摘 要

在快速进步的信息时代，人们在快速的生活节奏中能以很简单的方式用搜索引擎能够在茫茫互联网中找到较为准确的用户所需要的信息。在最近几年中，搜索引擎正在飞速发展。同时桌面搜索引擎也有了较为快速的发展。但在为用户提供个性化的服务方面还没有达到用户的要求，它们仅仅是找到用户的文件并作一些简单处理，在用户文件中所蕴含的知识的分析、聚合及潜在的知识的挖掘、搜索结果人性化展示等方面的功能有所欠缺。这几点正是本文的重点研究对象。

本文的工作源于四川省青年软件创新基金资助项目—U-CLASS，主要提供用户知识挖掘及管理、个性化搜索、用户兴趣学习及用户个性化服务等。本文主要工作如下：

（1） 从搜索引擎的基本概念及历史出发，分析了桌面（文件）搜索引擎的发展及长短处。同时结合 U-CLASS，提出本文的研究方向；

（2） 从 U-CLASS 项目的实际需求入手，研究了基于模糊信息处理的个性化分布式文件搜索引擎的基础理论，提出了基于并性模糊模糊蚂蚁的聚类算法、基于 Double-Array Trie 的中文分词算法及全新的用户模型等；在从理论角度研究相关模型和算法的同时，也采用了大量的数据及测试工具，对相关模型和算法作了较为详细的测试，进而验证算法和模型的准确性与高效性；

（3） 从设计和实现的角度作了较为详细的工作。主要采用自顶向下的方式，从系统架构到各个模块，再到各个软件包的设计，整个系统层次分明、结构清晰。主要包括：整个系统的架构、聚类算法的设计与实现、中文分词算法的实现、全新的用户模型的设计与实现及并发文件蜘蛛（Spider）的设计与实现等；

（4） 采用多个工具进行了相关测试，包括：功能测试、压力测试等；同时采用 JProfiler 对整个系统进行了性能分析和优化；

（5） 在总结本文目前的完成的工作的情况下，对后续工作也做了简明分析及展望。

关键词：个性化，搜索引擎，模糊蚂蚁，文本聚类，用户兴趣挖掘

ABSTRACT

In the fast developing age of Information Technology, People whose life is fast-paced can obtain precise useful information with very simple methods in the internet. In recent years, search engines are developing rapidly. The desktop search engines have also been developed rapidly but personalized services have not satisfied users' requirements. They are simply locating documents. Analyses of knowledge contained in users' documents, mining of potential knowledge and search results exhibition with humanization are their defects. These points are the focuses of this thesis.

The thesis stems from one of the Sichuan Provincial Youth Fund for software innovation projects —U-CLASS. The platform constructed in this thesis intends to provide users with knowledge mining and management, personalized search, users' interests learning and personalized service and etc. The work of this thesis is described as the following:

(1) From perspective of basic concepts and history of search engine, the development and advantages of desktop (document) search engine are analyzed. The research direction is proposed with U-CLASS;

(2) Clustering based on parallel fuzzy ants and some other algorithms are proposed; A large amount of data and some testing tools are used to assure accuracy and high efficiency of models and algorithms;

(3) More detailed work of design and implementation is done; top-down approach is mainly used; from the perspective of system structure of each module and package design, the thesis has a clear structure. The main module or algorithms are as the following: the clustering algorithm design and implementation and etc.;

(4) A number of tools related to software testing are used to do functional testing, pressure testing and etc.; JProfiler is used for performance analysis and optimization of the whole system;

(5) The follow-up work to be done is concisely analyzed and prospects are presented while the work accomplished in this thesis is summarized.

Keyword: personalization, search engine, fuzzy ant, text clustering, user profile mining

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。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作了明确的说明并表示谢意。

签名：冯周 日期：2007 年 05 月 13 日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、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。本人授权电子科技大学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。

（保密的学位论文在解密后应遵守此规定）

签名：冯周 导师签名：张扬

日期：2007 年 05 月 13 日

第一章 绪论

1.1 搜索引擎的基本概念

信息的生产、传播、搜集与查询是人类最基本的活动之一。考虑以文字为载体的信息，传统上有图书馆、相应的编目体系和专业人员帮助我们很快找到所需的信息，其粒度通常是“书”或者“文章”。随着计算机与信息技术的发展，有了信息检索（Information Retrieval，IR）学科领域，有了关于图书或者文献的全文检索系统，使我们能很方便地在“关键词”的粒度上得到相关的信息。

搜索引擎，是指因特网上专门提供查询服务的一类网站，是一种利用网络自动搜索技术，对因特网各种信息资源分门别类地进行标引建库，能够对检索者提出的各种检索查询做出响应的强有力的检索工具。它是为满足人们对网络信息搜索需求应运而生的网络工具，既是互联网信息查询的导航针，也是沟通用户与网络信息的重要桥梁[1]。从使用者的角度看，这种软件系统提供一个网页界面，让他通过浏览器提交一个词语或者短语，然后返回一个可能和用户输入内容相关的信息列表。此列表通常以分页且每页中包含有限数量条目的形式来表现。每一条目代表一个网页，至少包含三个元素：

- 标题

  网页内容的标题性概括。获得标题的方式不唯一，其中最简单的方式就是从网页的头信息中提取的内容。它是否能准确反映该网页的内容，取决于网页设计者对标题的概括。

- URL

  该网页的原始地址。该地址可能反映出该网页的一些特点。如：*http://www.chengdu.gov.cn* 为政府网站，且为中国政府的某个子网站，其内容主要为政治内容，其权威性高于其它诸如 *http://www.unamed.com* 的网站。同时根据网址的后缀可大概判断该网页的类型，比如：org 表示组织，com 表示公司等。

- 摘要

  网页内容的简要概括。最简单的一种方式就是将网页内容的头截取部分数据作为摘要。

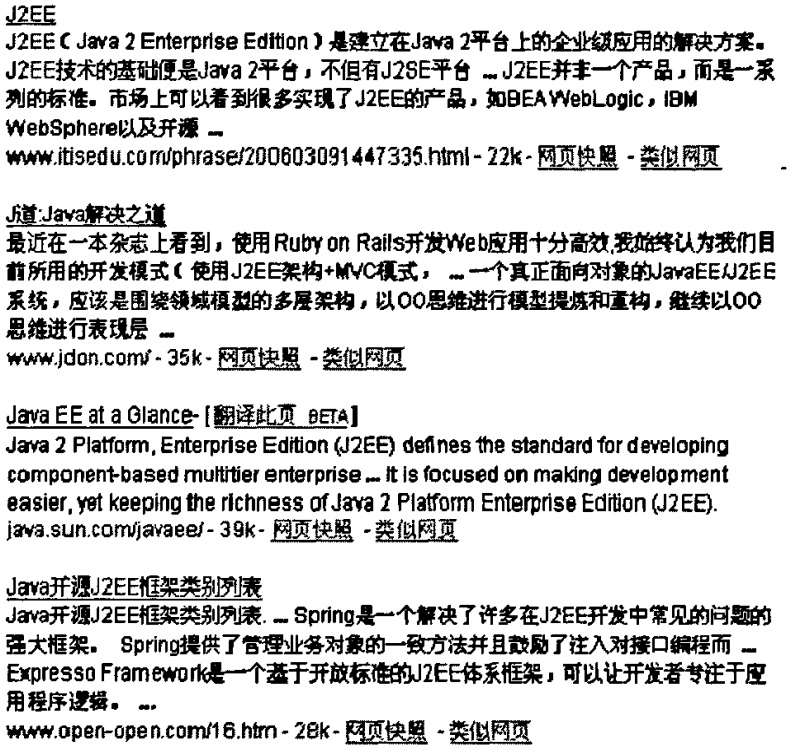
用户通过浏览网页的以上元素，进而判断是否真正包含其所需的信息。若用户根据上述三个因素判断出某个网页可能会符合其要求，则用户可点击相应的 URL 访问该网页。图 1-1 是 2007 年 3 月 23 日在搜索引擎 Google[4]上，以“J2EE”作为关键字搜索返回给用户的部分信息列表。列表中每一条目所含内容比上述要丰富些，但上述三个元素依旧是核心。虽然关键字相同，不同的搜索用户的所想知道的知识可能不同。如果用户只是想了解一下 J2EE 为何物，则第三条可能更符合用户的要求；如果用户是一个开源框架的爱好者，则 *http://www.open-open.com/16.htm* 为 URL 的网页可能能够提供一些他所需要的信息。

