或两个以上的汉字组成，而且词语与词语是连续的。

分词技术的运用在互联网内容搜索中极其普遍，如 Google、百度、Yahoo 等，这些搜索引擎将用户的搜索内容拆分成多个关键词，然后根据这些关键词进行数据搜索。Google 的中文分词系统选用的是美国 Basis technology 公司提供的中文分词技术，Baidu 采用的是百度内部研发的分词技术和算法。这些公司的中文分词算法大部分都是采用最小切分算法结合语义理解的方式，这样的设计可以让搜索引擎尽量找全用户关心的内容[2]。

1.2 中文分词研究现状

中文文本是以词作为最小的语言成分进行独立活动，但词与词之间并没有像英文中词与词之间存在空格这样明显的区分标记，因此对中文分词的深入研究就十分必要。概况的说，中文分词就是通过制定一些计算机可识别的规则，对中文文本进行处理，从连续的字序列组合切分成词序列的过程。由于中文词汇同义性、多义性等特征，相同的词汇在不同的语境下或者不同的词汇在相同的语境下都可能存在多种可能，导致中文分词技术的研究存在许多的技术难点。因而，目前针对中文分词的研究仍主要集中在自然语言的处理技术上。中文分词的广泛应用如搜索引擎、机器翻译、自动摘要、语音合成等，这些实际的需求使得尽管面对种种技术难题，中文分词技术依然稳步前进，进步明显。随着人们对网络数据的需求加大，搜索引擎作为一种重要的信息获取平台，同时作为中文分词的一项重要应用，极大的推动了中文分词技术的发展。就目前搜索引擎技术的发展现状来看，国外搜索引擎技术要优于中文搜索引擎技术一段距离，但鉴于中文词汇的复杂性，中文搜索引擎并不能直接采用国外的一些先进技术，而是必须首先经过中文分词处理。

目前，针对中文分词的深入研究主要集中在科研院校，清华大学、北京大学、中科院、复旦大学、东北大学、北京航空航天大学等都有其自身的研究小组。本文简要介绍几个影响较大且具代表性的中文分词系统，如下：

北京航空航天大学在 1983 年设计实现了我国第一个中文分词系统——CDWS 分词系统，该分词系统采用基于字符串的最大匹配算法，同时借助词尾字构词纠错技术进行歧义字段处理，在人工干预的前提下，分词精度达到 1/625，基本满足了应用需要。CDWS 分词系统是对中文自动分词的首次实践尝试，验证

了中文文本自动分词的可行性，对中文分词技术的研究具有极大的启迪作用。

SEG 和 SEGTAG 分词系统由清华大学开发，首次提出全切分的概念，采用带回溯的正向、反向、双向最大匹配法和全切分法。该分词系统的工作原理是穷举输入字串的所有可能性，取最佳的字串序列作为分词结果。经过封闭试验证明，该系统在切分精度上确实有很大提高，但由于算法复杂度的增加，切分速度明显减慢。

复旦大学研发的分词系统由四个模块构成[3]，分别是预处理、歧义识别、歧义字段处理和未登录词识别。该系统的设计思想是先分析中文分词技术中的主要难题，然后根据问题逐个改进。通过四个模块的细节改进，系统的切分速度和精度都有一定的提高，尤其是对未登录词的识别效果尤其明显。

中科院开发的 ICTCIAS 分词系统是一套获得广泛好评的分词系统，系统由中文自动分词、词性标注和未登录词识别三个模块构成。该分词系统是通过层叠型马尔科夫模型进行分词，得出概率最大的切分结果，然后利用角色标注方法识别未登录词，计算其概率并将未登录词加入到切分词图中，之后视它为普通词处理，最终进行动态规划优先选出最大概率切分标准结果。

综上所述，我国关于中文分词技术的研究已经步入相对成熟的阶段，处理技术的不断发展和分词算法的不断优化，使中文分词处理技术在切分精度和切分速度上都有了很大的提高。但科研院校的研究成果不能很快的产品化，而技术的研究最终是服务于应用，因此，中文分词技术还有很长的路要走。

1.3 搜索引擎概述

1.3.1 搜索引擎简介

搜索引擎[4]一般是指从网络等信息源中以某种规则收集所需信息，经过信息加工（如进行网页去重、提取信息、建立索引、生成关键词标引、生成自动文摘、对信息进行分类、聚类类似网页等），把使用者需要的信息根据一定的规则进行内容排序，以超链等形式呈现给使用者的系统。

看似简单的搜索引擎的背后涉及包括数据结构、索引、算法、知识表示、自然语言处理、信息检索、人工智能、计算机网络、分布式处理、数据库、数据挖掘等多方面的内容。一般地，搜索引擎主要包括信息采集、信息加工、信息检索与检索结果提供这几个部分。搜索引擎的信息采集系统根据一定的策略在因特网等信息源中收集相关内容；信息加工系统一般是指对网页内容进行标引、创建索引、制定摘要、实现信息分类等；信息检索模块则根据用户的检索提问对检索项

与索引项进行匹配运算以获取对应的检索结果集，有些系统为方便用户使用还提供高级检索功能，支持自然语言提问等；检索结果提供则是在进行必要的相关分析后以超链等形式给出检索结果。搜索引擎结构图，如图 1-2 所示。

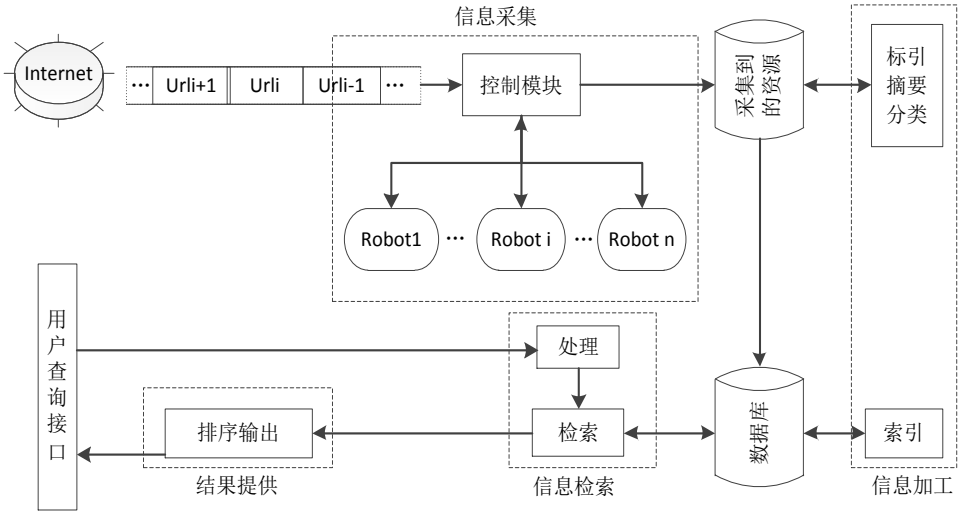

图 1-2 搜索引擎结构图

Fig.1-2 Structure of search engine

1.3.2 搜索引擎发展历史及现状

搜索引擎产生于 1990 年，前身为加拿大蒙特利尔大学学生开发的 Archie 系统。该系统通过定情搜索并分析 FTP 系统中存在的文件名，提供查找散布在各个分散的 FTP 主机中文件的服务。三年之后，美国内华达大学开发出功能更为全面的提供建设网页的 Veronica 系统。随着因特网规模的迅速扩大，研究人员引入网页自动采集器，即网络蜘蛛 Spider 来提高信息采集的性能。世界上首个 Spider 是由麻省理工学院的 Matthew Gray 开发的搜索产品 World Wide Web Wander，该产品最初用于统计因特网规模。1994 年初，美国华盛顿大学的学生 Brian Pinkerton 开发出了因特网上实现全文搜索的搜索引擎 WebCrawler。1994 年 7 月，Michael Mauldin 开发出基于 Spider 技术的搜索引擎 Lycos。同年，美国斯坦福大学的两名博士生 David Filo 和美籍华人 Jerry Yang 共同开发出目录索引式搜索引擎 Yahoo。从此，搜索引擎进入了快速发展的新时期。近年来，随着 Google、百度等搜索引擎使用率的逐渐提升，搜索引擎已成为网络用户检索信息的重要工具。

根据搜索引擎使用的技术不同[5]，可将搜索引擎的发展划分为分类目录和文本检索时代、链接分析时代、用户中心时代。分类目录和文本检索时代的代表为早期的 Yahoo、Alta Vista 和 Infoseek 等，这类搜索引擎一般索引量都较少，分类

目录搜索引擎采用人工对网页进行分类并建立索引，文本检索搜索引擎则采用较为成熟的信息检索模型。链接分析时代的搜索引擎以 Google、DirectHit 等为代表，也被称为第二代搜索引擎。在网页重要性评价方面，Google 认为一个网页的重要性取决于其被其他网页所链接的数量、等级等，率先使用 PageRank 分析技术；而 DirectHit 则认为被多数人访问的网站是较为重要的。这种由大众来确认网页重要性的方法有其合理性，但亦有不足：许多网站针对链接分析算法开发了作弊方案，导致搜索质量变差。当前搜索引擎已发展到用户中心时代，技术日趋成熟。现今的搜索引擎不仅索引规模大，而且更多地结合了自然语言理解、个性化等智能化技术以获取使用者的实际需求[6]。

互联网信息量在过去 20 多年获得了爆炸式的增长，信息过载的问题比较严重，搜索引擎是目前解决信息过载的有效方式，作为互联网网站和应用的入口，其地位在逐步加强。虽然国内外出现了一种现象，即一些大型的新互联网公司屏蔽某类搜索引擎爬虫，比如脸书对谷歌爬虫的屏蔽，国内电子商务网站淘宝对百度的屏蔽。类似情况只能说明这些商业公司基于自身发展的计划和策略，或者是垂直类搜索引擎和通用类搜索引擎的搜索引擎类别竞争，并不是搜索产品与非搜索产品的竞争。对于各自都拥有巨量用户的脸书和淘宝，也必须依赖搜索引擎来采集用户产生的大数据集，并以此拓展提升自身的竞争力。

搜索引擎技术发展迅速，一些优秀搜索引擎系统也纷纷涌现，并已成为人们生活中必不可少的互联网工具。目前最有代表性的搜索引擎首推 Google 和百度。

Google：在 1998 年 10 月以前，Google 只是斯坦福大学的一个小项目 BacnRub。1995 年斯坦福博士生 Larry Page 开始学习搜索引擎设计，并于 1997 年 9 月 15 日注册域名 google.com。1997 年底，在 Sergey Brin 和 Scott Hassan、Alan Steremberg 的共同参与下，BachRub 开始提供 Demo。1998 年 10 月 Google 正式推出，次年 2 月完成了从 Alpha 版到 Beta 版的蜕变。2000 年 Google 数据库升级，又借被 Yahoo 选作搜索引擎的东风，一飞冲天迅速成为业界的领头羊。Google 在网页排序、动态摘要、网页快照、每日更新、多文档格式支持、地图股票词典人物搜索等方面的创举，令全世界为之一惊。

百度：进入 2000 年后，互联网高速发展，网上信息量以指数级速率增长，这再次为搜索引擎的发展提供了良好的背景，搜索引擎成为继门户网站后的又一重心。但是同时，在中文搜索领域，全世界还没有任何一个搜索引擎有着令人满意的效果，这其中的原因是多方面的：一方面，国内搜索引擎技术起步较晚，水平与国外还有较大差距；另一方面，中文的复杂性和中文分词的不确定性制约了中文搜索引擎的发展。

2000 年 1 月，前 Infoseek 资深工程师李彦宏与好友加州伯克利分校博士徐勇

从美国硅谷回国，在北京中关村创立百度（Baidu）公司，定位于打造中国人自己的搜索引擎。2000 年 5 月，百度首次为门户网站“硅谷动力”提供搜索技术服务，之后迅速占领中国搜索引擎的市场，成为国内最主要的搜索技术供应商。2001 年 8 月，百度从后台服务转向独立提供搜索服务，并在中国首创竞价排名的商业模式。2005 年百度在纳达斯克上市，中国搜索引擎市场由此进入一个崭新阶段。据统计，百度目前收录中文网页超过 2 亿，是最大的中文数据库。百度搜索引擎的其他特色包括网页快照、网页预览、相关搜索词、错别字纠正提示、新闻搜索等。

1.4 本文研究内容

中文分词作为计算机文本处理领域的重点，是广受关注的研究热点。由于中文信息的特殊性，极高的准确率和极快的分词速度并不容易同时实现。中文分词模块的好坏直接影响搜索引擎的性能，如何做好分词速度和分词精度的平衡对于搜索引擎至关重要。本文主要研究如何改进分词算法以降低中文分词时间复杂度，提高切词准确率，并根据优化算法和词典设计更高效的中文分词模块替换 Lucene 中自带的中文分析模块，提高以 Lucene 为基础搭建的搜索引擎系统性能。本文主要研究内容如下：

1.了解中文分词技术、搜索引擎的发展现状。系统学习搜索引擎的架构，对当前成熟的中文分词技术进行学习和研究，分析最大匹配算法存在的问题，并针对每个问题提出行之有效的解决途径。

2.根据最大匹配算法的不足设计更合适的算法流程，并针对提出的改进算法需求优化传统的双字哈希词典机制，提出基于双字哈希词长分组词典机制的正向最大匹配改进算法。该算法对于每一次匹配都能根据具体情况合理的选择匹配初始位置和匹配长度，优先选择待匹配文本全域内的最长词，减少了不必要的匹配消耗，并一定程度上解决最大匹配算法造成的分词歧义问题。

3.对于改进后的算法可能存在的切分歧义，根据算法的匹配过程，结合最大匹配算法+回退一字法算法思想，构建歧义处理模块。经过歧义处理，有效消除部分交集型歧义，较大提高分词准确率。

4.通过对搜索引擎知识和 Lucene 开发包的研究，以 Lucene 为基础搭建基本的搜索引擎系统。根据基于双字哈希词长分组词典机制的正向最大匹配改进算法重新设计 Lucene 的中文分析模块，提高该模块的分词能力，进而提升基于 Lucene 的搜索引擎系统的应用性能。

5.对算法改进进行实验评价。首先采用不同词典机制对相同语料进行分词，验证本文选取的双字哈希词典性能；然后分别使用本文改进算法及正向最大匹配

算法对相同语料进行分词，比较结果。实验结果显示，本文提出算法在分词速度及分词精度上都优于正向最大匹配算法，达到了改进目的。

1.5 论文结构安排

本研究课题的论文结构分为五章，各章的主要内容安排如下：第一章系统介绍了本课题的研究背景和研究意义，介绍了中文分词技术和搜索引擎的发展现状。第二章重点学习中文分词的核心知识基础，包括对词典结构和分词算法的学习，并介绍中文分词技术的难题。其中最大匹配算法和双字哈希词典机制是本文的研究改进重点。第三章重点分析最大匹配算法存在的不足并提出相应优化方法，结合双字哈希词典机制设计基于双字哈希词长分组词典机制的正向最大匹配改进算法，同时针对改进算法设计有效的歧义处理机制。第四章使用搜索引擎开发包 Lucene 搭建基础搜索引擎系统，结合基于双字哈希词长分组词典机制的正向最大匹配改进算法改进 Lucene 分析器 Anlyzer（主要改进分词器 Tokenizer）。第五章为对文本工作的实验验证。

2 中文分词概述

字是汉语文本的最小组成元素，词是中文组成中具有完整意义的最小单位，但中文文本是以一句话加一句话的形式罗列而成，在表述中文信息时，词与词之间并不设置有效的分隔符。要想正确的理解中文文本信息，首先需要将汉字序列分割成正确的词语序列。比如句子“The early bird catches the worm”与句子“早起的鸟儿有虫吃”具有相同的含义，在前句中通过空格可以简单的将单词隔开，但后一句就没有自然的分隔符。中文分词就是通过一定的算法和规则，使计算机具有将汉字序列正确分隔为词语组合的能力，这种能力是各类应用中文处理的基础。

2.1 中文分词词典机制

要提高中文分词的执行效率，就必须降低匹配算法的时间复杂度，而一个匹配算法的时间复杂度，在很大程度上取决于匹配数据的存储结构[7]。因此，一个好的分词方法，它所使用的词典也必须具有好的存储结构。文献[8]使用的是首字哈希词条按词长分组的词典机制；文献[9]提出了分层逐字二分词典机制；文献[10]提出一种基于全哈希的整词二分词典机制；文献[11]提出了一种具有三级索引的新词典结构。以上改进的词典结构都在一定程度上加快了分词速度。目前，常用的中文分词词典机制主要基于整词二分、Trie 索引树、逐字二分和双字哈希[12]。

2.1.1 基于整词二分词典机制

基于整词二分词典机制是一种被广泛应用的分词词典机制[13]，这种词典机制通常由三部分构成，分别是首字哈希表、词索引表以及词典正文。其中，首字哈希表和词索引表为具有指针表的索引，首字哈希表指针指向的是词索引表，词索引表指针指向的是词典中的词条，该词典机制的正文部分是词条内容的线性表。