由于搜索引擎因为数据量巨大，且需要对用户的搜索做出快速、稳定的反馈等特点，因而具有以下两点需要注意。其一，当用户提交查询时，搜索引擎并不是实时从整个互联网中搜索，发现与用户所提交的关键字相关的网页，从而形成列表反馈给用户；而是事先搜索引擎使用爬虫(Spider)抓取了一定范围内的网页，以索引等形式存放在系统中，此时的搜索只是在系统内部进行而已。其二，当用户发现在反馈的网页中某个网页符合其要求，从而点击该网页的 URL，获得全文的时候，用户此时访问的则是网页的原始出处。于是，搜索引擎并不保证用户在返回结果列表中所看到的标题和摘要内容与他点击 URL 所看到的内容一致（图 1-1 也是如此!），甚至不能保证改网页的存在。这也是搜索引擎和传统信息检索系统的一个重要区别。这种区别源于 Web 信息随时可能在更新，而搜索引擎未能及时同步。为了弥补这个差别，现代搜索引擎都保存网页搜集过程中得到的网页全文，并在返回结果列表中提供“网页快照”或“历史网页”链接，保证让用户能看到和摘要信息一致的内容。同时，搜索引擎也会采取一定算法及技术来及时更新保存在系统中网页，这也是现代搜索引擎性能评价的一个重要标准。

从图 1-1 的搜索例子可以看出，搜索引擎提供信息查询服务仅是以查询词为基础的。虽然不同背景的人可能会提交相同的关键词，但他们可能关心与这个查询词相关的不同层面的信息。但搜索引擎通常不知道用户背景，因而搜索引擎既要争取不漏掉任何相关的信息，还要争取将那些最受关注的信息排在列表的前面。这也就是对搜索引擎的根本要求[2]。Google 采用了 PageRank[3]等技术来解决网页排名问题，它也是 Google 成功的重要技术因素之一。除此以外，考虑到搜索引擎的应用环境是 Web，因此对大量并发用户查询的响应性能也是一个不能忽略的方面。

正如上述问题，搜索引擎对用户的情况是未知的。对于不同的用户在关键字相同的情况下，搜索引擎会反馈相同的结果。尽管反馈的结果是经过优化选择的，

但可能离用户的需求有较大的差别。根据《基于客户体验的搜索引擎相关性改进报告》的结果，调查者发现：一是无论是标题、文档还是按查询关键词分类来看，三家搜索引擎的质量不分仲伯；二是三家搜索引擎的结果重合率很低，但没有一家搜索引擎能够满足用户的完整搜索需求[6]。进而对用户提供个性化服务、为用户提供更加精确的服务是未来搜索引擎的发展方向之一，也是本文研究的方面之一。


J2EE
J2EE（Java 2 Enterprise Edition）是建立在Java 2平台上的企业级应用的解决方案。J2EE技术的基础便是Java 2平台，不但有J2SE平台 ... J2EE并非一个产品，而是一系列的标准。市场上可以看到很多实现了J2EE的产品，如BEA WebLogic，IBM WebSphere以及开源 ...
www.itisedu.com/phrase/20060309144733.html - 22k - 网页快照 - 类似网页

J道:Java解决之道
最近在一本杂志上看到，使用Ruby on Rails开发Web应用十分高效,我始终认为我们目前所用的开发模式（使用J2EE架构+MVC模式， ...一个真正面向对象的JavaEE/J2EE系统，应该是围绕领域模型的多层架构，以OO思维进行模型提炼和重构，继续以OO思维进行表现层 ...
www.jdon.com/ - 35k - 网页快照 - 类似网页

Java EE at a Glance - [翻译此页 BETA]
Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing component-based multitier enterprise ... It is focused on making development easier, yet keeping the richness of Java 2 Platform Enterprise Edition (J2EE).
java.sun.com/javaee/ - 39k - 网页快照 - 类似网页

Java开源J2EE框架类别列表
Java开源J2EE框架类别列表. ... Spring是一个解决了许多在J2EE开发中常见的问题的强大框架。 Spring提供了管理业务对象的一致方法并且鼓励了注入对接口编程而 ... Expresso Framework是一个基于开放标准的J2EE体系框架，可以让开发者专注于应用程序逻辑。 ...
www.open-open.com/16.htm - 28k - 网页快照 - 类似网页

图 1-1 以关键字“J2EE”在 Google 搜索到的前四条结果

1.2 搜索引擎的发展历史

在互联网发展初期，网站相对较少，信息查找比较容易。然而伴随互联网爆炸性的发展，普通网络用户想找到所需的资料简直如同大海捞针，这时为满足大众信息检索需求的专业搜索网站便应运而生了。

现代意义上的搜索引擎的祖先，是 1990 年由蒙特利尔大学学生 Alan Emtage 发明的 Archie。虽然当时 World Wide Web 还未出现，但网络中文件传输还是相当频繁的，而且由于大量的文件散布在各个分散的 FTP 主机中，查询起来非常

不便，因此 Alan Emtage 想到了开发一个可以以文件名查找文件的系统，于是便有了 Archie。

Archie 工作原理与现在的搜索引擎已经很接近，它依靠脚本程序自动搜索网上的文件，然后对有关信息进行索引，供使用者以一定的表达式查询。由于 Archie 深受用户欢迎，受其启发，美国内华达 System Computing Services 大学于 1993 年开发了另一个与之非常相似的搜索工具，不过此时的搜索工具除了索引文件外，已能检索网页。

当时，“机器人”一词在编程者中十分流行。电脑“机器人”（Computer Robot）是指某个能以人类无法达到的速度不间断地执行某项任务的软件程序。由于专门用于检索信息的“机器人”程序象蜘蛛一样在网络间爬来爬去，因此，搜索引擎的”机器人”程序就被称为”蜘蛛”程序。

世界上第一个用于监测互联网发展规模的“机器人”程序是 Matthew Gray 开发的 World wide Web Wanderer。刚开始它只用来统计互联网上的服务器数量，后来则发展为能够检索网站域名。

与 Wanderer 相对应，Martin Koster 于 1993 年 10 月创建了 ALIWEB，它是 Archie 的 HTTP 版本。ALIWEB 不使用“机器人”程序，而是靠网站主动提交信息来建立自己的链接索引，类似于现在我们熟知的 Yahoo!。

随着互联网的迅速发展，使得检索所有新出现的网页变得越来越困难，因此，在 Matthew Gray 的 Wanderer 基础上，一些编程者将传统的“蜘蛛”程序工作原理作了些改进。其设想是，既然所有网页都可能有连向其他网站的链接，那么从跟踪一个网站的链接开始，就有可能检索整个互联网。到 1993 年底，一些基于此原理的搜索引擎开始纷纷涌现，其中以 JumpStation、The World Wide Web Worm（Goto 的前身，也就是今天 Overture），和 Repository-Based Software Engineering (RBSE) spider 最负盛名。

然而 JumpStation 和 WWW Worm 只是以搜索工具在数据库中找到匹配信息的先后次序排列搜索结果，因此毫无信息关联度可言。而 RBSE 是第一个在搜索结果排列中引入关键字串匹配程度概念的引擎。

最早现代意义上的搜索引擎出现于 1994 年 7 月。当时 Michael Mauldin 将 John Leavitt 的蜘蛛程序接入到其索引程序中，创建了大家现在熟知的 Lycos。同年 4 月，斯坦福（Stanford）大学的两名博士生，David Filo 和美籍华人杨致远（Gerry Yang）共同创办了超级目录索引 Yahoo!，并成功地使搜索引擎的概念深入人心。从此搜索引擎进入了高速发展时期。目前，互联网上有名有姓的搜索引擎已达数

百家，其检索的信息量也与从前不可同日而语。比如最近风头正劲的 Google，其数据库中存放的网页已达 80 亿之巨！

随着互联网规模的急剧膨胀，一家搜索引擎光靠自己单打独斗已无法适应目前的市场状况，因此现在搜索引擎之间开始出现了分工协作，并有了专业的搜索引擎技术和搜索数据库服务提供商。象国外的 Inktomi（已被 Yahoo!收购），它本身并不是直接面向用户的搜索引擎，但向包括 Overture（原 GoTo，已被 Yahoo! 收购）、LookSmart、MSN、HotBot 等在内的其他搜索引擎提供全文网页搜索服务[5]。

与国外搜索引擎相比，国内搜索引擎起步较晚。由于中文信息处理较英文信息更为复杂，中文搜索引擎的研制也是最近几年刚刚兴起，但其发展却十分迅速。据“中文导航及搜索引擎指南”统计，目前国内较完备的搜索引擎有 898 个，中文繁体搜索引擎 18 个，这些还不包括一些开展信息服务的图书馆等其他信息服务网站。其中，用户常用的搜索引擎有 Google、百度、新浪、搜狐、雅虎中文、天网等。我国港台地区拥有更为普及的网络基础，台湾地区采用 Big5 编码的中文信息也十分丰富，并有一批著名的搜索引擎，如番薯藤等；香港较优秀的搜索引擎有茉莉之窗等[1]。近年来国外一些网络企业也开始瞄准中国互联网这一潜在的巨大市场，先后推出了一些中文版的搜索引擎，如 Google 推出的中文版–谷歌等。

1.3 桌面(文件)搜索的发展形势

1.3.1 现状

随着各大搜索引擎的快速发展及互联网的不断普及，搜索引擎所带来的商业价值及给人类生活所带来的正面影响越来越得到人们的肯定。同时，随着 PC 的普及以及 PC 存储设备等方面的发展，人们在 PC 上存储的信息越来越多，快速准确的定位信息已经成为 PC 用户的一个重要需求。因此，各大搜索引擎的战火开始从网络延伸到 PC 桌面，纷纷推出各具特色的桌面搜索（以下简称 DS）工具。比如：Google 的 Google Desktop Search， 百度硬盘搜索，Yahoo!的 Desktop Search 等。

1.3.2 优劣势分析

据文献[7][8]中的分析报告，如图 1-2 所示，各大主流搜索引擎及一些小公司所

推出的DS各具特点，在不同的方面占有不同的优势。如：在界面和易用性方面，百度和Google DS(以下简称GDS)则表现相对突出。同时，GDS拥有灵活的插件机制，使得它的扩展性和灵活性较其它DS有突出的表现。但GDS在索引文件管理方面就差强人意了。在文件类型支持方面，由于部分DS来自“番邦”，在本地化方面与本地DS相比就稍差一些。如：中搜的网络猪则支持对Foxmail的邮件进行索引、搜索；而Yahoo！DS在各个方面都有优秀的表现，但在对中文的支持太差，这一重要缺陷导致了Yahoo! DS目前较难占领国内的PC桌面搜索市场。

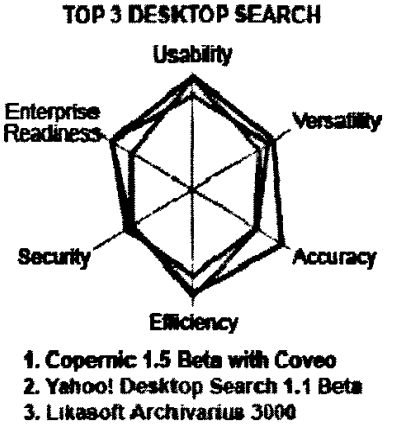


图1-2 主流桌面搜索的评测结果[8]

据上所述，虽然各大DS在不同方面都有不俗的表现，但它们在为用户提供个性化的服务方面还没有达到用户的要求，大多数DS仅仅是一个搜索引擎的桌面简化版。它们仅仅是把用户的文件找出来并作一些简单处理，走出的仅是搜索的第一步。在用户文件中所蕴含的知识的分析、聚合及潜在的知识的挖掘、搜索结果人性化展示等方面的功能有所欠缺。这几点是本文的研究的部分重点。

1.4 搜索引擎的发展趋势

搜索引擎在经过10多年的飞速发展，在多方面已经影响到人类的日常生活及工作。但目前搜索引擎所提供的功能越来越难以满足人类的需求。主流的搜索引擎公司指出增强的个性化搜索会是最近发展的一个重要的目标。而改善多媒体搜索、本地化、问答技术、可视化搜索结果展示及除PC之外的其它设备，如手机等的搜索也是未来的发展方向[9]。

(1)个性化搜索

它是搜索引擎的一个重要优势但也是目前搜索引擎缺陷之一。搜索引擎能够搜集用户的个人信息、记录用户的搜索历史，记录用户的点击历史，进而过滤搜索结果，使用户得到更有用的搜索结果；但由于一些用户对个人隐私存在疑虑，这是推行个性化技术的障碍之一。因此，隐式的个性化和显式的个性化搜索相结合能够一定程度上解决隐私问题；在本文中，即采用此方式为用户提供个性化服务；

(2)解析用户意图

根据用户的历史搜索记录、其它人的历史搜索及用户当前的查询解析用户的搜索目的，进而更准确的分析用户的搜索意图。同时，也可加入其它用户的相关信息，搜索的结果就可能更趋近用户需要的信息。在本文中，采用模糊方法，动态处理用户的搜索记录、其它用户的搜索记录、当前查询已经公用用户兴趣知识库，进而更好地解析用户的搜索意图；