该词典机制的取词思想如下：首先通过首字哈希散列函数和词索引表找到目标词条在词典正文中的大体位置，接着通过整词二分定位词条在词典正文中的准确位置。结构如下：

（1）首字哈希表：每个首字哈希表单元都包括入口项个数和入口项指针两部分，每部分分别占用 4 个字节。入口项个数记录了首字对应的词的数量；入口项指针指向词索引表中第一入口项的具体位置。

（2）词索引表：由于读取词典时必须保证能够随机访问词条，所以词索引表是非常必要的。而且词条长度多种多样并不固定，因此必须使用不定长存储方式。词典正文指针是词索引表的唯一信息，占用 4 个字节，指向词语在词典中的具体位置。

（3）词典正文：以词为单位的有序表。

基于整词二分词典机制的优点是词条存储方便易于实现，词典占用的内存空间较小，词典的建立和后期的维护都比较简单。缺点是采用全词匹配方式进行查询，查询效率很低。

该词典机制结构，如图 2-1 所示。

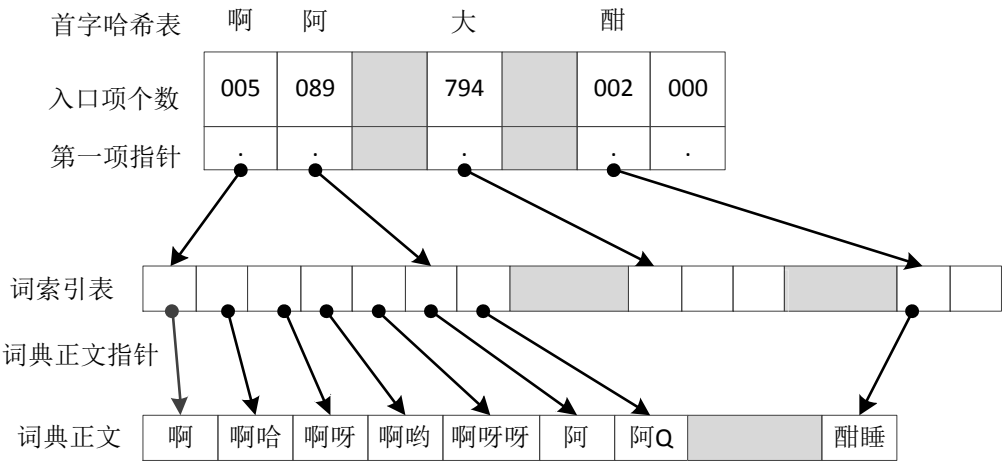

图 2-1 整词二分词典机制结构

Fig.2-1 Binary dictionary mechanism structure

以在词典中查找“青岛市政府”为例说明：首先根据哈希函数在词典中找到“青”，再根据指针找到词索引表中首字为“青”对应的词典正文指针，然后在该范围内二分查找“青岛市政府”。

2.1.2 基于 Trie 索引树词典机制

Trie 索引树是键树，以多重链表表示。在英文中，每个字母即是英文 Trie 索引树的关键字，树的节点包含数量相同的指针。但是对中文来说，汉字的数量远不止英文字母的 26 个，如果对约 7000 个汉字使用和英文同样的办法构造中文词典，势必会浪费大量指针，所以汉语 Trie 索引树的指针个数不统一。

基于 TRIE 索引树[14]的词典机制包括首字哈希表和 Trie 索引树节点，具体描述如下：

（1）首字哈希表：是 Trie 索引树的根节点，结构和整词二分词典的首字哈希表相同。

（2）Trie 索引树节点：根据关键字排序产生的数组，包括关键字、子树大小及子树指针三个基本单元。前两个单元分别占 2 个字节，其中子树大小的定义为以从根节点到当前节点组成字串为前缀的词条数目。子树指针占 4 个字节，根据子树大小是否为零分别指向叶子和子树。

该词典机制的优点在于扫描时不需提前知道待匹配字符串长度，只需要沿着树节点逐字匹配。缺点在于树结构构造复杂，维护也比较麻烦，并且一条树枝只能代表单一词语，空间浪费严重。Trie 索引树结构，如图 2-2 所示。

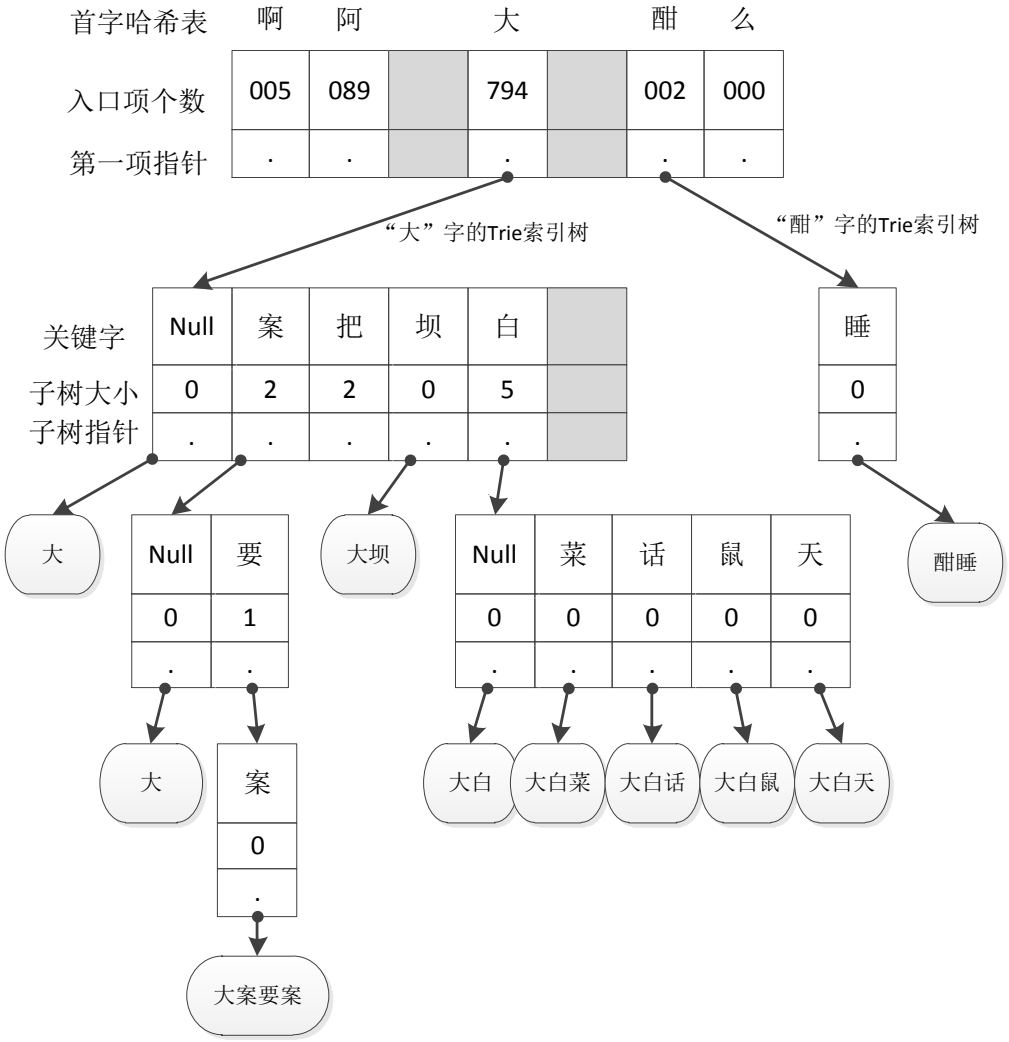

图 2-2　Trie 索引树词典机制结构

Fig.2-2 Trie indexing tree dictionary mechanism structure

2.1.3 基于逐字二分词典机制

逐字二分的词典结构与整词二分完全一样，但查询过程不同。逐字二分采取

了 Trie 索引树的优点“逐字匹配”进行查询，在一定程度上提高了匹配效率[12]。逐字二分词典机制本质上和整词二分的词典结构并无差异，故而其效率的提高也有较大的局限性。

基于逐字二分的词典机制是对整词二分和 Trie 索引树两种词典机制的改进，其结构如图 2-3 所示。

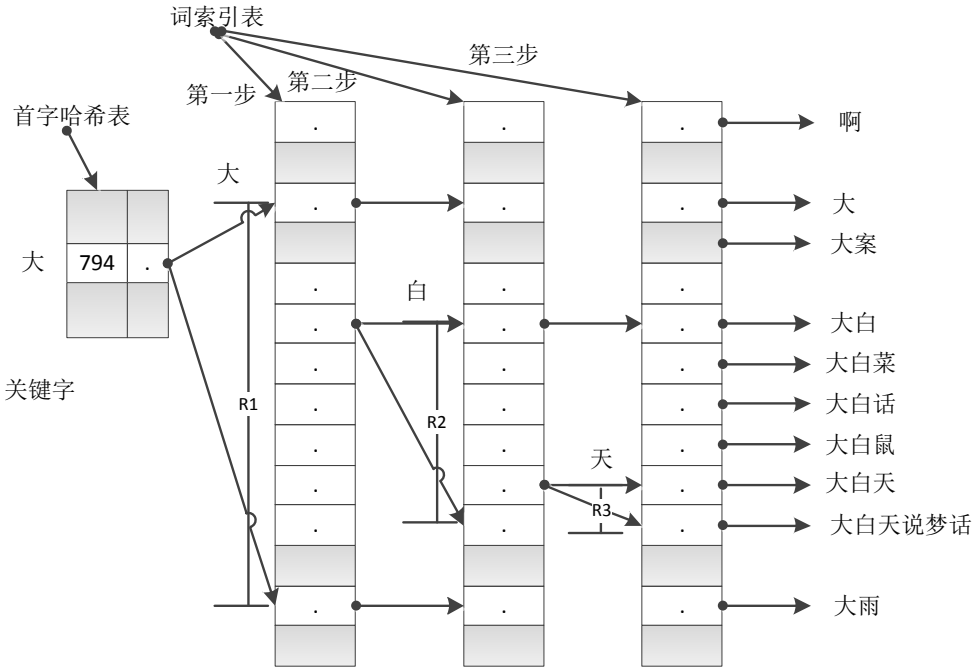

图 2-3 逐字二分词典机制结构

Fig.2-3 Binary word by word dictionary mechanism structure

2.1.4 基于双字哈希的词典机制

文献[15]对一个通用词典信息进行的统计信息如下：词典中共有词条数量为 42425，其中单字词条数量为 1650，二字词条数量为 30770，单字词条和二字词条的数量占了词条总数的 76.42%，二字以上词数量仅占词条总数的 23.58%并且词频占比也较低。

根据中文词典的词长及词频分布规律，文献[15]提出了基于双字哈希的分词词典机制。该词典整合了整词二分分词词典和 Trie 索引树分词词典的优点，采用多次哈希循环，分别对多字词的首字和次字建立哈希索引表，构成深度为 2 的 Trie 子树，而该词条的剩余字则根据一定的排序规则组成词典正文。基于双字哈希的词典结构既提高了查找速度，又有较好的词典空间利用率，是一种高效简洁的词典形式。

双字哈希的词典机制由首字哈希索引、次字哈希索引、词典正文三部分组成。

首字哈希索引和次字哈希索引分别包括三项内容，分别是关键字哈希表、是否为词标志、指针。其中首字哈希索引中指针指向次字哈希索引，次字哈希索引中指针指向词典正文地址。词典正文包括以首字次字开始的余字部分及是否成词标志，其中余字部分采用有序数组存储。双字哈希的词典机制很适合正向最大匹配算法，该词典与首字哈希词典机制相比，极大的提高了匹配速度。目前基于双字哈希的词典机制已成为大多数基于字符串匹配分词算法的首选词典机制[16]。

基于双字哈希的词典结构，如图 2-4 所示。

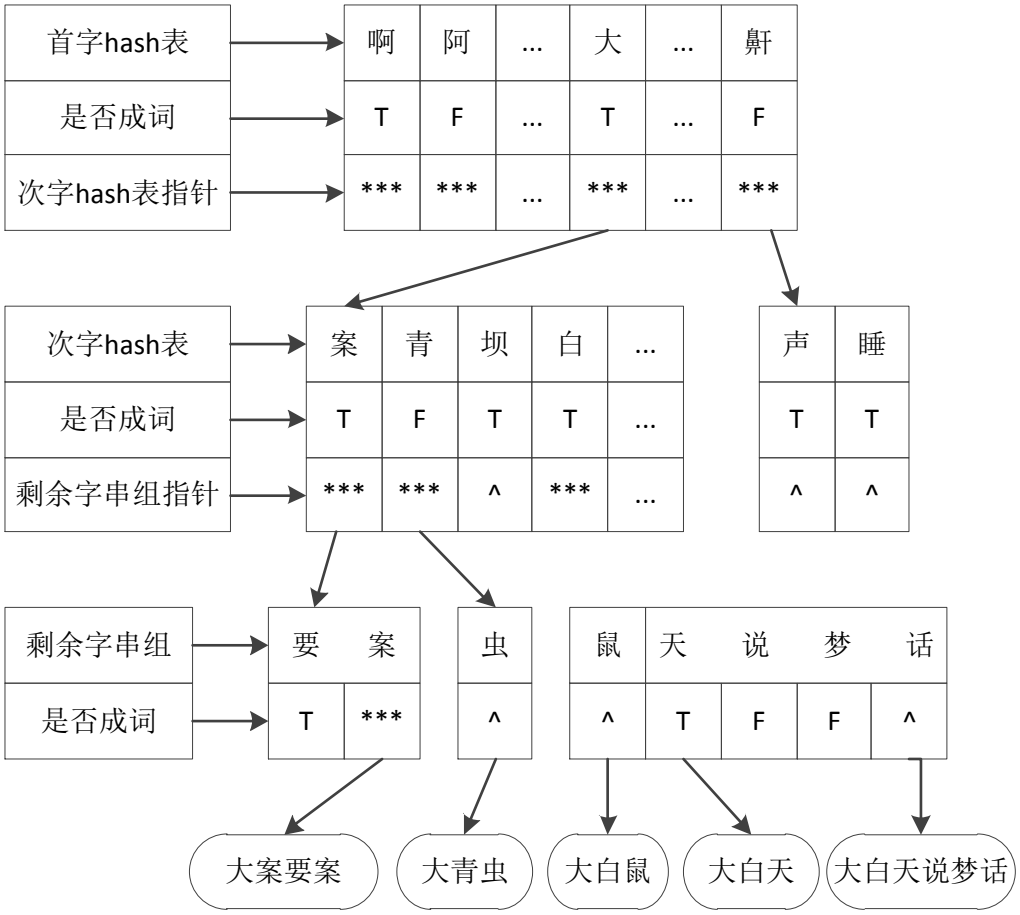

图 2-4 基于双字哈希的词典结构

Fig.2-4 Dictionary based on double word hash structure

2.2 中文分词主要方法

中文分词的算法多种多样，常见的像正向最大匹配、逆向最大匹配、双向最大匹配、最佳匹配法、最少分词法、设立切分法、有穷多层次列举法、逐词遍历法、二次扫描法、词网格算法、邻接约束法、邻接知识约束法和专家系统法等。现有的分词算法可分为 3 大类：基于字符串匹配的分词方法、基于理解的分词方法和基于统计的分词方法[17]。

三类分词方法的对比[18]，如表 2-1 所示。

表 2-1 常用分词方法对比

Tab.2-1 Commonly used word segmentation

分词方式	基于字符串匹配的方式	基于统计的方式	基于理解的方式
分词速度	快	一般	慢
分词准确性	一般	较准	准
算法复杂度	容易	一般	难
实现难度	容易	一般	难
技术成熟度	成熟	成熟	不成熟
歧义处理能力	弱	较强	强
新词识别能力	弱	较强	强
是否需要词典	是	否	否
是否需要语料库	否	是	否
是否需要规则库	否	否	是

2.2.1 基于字符串匹配的分词算法

基于字符串匹配的分词算法方法又叫做机械分词方法，是目前使用最广泛的中文分词算法。这种方法基于相对完备的词典，按照一定的策略将待匹配字符串与该“充分大的”机器词典中的词条进行匹配，若在词典中找到某个字符串，则匹配成功（识别出一个词），将该字符串作为词从待匹配文本中切分出来。机械分词算法是通过将字符串与词典中词比对的方式进行分词，对词典的依赖性非常高，词典的好坏直接影响算法切分结果。而且由于词典总是有限的，该算法对于新词识别的能力比较弱。机械分词算法的优点是算法简单易于实现且分词速度较快，不需要建立规则库或统计语料库。基于字符串匹配的分词方法根据从左向右扫描还是从右向左扫描的不同划分为正向匹配和逆向匹配；根据选取多少相邻字符进行优先匹配的情况，可划分为最大（最长）匹配和最小（最短）匹配；根据分词过程是否基于词性标注，又可以分为单纯分词方法和分词与标注相结合的一体化方法。常用的几种机械分词方法如下所示。

（1）正向最大匹配法（Forward Maximum Matching method，FMM）：假设一个已知词典中，最长的词条有 m 个汉字，则截取待匹配文本中前 m 个字作为匹配字段与分词词典进行匹配。若词典中存在该词则匹配成功，将匹配字段切分出去，然后从切分出去的字符串的下一字开始在待匹配文本中再取前 m 个字作为匹配字段重新与词典匹配。若匹配不成功，则去掉待匹配字段的最后一个字，用