(3)搜索本地化

相同关键字对于不同地区的用户可能会需要不同的结果。如：搜索“天府广场”、”锦江宾馆”可能会有很多搜索结果，但用户可能会对成都的天府广场或锦江宾馆的地址或者收费情况更感兴趣，搜索引擎未对用户的位置(本地化)作分析，则可能会返回所有结果而未过滤。因此，搜索本地化是搜索引擎的重要发展方向之一。如：Google、百度的地图等。在本文中，采用动态搜索查询用户的 IP 等方式，为用户提供本地化的搜索结果；

(4)搜索引擎的运行平台的多样化

PC 由于大小及其它方面的限制，移动性能欠佳。同时移动平台的不断发展，用户对搜索引擎的移动性能要求越来越高。因此，适应移动平台的需要和限制也是搜索引擎的发展重点。在本文中，拟采用移动客户端（手机）支持移动查询，但由于移动客户端属于本文所在项目组中的另外一个项目，所以在本文中省略了其详细信息。

1.5 课题的来源及研究背景

本课题是 U-CLASS 项目的一部分，为 U-CLASS 提供个性化知识管理。关于 U-CLASS 会在后文第二章详细介绍。本课题拟为 U-CLASS 平台提供核心搜索功能及用户个性化知识管理，主要提供以下功能：

- 用户知识挖掘及管理
- 个性化搜索
- 用户兴趣学习
- 用户个性化服务

1.6 本课题相关研究情况

从前文可知，搜索引擎及桌面文件搜索引擎都有较大的发展，且在市场上都存在各种不同的产品。但本文所指的基于模糊信息处理的个性化分布式文件搜索引擎无论是文献中还是市场中都未有涉及。而本系统的核心算法，包括：基于并行模糊蚂蚁的文本聚类算法、基于 Double-Array Trie 的中文分词算法及用户模型挖掘算法等，无论从理论角度，还是从实际应用角度都具有一定的价值。同时，本系统可与 U-CLASS 系统无缝集成，从而使 U-CLASS 的市场应用前景更为广阔。

几乎任何科研工作都是建立在前人的工作之上、“站在巨人的肩膀上”。本文的工作也是建立在无数前人的科研成果之上。文献[1][2][3][4][5][6][7][8][9]对搜索引擎及桌面搜索的知识作了介绍，同时对用户行为作了分析，并在搜索引擎的发展方向作了一定的预测；文献[11][12]不但对模糊数学作了框架性的介绍，同时也全面地讲解了模糊数学的理论及在相关领域的应用；而文献[10] [13] [14] [15] [16] [17]对模糊蚂蚁、模糊推理及串行模糊蚂蚁文本聚类算法等作了较为详细的介绍。在文献[2][18][19] [20] [21] [22] [23]对 Double-array trie、基于 Double-array trie 的字典查询方法及中文分词等知识作了介绍及实现等等；同时，在本系统的实现上参考了大量关于 J2EE 的文献，包括[33] [34] [35] [36] [37] [38] [39] [40] [41]等。其中包括纸资资料及电子文档，数量众多，所以部分省略。这些参考文献从多个角度阐述了 J2EE 的相关知识，包括 Lucene、Spring、线程池等；由于本文需要对多种文件格式的内容进行分析，当然也包括了对多种文件格式说明的参考，但由于格式众多且不是本文的重点，在此省略。

1.7 本课题工作及本文内容安排

本文的工作源于四川省青年软件创新基金资助项目—U-CLASS，拟为 U-CLASS 平台提供核心搜索功能及用户个性化知识管理，主要提供用户知识挖掘及管理、个性化搜索、用户兴趣学习及用户个性化服务等。本文主要工作如下：

（1） 从搜索引擎的基本概念及历史出发，分析了桌面（文件）搜索引擎的发展及长短处。同时结合 U-CLASS，提出本文的研究方向；

（2） 从 U-CLASS 项目的实际需求入手，研究了基于模糊信息处理的个性化分布式文件搜索引擎的基础理论，包括：模糊理论、模糊推理、模糊蚂蚁、文本聚类、Double-Array Trie、中文分词、用户模型等；在从理论角度研究相关模型和算法的同时，也采用了大量的测试数据及测试工具，对相关模型和算法作了较为详细的测试，进而验证算法和模型的准确性与高效性；

（3） 从设计和实现的角度作了较为详细的工作。主要采用自顶向下的方式，从系统架构到各个模块，再到各个软件包的设计，整个系统层次分明、结构清晰。其中包括：整个系统的架构；聚类算法的设计与实现；中文分词算法的实现；全新的用户模型的设计与实现；并发文件蜘蛛（Spider）的设计与实现等；

（4） 从整个系统的角度，采用多个工具，进行了相关测试，包括：功能测试、压力测试等；同时采用 JProfiler 对整个系统进行了性能分析和优化；

（5） 在总结本文目前的完成的工作的情况下，对后续工作也做了展望和简明的分析。

本文内容安排如下：

第一章为绪论。从搜索引擎的基本概念及历史出发，分析了桌面（文件）搜索引擎的发展及长短处。同时结合 U-CLASS，提出本文的研究方向；

第二章为需求分析。从实际应用角度出发，对 U-CLASS 在搜索方面的需求做了分析。但由于不是本文的重点，在内容的阐述上，则较为简略；

第三章为理论分析。这一章是本文的核心，从理论的角度对模糊理论、模糊推理、模糊蚂蚁、文本聚类、Double-Array Trie、中文分词、用户模型等作了一定深度的研究，并作了较为详细的测试来验证模型及算法的准确性及高效性；

第四章为系统设计及实现。主要采用自顶向下的方式，从系统架构到各个模块，再到各个软件包的设计，整个系统层次分明、结构清晰。在本章对聚类算法的设计与实现、中文分词算法的实现、全新的用户模型的设计与实现及并发文件蜘蛛（Spider）等核心算法及模型作了实现；

第五章为总结与展望。本章对本文所做的工作做了总结，并对未来的工作作了展望。

第二章　需求分析

本章主要对首先简要分析U-CLASS，然后再对U-CLASS在搜索方面的需求作简略分析，为本文的后续章节中的理论分析与系统实现等做好前提准备。下面首先对U-CLASS作简要分析。

2.1 U-CLASS

U-CLASS 通用电子课件教学管理平台是*四川省青年软件创新基金资助项目*之一，目的是为解决教育信息化日益增长的大容量数据存储、移动共享的迫切需求而产生的；针对学校对数字化教学的需要，U-CLASS提供电子课件的组织、存储、传输、发布服务。通过互联网（教育网），让老师和学生在任何地方存储和访问教育资源，实现真正意义了上的3A服务（即Anytime、Anywhere、Anyone）。它主要提供以下服务：

（1）大容量数据的高速传输

在传输大文件时尤其明显。OETP协议采用字节流的方式进行数据传输，使传输的数据量比电子数据的字符流方式减少很多。同时，优化后的程序能够节省大量的编解码数据时间。此外，它采用了分布式存储结构，用户只需发送一个短信息或一个很小的报文就可以将数据重发或转发给同一个人或不同的人。整个数据发送过程只需要几秒甚至更短的时间(无论重发的数据有多大)。

（2）支持断点续传功能

无论是发送还是接收，都可以在异常中断后继续未完成的传输，而不是从头开始。所有数据传输过程是自动完成的。与E-mail的排队机制相比，它将节省用户大量的时间和操作。

（3）拒绝垃圾数据

垃圾数据大量浪费了用户的时间、精力和金钱，有时还会导致用户的使用空间爆炸。用U-CLASS就可以避免这种状况的发生，因为垃圾数据发送者发信时，数据不是发送到用户的邮箱，而是发到他自己的邮箱，用户收到的只是一个短消息，如果用户发现这是一个垃圾数据，他只需把这个短消息删除即可，就没有必要再接收垃圾数据了。

（4）具有挂号信功能

对方收到信件后，可以进行数据确认，这下你就不用担心对方是否收到数据了，它克服了传统 E-Mail 只管发不管收的状况。

（5）更高的安全性

U-CLASS 采用了独特的加密、签名技术，任何人都无法获知你的密码，它在管理上采用了国际上公认的不可破解方式，可为用户提供军事级的安全性和保密性，保证传输内容的安全，避免伪造和假冒信件的情况。另外，由于 U-CLASS 并不将数据直接发到用户的邮箱，且为用户提供黑名单功能，从而能够彻底杜绝垃圾数据现象。在多重加密措施的保护下，该系统能提供更好的安全性。

（6）兼容目前的 Email

U-CLASS 可以用向 Email 发信，收发 Email 数据，支持各种 Email 编码，二者的操作完全相同，U-CLASS 会自动识别。

（7）个性化知识库搜索引擎

（8）基于 P2P 技术的大容量课件发布模块

（9）分布式注册模块

（10）分布式存储模块

2.2 基于 U-CLASS 的扩展

U-CLASS 主要是针对电子课件教学管理，但由于其良好的伸缩性、扩展性等，使其很容易扩展成为通用的、易用的、功能强大的在线储存、共享和资料保护、备份的在线服务。进而可以满足用户随时随地存储并管理数据的需求，并运用先进的互联网技术，实现 Internet 及局域网上的“网络邻居”，从而快速方便地交流和共享信息，实现真正意义上“天涯若比邻”。在本文以下内容所指的 U-CLASS 均是基于 U-CLASS 的扩展，在后文不再提及。

2.3 必要性分析

2.3.1 迅速定位文件

因 U-CLASS 为用户提供的基本功能之一是大容量在线存储，所以用户可以存

放大量文件在 U-CLASS 系统中。根据 U-CLASS 的实际运行过程中所得的数据，大多数用户存储的文件数量较多。在文件大小方面，服从以中等文件大小为均值的正态分布。因此，用户要从众多的文件中找到自己需要的文件是相当繁琐的，既耗时又耗费精力，且效果欠佳。因而为用户提供各类查询接口进而迅速定位文件是 U-CLASS 非常重要的功能之一，也是个性化服务的重要特点之一。

2.3.2 用户知识挖掘

因用户在 U-CLASS 存放了相当数量的文件，其中包括各类文档、音视频文件、图片等，在其中蕴含了丰富的知识。既然用户存放了文件，那么用户对其中的文件可能存在一定的兴趣。然后对于数量庞大的文件以及人类遗忘的生理特点，用户可能会对其存放的文件中所包含的知识逐渐遗忘，从而造成了一定的浪费，同时用户也失去了对文件中蕴含的知识学习的适当机会。因此，挖掘用户所存放的文件中蕴含的知识，并能快速、准确地展现给用户，并以一定的方式使用户能够更快速的学习知识是 U-CLASS 的重要功能之一，也是 U-CLASS 的亮点之一。

2.3.3 用户兴趣学习

U-CLASS 是功能强大的在线服务系统，其最重要的服务对象就是系统的用户。为用户提供个性化的优秀服务是系统最重要的目的之一。同时，准确、及时是系统最重要的服务特点。因而，准确的把握用户的兴趣、掌握用户的个性化需求动向是实现个性化服务的必要条件。为了达到上述目的，U-CLASS 会从用户的个人信息、行为、用户存放的文件中智能学习用户的兴趣。在掌握了用户的兴趣之后，系统可以为用户提供各类个性化服务，使服务更趋人性化。同时，随着用户和系统交付的进行，系统会不断的学习用户的兴趣，使得系统越来越能理解用户，从而提高服务质量，也提高了用户对系统的参与感与粘度。因为，在用户在注册时，可选择性提供少量个人信息，且不涉及用户比较敏感的数据，所以在用户隐私方面 U-CLASS 是比较安全的。

2.3.4 用户个性化服务

个性化服务是当前众多行业的发展方向之一，为用户提供贴心、及时、准确的信息或实体服务，是赢得客户的最重要的方式之一。

（1）个性化搜索

当用户要从在U-CLASS中存放的大量文件中找到其需要的文件，需要通过一定的方式进行查询。但由于一般的查询都是基于关键字且大多数用户并非专业人员，一般难以形成准确的关键字表达式；同时由于知识本身的模糊性，仅仅基于关键字的查询难以满足用户的需求；而在搜索结果排序上，传统PageRank等排序技术并未考虑单个用户的需求，所以不适合直接应用在U-CLASS中。因此，U-CLASS需要以相关技术为基础，改变传统技术，从而更加准确的定位文件。

由于搜索条件的不确定性及数量庞大的文件，有可能返回大量的搜索结果。在传统的分页方式下，用户并不能从总体上了解搜索结果的类别，从而降低了定位文件的效率。因此，在个性化搜索的另外一个方面即是自动将搜索结果分类（聚类），并按照一定的顺序排列类别，使得用户在总览全貌的情况下迅速找到自己所需要的文件。

（2）个性化推荐

U-CLASS的用户是既是一个网络存储平台，也是一个网络资源分享的平台。在此平台上，用户可以将其资源共享给平台的用户或者平台外的用户。而用户共享的资源是用户的重要知识来源之一，也能够让平台上的资源能够得到更好的利用，从而使平台用户在管理自己的知识的前提下，能够更容易获得更多与自己知识需求相关的知识。因此，U-CLASS会根据用户的搜索条件、下载、共享等方面的信息，为用户提供与用户兴趣相关的共享资源。

在U-CLASS中，用户的个性化服务是以用户搜索条件、搜索结果、文件下载、文件共享、用户知识挖掘、用户兴趣学习等方面为基础，并结合U-CLASS的公共资源而形成一套完整的个性化搜索服务。同时，它将与U-CLASS的其它组件紧密结合，从而使U-CLASS更加完善、友好。

2.4 功能分析

在本节中，将对U-CLASS的用户需求及系统需求作分析。用户需求是指用户在使用U-CLASS平台时，平台为其所提供的功能。这些功能不但能够使用户正常使用平台，更让用户有很好的体验。值得注意的是，本文的重点是理论分析，因此下文仅仅提出要点，不做详细的分析。

2.4.1 用户需求

用户的需求主要包括以下几点：

（1） 根据关键字搜索文件；

（2） 选择文件类型搜索文件；

（3） 根据文件的创建日期范围搜索文件；

（4） 根据文件的大小搜索文件；

（5） 指定文件路径搜索；

（6） 察看搜索历史纪录；

（7） 对历史纪录进行分析，并给出搜索的相关结果；

（8） 现有文档中所包含的知识，以知识点的重要性排列；

（9） 根据用户所存放的照片，找到自己拍照片所用的相机型号及照片类型等包含用户个人兴趣等信息；

（10） 根据视频、音频等文件所包含的元信息，用户可以快速定位视频或音频文件并欣赏；

（11） 根据用户搜索的历史纪录及用户所存放的文件中所包含的信息，可以挖掘用户的个人兴趣并自动学习，让用户能够在获得更准确、更快速的找到自己所需要的文件的同时，我们系统也能够根据大多数用户的爱好或需求，为用户提供更多更友好的服务。