剩下的 m-1 个字组成的字符串与分词词典进行匹配，直到匹配成功为止。

（2）逆向最大匹配法（Backward Maximum Matching method，BMM）：逆向最大匹配法也是基于字符串的一种分词方法，算法和正向最大匹配法相似，不同之处在于逆向最大匹配算法是从待匹配文本的最后一个字开始，从右向左进行匹配，当匹配不成功时去掉待匹配字段最前面一个字，直到匹配成功为止。根据统计结果，单纯使用正向最大匹配算法的错误率为 1/169，单纯使用逆向最大匹配算法的错误率为 1/245，逆向匹配算法的精确度高于正向匹配算法。在实际需求中，对中文分词系统的切分精度要求较高，在使用最大匹配算法时需对算法进行改进或结合其他方法提高切分精度。

（3）双向匹配法（Bi-direction Maximum Matching method，BM）[19]：将 FMM 法和 BMM 法结合起来的算法称为双向匹配法，该算法的最终结果是基于两种匹配方法的结果来确定的，可以方便的识别交集型歧义字段。

（4）逆向最小匹配算法：该算法的匹配方向与逆向最大匹配算法相同，也是从右向左进行匹配。不同点是逆向最小匹配算法规则为加字匹配，即先取待匹配文本最末两字与词典中的词进行匹配，若匹配成功则切出该词，否则取最末三字在词典中进行匹配，直到匹配成功。若匹配至最右端字仍未匹配成功，则切分文本最末字为词。

（5）最少匹配法：最少匹配算法的基本思想与在有向图中寻找最短路径的思想很像，该算法要求分词结果中的词数量达到最少。算法第一步要对待处理的文本进行分段，接着逐段计算最短路径，并得到多个分词结构，然后进行统计排歧，确定最后的分词结果。

2.2.2 基于统计的分词算法

从组合形式上分析，词是一到多字的稳定结合。在中文文本，相邻的字组合出现的次数越多，该相邻字组成词的可能性就越大。相邻若干字成词的可信度可以根据这些相邻字共同出现的频率和几率来计算，字与字的互现信息可以依据对文本中相邻且共同出现的若干字的组合频率加以统计得到。可以根据统计和计算汉字 A、B 的相邻共现概率来定义 A、B 两字的互现信息，因此，字与字之间组合关系的紧密度能够由互现信息表现。当若干个相邻字的紧密度高于规定的数值时，就可以认定此汉字序列组合可以组成一个语。这种分词算法不依赖于分词词典，仅需要根据对所选语料中的相邻字组合频度统计结果进行切分，所以也被称作做无词典分词法或统计取词方法。基于贝叶斯公式的模型是统计分词系统中最初常用的统计语言模型，后又出现 N 元统计模型、最大熵模型和隐马尔科夫模型

等多种模型。

基于统计的分词算法同样存在局限性，因为该方法基于数学统计模型，会频繁切出一些虽然共现频度高，但并非词的常用字符组合，譬如“这就”、“这一”、“你的”“许多的”等。同时统计分词时空开销都比较严重，对于常用词的识别敏感度也不理想。因此，在实际使用中一般会以将统计分词方法和基于字符串匹配的方法结合的方式进行分词，这样既可以发挥统计分词方法善于消除歧义识别新词的优点，又能发挥机械分词方法切分速度快的优点。

2.2.3 基于理解的分词算法

基于理解的分词方法指通过计算机模拟和实现人对汉字串的理解，实现中文分词的效果。其基本思想就是在分词的同时进行句法、语义分析，利用句法信息和语义信息来处理歧义现象。它通常包括 3 个部分：分词子系统、句法语义子系统、总控部分。在总控部分的协调下，分词子系统可以通过获得有关词、句子等的句法和语义信息来对分词歧义进行判断，即它模拟了人对句子的理解过程。这种分词方法需要使用大量的语言知识和信息。由于汉语语言知识的笼统、复杂性，难以将各种语言信息组织成机器可直接读取的形式，因此目前基于理解的分词系统还处在试验阶段。

2.3 中文分词难点

中华文化博大精深，汉语的规则和形式复杂多变，让计算机去模仿人进行汉语语言分析非常困难。虽然中文分词技术已被研究多年，仍旧没有一种中文分词方式能完全正确切分，任何算法都会或多或少产生切分错误。对于中文分词，目前有三个难点尚未完全解决，一是分词歧义，二是新词识别，三是分词规范问题。

2.3.1 分词歧义

有个经典笑话：护士看到病人在喝酒，就善意叮嘱说：“小心肝！”不料病人暧昧回答：“小宝贝！”在这里两个人对“小心肝”存在不同的理解，护士的原意是“小心/肝”，而病人的解读是“小心肝”。

歧义句指的是存在多种理解方式的句子。分词歧义是指对于同样的汉字串可以切分成两种或两种以上的的结果，分别将每个结果集的元素都能够组合结果相同，但每种结果集表达的句意却不相同。语言是人与人交流最主要的媒介，对于同样的语言，保证表达者要表达的内容和接收者理解的内容统一是人们关心的问

题。在中文分词系统中，机器对信息的正确切分是保证信息有效识别的关键，也是历年来中文分词的在研究重点。目前任何一种分词方法都不能避免分词歧义问题，识别分词歧义是中文分词技术的常见问题，根据歧义的形成原因分类，歧义可分为交集型歧义、组合型歧义和真歧义三种[20]。

（1）交集型歧义

交集型歧义定义：在字段 JKL 中，JK∈C 并且 KL∈C，J、K、L 为字串，C 为词表，则 JKL 称为交集型歧义字段[21]。

交集型歧义的链长是指在交集型歧义字段中含有交集型歧义的个数，交集型歧义字段的交集字段长度是指交集歧义字段中汉字的数量，J，K，L 的长度是大于等于 1 的。

例如“门把手”一词，因为“门把”和“把手”都是词，那么这个短语就可以分成“门把/手”和“门/把手”，该字段就是链长为 1、交集字段长度为 1 的交集型歧义。交集型歧义非常常见，例如“化妆和服装”可以分成“化妆/和/服装”或者“化妆/和服/装”。交集型歧义是由于计算机分词产生的，计算机不具备像人一样利用知识去智能判断的能力，很难知道到底哪个切分方案正确。文献[22]研究结果显示，交集型歧义的占比超过歧义总量的 85%，如果能利用算法合理的避免交集型歧义，分词准确率一定会大幅度提升。文献[23]列出了对从因特网上任意提取的 510 万字《人民日报》标本分词结果统计，其中共切分出 7.8 万个交集型歧义字段，约占总标本字数的 1.5%。统计结果直观的呈现了该统计中交集型歧义字段的分布规律，如下表 2-2 所示。

表 2-2 交集型歧义字段分布规律

Tab.2-2 Features of crossing ambiguity fields

链长	1	2	3	4	5	6	7	8	总计
交集型歧义字段数	47402	28790	1217	609	28	18	3	2	78069
比重（%）	60.71	36.88	1.56	0.79	0.04	0.02	0.00	0.00	100
歧义字段出现次数	12689	10136	745	334	29	3	2	2	23940
比重（%）	53.00	42.33	3.11	1.40	0.12	0.02	0.01	0.01	100

（2）组合型歧义

组合型歧义（也叫多义型歧义）定义：在字段 XY 中，XY∈C，X∈C，Y∈C，X、Y 为字串，C 为词表，则称 XY 是组合型歧义字段[24]。

相对于交集型歧义，组合型歧义更依赖于对句子的整体理解，这种歧义必需根据整个句子环境来具体判断。例如，在句子“这个门把手坏了”中，“把手”是个词，但在句子“请把手拿开”中，“把手”就不是一个词；在句子“北海舰队有多名少将”中，“少将”是个词，但在句子“词典中词语的多少将影响分词结果”中，“少将”就不能切分成一个词。中文文本中组合型歧义字段主要为 XY 双字型，也存在少量 2 字以上的组合型歧义字段。组合型歧义也是由于计算机分词产生。

（3）真歧义

除了交集型歧义和组合歧义，还有一种计算机非常难识别的歧义，这种歧义被称作真歧义，是由于汉语语言的二义性造成的。即便由人去切分，这种歧义类型的句子依然具有有两种以上的切分结果。例如对于句子“签名网球拍卖完了”，可以理解为“签名/网球拍/卖/完/了”，也能理解为“签名/网球/拍卖/完/了”，倘若不根据上下文其他信息，没有人能够确定“拍卖”在这里能不能切分成词。解决这种歧义只能通过上下文的信息进行判断。

2.3.2 新词识别

新词也叫未登录词[25]，通常指不存在于分词词典中，又必须按词切分的汉字串。未登录词种类多种多样，涵盖的范围也很多，很多未登录词恰恰被经常使用，例如专业名词、人名、地址名、商标名、机构名、省略用语等。未登录词是影响分词精度的重要因素[26]，未登录词的识别是中文分词技术研究的重点和热点问题之一。

最典型的未登录词是人名。中文人名的形式比较多，组成结构也相对复杂，而且不像英文名可以通过大小写字母加以识别。在“公司元老刘总在开会”中，“刘总”代表人名可以切分为词；在“老刘总是乐于助人”中“刘总”便不能切分成词。目前解决人名带来的切分问题有两种方式[27]，一是建立人名词典库，但词典不能无限大，也不可能包含所有姓名。二是把姓氏和常见的名集合起来建立人名库并辅以人名规则，在句子中使用特定的评价机制进行判定。

穷举法可以部分解决未登录词带来的切分问题，但还远远不够。只有通过恰当的数学模型建立统计机制，并辅以合适的识别规则，才是对这个问题更好的解决办法[28]。

2.3.3 分词规范

目前的分词规范没有能力对词确定定义方式和界定标准，而中文分词结果的最小单位是词，这一定程度上造成了分词结果的不确定性。导致该问题的原因主要集中在两方面，一是不同的人对中文词语的认识并不一致，二是不同领域的分词系统对“分词单位”的划定也不统一。一方面，虽然业界对分词规范进行了大量的研究工作，但对于词仍旧没有统一的定义和划分标准，人与人对词语的认识并不一样。例如对于“归纳总结”是整体划分为一个词还是划分为“归纳/总结”，不同的人可能会得出不同的答案，并没有划分的对错。另一方面，对于不同需求的分词系统，通常会使用不同的分词标准。

2.3.4 歧义采集方法

目前主流的歧义采集方法有以下几种：双向扫描算法、逐词扫描算法、最长词次长词发现法、正向最大匹配+回退一字法、有向图切分法等。

（1）双向扫描算法

双向扫描算法[29]是指分别使用正向匹配算法和逆向匹配算法对同一字段进行切分，并将两种分词结果做比较。若切分结果相同则无歧义产生，切分结果不同则标记为歧义字段。该算法可以识别大多数的交集型歧义，并不善于识别组合型歧义字段。

以切分“当中间的门打开时”为例，正向最大匹配算法切分结果为“当中/间/的/门/打开/时”，逆向最大匹配算法的切分结果为“当/中间/的/门/打开/时”。显然两种算法匹配结果不同，“当中间”三个字为链长为 1 的交集型歧义字段。

以切分“校长将来宿舍走访”为例，正向最大匹配算法和逆向最大匹配算法的切分结果均为“校长/将来/宿舍/走访”，与正确切分结果“校长/将/来/宿舍/走访”不一致。“将来”为组合型歧义，双向扫描算法并未识别。

（2）逐词扫描算法

逐词扫描算法[30]一般是指使用正向最大匹配算法，将最新切分出来词与上次切分出的词做比较，并标识歧义类型。重复比较所有根据算法切分出来的相邻词，直到对目标文本切分结束。

以“地产业发展迅速”为待匹配文本，假设所用词典中最长词词长为 3，逐词扫描算法可描述为：

①先取“地产业”为初始匹配字符串，与词典进行匹配，匹配成功；

②去掉“地产业”最末尾的字，待匹配字符串变为“地产”重新在词典中查

询，成功找到该词，与上次切分出的“地产业”比较，无歧义；

③将指针后移一位，取“产业发”为待匹配字符串在词典中进行查询，匹配失败，取“产业”在词典中查询，匹配成功；

④“产业”与上次切分出的“地产”比较，标记“地产业”为交集型歧义字段，以此类推。

该算法是双向扫描算法的改进算法，可以标记出全部交集型歧义，但同样不能检测全部组合型歧义。该算法实现耗时较长，中文分词系统中若对时间复杂度有要求一般不采用此方法。

（3）最长词次长词发现法

最长词次长词发现法[31]的实现原理是：对于待匹配文本，根据基于字符串的匹配方式首先得到词典中以待匹配文本首字为首字的最长词 W_i 和次长词 W_i'，然后从待匹配文本中 W_i' 后的首个字开始，在词典中查询以该字为首字的最长词 W_{i+1} 和次长词 W_{i+1}'。以此类推得到整个待匹配文本的切分信息，最后将最长词 W_{i+1} 的字数 n_{i+1} 和 W_i 与 W_i' 的字数之差 $n_i - n_i'$ 作比较，则

① 果 $n_{i+1}-（n_i - n_i'）< 0$，则最长词条 W_i 无歧义；

② 果 $n_{i+1}-（n_i - n_i'）= 0$，则最长词条 W_i 为组合型歧义；

③ 果 $n_{i+1}-（n_i - n_i'）> 0$，则最长词条 W_i 为交集型歧义。

最长词次长词发现法可以识别部分组合型歧义，也可识别部分交集型歧义，但不能检测到所有歧义。最长词次长词算法示意图，如图 2-5。

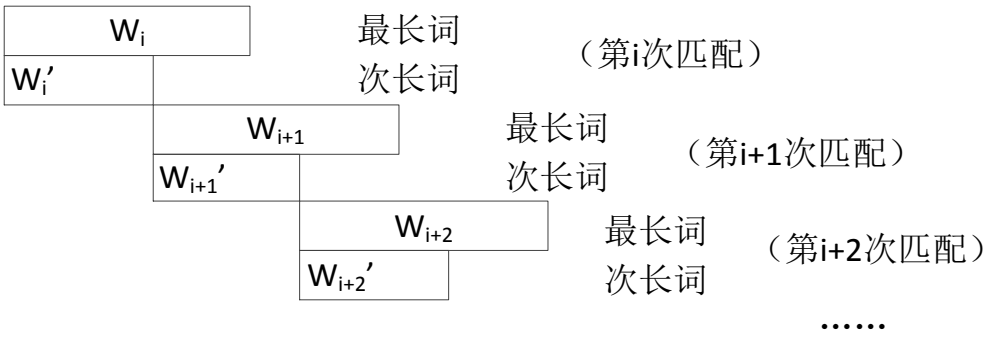

图 2-5 最长词次长词算法

Fig.2-5 The longest term and long term algorithm

（4）正向最大匹配+回退一字法[32]

该歧义检测算法的基本思想如下：当使用在词典中查找到一个词时，将该词回退一个字，组成新的待匹配字符串重新在词典中进行匹配，若匹配成功，则标记为歧义字段；若匹配不成功，则表明不存在歧义，把之前匹配成功的词切分出来，剩余字符串继续进行匹配。以此类推，直到待匹配文本匹配结束。算法流程

图如图所示。

该算法思想简单，实现也并不困难，缺点是只能找出链长为 1 的交集型歧义字段，同时并不善于识别组合型歧义。正向最大匹配+回退一字法流程图，如图 2-6 所示。

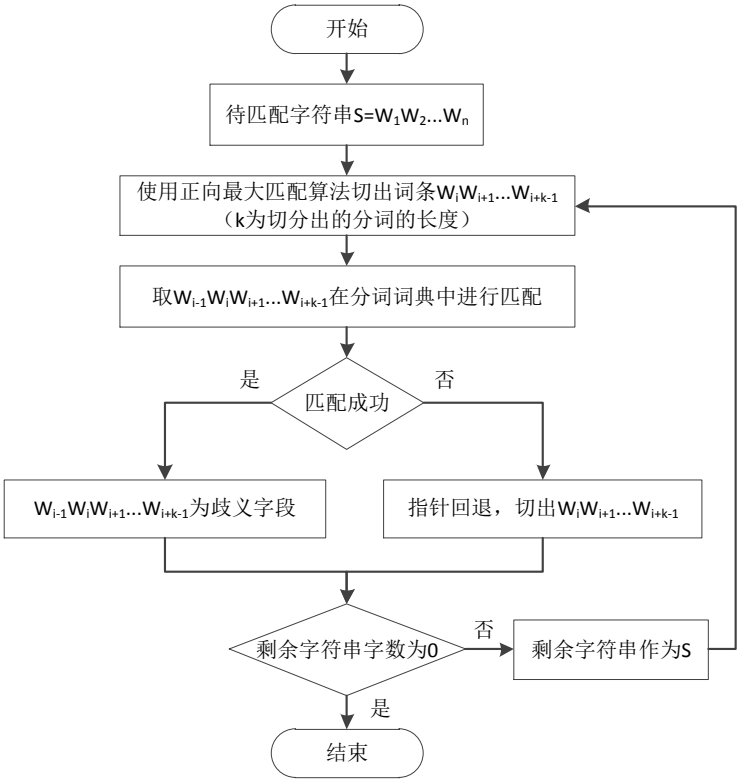

图 2-6 正向最大匹配算法+回退一字法算法流程

Fig.2-6 Positive maximum matching algorithm and back word algorithm

2.4 本章小结

本章分别从中文分词算法、中文分词词典、中文分词难点和常用的分词算法衡量标准等方面对中文分词的基础知识进行了研究，比较了三种主要算法的优缺点，了解了常用的分词词典机制和歧义采集方式，重点研究了正向最大匹配算法，并对最大匹配算法存在的不足加以分析。