2.4.2 系统需求

1. 根据所有用户的搜索记录，统计当前的搜索排行；
2. 根据用户的个人信息及用户的搜索记录，更新及统计用户的个人兴趣；
3. 根据用户的兴趣，优化用户的搜索结果排名；
4. 根据用户的兴趣即搜索记录，统计用户的知识需求，以便为用户提供最需要的服务及公用资源的调整；
5. 分布式支持
   （1）存在根服务器，维护各个节点服务器的列表；
   （2）各个节点服务器需要提供分布式查询接口；
   （3）各个节点服务器能够自动处理来自各个节点的数据；
   （4）根服务器维护通用用户兴趣列表，各个节点服务器保留本地备份。根服务器定期更新通用用户兴趣列表。各个节点服务器需要提供相

关接口；

6. 实时监控支持

（1）当前正在处理的文件数；

（2）当前处理完成的文件数；

（3）当前并发任务数；

7. 异构系统支持

（1）本系统存在多个层次、多个系统协作工作；

（2）向其它可能接入的系统提供接口。

在对 U-CLASS 的需求做简要分析之后，下一章将对本文的理论基础展开讨论，它是本文的核心。

第三章　系统理论分析

本章将对本文的基础理论进行了分析，其中包括模糊数学等已广泛使用的理论；同时本文也在前人的基础之上提出了基于并性模糊蚂蚁的聚类算法、基于Double Array Trie 的中文分词算法及用户模型等三个重要算法，并作了相关测试及分析；同时对搜索引擎中几个常用的信息获取模型，如：布尔模型、空间向量模型等作了一定深度的分析。下面首先将对模糊数学进行分析。

3.1 模糊数学

3.1.1 基本概念

在人类的日常生活中，几乎处处都存在模糊（Fuzzy）现象或模糊概念。例如："年轻人"，"高个子"，"胖子"，"性能良好"等等。这个世界上的现象，或许模糊性是绝对的，而清晰性和精确性是相对的[11]。人类和动物的根本区别之一就是拥有思考和处理模糊信息的能力。即人类能够从真实世界提取抽象的概念而用来解决实际而具体的问题。而建立在经典集合论基础上的模糊集合理论使用了数字表示模糊语义词汇从而模仿人类处理模糊信息的能力，这使得它具有广泛应用的能力。

定义 1　在模糊集合理论中，被讨论的全体对象被称为论域。

定义 2　在给定的论域 U 上的一个模糊集 A 的定义为：对任何 $u \in U$，都指定了数 $\mu_A(u) \in [0,1]$ 与之对应。这个数 $\mu_A(u)$ 被称为 u 对 A 的隶属度。这意味着构造了一个映射，此映射成为 A 的隶属函数 [11]。

$$\mu_A : U \mapsto [0,1] \tag{3-1}$$

$$u \mapsto \mu_A(u) \tag{3-2}$$

定义 3　模糊集运算。若 A、B 为 U 上两个模糊集，它们的并集、交集和 A 的余集都是模糊集，　其隶属函数分别定义为：

$$C = A \cup B \Leftrightarrow (\forall u \in U)(\mu_C(u) = \mu_A(u) \vee \mu_B(u)) \tag{3-3}$$

$$C = A \cap B \Leftrightarrow (\forall u \in U)(\mu_C(u) = \mu_A(u) \wedge \mu_B(u)) \tag{3-4}$$

$$E = A^c \Leftrightarrow (\forall u \in U)(\mu_E(u) = 1 - \mu_A(u)) \tag{3-5}$$

$$\vee \triangleq MAX\,,\quad \wedge \triangleq MIN$$

关于模糊集的并、交等运算，可以推广到任意多个模糊集合。

定义 4 若一个矩阵元素取值为[0，1]区间内，则称该矩阵为模糊矩阵。同普通矩阵一样，有模糊单位阵，记为I；模糊零矩阵，记为 0；元素皆为 1 的矩阵用J表示；当矩阵元素仅为 0 或 1 时，该矩阵称为布尔矩阵。

定义 5 若$A=(a_{ij})_{m\times n}$，$B=(b_{jk})_{n\times l}$，$C=(c_{ik})_{n\times l}$均为模糊矩阵，则A与B的复合记为：$C=A\circ B$。其中：

$$c_{ik} = \bigvee_{j=1}^{n}(a_{ij} \wedge b_{jk}) \tag{3-6}$$

模糊矩阵的复合运算的性质包括：

性质 1 $A\circ I = I\circ A = A$

性质 2 $(A\circ B)\circ C = A\circ(B\circ C)$

等等。

3.1.2 经典集合与模糊集合

对于一个经典集合A及任一元素x，要么$x\in A$，要么$x\notin A$，二者必居其一。这一特征可用一个函数表示为：

$$A(x) = \begin{cases} 0 & x\in A \\ 1 & x\notin A \end{cases} \tag{3-7}$$

$A(x)$即为集合A的特征函数。将此特征函数推广到模糊集，在普通集合中只取 0、1 两值推广到模糊集中为[0，1]区间。即当上节中的定义 2 的映射定义为：A中的任何一个元素，都映射到 0 或 1，那么经典集合就转化为模糊集合的一个特例。

3.1.3 模糊关系

设U，V是两个论域，由U，V产生一个新的论域$U\times V$。$U\times V$上任何一个模糊集R都成为U与V之间的模糊关系 [11]，即

$$\mu_R : U\times V \rightarrow [0,1] \tag{3-8}$$

$$(u,v) \mapsto \mu_R(u,v) \tag{3-9}$$

其中$\mu_R(u,v)$成为u与v关于R的关系强度。当$U=V$时，称R为U上的模糊关系。

设R为$U\times V$上的一个集合，且满足：

（1）反身性：$(x_i,y_j)\in R$，即集合中每个元素和它自己同属一类；

（2）对称性：若$(x,y)\in R$，则$(y,x)\in R$，即集合中(x,y)元素同属于类R时，则(y,x)也同属于R；

（3）传递性：$(x,y)\in R$，$(y,z)\in R$，则有$(x,z)\in R$。

上述三条性质称为等价关系，满足这三条性质的集合R为一分类关系。聚类分析的基本思想是用相似性尺度来衡量事物之间的亲疏程度，并以此来实现分类，模糊聚类分析的实质则是根据研究对象本身的属性来构造模糊矩阵，在此基础上根据一定的隶属度来确定其分类关系。

3.1.4 模糊聚类分析

1. 文本聚类

聚类是最常见的无监督机器学习形式之一，已在很多领域如信息获取(IR)中得到了应用。聚类的目的是把一个数据集分解成为多个自我相似的数据分组。在这些分组中，同一个组的数据点比在其它组中的数据点有更强的相似度。即聚合是创建具有强内聚性、弱耦合性的簇（cluster）。在一些常见数据分组算法中，如：模糊C均值(Fuzzy c-Mean，FCM)，存在一些缺点。它们需要分组数量等先验知识，同时还对分组中心的初始化值比较敏感，而这些信息对于新的环境来讲是未知的，这使得这些算法的应用受到了限制。

2. 相似度度量算法

进行模糊聚类之前，首先要建立模糊相似矩阵，该矩阵是以相似度为基础的。相似度反映了元素之间某些属性的相似程度，有了相似度这个概念，我们就可以根据元素与模糊块之间的相似度来进行模糊划分了。相似度的计算方法有很多种，主要有以下几种：