3 最大匹配算法改进及词典优化

本章从正向最大匹配算法入手，先是通过对该算法的分析，找到有效改进算法的思路；然后选择对高效率的双字哈希词典机制进行优化，使其适用于改进的算法要求；接着把对正向最大匹配算法的改进和词典的优化融为一体，提出基于双字哈希词长分组词典机制的正向最大匹配改进算法；最后结合最终的改进算法设计合理的分词歧义处理方式。

3.1 最大匹配算法分析及改进

最大匹配算法是使用最为广泛的机械分词算法。本节对最大匹配算法的进行研究，分析该算法的不足，并给出相应改进思路。正向最大匹配算法流程图，如图 3-1 所示。

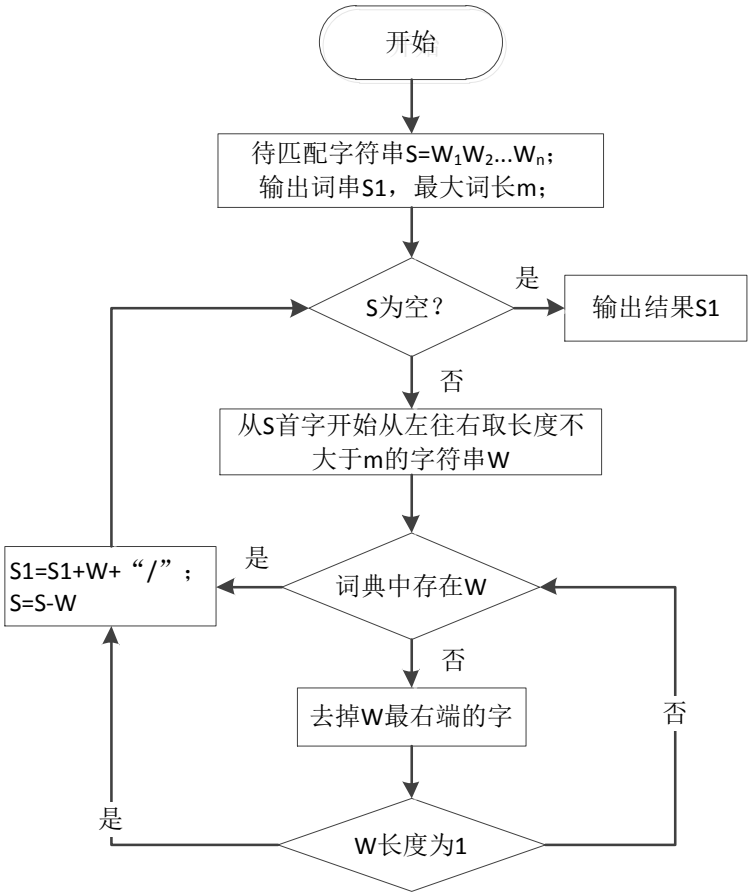

图 3-1 正向最大匹配算法流程图

Fig.3-1 Positive maximum matching algorithm flow chart

3.1.1 最大匹配算法分析

分析最大匹配算法的原理和算法流程，可总结最大匹配算法有以下不足[33]：

（1）初始最大词长设置不合理

在最大匹配算法中，匹配的初始最大词长是所用词典中最长词的词长。而根据文献[34]统计，汉字词中二字词占比最多，三字词、四字词较多，五字以上词所占比例很少，这样在匹配较短词语时，会造成多次无意义的匹配循环，极大的增加了时间和空间开销。

以“学校食堂的饭很便宜”为例句说明，假设所用词典中最长词的长度为 9，则初始最大词长设置为 9。使用最大匹配算法，该句的正确切分结果为“学校/食堂/的/饭/很/便宜”，结果中词长均为 1 或 2，显而易见以 9 作为句子中所有字的匹配初始最大词长开始匹配至切分出最后结果，程序会进行多次毫无意义的循环。

将匹配初始最大词长设置为词典中的最长词词长会严重影响分词速度，若人为规定匹配初始最大词长为某个数，则虽有可能减少部分无意义匹配，但当正确结果词长度大于人为设置的长度时，就会无法分出该词影响分词结果的正确性。如何合理的设置匹配初始最大词长是提高最大匹配算法效率的重点问题之一。

（2）“长词优先”原则覆盖范围不合理

最大匹配算法是基于“长词优先”原则的算法。所谓“长词优先”，是指分词时优先切分以某字作为首字的最长词，这样切分的结果词数也相对少。根据这一规则，算法能实现的最理想的状态是优先切分出待匹配字符串中的最长词，然后切分出剩余字符串的最长词，按词长序列依次递减逐条切分。而在最大匹配算法中，“长词优先”原则存在覆盖范围的问题。在进行最大匹配时，根据匹配方向不同一般是从待切分文本的首字或末字进行最大匹配，忽视了对字符串中间子串的最大匹配，如此一来可能产生错误切分 [35]。

以正向最大匹配算法切分“当中国共产党召开党内大会时”为例，切分结果为“当中/国共/产/党/召开/党内/大会/时”,而正确切分结果是“当/中国共产党/召开/党内/大会/时”，显然正向最大匹配切分的结果不正确。该句出现错误切分正是由于正向最大匹配算法的“最大词长”原则对于每次匹配仅仅是对待匹配字符串的首字而言，并不是针对待匹配文本中的所有字符。

（3）匹配词长重设流程不合理

使用最大匹配算法对以某字为首字的待匹配字符串进行最大词长截取匹配时，如果首次匹配失败，则用初始匹配最大词长减 1 重新赋值匹配词长继续进行匹配，若匹配不成功再减 1，以此循环直至匹配成功。该算法流程可保证切分出的词一定是词典中以待匹配字符串首字为首字的最长词，但逐一递减依次匹配的

方式同样影响了分词速度。在正向最大匹配算法使用的词典中，若将以某字为首字的词按长度排序，排序后长度大小间隔并不总是 1。所以当因匹配失败而对匹配词长进行重新设置时，匹配词长逐一递减的方式并不不合理。

以使用正向最大匹配算法切分“中文与英文不同”为例，切分结果为“中文/与/英文/不同”，假设初始匹配长度为 7，采用匹配长度逐一递减的方式，在以“中”为首字进行匹配时，程序需从 7 递减到 2 进行 5 次无意义匹配才能匹配成功。假设词典中以“中”为首字并无长度为 6、5 的词，匹配词长逐一递减的方式也造成了时间浪费。

3.1.2 最大匹配算法改进思路

根据上述最大匹配算法的不足，总结本文所引用文献经验并加以改进，可将最大匹配算法做如下优化：

（1）根据不同字动态设置匹配初始最大词长

最大匹配算法中，匹配初始最大词长设置为词典中最长词长度或者人为统一设置为某个数都会影响分词效率和效果。针对这一问题，文献[26]提出一种改进方法：取词典中以待匹配字符串任意字为首字的词长最大值与字符串长度作比较，取两者的较小值最为匹配初始最大词长，该方法在没有影响分词结果的前提下成功简化了匹配流程。文献[36]提出针对每一个字，取词典中以其为首字的词的最大词长与待匹配字符串长度进行比较，取较小值作为该字对应的初始匹配最大词长。该方法结合不同首字做了相应处理，进一步优化了算法。

如果能根据待匹配字符串任意字及该字在待匹配字符串中的位置合理的设置匹配初始最大词长，则可进一步减少程序循环次数。根据在词典中以当前待匹配字符串首字为词的词长度 m 与待匹配文本长度 n 做比较，选择不大于 n 的 m 值作为该字匹配初始最大词长，其余字符的匹配初始最大词长根据其在字符串中的位置动态变化。这样根据具体的首字动态设置的匹配初始最大词长对于任意字符串的首次匹配都是合适的，由于设置值对应的词长在词典中真实存在，既不会因取值过大造成时间浪费，又不会因取值偏小造成错误切分。

基于正向最大匹配算法进行改进描述：对于在待匹配字符串中位置为 i（$1 \leqslant i \leqslant n$，$i \in N$，$n \in N$ 且 $n \geqslant 1$）的字，选取以该字为首字的词的词长取值范围限定为不大于 n-i+1。这样选取的词长无论对于该首字本身还是对于整个字符串，都是最为合适的取值，因为最终取值一定能对应到词典中以该字为首字的相关词长度，且该词是可能匹配成功的最长词，同时还保证该取值不大于当前字位置对应的剩余字符串长度。

（2）根据词长序列重设匹配长度

最大匹配算法在匹配失败需对最大词长进行更改时，采用了逐次减一的方式，而词典中词长往往是跳跃的，逐次减一的算法一定程度上影响了分词速度。如果当匹配失败对最大词长 m 进行更改时，选择在词典中以当前待匹配字符串某字为词的下一最大词长度对 m 重新赋值，则可以减少不必要的匹配次数。假设初始待匹配文本长度为 12，词典中以当前待匹配字符串首字为首字的词长序列为 7、5、3，则初始匹配字符串长度为 7，当待匹配字符串与词典中对应的 7 字词无法匹配时，将待匹配字符串长度重新设置为 5，当 5 字词匹配不成功时，直接选择 3 字词继续进行匹配。

以字符串“今天星期天”为例采用正向最大匹配算法说明改进（1）和改进（2）：字符串长度 n=5，对于“今”、“天”、“星”、“期”、“天”五个字来说，其在字符串中位置分别为 1、2、3、4、5，假设以每个字为首字的词长序列分别为{4,2}、{9,4,2}、{4,3,2}、{3,2}、{9,4,2}。根据改进算法，选取对应字为首字的词长取值应分别不大于 5、4、3、2、1，则以每个字为首字的最大词长取值为 4、4、3、2、空集，如此以每个字为首字的词长取值范围分别限制为{4,2}、{4,2}、{3,2}、{2}、空集。正向最大算法开始首先将“今”字在词典中对应的 4 字词与“今天星期”进行匹配，匹配不成功再将“今”在词典中对应的 2 字词与“今天”进行匹配，切出词“今天”；然后将“星”字在词典中对应的 3 字词与“星期天”进行匹配，切出词“星期天”。通过上述两种优化策略，实现了根据不同字动态设置合理的匹配初始最大词长及当匹配不成功时根据词长序列重设匹配长度，这样可以大大优化算法流程。

（3）待匹配字符串整体“长词优先”

根据匹配方向不同，最大匹配算法是从待匹配字符串首字或尾字开始进行匹配，这样“长词优先”的规则仅在对应字符串首字或尾字范围内使用，并没有在整个待匹配字符串范围内整体应用。若能通过某种方法将“长词优先”原则对于每一轮匹配都优先应用到整个待匹配字符串中，则切分出的词数可能会更少。根据最少切分原理，分词结果词数越少正确率越高，可以避免一定数量的错误切分。

最理想的情况下，每一轮的切分结果都应该是词典中以待匹配字符串中任意字为首字的最长词。通过标记词典中词条的长度，列出以待匹配字符串中的任意字为首字对应的所有词长，比较出最大值，并选择以此词长对应的字为首字、长度为该词长长度的字符串优先进行比较，这样就能保证“长词优先”原则应用到了整个待匹配字符串。为了进一步避免无意义的比较，某个字对应的最大词长需要根据该字在待匹配字符串中的位置进行限制，该最大词长并不一定是词典中以该字为首字的最大词长。

3.1.3 改进后的正向最大匹配算法

假设待匹配字符串 $Str=W_1W_2\cdots W_iW_{i+1}\cdots W_n(1\leqslant i\leqslant n,\ i\in N,\ n\in N)$，则待匹配字符串长度为 n。对于 $W_1W_2\cdots W_iW_{i+1}\cdots W_n$ 这 n 个字来说，其在字符串中的位置分别为 1、2、…i、i+1、…n，以字符串中任意字为首字的词长上限为 n-i+1。在词典中分别取 $W_1W_2\cdots W_iW_{i+1}\cdots W_n$ 这 n 个字的词长序列，对应每个字分别将该字符在词典中的所有词长值与该字符在字符串中的词长上限作比较，留下不大于对应词长上限的词长，并将相关信息按词长值大小排序，如果词长值相等则按对应首字位置从左往右排序。算法匹配过程中需要用到的信息包括相关字、该字在字符串中的位置、以该字为首字的词长上限、词典中以该字为首字的词条长度、该词条长度对应的词典正文指针。将这些信息集建表，格式如表 3-1 所示。

表 3-1 匹配信息表

Tab.3-1 Matching information table

字符	该字在字符串中位置	该字为首字的词长上限	该字为首字的词长	词长对应的词典正文指针
W_1	**1**	**n**	$L_{1,1}$	…
W_1	**1**	**n**	$L_{1,2}$	…
…	…	…	…	…
W_2	**2**	**n-1**	$L_{2,1}$	…
…	…	…	…	…
W_i	**i**	**n-i+1**	$L_{i,1}$	…
…	…	…	…	…
W_n	**n**	**1**	$L_{n,1}$	…
…	…	…	…	…

其中，$L_{1,2}$ 代表在词典以 W_1 为首字的词中，词长不大于相应上限的第 2 个词长。根据算法要求，将表中信息以词长排序，结果如表 3-2 所示。

表 3-2 中，W_a、W_b 为待匹配字符串中的任意字，且 $1\leqslant b\leqslant a\leqslant n$；d、e、f 代表词典中以对应字为首字的词长度，且满足关系 $d>e>f>0$；指针 $P_{(b,d)}$指向以待匹配字符串中位置为 b 的字符 W_b 为首字、词长为 d 的词典正文。

对于待匹配字符串 $Str=W_1W_2\cdots W_iW_{i+1}\cdots W_n(1\leqslant i\leqslant n,\ i\in N,\ n\in N)$，改进后的正向最大匹配算法流程如下：

（1）判断待匹配字符串长度 n，若 n=1，切出 Wn 结束循环；否则转（2）。

（2）在词典中查询以字符串中每个汉字 W_i 为首字的所有词长 $L_{i,1}$、$L_{i,2}$、$L_{i,\cdots}$，

找出不大于对应字符词长限度的词长值，与汉字、字符位置、词长上限、对应指针等信息存入表中；将信息表按词长由大到小排序。

（3）在待匹配字符串中找到表中第一条记录所指位置的汉字，取待匹配字符串中以此字为首字、长度为该条记录对应词长的子字符串，与词典中以此字为首字、长度为该条记录对应词长的词条进行比对。若匹配成功则切出该词，匹配不成功则根据下一条表信息重新进行匹配，直到匹配成功，（4）。

（4）将切出词后的剩余字符串左右两部分分别作为新的待匹配字符串 Str，转（1）。

表 3-2 匹配信息词长排序表

Tab.3-2 Matching information table according to length

字符	在字符串中位置	该字为首字的词长上限	该字为首字的词长（降序排列）	词长对应的词典正文指针
…	…	…	…	…
W_b	**b**	**n-b+1**	**d**	$P_{(b,d)}$
…	…	…	…	…
W_a	**a**	**n-a+1**	**e**	$P_{(a,e)}$
W_b	**b**	**n-b+1**	**e**	$P_{(b,e)}$
…	…	…	…	…
W_a	**a**	**n-a+1**	**f**	$P_{(a,f)}$
…	…	…	…	…

以字符串“学校就业率名列前茅”为例说明本文的正向最大匹配改进算法，如表 3-3 所示。

字符串长度 n=9，“学”、“校”、“就”、“业”、“率”、“名”、“列”、“前”、“茅”九个字在字符串中位置分别为 1、2、3、4、5、6、7、8、9，假设词典中以每个字为首字的词长序列分别为{7,4,2}、{5,4,2}、{4,3,2}、{7,4,2}、{4,2}、{6,4,2}、{4,2}、{4,2}、{4,2}，所有集合中元素数为 3+3+3+3+2+3+2+2+2=23 个。

根据改进算法，选取对应字为首字的词长取值应分别不大于 9、8、7、6、5、4、3、2、1，则以每个字为首字的词长取值集合分别限制为{7,4,2}、{5,4,2}、{4,3,2}、{4,2}、{4,2}、{4,2}、{2}、{2}、空集。将这些集合元素从大到小排序，结果为{7（1）,5（2）,4（1）,4（2）,4（3）,4（4）,4（5）,4（6）,3（3）,2（7）,2（8）}，结果中元素数为 12 个。其中结构 7（1）中 7 为词长，（1）代表该词长对应字在字符串中位置为 1。根据排序结果，算法开始首先将长度为 7 的 字符串“学校就业率名列”与位置为 1 的字“学”在词典中对应的 7 字词进行匹配，匹配不成功

再将长度为 5 的字符串“校就业率名”与位置为 2 的字“校”在词典中对应的 5 字词进行匹配，直至长度为 4 的字符串“名列前茅”与词典中以位置为 6 的字“名”为首字的词条“名列前茅”匹配成功，切出词语“名列前茅”。将切分后剩余 “学校就业率”的 5 字字符串重新进行上述匹配，先切分出“就业率”，再切分出“学校”，得到分词结果为“学校/就业率/名列前茅”，剩余字符串长度为 0，切分结束。

表 3-3 匹配信息表示例

Tab.3-3 Example of matching information table

字符	在字符串中位置	该字为首字的词长上限	该字为首字的词长（降序排列）	词长对应的词典正文指针
学	1	9	7	$P_{(1,7)}$
校	2	8	5	$P_{(2,5)}$
学	1	9	4	$P_{(1,4)}$
校	2	8	4	$P_{(2,4)}$
就	3	7	4	$P_{(3,4)}$
业	4	6	4	$P_{(4,4)}$
率	5	5	4	$P_{(5,4)}$
名	6	4	4	$P_{(6,4)}$
就	3	7	3	$P_{(3,3)}$
列	5	3	2	$P_{(5,2)}$
前	7	2	2	$P_{(7,2)}$
矛	8	1	0	无

3.2 双字哈希词长分组词典机制设计

对于基于词典的机械分词算法，分词速度除跟算法本身的运行流程有关系外，合适的词典结构也对效率有举足轻重的作用。机械分词算法与分词词典的有效配合才能保证良好的分词性能。

传统的汉字词典中，词长以 2 为主，且普遍存在不定长的现象。选用文献[37]对人民日报语料库进行的统计和分析可知，二字词所占比例很大，使用频率很高，四字以上多字词语占比较低，使用频率也较低。统计与分析如表 3-4 所示。

表 3-4 人民日报语料库词语统计信息

Tab.3-4 The People's daily corpus word statistics

类别	词数/个	词数占比/%	词频/个	词频占比/%
一字词	2873	5.164	361751	40.971
二字词	30939	55.616	463077	52.447
三字词	10658	19.159	38082	4.313
四字(含)以上词	11160	20.061	20032	2.269
总词数/总词频	55630	100	882942	100

汉字在计算机内以内码的形式进行存储[38]，通过哈希函数可以快速的定位汉字。基于汉字词中二字词所占比例较大，为了提高词典中词条的查找速度，本文选用双字哈希的词典机制，并根据算法需求加以优化。该机制吸纳了“整词二分”和“Trie 索引树”的优点，只对词条的前两个字建立 Hash 索引，构成深度为 2 的 Trie 子树，而词的剩余字串则按词长降序排列组成词典正文，这样既可以提高匹配效率又可以兼顾空间利用率。

本文设计的词典中不包含词长为 1 的词，因为若在词典中找不到以某字为首字的词，在分词时可直接切分该字为词，并不需要在词典中分配单字词位置，这样可以有效减小词典机制所占存储，避免空间浪费。

为进一步提高查询效率，词典采用分组思想，将首字相同的所有词条按照不同的词长进行分组，且标记长度。这种情况下最小组是首字、次字相同且词长相同的词条集合，查找范围更加小的同时满足了算法动态选择最大词长的要求。本文所用词典词词长索引表中标记的最小词长为 2， 改进后的词典结构，如图 3-2 所示。

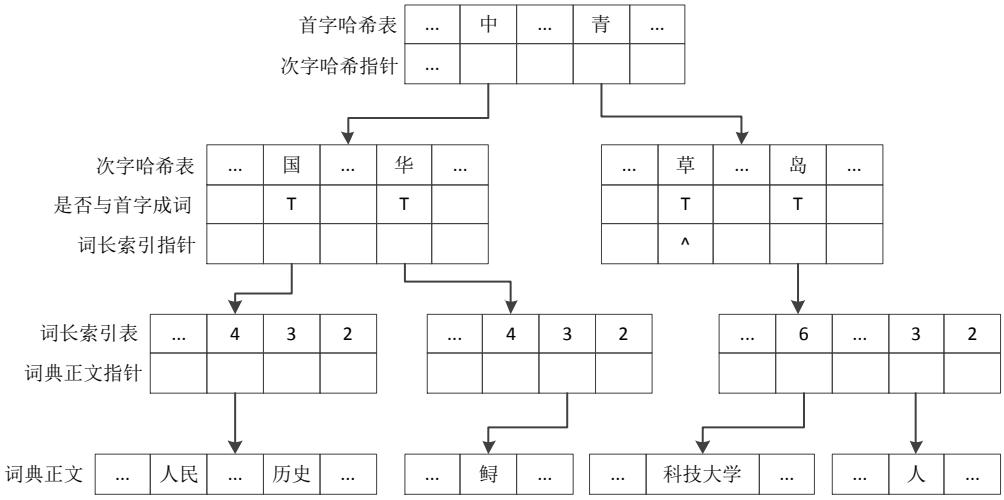

图 3-2 改进的词典结构图

Fig.3-2 Improved dictionary body figure

改进的词典结构分为 4 部分：首字哈希表、次字哈希表、词长索引表、词典正文。

（1）首字哈希表：由散列表的关键字及次字哈希指针构成，用于确定词语首字的具体位置。

汉字在计算机中以内码形式存在，将内码计算后转为区位码，再将区位码转换为一个十进制的数字，这个数字就是该汉字在哈希表中的序号。该序号指示了以该字为首字的词的词长表入口地址。根据 Hash 函数[39]，入口地址的计算，如公式 3-1 所示。

$$Offset = (ch1 - 0xB0) * 94 + (ch2 - 0xA1) \quad (3\text{-}1)$$

其中，Offset 为该字在哈希表中的序号，ch1、ch2 分别为该字机内码的高字节和低字节。

（2）次字哈希表：包括关键字、是否能与首字组合成词、以此首字和次字开头的词长索引指针三部分。

（3）词长索引表：对应以前述二字开头的词的剩余内容，包括以前述二字开头的词的长度及所有该长度词集合构成的词典正文指针两部分。

（4）词典正文：顺序存储某长度词语除首字、次字外的剩余字符。

为提高词典查找速度，本文采用了双字哈希的结构。为适应改进后的正向最大匹配算法合理选取匹配词长的需求，词典在两层哈希结构后增加了词长索引表。该词典机制根据词长对词语进行了恰当的分组，使匹配的范围进一步缩小，大大降低了算法的时间开销。

本文上节根据正向最大匹配算法的不足做了具体改进设计，本节选择高效的双字哈希词典进行优化，在下节将会把算法改进和词典优化融为一体，设计基于双字哈希余字分组词典的正向最大匹配改进算法。

3.3 基于双字哈希词长分组词典结构的最大匹配改进算法

上文提出的最大匹配改进，相较于最大匹配算法有以下改进：一是每次匹配优先选择待匹配字符串全域内的最长词；二是根据字符串中所有字的位置及在词典中对应词的词长选择合适的匹配初始位置和匹配初始字符串长度；三是当匹配不成功时选择待匹配字符串全域内对应下一词长的子串进行匹配。理论上经过改进的算法分词速度更快且切分精度更高。

根据汉字词中二字词最多的特点，选用双字哈希的词典结构。针对算法需要词典中词长度信息的需求，对双字哈希词典加以改进，设计双字哈希余字按长度分组的词典结构。双字哈希的结构可以保证词语的查找速度，同时优化后按词长

分组的词典将程序进行匹配时的范围大大缩小。

基于双字哈希词长分组词典结构的正向最大匹配算法流程具体步骤如下：

假设待匹配字符串 $Str=W_1W_2\cdots W_iW_{i+1}\cdots W_n(1\leqslant i\leqslant n,\ i\in N,\ n\in N)$，待匹配字符串由包括首字为 W_1 的 n 个字符构成，i 初始值为 1。

（1）若 n=1，则切出 W_n 算法结束；否则转（2）。

（2）通过哈希函数确定待匹配字符串首字 W_i 在词典中位置，若存在则转（4）；不存在则切出 W_i，i=i+1，转（3）。

（3）若 i=n，切出 W_n 算法结束；否则转（2）。

（4）确定以 W_i 为首字的词典次字哈希表中是否存在 W_{i+1}，若存在则转（5）；不存在则切出 W_i，i=i+1，转（3）。

（5）依次计算词典中以 W_iW_{i+1} 为首二字的不大于 s 的所有词长 $L_{2i+1,j}$，其中 $2\leqslant j\leqslant s$；i=i+1，转（6）。

（6）以 W_i 为首字的剩余字符串的长度 s=n-i+1，若 s=1，转（9）；否则转（7）。

（7）通过哈希函数确定 W_i 在词典中位置，若存在则转（8）；不存在则 i=i+1，转（6）。

（8）确定以 W_i 为首字的词典次字哈希表中是否存在 W_{i+1}，若存在则转（5）；不存在则 i=i+1，转（6）。

（9）若词长取值为空集，则将字符串逐字切分，算法结束；否则将所有符合条件的词长按值从大到小排序，令程序初始匹配最大长度 $wordmax=maxL_{2i+1,j}$，转（10）。

（10）待匹配字符串初始匹配位置为 i，取以 W_i 为首字长度为 wordmax 的字符串 $W_i\cdots W_{wordmax-1}$ 与词典中以 W_iW_{i+1} 为首二字词长为 wordmax 的词进行匹配。若匹配成功则转(11)；匹配不成功则转(12)。

（11）切出词 $W_i\cdots W_{wordmax-1}$，字符串左右剩余两部分分别作为新的 Str，转（1）。

（12）存在下一词长，转（13）；否则转（14）。

（13）令 wordmax 为符合条件的下一词长值，转（10）。

（14）逐字切分，算法结束。

结合正向最大匹配改进算法及词典机制优化，得到基于双字哈希词长分组词典结构的正向最大匹配算法。该算法与最大匹配算法的分词结果比较，如表 3-5 所示。

表 3-5 两种算法分词结果示例

Tab.3-5 Word segmentation examples of two algorithms

例句/分词算法/实验结果	最大匹配改进算法	正向最大匹配算法
从中国共产党成立起	从/中国共产党/成立/起	从中/国共/产/党/成立/起
会所罗门群岛语言	会/所罗门群岛/语言	会所/罗/门/群岛/语言
独立自主和平等互利	独立自主/和/平等互利	独立自主/和平/等/互利

基于双字哈希词长分组词典结构的正向最大匹配算法流程图如图 3-3 所示。

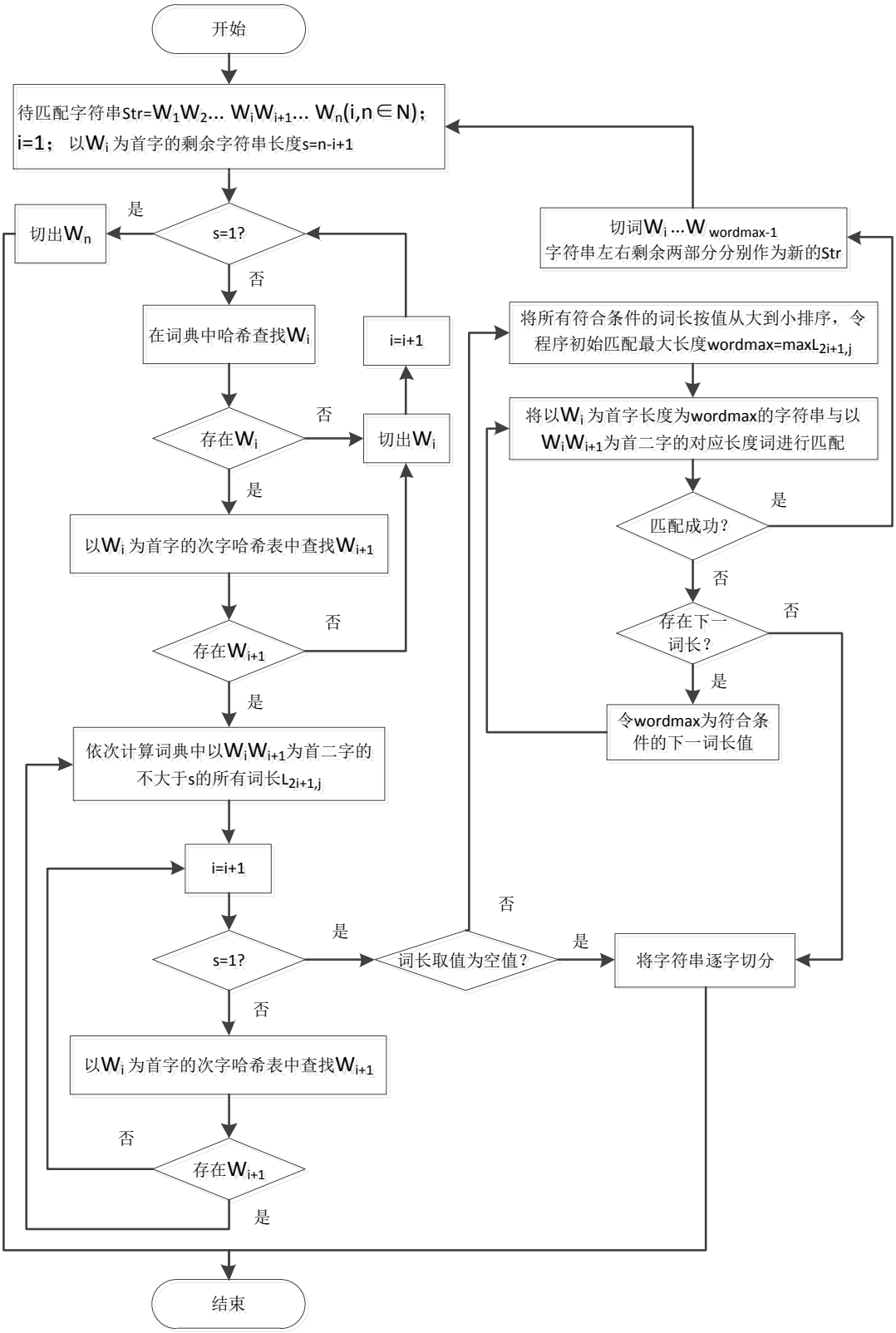

图 3-3 改进算法流程图

Fig.3-3 The improved algorithm flow chart

3.4 分词歧义处理

上节提出的基于双字哈希词长分组词典结构的正向最大匹配算法实现了在待匹配文本全域内优先选择最长词的目的，该种方法切分出的结果集合是能组成待匹配文本的最少词数。一般来说，整体优先选择最大词长的方法可以避免大部分的错误切分，但仍有一些歧义无法避免。例如句子“他开了个种子公司”，正确切分结果为“他/开/了/个/种子/公司”,而根据“长词优先”算法会切分为“他/开/了/个/种/子公司”。要想得到更高的分词精度，就需要对切分结果进行歧义识别和处理。

进一步对上节提出的基于双字哈希词长分组词典结构的正向最大匹配算法进行研究发现，在分词过程中为了缩小查找范围减少匹配词数，采用了双字哈希词典并且判断了所有 W_iW_{i+1} 的连接情况以确定是否有必要继续进行匹配。

汉字词语四字及四字以上词语出现歧义的可能性非常小。根据文献[37]的统计，一字词、二字词、三字词的词频分别为 40.971%、52.447%、4.313%，则对于 4 字字符串，正确组合方式为一字词+三字词的概率远小于二字词+二字词的概率。根据文献[23]得知，尽管交集型歧义有多种样式，歧义样式分布同样有规律可循:其中链长为 2 的交集型歧义出现频率占到交集型歧义字段总数的接近一半；且对于链长为 2 的歧义字段 JKLM，只有 2%左右不是按照 JK/ML 切分的。若能采用恰当方法处理 4 字字符串的切分，在时间代价较小的前提下可大大提升分词准确率。文献[16]对基于词典的中文分词歧义算法进行了研究，其中提到若歧义字段长度等于 4，则直接从中间切分为两个二字词组合，如此切分正确率最高。本文歧义处理阶段主要借鉴正向最大匹配+回退一字法的思想对切分结果中的三字词进行识别和歧义消除。