- 数量积法
- 夹角余弦法
- 统计相关系数法
- 最大最小法
- 算术平均最小法

- 几何平均最小法
- 绝对值指数法
- 指数相似系数法
- 绝对值倒数法
- 绝对值减数法
- 贴近度法
- 专家打分法

3. 模糊聚类

由于聚类是无监督的机器学习，同时在真实世界中，事物的分界并不一定非常清晰。这正是模糊数学处理的一个方面。而模糊聚类正是利用了事物的这一特征，对数据的关系进行模糊处理，进而对数据进行聚类。模糊聚类相比其它的聚类算法具有对分组中心初始化值不敏感等特点，使其得到越来越广泛的应用。

3.2 模糊推理

3.2.1 Mamdani 推理系统

除了把现实世界中的模糊性采用数字表示之外，近似推理是模糊集合理论的另外一个重大贡献。基于模糊规则的系统（FRBS，Fuzzy rule－Based System）是模糊集合理论应用最广泛的领域之一[10]。FRBS 由知识库(KB，knowledge base）和推理机（IE，inference engine)。而知识库由规则库（RB，rule base）和数据库（DB，database）构成。在 FRBS 中有关求解问题的知识是通过“IF-THEN”形式的模糊语言规则表示为：

$$R_i: \text{IF } X_1 \text{ is } A_{i1} \ldots \text{and } X_n \text{ is } A_{in} \text{ THEN } Y \text{ is } B_i$$

$$i \in \{1 \ldots n\}$$

其中，X_i 和 Y 分别是输入和输出变量，他们的值取自各自的论域 U_i 和 V。$A_{i1} \ldots A_{in}$ 和 B_i 均是由模糊集合所定义的语言变量。推理机的目的是对于任意 A_i 都能推导出 Y 的值。处理此推理问题的有很多不同的方法，在本文中选择了简单、有效的 Mamdani 算法[13]。在本文的算法中，因只有两个输入，所以 n=2。同时因为存在多条推理规则，所以选择了多重多维推理算法。 整个推理过程描述如下[14]：

1. 对于X_1和X_2的第i个值(实数)，执行第i条规则，计算$A_{i1}(x_1)$及$A_{i2}(x_2)$并计算$A_{i\min}$。$A_{i\min}=\min\{A_{i1}(x_1),A_{i2}(x_2)\}$。计算$A_{i1}(x_1)$及$A_{i2}(x_2)$的过程称为模糊化过程；
2. 对于第i条规则，使用$A_{i\min}$截断B_i，其中$B_i'=\min(A_{i\min},B_i)$；
3. 执行所有规则，并得到全局推理结果B，$B=\underset{i=1}{\overset{n}{MAX}}B_i'$；
4. 最终的推理结果可以由B和重心法(COG，Center of Gravity)计算得到。其中重心法是将模糊结果转换成精确数字的流行算法之一。COG可以利用下面公式计算：

$$x=\frac{\sum\mu(x_i)\cdot x_i}{\sum\mu(x_i)} \tag{3-10}$$

若为连续变量，则可用以下公式计算：

$$x=\frac{\int x\mu(x)\,dx}{\int\mu(x)\,dx} \tag{3-11}$$

由于在本文中的反模糊化中的变量是离散的，所以采用前式计算。

3.3 模糊蚂蚁聚类算法

尽管单个蚂蚁的行为和结构都非常简单，而且一个工蚁只能完成不超过50个动作，但蚁群却能完成具有相当复杂度的任务[10]。科学家们通过观察蚂蚁怎么对尸体进行分组及排列幼虫，进而总结出一些有趣的结论。在蚁群中，工蚁们通过尸体相互之间的相似性从而排列形多堆尸体。通过在堆中以前尸体相似性信息形成的正反馈，工蚁进而形成越来越大的尸体堆。即当蚂蚁发现了小堆，他们会倾向于把越来越多的尸体放入堆中。

而蚂蚁们也以同样的方式，通过观察幼虫的大小，进而把幼虫收集起来。不同的、具有相同大小的幼虫会以一定概率被收集到一起。当蚂蚁发现与它背着的幼虫大小相似的幼虫时，蚂蚁会将背着的幼虫放下；同时，当蚂蚁没有背负幼虫且发现跟周围幼虫相似度较小的幼虫时，蚂蚁会将“与众不同”的幼虫背走[15]。因为蚁群的这个特点，许多工蚁可以参与聚类的过程。

有感于如此有趣的现象，本文提出一个简单、不需要中心统一控制及不需要关于分组中心等先验信息的聚类算法。正因为不需要关分组中心的先验知识，因此该算法可以很好的弥补FCM对分组中心等初始化信息比较敏感这个缺陷。当把

多个蚂蚁找到的分组中心作为 FCM 的输入，从而使分组更加优化. 因为不同的蚂蚁在爬行的时候，可以没有相互影响，所以比较适合并行执行。在下文中，将对 FCM 进行简单介绍，之后对串行及并行算法进行分析对比。

3.3.1 模糊 C 均值（FCM）算法

模糊 C 均值聚类（FCM），即众所周知的模糊 ISODATA，是用隶属度确定每个数据点属于某个聚类的程度的一种聚类算法。1973 年，Bezdek 提出了该算法，作为早期的硬 C 均值聚类（HCM）方法的一种改进。

FCM 把 n 个向量 x_i（i=1，2，…，n）分为 c 个模糊组，并求每组的聚类中心，使得非相似性指标的价值函数达到最小。FCM 与 HCM 的主要区别在于 FCM 用模糊划分，使得每个给定数据点用值在 0，1 间的隶属度来确定其属于各个组的程度。与引入模糊划分相适应，隶属矩阵 U 允许有取值在 0，1 间的元素。不过，加上归一化规定，一个数据集的隶属度的和总等于 1：

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1,...,n \tag{3-12}$$

那么，FCM 的价值函数（或目标函数）即为：

$$J(U,c_1,...,c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^m d_{ij}^2 \tag{3-13}$$

这里 u_{ij} 介于 0，1 间；c_i 为模糊组 i 的聚类中心，$d_{ij} = |c_i - x_j|$ 为第 i 个聚类中心与第 j 个数据点间的欧几里德距离；且 $m \in [1,\infty)$ 是一个加权指数。

构造如下新的目标函数，可求得 $J(U,c_1,...,c_c)$ 达到最小值的必要条件：

$$\begin{aligned} \bar{J}(U,c_1,...,c_c,\lambda_1,...,\lambda_n) &= J(U,c_1,...,c_c) + \sum_{j=1}^{n}\lambda_j(\sum_{i=1}^{c} u_{ij} - 1) \\ &= \sum_{i=1}^{c}\sum_{j}^{n} u_{ij}^m d_{ij}^2 + \sum_{j=1}^{n}\lambda_j(\sum_{i=1}^{c} u_{ij} - 1) \end{aligned} \tag{3-14}$$