算法流程描述如下：

（1）从左到右寻找切分结果中的三字词 $W_iW_{i+1}W_{i+2}$，若存在转（2）；否则处理结束。

（2）若 i-1>0，转（3）；否则转（6）。

（3）判断切分结果中 W_{i-1} 是否被切分为单字词，若是转（4）；否则转（6）。

（4）判断 $W_{i-1}W_i$ 是否为词，若是转（5）；否则转（6）。

（5）$W_{i-1}W_iW_{i+1}W_{i+2}$ 重新切分为 $W_{i-1}W_i/W_{i+1}W_{i+2}$，转（1）。

（6）若 i+2<n，转（7）；否则转（1）。

（7）判断切分结果中 W_{i+3} 是否被切分为单字词，若是转（8）；否则转（1）。

（8）$W_{i+2}W_{i+3}$ 是否为词，若是转（9）；否则转（1）

（9）$W_iW_{i+1}W_{i+2}W_{i+3}$ 重新切分为 $W_iW_{i+1}/W_{i+2}W_{i+3}$，转（1）。

本文的歧义处理流程，如图 3-4 所示。

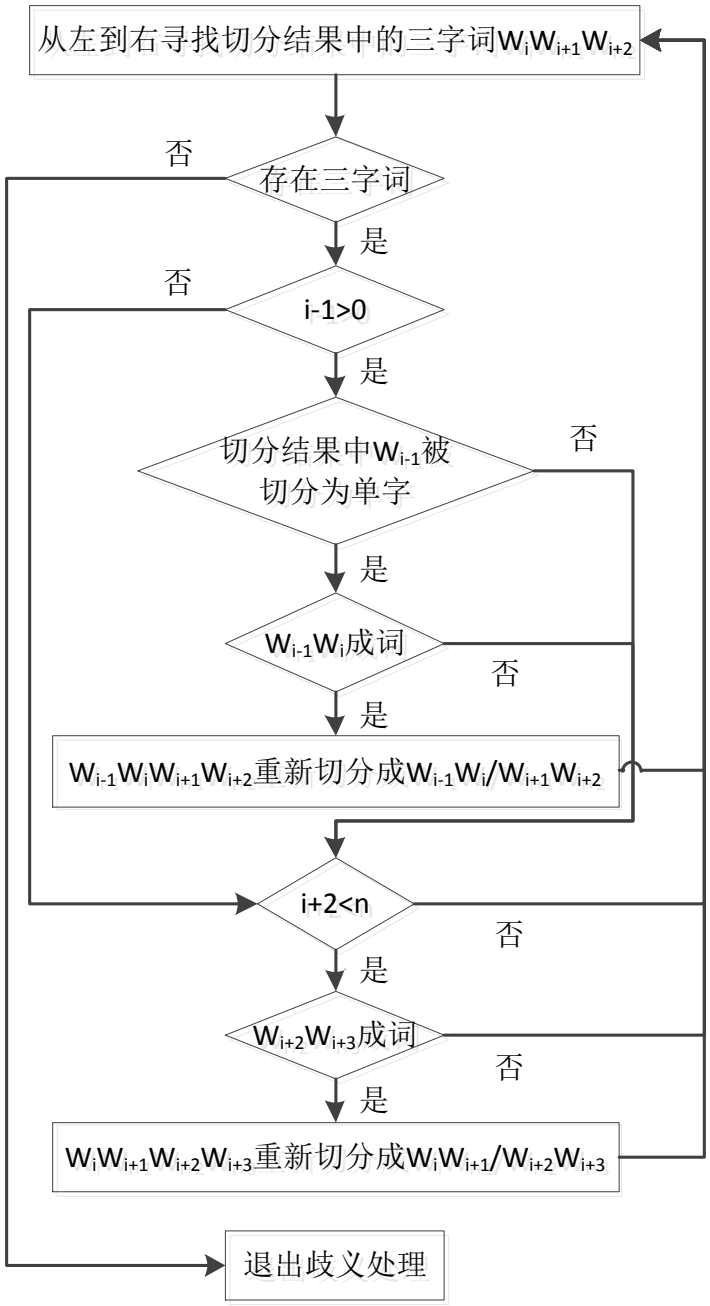

图 3-4 歧义处理流程图

Fig.3-4 Ambiguity processing flow chart

以“贸易进出口岸”为例说明歧义处理过程：该句通过上节的算法切分结果为“贸易/进出口/岸”。（1）从左到右找到切分结果中为三字词为“进出口”；（2）“进”字并非字符串首字；（3）“进”字前一字“易”并未被切分为单字词；（4）“口”字并非字符串尾字；（5）“口”字后一字“岸”被切分为单字词；（6）“口岸”为词；（7）“进出口岸”被重新切分为“进出/口岸”；（8）原切分结果中再无三字词，歧义处理过程结束。

3.5 本章小结

本章首先根据最大匹配算法的不足之处改进了算法，算法改进后可以根据待匹配文本中所有汉字在字符串中的位置及该字在词典中对应的词长动态选择合适的匹配点和匹配长度，可以同时避免不必要的算法消耗和歧义结果。然后根据改进算法的需求优化传统词典结构，双字哈希余字按词长分组的词典结构适用于改进算法。最终结合词典提出基于双字哈希余字分组词典结构的最大匹配改进算法，并针对算法结果设计歧义处理模块，通过歧义处理可较好的识别和避免部分交集型歧义字段。

4 改进算法在 Lucene 中的应用实现

Lucene 是 Apache 基金会的重要研究成果之一，是当前使用最为广泛的开源全文搜索引擎工具包，该工具包能够方便的被移植嵌入和二次开发。Lucene 中自带的中文分词器性能较差，开发人员可通过其丰富的 API 接口根据需要在 Lucene 上添加定制的分词器，并以此为基础建立完整的搜索引擎。

本章将根据第三章提出的基于双字哈希余字分组词典的正向最大匹配改进算法设计 Lucene 的中文分词模块，使其具有更高的可用性。

4.1 搜索引擎开发平台 Lucene

Lucene 工具包目前有 Java、Perl、Python 等多语言版，其中 Java 版本使用最为普遍。Lucene 没有爬虫功能，不是具有完全意义的搜索引擎应用程序，而是一个全文搜索引擎的框架，可提供系统的索引和检索功能。作为全文索引的引擎工具包，Lucene 基本可应用到所有需要全文检索的系统，尤其可以适用于跨平台的应用、检索模块实现[40]。相对于数据库应用系统来说，使用 Lucene 完成全文检索具有一定的优势。首先，多种信息载体文件通过预处理可转换为 Lucene 能够处理的数据（像常见的网页、PDF、Word 文件、数据库中的数据表等）。其次，由于数据源信息量巨大能检索到的相关内容也非常多，Lucene 通过某些策略只将最相关的搜索结果提取出来，而不是将搜索到的所有内容呈现出来。虽然检索匹配总数很多，通过 Lucene 得到的结果并不会使用很多存储资源。

当前，许多应用都基于 Lucene 搭建了应用自身后台的全文索引引擎，比较有名的项目有：Eclipse（著名 Java 开发平台，该开放平台内部以 Lucene 框架实现了对帮助文档的检索）；Nutch（基于 Lucene 的分布式搜索引擎系统）；Hadoop（使用 Lucene 搭建的的分布式计算平台）。由于搜索和索引架构良好、代码开放且易于开发，Lucene 在国内外有很多忠实粉丝，除了个人用户外，像 Twitter、搜狐、搜房网、亚商等大体量公司也是 Lucene 的使用者。

4.2 Lucene 系统结构

Lucene 的系统结构主要功能由 5 个包完成，如图 4-1 所示。

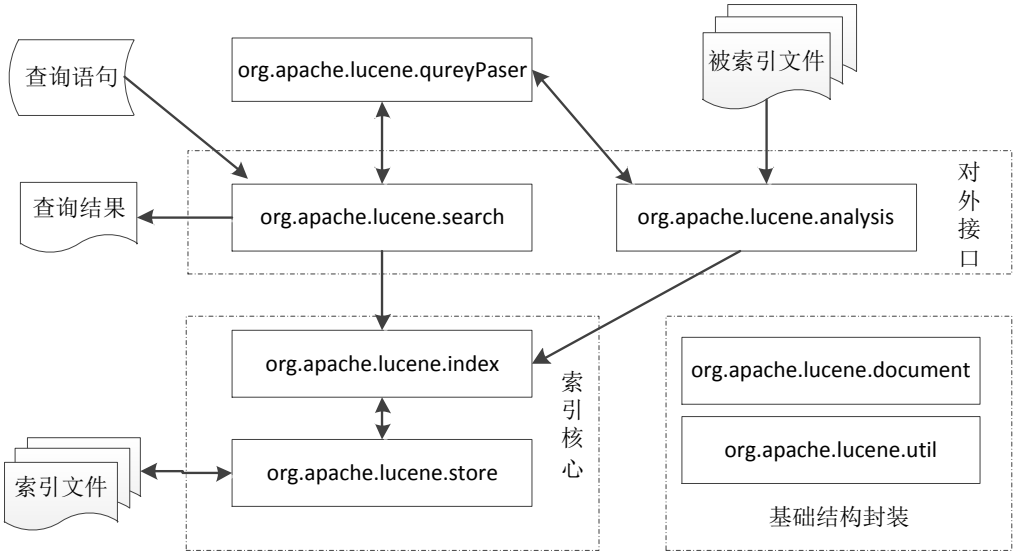

图 4-1 Lucene 系统结构图

Fig.4-1 Lucene system structure

在核心组件中，org.apache.Lucene.store 包着眼于底层的 I/O 存储结构并提供对索引存储的支持，其中的 FsDirectory 和 RAMDirectory 类是 Lucene 中重要的两种索引方式：org.apache.Lucene.index 包完成索引。在对外接口组件中，org.apache.Lucene.search 完成对索引数据的检索：org.apache.Lucene.analysis 包则负责完成对索引和检索时检索项的分词、语言分析、过滤等功能；org.apache.Lucene.queryParser 包负责对使用者的输入文本进行分析（例如明确各检索关键词间的逻辑关系等）。在基础结构封装组件中，org.apache.Lucene.util 包包含数个具体的公用工具类。例如可以通过 org.apache.Lucene.document 完成对 Document 和 Field 的各种操作。

分词、索引、检索处理在信息检索中占有重要的地位。分词的准确与否直接影响到后续的检索。Lucene 提供了分析器用于处理词汇，此外也有一些开源的分析处理模块可供方便地集成到信息检索系统中；Lucene 的索引是利用预设的索引项字典建立按索引项排列的链表来对文件进行索引，它提供了多种不同的建立索引的方式，并提供了添加、删除、更新、管理索引文件的机制；Lucene 在检索时可以用抽象类 Diretory 来表示这个索引并把该 Diretory 类实例传递给 IndexSearcher 类，然后通过将封锁在 Query 对象中的用户提交的检索项传递给 IndexSearcher 类的 search 方法，并获得匹配检索语句的文档集 Hits。一般来说，完成信息检索任务要经过如下几个步骤[41]：

1.文档解析。此步骤由 Lucene 之外的特定解析工具进行。

2.文本分析。Lucene 内部文本分析处理由抽象类 Analyzer 实现，对于不同语

言的文本，Lucene 会派生出特定的 Analyzer。因为 Lucene 采用的索引数据结构为倒排索引机制，所以 Lucene 在构造索引前需要首先把待处理的文本内容切分成若干个信息单元（即 Token），然后利用这若干个信息单元构造索引。

3.生成索引。通过使用 IndexWriter 类可以将文档转换成 Token，并将 Token 写入到倒排文件中。Lucene 会首先得到某个 Token 在相应语料中出现的位置统计和频率统计等信息，然后才进行内容分析，该 Token 的相关信息需要依据倒排文档的格式写入到文件中。Lucene 在生成索引时，可能会进行若干压缩等操作。

4.处理用户提交的检索词。该部分同需要使用 Query 和 QueryParser 类来实现，具体内容包括完成模糊查询、语义查询、范围查询、组合查询、短语查询等多种检索操作。

5.完成检索。Lucene 通过 IndexSearcher 完成检索工作。构建 IndexSearcher 对象的方式多种多样，常用的如基于 Directory 的对象构建、基于文件系统的路径构建等。

6.得到检索结果。在 Lucene 中 Hits 负责返回结果集。与输入项检索项相关的信息可以通过 Hits 得到。

4.3 Lucene 索引和检索实现

4.3.1 Lucene 索引实现

索引是搜索引擎的第一步，是 Lucene 最重要的一个过程，Lucene 的索引不依赖于任何数据库平台，采用倒排索引机制。倒排是一种面向 Term 的索引机制，通常由关键字和出现位置两部分组成。对于索引中的每个关键字，都跟随一个用来记录这个关键字在文档集合中出现位置的位置表。通过 IndexWriter 的 addDocument 接口，可以将构建好的 Document 加入索引。

常用的 Lucene 索引有两种方式，一种是采用 Directory 类的 RAMDirectory 方法，这种方法在内存中建立索引效率较高，但关机后索引会消失；另一种索引方式采用 FSDirectory 方式，它在硬盘目录中创建索引文件，其优点是可以长期保存，缺点是检索时需要进行 IO 操作，速度较慢。

Lucene 的索引文件由若干个子索引段 Segment 组合而成，每个 Segment 包含若干个 Document，同一个 Segment 中的全部索引文件具有共同的前缀（索引后缀不同）。索引文件的前缀名由两部分构成，一部分是待处理的 Document 的个数（转化成三十六进制数），第二部分是该三十六进制数前面的“_”。需要注意的是每个 Index 中有且只有一个无后缀的 Segment_*文件，它记录了当前索引中全部

Segment 情况[42]。

Lucene 中实现索引步骤如下：（1）构建索引，实例化 IndexWriter；（2）构建 Document 并创建 Field；（3）调用 IndexWriter 实例的 addDocument()方法将 Document 添加到索引。

Lucene 中往索引文件中添加文档的流程图[43]，如图 4-2 所示。

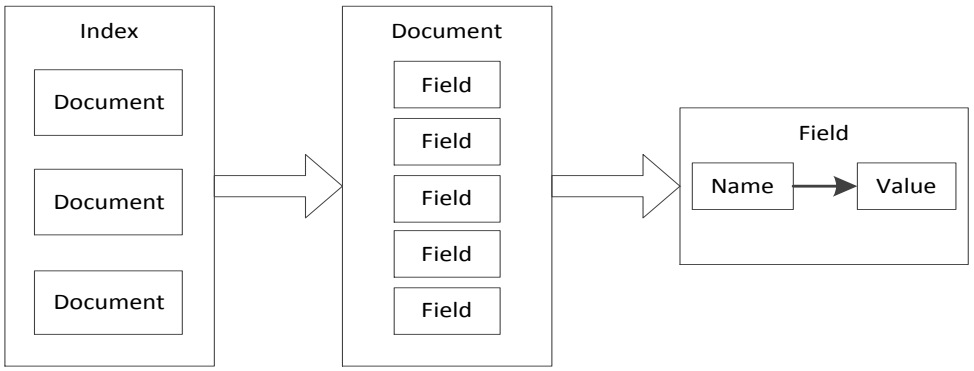

图 4-2 在索引中添加文档流程图

Fig.4-2 Flow chart of added document to the index

IndexWriter 的构造函数中含有三个参数，第一个参数代表索引位置，第二个参数是所调用的分析器 Analyzer，第三个参数的值代表构建的索引是否为增量索引。Lucene 通过 IndexReader 中的方法来管理已经建立好的索引，使用 IndexWriter 中第三个参数的取值来设定增量索引（在已有的索引上追加新的文档，不覆盖原索引），通过调用 IdexWriter 中的 updateDocument 函数来更新索引。创建索引部分代码，如图 4-3 所示。

```
public void createIndex(String inputDir) {
   try {
        IndexWriter writer = new IndexWriter(INDEX_STORE_PATH,
               new GaiJinAnalyzer(), true);
        File filesDir = new File(inputDir);
        取得所有需要建立索引的文件数组;
        遍历数组;
        for (int i = 0; i < files.length; i++) {
        获取文件名;
        判断文件是否为txt类型的文件;
        创建一个新的 Document;
        为文件名、文件内容等分别创建 Field;
        doc.add(field);
        //Field 加入 Document
        writer.addDocument(doc);
        // Document 加入 IndexWriter
       }
   }
        //关闭IndexWriter
        writer.close();
   }catch (Exception e) {
    异常处理;
  }
}
```

图 4-3 创建索引核心代码

Fig.4-3 Code of creating index

4.3.2 Lucene 检索实现

Lucene 建立的强大的检索机制，有丰富的检索 API 接口。Lucene 里与搜索相关的 API 多数都被包含在 org.apache.Lucene.search 包中，IndexSearcher 类是该包中最重要的类，Lucene 完成检索的主要步骤如下：

（1） 创建 IndexSearcher 实例实现对索引的访问。

（2）通过构造 Term 和 Query 对象查询信息。也可使用 QueryParser 的 Parse 方法来直接根据检索词构造。

4.4 中文分词模块框架设计

完整的中文分词模块结构一般包含三个阶段：预处理阶段、分词算法切分阶段、歧义处理阶段。如图 4-4 所示。

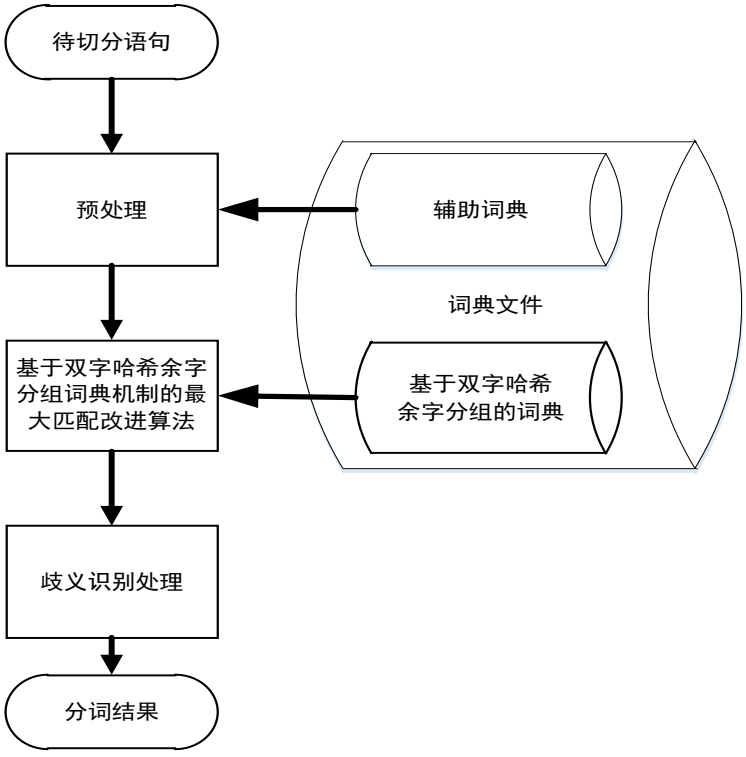

图 4-4 中文分词模块工作流程

Fig.4-4 Chinese word segmentation module workflow

中文分词预处理阶段主要任务是将待处理文本粗切分成更小的字符串集合，方便系统进行进一步切分；分词算法切分阶段是指使用分词算法将粗切分成的字符串集合精确切分成有意义的词的集合；歧义识别处理阶段是针对分词算法切分出的词集合，基于特定的策略进行错误切分的检查和处理。

分词预处理包括三个过程：一是根据标点符号对待切分的文本进行断句，将文本划分为意思相对完整的句子或字符串序列集合；二是根据字符串类型进行判断，将汉字、字母、数字等不同字符串类型进行筛选分隔，将复杂的句子组成进一步简化成纯汉字序列；三是根据某种策略结合噪声词典、量词词典、汉语数词词典等词典对文本过滤。预处理是进行中文分词的第一步，恰当的分词预处理可有效较少分词算法切分阶段的负担。预处理的过程，如图 4-5 所示。

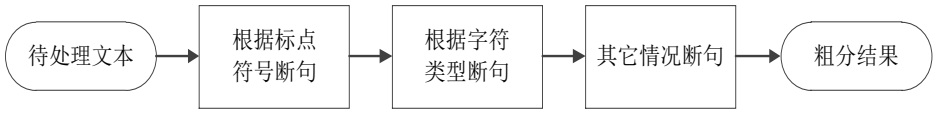

图 4-5 中文分词预处理过程

Fig.4-5 Chinese word segmentation pretreatment process

预处理的过程是将系统识别出的断句标识直接切分并进行断句处理，在分词结束时与精确切分出的词一起按位置组合输出最终的切分结果。以预处理句子“目前 UI 设计人才非常紧缺，该行业人员流动大、岗位要求高，经验丰富的从业者供不应求”举例说明，经预处理后的结果为“目前/设计人才非常紧缺/该行业人员流动大/岗位要求高/经验丰富的从业者供不应求”。预处理完成后再使用分词算法对切割成的字符串段进行进一步切分。

4.5 构造 GaiJinAnalyzer 分词器

分析器的功能是过滤和切分句子为词序列。分析器是文本信息与索引库的桥梁，只用经过分析器分析的文本才能建立索引。Lucene 提供了以下几种分析器，如 WhitesapceAnalyzer 分析器，用于去除文本中的空格；SimpleAnalyzer 分析器用于进行单词首字母大小写转换；StopAnalyzer 分析器，此分析器是在 SimpleAnalyzer 的基础上增加了过滤停用词的功能；StandAnalyzer 分析器对于英文的处理能力与 StopAnalyzer 一致。

为了进行中文分词，Lucene 提供了针对中文文本处理的工具包 ChineseAnalyzer、CJKAnalyzer 以及 MMAnalyzer。ChineseAnalyzer 采用的是切分单字为词的方式，CJKAnalyzer 采用的二元覆盖的方式分词，这两个工具包的分词效果非常差。MMAnalyzer 则采用的是最大正向匹配算法，相对于 ChineseAnalyzer、CJKAnalyzer 分词性能要好得多。想要获得较好的中文处理功能，必须使用分词效果好的中文分析器。良好的分词模块质量对于搜索引擎性能至关重要，分词精度直接决定搜索引擎的搜索精度，分词速度也同样影响搜索引擎的响应速度。

本文设计的分析器 GaiJinAnalyzer 通过实现 Analyzer 类的抽象方法

tokenStream(String,Reader)继承原 Lucene 中的 Analyzer 类分析器，GaiJinAnalyzer 包含分词器 GaiJinTokenizer 和过滤器两部分。上章提出的改进算法主要针对于分析器中的分词器 GaiJinTokenizer，至于 GaiJinAnalyzer 中的过滤器，则依旧使用 Lucene 中内置的 TokenFilter。Analysis 包是系统中负责语言分析处理的模块，主要功能有两点，第一个是将待匹配文本解析为词语单元（Tokenization，包括去除标点、移除常用词、字母规格化等），第二个是将各词语单元与关联 Field 相结合形成 Term。Lucene 在按关键词搜索和建立索引文档时调用 GaiJinAnalyzer，Lucene 工作后台简图，如图 4-6 所示。

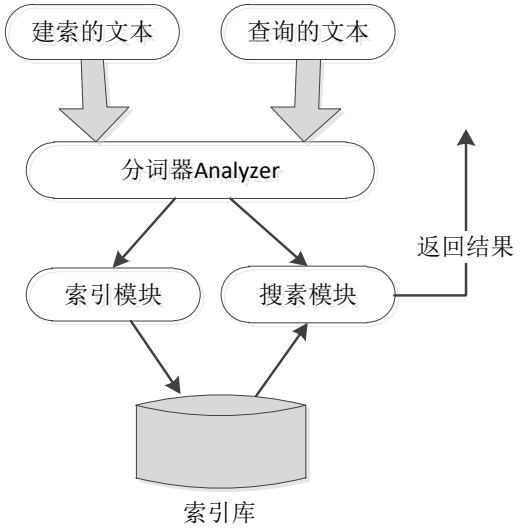

图 4-6　Lucene 工作后台简图

Fig.4-6 Working backstage diagram of Lucene

分词器构造过程如下：

1.完成对优化后的词典机制的实现和词典初始化。

2.完成待处理文档的分词预处理

3.根据本文提出的基于双字哈希余字分组词典结构的最大匹配改进算法思想实现算法。将本文算法嵌入到 Lucene 中，定义改进的分词器类 GaiJinTokenizer 继承 Tokenizer 类，定义改进的析器类 GaiJinAnlyzer 继承 Analyzer 类。GaiJinTokenizer 类的主要功能为把分词预处理过后的文本流转换为 TokenStream 流，返回的结果为 Token 对象。

TokenStream 类定义了 next()和 close()两个抽象方法，其中 next()方法用于返回下一个 Token，close()方法用于关闭流。TokenStream 的定义，如图 4-7 所示。

```
public abstract class TokenStream {
        public abstract Token next() throws IOException;
        //取出下一个Token
        public void close() throws IOException {};
        //关闭流
}
```

图 4-7 TokenStream 定义

Fig.4-7 Definition of GaiJinTokenizer

GaiJinTokenizer 类是文本分析模块的核心类，其构造函数接收的是 Reader 对象，使用 next()方法返回分词结果。

部分代码展示如图 4-8、图 4-9、图 4-10 所示：

```
public abstract class GaiJinTokenizer extends TokenStream {
//分词器基类

protected Reader input;
protected GaiJinTokenizer() {}
//表示传入的文本流
protected GaiJinTokenizer(Reader input) {
        this.input = input;
}
public void close() throws IOException {
input.close();
}
}
```

图 4-8 GaiJinTokenizer 代码

Fig.4-8 Code of GaiJinTokenizer

```
public class GaiJinAnalyer extends Analyzer
{
     public TokenStream tokenStream(String fieldName, Reader reader)
     {
        使用 GaiJinAnalyer 对文本进行分词;
      }
 }
```

图 4-9 使用 GaiJinAnalyzer 进行分词实现代码

Fig.4-9 Code of segmenting by GaiJinAnalyzer

```
private ArrayList getTokens() throws IOException{
//将文本读入内存
ArrayList resList = os.analyzeSens();
//将分词结果添加到结果集
return this.tokenList;
 }
```

图 4-10 将文本读入内存代码实现

Fig.4-10 Code of reading text into memory

4.6 优化的词典机制实现

4.6.1 词典的结构

词典是机械分词工作的基础，本文研究是正向最大匹配算法的改进，在设计的中文分词系统工作过程中，需要在分词词典进行大量的查找工作合理的词典构造是实现本文设计的文本分析模块的先决条件。

本文的分词词典的结构中由 4 个表组成：首字哈希表、次字哈希表、词长索引表以及余字词典正文线性表。其中首字哈希表中的内容包括：词条首字关键字（FirstWord）和从首字哈希表指向次字哈希表的指针 pSecondWord；次字哈希表中的内容包括：以首字哈希表关键字为首字的词的第二个关键字（SecondWord）、该关键字是否与首字关键字结合为词的标记状态（True/False）以及从次字哈希表指向词长索引表的指针 pLenghIndex；词长索引表中包含以首字关键字和次字关键字为首二字词的所有词长 Length 和由词长索引表指向余字词典正文中首个词条的指针指针 pDic 指向词典正文中末字为 KeyWord 长度为 length 的首个词条。

本文使用的是改进的双字哈希词典，首字和次字索引表通过 Java 类 java.util.HashMap 实现。

4.6.2 词典的建立

通常词典文件以.txt 的形式存储，词典中的词数越多，机械分词的精度就越高。但同时随着词典内容的增多，在进行查找时匹配次数会相应增加，增加了词语的查找时间。在查询词典前，系统会将词典内容载入到内存中以方便查询使用。词典加载实现如图 4-11 所示。

```
InputSteam file = null;
if (System.getProperty("dic.dir")==null)
//未指定词典存放路径时，从默认路径加载
file = getClass().getResourceAstream(Dictionary.getDir() + dic);
else
file = new FileInputStream(new File(Dictionary.getDir() + dic));

BufferedReader in = new BufferedReader(new InputStreamReader(file,"GBK"));
string word;
while ((word = in.readLine()) != null) {
//按行处理读入的文本格式的词典
}
in.close();
```

图 4-11 词典加载代码实现

Fig.4-11 Dictionary loading code implementation

改进的词典为首字和次字固定，词条剩余字按词长排序的机制。其中

FirstWord 标识首字关键字，SecondWord 标识次字关键字。True/False 标识首字和次字是否成词，LenghIndex 标识首字、次字相同的词条词长及对应余字位置，Word 标识首字、次字相同词条的剩余关键字。词典加载完成后结果，如图 4-12 所示。

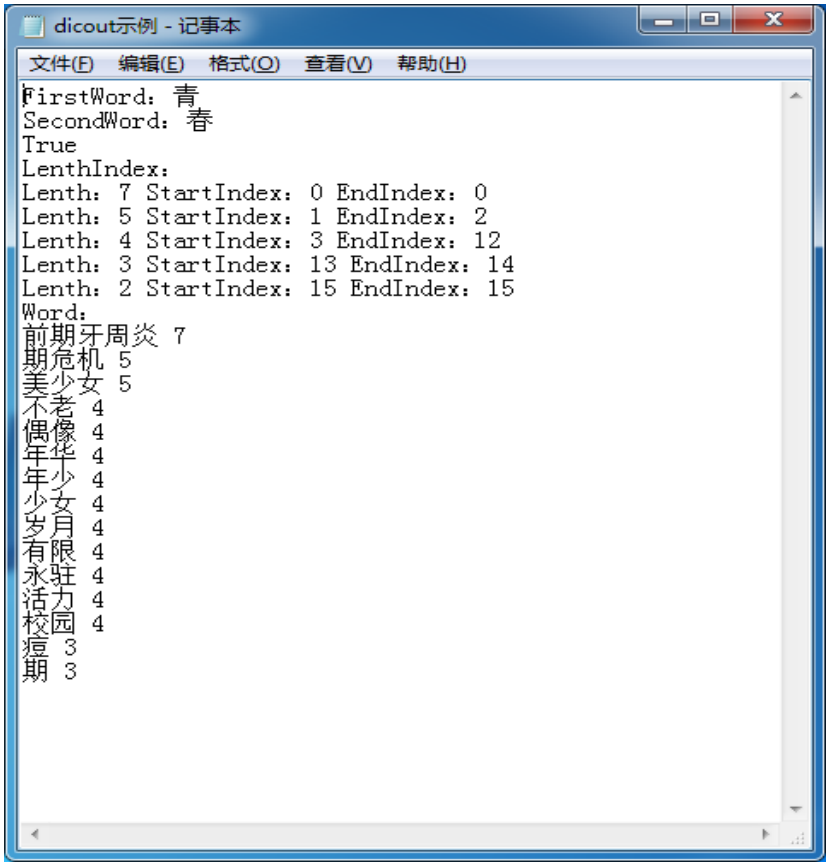

图 4-12 词典加载示例

Fig.4-12 Dictionary load sample

4.7 基于改进算法的分词工具

为了验证本文用基于双字哈希余字词长分组词典结构的正向最大匹配改进算法实现的中文分词模块的性能，以该模块为基础建立了一个中文分词工具。该中文分词器的主要功能是将输入的文本信息经过正向最大匹配改进算法切分并输出分词结果，主要有输入、输出两大功能。其中输入有直接打字输入、粘贴文本和设置路径读取文件 3 种方式，并设有重置选项；输出功能显示与输入对应的切分结果，设有复制结果和保存结果选项。分词工具的工作界面，如图 4-13 所示。

图 4-13 分词工具界面

Fig.4-13 Interface of word segmentation

分词工具首先将输入内容传递到后台算法接口，经过算法切分后将结果传递给上层接口，通过输出模块显示。随机选择一段文本内容对该工具进行性能测试，分词界面如图 4-14 所示。

图 4-14 分词结果界面

Fig.4-14 Interface of word segmentation results

测试所选词典共有 427450 条词语的纯词语.txt 文件，占用空间 4636672 字节，

经过分词得到词语 245 个，分词结果如下：

近年/来/，/国家/高度重视/文化产业/发展/，/动漫/产业/作为/创意/文化产业/的/重要/组成部分/，/迎来/了/前所未有/的/发展/机遇/。/我/校/紧紧围绕/国家/产业结构/调整/学科/专业/布局/，/于/2007 年/适时/设立/了/动画/专业/并于/当年/招生/。/同年/国家/动漫/创意/产业基地/在/青岛/揭牌/，/青岛科技大学/同时/被/授予/“/国家/动漫/创意/产业基地/人才培养/与/研发/基地/”/，/这/为/我/校/动漫/专业/的/快速/发展/搭建/了/更/广阔/的/平台/。

学校/依托/国家/动漫/创意/产业基地/，/广纳/贤良/才俊/，/投入/大量/人力/、/物力/加快/动画/专业/学科建设/，/并/取得/可喜成绩/。/先后/投资/300 余万/建立/了/定格/动画/制作/实验室/、/影视制作/实验室/、/影视/编辑/合成/实验室/、/影视广告/制作/实验室/、/手绘/动画/实验室/等/7 个/实验室/。/由/袁/志/发/院长/任/总/编剧/、/学院/动画/专业/教师/参与/前期/策划/的/《/小牛/向前/冲/》/和/《/大角/牛/梦工厂/》/登陆/央视/一套/黄金/时段/，/并/被/多家/地方/电视台/播出/，/广/受/好评/，/该剧/主人公/“/大角/牛/”/被/评为/“/首届/中国/十大/卡通/形象/”/之一/。/两部/原创/动画片/分别/荣获/第十二届/、/第十三届/全国/精神文明/建设/“/五个一工程/”/奖/。/

特色/办学/能/优化/教育资源/，/提高/办学/的/质量/和/效益/，/从而/提高/学校/的/社会/声誉/。/没有/特色/就/没有/活力/，/没有/活力/就/难以/发展/。/今年/是/“/十三五/”/的/开局/之年/，/面对/高校/综合/改革/的/新/形势/、/新/任务/，/我们/要/一如既往/地/坚持/特色/发展/的/理念/，/在/学科建设/、/专业/建设/中/合理/谋篇布局/，促进/学校/又好又快/发展/。

4.8 本章小结

本章详细说明了Lucene的架构及基于Lucene搭建简单搜索引擎系统的过程，其中重点描述了 Lucene 的索引实现和检索实现、Lucene 中文分析模块设计、双字哈希余字分组词典实现、使用基于双字哈希余字分组词典结构的最大匹配改进算法设计 Analyzer 等内容。在对 Lucene 分词器的实现阶段，首先设计 GaiJinAnalyzer 类，实现文本输入和结果输出；之后设计 GaiJinTokenizer 类，实现将文本转化为 Token 并分词。为了验证本文用基于双字哈希词长分组词典结构的正向最大匹配改进算法实现的中文分词模块效果，以该模块为基础建立了中文分词工具。

5 实验结果及性能测试

5.1 分词算法衡量标准

中文分词系统的切分准确率（P）、切分召回率（R）以及切分速度（T）是衡量系统优劣的最重要的标准。各标准的计算方式如下所示：

1. P=（分词结果中正确切分词数/分词结果词语总数量） ×100%
2. R=（分词结果中正确切分词数/正确结果词语总数量） ×100%
3. T=切分结束时间-切分开始时间

5.2 实验结果及性能比较

Lucene 自带分词器的中文分词算法分别为单字切分、二元切分以及正向最大匹配算法，其中性能最好的是采用正向最大匹配算法的 MMAnlyzer。

本文的实验分为两部分，第一部分是验证双字哈希词典机制的性能；第二部分是验证 GaiJinAnalyzer 的分词性能对 MMAnlyzer 分词性能的提升，分别从查询速度和查询精度两方面进行比较。

5.2.1 验证词典性能

同一种分词算法在不同词典机制下分词的速度差距较大，本文算法采取的是双字哈希的词典机制。该实验目的在于验证不同分词词典机制对算法速度的影响，实验方式是通过分词系统加载本文第二章介绍的基于整词二分的词典机制、基于 Trie 索引树的词典机制、基于逐字二分的词典机制和基于双字哈希的词典机制四种词典，对同样的文本进行多组分词。

实验结果显示双字哈希词典结构是分词速度较快的词典机制。其中 3 组实验结果，如图 5-1 所示。

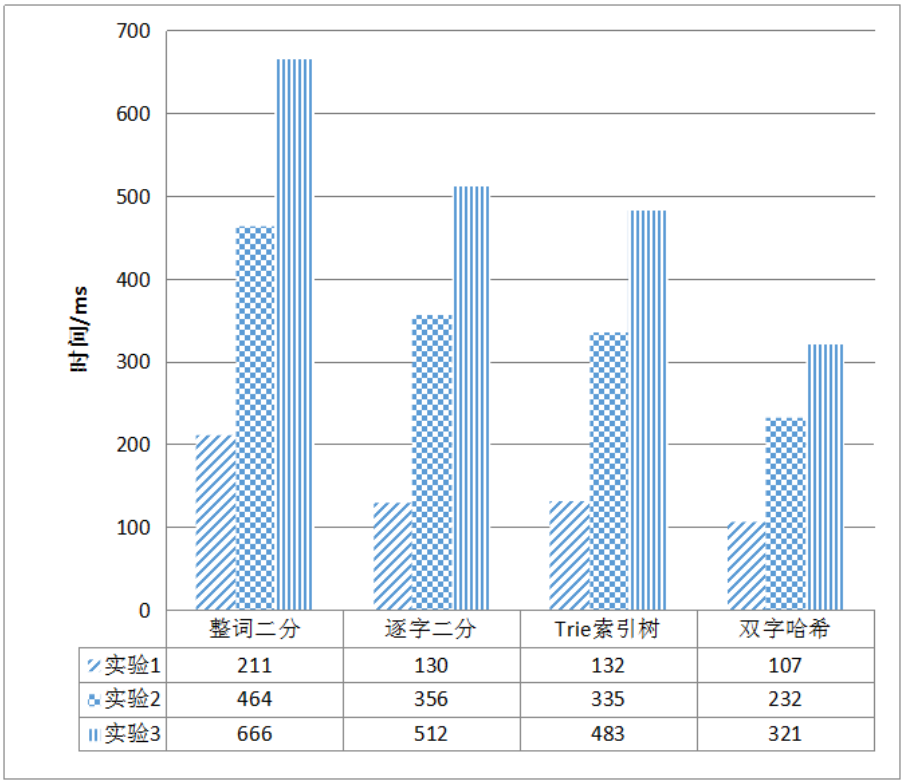

图 5-1 四种不同词典机制对分词速度影响

Fig.5-1 Compare of word segmentation speed with four dictionary

5.2.2 分词算法比较

为了比较 GaiJinAnlyzer 与 MMAnlyzer 的分词精度及算法环境适用性，本次实验采用多组不同领域的随机样本进行实验，在该次试验中两种算法均采用双字哈希的词典机制。实验结果，如表 5-1 所示。

表 5-1 实验结果比较

Tab.5-1 Comparison of experiment results

评价标准	分词算法	实验 1	实验 2	实验 3	实验 4	实验 5
准确率 P（%）	MM	91.62	91.43	92.04	92.75	91.89
	GaiJin	93.88	93.51	94.06	94.51	93.97
召回率 R（%）	MM	92.57	92.39	92.92	93.18	93.12
	GaiJin	93.85	93.62	94.03	94.50	94.05
分词速度 T（ms）	MM	1352	2512	1791	964	5143
	GaiJin	1175	2259	1582	810	4536

注：MM 表示 MMAnlyzer 所用的正向最大匹配算法；GaiJin 表示本文改进后的算法。

由表 5-1 中数据可得，使用两种算法对同样的文本进行多组分词，本文提出

的基于双字哈希词长分组词典结构的正向最大匹配改进算法的分词准确率 P 和召回率 R 都要高于正向最大匹配算法；分词速度方面本文提出算法略快。实验中最大匹配算法分词精度在 93%上下，改进算法的分词精度在 94%上下，两种算法的分词精度都远低于目前主流的分词方法。分析原因有以下几点：一是基于字符串匹配的分词算法的分词精度比较依赖词典，所用词典固有缺陷影响了分词精度；二是未登录词是影响分词精度的重要原因，本文主要工作为分词算法改进，对于未登录词并未涉及。

5.3 实验结果总结

本章针对词典性能和算法性能进行了多组实验。实验证明双字哈希结构的词典确实是较理想的词典机制。本文设计的基于双字哈希词长分组词典结构的正向最大匹配改进算法相较于传统最大匹配算法的改进，包括可以合理的选择初始匹配最大词长，以及当匹配不成功时待匹配字符串长度变为下一词长。这两点改进有效的减化了原算法的流程，在不影响分词结果的情况下明显缩短分词耗时。针对最大词长应用范围的改进是对分词精度的优化，当句子中含有长词时效果尤为明显，但此举是以增加算法复杂度为代价提高分词精度。从实验结果看，本文所提算法的分词速度略快于最大正向匹配算法。基于双字哈希词长分组词典结构的正向最大匹配改进算法能同时提高分词精度和分词速度，确实是一种较为理想的分词方法。

总结与展望

中文分词是计算机文本处理领域的重点和难点问题之一，中文分词技术的发展进步直接影响具有中文分析模块应用的实用性。本文的主要研究内容是如何通过算法改进和词典优化降低中文分词时间复杂度，目的是提高中文分词的查全率、查准率，并根据改进算法和优化词典设计高效率的中文分词模块替换 Lucene 中自带的汉语分析模块，提高基于 Lucene 搭建的搜索引擎应用性能。针对本文研究内容，对论文工作做以下总结：

1.本文对中文分词技术的分词算法（主要是是基于字符串匹配的正向最大匹配算法）、分词词典结构（主要是双字哈希的词典机制）、歧义识别和处理等内容进行了深度的研究和学习，并根据最大匹配算法的不足提出具体的算法改进途径。

2.根据总结出的中文分词算法改进途径，设计正向最大匹配算法的改进机制。通过对该机制的分析，优化传统的词典结构，本文算法更适合使用双字哈希余字按词长分组的词典结构。结合优化后的词典提出最终的改进算法——基于双字哈希词长分组词典结构的正向最大匹配算法改进算法，该算法的任一次匹配优先选择待匹配文本域内最长词，一定程度上避免了传统最大匹配算法的歧义切分。

3.目前任何一种分词算法都没有百分之百的切分准确率。对于改进后的算法机制仍可能产生的歧义问题，通过对本文提出算法的进一步分析，结合正向最大匹配+回退一字法检测歧义的算法思想，设计简单易实现的歧义处理模块。歧义处理模块对部分交集型歧义字段有较好的识别和处理功能，较大的提高了分词结果的准确率。

4.系统学习了搜索引擎的架构和搭建方法，对最热门的搜索引擎开发工具包 Lucene 的结构和基本工作原理进行研究，基于 Lucene 搭建基础的信息检索系统。本文主要研究内容为中文分词技术的优化与应用，对于搜索引擎中的网络爬虫功能，本文并未涉及。为验证本文提出算法的改进效果，利用优化后的正向最大匹配算法思想，重新设计 Lucene 中的中文分词处理模块，并实现简单的分词工具对算法进行测试和评价。

5.实验验证算法性能。首先通过采用不同词典机制对同样的语料进行分词，验证双字哈希词典的性能优势；然后通过使用不同算法对同样的语料进行分词，比较改进后的算法与正向最大匹配算法的优劣。实验结果显示，基于改进算法设计的分词模块有效提高了中文文本分析的工作效率，使其具有更高的可使用度。

本文提出了基于双字哈希词长分组词典结构的正向最大匹配算法改进算法，随着本课题研究的日渐深入，一些新的问题也随之产生：

1.一般说来最大逆向匹配算法的分词精度高于正向最大匹配算法。因本文提出的算法是基于待匹配字符串全域的最长词优先匹配，理论上不会出现传统正向最大匹配算法和最大逆向匹配算法的分词精度差别，但由于研究时间有限并未对该问题加以验证。

2.本文构建的信息检索系统不具备网络爬虫功能，结果虽然经过大量数据验证，但对于因特网上的巨量信息，本文使用的数据标本显然不够充分。且因特网上的信息样式更多变，本文提出的中文分词算法改进是否同样适用于环境更复杂的因特网还需进一步探究。

3.本文对分词歧义的探究仅限于一部分交集型歧义，对于组合型歧异没有给出解决方案；而且本文实现算法也无法对未登录词进行有效识别。日后如能将本文提出的算法与统计方法和语义标注相结合，提高对未登录词和歧义的识别处理能力，系统性能会更上一个台阶。

参考文献

[1] 张启宇，朱玲，张雅萍，中文分词算法研究综述[J]，情报探索，2008(11)：53-56

[2] 陈建英，面向中文地址的分析引擎设计及实现[D]，北京：中国科学院大学，2015

[3] Mikio Yamamoto, Kenneth Church, Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus[J], Association for Linguistics, 2000, 27(1)

[4] 高凯，郭立炜，许云峰，网络信息检索技术及搜索引擎系统开发[M]，北京，科学出版社，2010，34-35

[5] 张俊林，这就是搜索引擎核心技术详解发[M]，北京，电子工业出版社，2012，1-11.

[6] 焦玉英，信息检索进展[M]，北京，科学出版社，2003，17-106

[7] Sproat R, Gale W, Shih C, A stochastic finite-state word-segmentation algorithm for Chinese[J], Computational Linguistics,1996,22(3): 377-404

[8] 丁振国，张卓，黎靖，基于 Hash 结构的逆向最大匹配分词算法的改进[J]，计算机工程与设计，2008(12)：3208-3211+3265

[9] 周程远，朱敏，杨云，基于词典的中文分词算法研究[J]，计算机与数字工程，2009，03：68-71+87

[10] 彭焕峰，丁宋涛，一种基于全 Hash 的整词二分词典机制[J]，计算机工程，2011，21：40-42

[11] 叶继平，张桂珠，中文分词词典结构的研究与改进[J]，计算机工程与应用，2012(23)：139-142

[12] 孙茂松，左正平，汉语自动分词词典机制的实验研究[J]，中文信息学报，2000，14(1)：1-6

[13] 马志强，周长胜，杨娜，等，基于中文搜索引擎的分词词典的设计与实现[J]，铁路计算机应用，2006，15(12)：45-47

[14] 严蔚敏，吴伟民，数据结构[M]，北京，清华大学出版社，2005，210-221

[15] 李庆虎，陈玉健，孙家广，一种中文分词词典新机制——双字哈希机制[J]，中文信息学报，2003，17(4)：13-18

[16] 刘延吉，基于词典的中文分词歧义算法研究[D]，吉林：东北师范大学，2009

[17] 袁津生，李群，搜索引擎基础教程[M]，北京，清华大学出版社，2010，78-83。

[18] 纪晓阳，基于 Nutch 搜索引擎系统数据处理的中文分词技术的研究[D]，成都：成都理工大学，2014

[19] Rongyou Huang, Xinjian Zhao, A Chinese Web Page Automatic Classification System[J], Web

Information System and Mining,2010,6318:61-66

[20] Zheng Jiaheng, Zhang Jianfeng, Tan Hongye. Segmentation Strategies on Ambiguity String in Chinese Word Segmentation. Journal of Shanxi University[J],2007,30(2):163-167

[21] 张锋，樊孝忠，基于最大熵模型的交集型切分歧义消解[J]，北京理工大学学报，2005，25(7)：590-593

[22] 陈小荷，现代汉语自动分析[M]，北京，北京语言文化大学出版社，1999

[23] 闫引堂，周晓强，交集型歧义字段切分方法研究[J]， 情报学报，2000，19(6)：637-643

[24] 冯素琴，陈惠明，基于语境信息的汉语组合型歧义消歧方法[J]，中文信息学报，2007，21(6)：13-16+42

[25] Wenli Gao, The word segmentation system based on the dictionary search mechanism of three array tire index tree[J],Journal of information,2009,2:69-71

[26] 黄昌宁，赵海，中文分词十年回顾[J]，中文信息学报，2007(03)：8-19

[27] Jiansheng, Tan, Design of Chinese word segmentation dictionary based on traditional dictionary[J],Information technology,2009(5):42-45

[28] Jian Yun Nie, Unknown Word Detection and Segmentation of Chinese Using Statistical and Heuristic Knowledge[J], Communications of CLSIPS, 1995, 5(1): 47-57

[29] Luo Zhiyong, Song Rou, Disambiguation in a Modern Chinese General-Purpose Word Segmentation System[J], Journal of Computer Research and Development, 2006, 43(6): 1122-1128

[30] 张培颖，李村合，一种改进的上下文相关的歧义字段切分算法[J]，计算机系统应用，2006(5)：46-48 +14

[31] 王中立，汉语自动分词中切分歧义及处理技术[J]，许昌学院学报，2006，25(2)：118-121

[32] 李家福，张亚非，一种基于概率模型的分词系统[J]，系统仿真学报，2002，14(5)：544-550

[33] 康晨阳，基于避免交集型歧义的最大匹配算法改进的研究与实现[D]，陕西：西安电子科技大学，2012

[34] 王瑞雷，栾静，潘晓花，等，一种改进的中文分词正向最大匹配算法[J]，计算机应用与软件，2011，28(3)：195-197

[35] 莫建文，郑阳，首照宇，等，改进的基于词典的中文分词方法[J]，计算机工程与设计，2013，34(5)：1802-1807

[36] 张贤坤，李亚南，田雪，基于双哈希结构的整词二分词典机制[J]，计算机工程与设计，2014，35(11)：3956-3960

[37] 褚敬年，面向企业信息检索的中文分词系统的研究与实现[D]，辽宁：东北大学，2008

[38] 姚兴山，基于哈希算法的中文分词算法的改进[J]，图书情报工作，2008，52(6)：60-62

[39] 张庆扬，柴胜，使用二级索引的中文分词词典[J]，计算机工程与应用，2009，45(19)：

139-141

[40] 高琰，谷士文，谭立球，等，基于 Lucene 的搜索引擎设计与实现[J]，微机发展，2004，14(10)：42-44

[41] 刘挺，秦兵，张宇，等，信息检索系统导论[M]，北京，机械工业出版社，2008，161-184.

[42] 邱哲，符滔滔，开发自己的搜索引擎[M]，北京，人民邮电出版社，2007，21-258

[43] 罗刚，解密搜索引擎技术实战 Lucene&Java 精华版第二版[M]，北京，电子工业出版社，2014

致 谢

奋斗的日子总是过的特别的快，三年研究生求学生涯转瞬即逝，在此我想对三年里给予我无私教导和帮助的导师、同学朋友们以及我的家人表示感谢。

首先，特别要感谢我的导师***教授，本篇论文的工作是在刘老师的细心指导下完成。在实验室科研期间，刘老师不断的提点我，为我理清思路指明方向。在我遇到科研瓶颈时，刘老师总是更多的给予我鼓励，默默的支持，她对科研的严谨认真和令人尊敬的行事作风让我感悟很多道理，受益匪浅。

其次，感谢曾给予我无私帮助的学院老师们。感谢***在我迷茫找不到方向时，为我理性分析形势现状，助我抚平心态稳步前行；感谢***老师、***老师、***老师以及***老师，在我多年学习生涯中，给我学习、工作、生活等各个方面的教诲、帮助和指导。

另外，感谢一直陪伴的同学朋友们。感谢***师兄、***师姐、***师姐、***师妹和***师弟，他们热心的和我探讨研究方向，主动与我分享他们的学习经验和技巧，在论文撰写过程中，他们在查阅文献、测试数据和研究成果分析等方面给予了很大帮助；感谢朝夕相伴的同窗好友，感谢生活中他们一直的陪伴和无私的照顾，感谢学习中有他们同我互相鼓励、共同进步，感谢工作中有他们对我的默默扶持和付出。

最后，感谢我的家人，尤其是我的父母。感谢他们多年来对我的教育和辛苦付出，无论是精神上还是物质上，感谢他们为我所做的一切，感谢他们成为我身后强有力的支撑，让我能够在就读期间无忧无虑的专心于学业研究。

攻读学位期间发表的学术论文

[1] ***，***，基于汉语框架的语义标注方法[J]，计算机科学

[2] ***，***，数据结构教学的研究与改革实践[J]，电脑知识与技术

独创性声明

本人声明所呈交的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，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含本人已用于其他学位申请的论文或成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：

日　　期：　　年　月　日

关于论文使用授权的说明

本学位论文作者完全了解青岛科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权学校可以将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、汇编学位论文。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为青岛科技大学。(保密的学位论文在解密后适用本授权书)

本学位论文属于：

保密 □，在　　年解密后适用于本声明。

不保密□。

(请在以上方框内打“ √ ”)

本人签名：　　日期：　　年　月　日

导师签名：　　日期：　　年　月　日