这里λ_j，j=1 到 n，是（3-12）式的 n 个约束式的拉格朗日乘子。对所有输入参量求导，使式（3-13）达到最小的必要条件为：

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{3-15}$$

及

$$u_{ij}=\frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \tag{3-16}$$

由上述两个必要条件，模糊C均值聚类算法是一个简单的迭代过程。在批处理方式运行时，FCM用下列步骤确定聚类中心c_i和隶属矩阵U：

步骤1：用值在0，1间的随机数初始化隶属矩阵U，使其满足式（3-12）中的约束条件；

步骤2：用式（3-15）计算c个聚类中心c_i，i=1，...，c；

步骤3：根据式（3-13）计算价值函数。如果它小于某个确定的阀值，或它相对上次价值函数值的改变量小于某个阀值，则算法停止；

步骤4：用（3-16）计算新的U矩阵。返回步骤2。

上述算法也可以先初始化聚类中心，然后再执行迭代过程。由于不能确保FCM收敛于一个最优解。算法的性能依赖于初始聚类中心。因此，我们要么用另外的快速算法确定初始聚类中心，要么每次用不同的初始聚类中心启动该算法，多次运行FCM。

通过上面的讨论，我们不难看出FCM算法需要两个参数一个是聚类数目C，另一个是参数m。一般来讲C要远远小于聚类样本的总个数，同时要保证$C>1$。对于m，它是一个控制算法的柔性的参数，如果m过大，则聚类效果会很次，而如果m过小则算法会接近HCM聚类算法。

算法的输出是C个聚类中心点向量和$C*N$的一个模糊划分矩阵，这个矩阵表示的是每个样本点属于每个类的隶属度。根据这个划分矩阵按照模糊集合中的最大隶属原则就能够确定每个样本点归为哪个类。聚类中心表示的是每个类的平均特征，可以认为是这个类的代表点。

从算法的推导过程中我们不难看出，算法对于满足正态分布的数据聚类效果会很好，另外，算法对孤立点是敏感的。

3.3.2 串行算法分析

在文献[15] [16] [17]中，模糊蚂蚁已经有过讨论。本节主要给出一些定义和讨论串行算法，目的是为了方便后文讨论并行算法。因此省略了串行算法的细节。

定义1　一个堆是指两个或多个对象[15]。在本文中，每个堆还添加了一个唯一的

标识；

定义 2 当蚂蚁分泌的信息素为 s、响应门限值为 θ 时，蚂蚁开始工作的概率为[16]：

$$T_n(n;s)=\frac{s^n}{s^n+\theta^n} \quad , \ n \text{ 为正整数；} \tag{3-17}$$

定义 3 蚂蚁丢掉它背着的东西的概率为[16]：

$$P_{drop}=T_{n_i}(s_{drop};\theta_{drop}) \tag{3-18}$$

$$i\in\{1,2\}, \ n_1, \ n_2 \text{均为正整数}$$

其中 s_{drop} 是与任务相关的信息素，θ_{drop} 是响应的门限值；

定义 4 蚂蚁背上一个东西和背上所有东西的概率为[16]：

$$P_{pickup\_one}=\frac{s_{one}}{s_{one}+s_{all}}\cdot T_{m_1}(s_{one};\theta_{one}) \tag{3-19}$$

$$P_{pickup\_all}=\frac{s_{all}}{s_{one}+s_{all}}\cdot T_{m_2}(s_{all};\theta_{all}) \tag{3-20}$$

其中 s_{one} 和 s_{all} 是信息素，θ_{one} 和 θ_{all} 为响应的门限值。

定义 5 蚂蚁丢掉背上的东西的模糊推理规则请参考文献[16]

定义 6 蚂蚁背上堆的模糊推理规则请参考文献[16]

在文献[16]中讨论的串行算法与其它算法，如：FCM，相比有如下优点：

1. 首次在数据聚类中采用模糊规则进行推理
2. 对簇中心的初始化值不敏感
3. 鲁棒性、有效性较好

但在该算法与其作为 Carrot2 的组件的实现存在一些缺点：

1. 只使用了一只蚂蚁，将蚁群本身的并发性丢失了；
2. 基于粗糙集合理论的相似度计算太过复杂且比较慢；
3. 在该算法的Java实现中，缓存(Cache)的使用存在不合理之处，导致了该实现效率较低。

在本文中，将上述问题得到了较好的处理。

3.3.3 并行算法

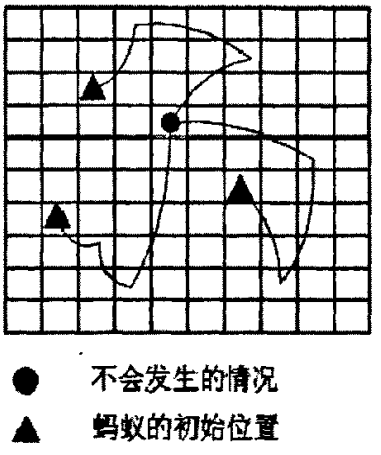


图 3-1 蚂蚁并行移动

图 3-2 堆的状态转换

有感于文献[15]，数据项和蚂蚁均随机地分散在一个离散的 2D 板上。该板是环形的，因而蚂蚁可以自由地从板上的一个单元格移动到另外一个。板的大小由数据项的数量确定且规定为 $m \times m$ 大小，其中 $m^2 = 4n$，n 为数据项的数量。在每个单元格中，只能放置一个堆，且初始化时，每个堆中只放置一个数据项。该堆能被蚂蚁背上，或者新的数据项添加到该堆，从而形成新的堆。 在任意时间t， 在一个单元格中，只能有唯一堆存在且任意堆只能被唯一的蚂蚁占用。板和蚂蚁的布局在图 3-1 中作了图示。其中，任意两条曲线的交点为被蚂蚁占用的堆所在的单元格的位置。

在本文的算法中，维护了两个堆的列表--空闲列表(Idle List，IL)及占用列表(Occupied List，OL)。其中，IL 包含了没有被占用且等待提取的堆；而 OL 包含了已经被提取且正在被蚂蚁处理的堆。这两个列表均被同步，因为二者可能同时被两个或者多个蚂蚁访问。

如图 3-2 所示，任意堆或数据项都有其状态。当 2D 板上的堆在初始化时，只有一个数据项会放入堆中且该数据项会同时加入到 IL 中。当蚂蚁移动到某个单元格时，该单元格中的堆会被蚂蚁提取，该堆会从 IL 中移出并添加到 OL 中，然后蚂蚁会在模糊推理机的指导下，检查是否可以背上该堆或者将其与蚂蚁背着的堆进行相似性对比，从而作进一步的处理。当蚂蚁发现可以背上与堆中其它数据项相似度较低的数据项时，蚂蚁会背上该数据项，且该数据项所在的堆会从 OL 中移出，然后返回到 IL 中。当堆符合 Mamdani 模糊推理系统中被蚂蚁背上的规则(3.3.1

中定义 6）及式(3-19)或(3-20)中定义的概率，蚂蚁会背起其中的数据项或者整个堆；当蚂蚁发现有堆符合预定义的蚂蚁放下背着的堆的规则（3.3.1 中定义 6）时，蚂蚁会根据公式(3-18)中定义的概率放下背着的堆。该放下的堆会被加入到板上单元格中的堆，进而形成一个新的堆替代原来的堆。

伪代码如下所示：

```
Initialize {Idle List and Occupied List,  data items,  ants,
        thread pool and tasks for ants,  NumberOfIteration =0}
Parallel for each ant (task)
While NumberOfIteration <  NumberOfIteration max
    Get a random heap from idle list and add it to occupied list
    If the current ant load a heap
        Check whether to unload the loaded item
        Unload and form a new heap
        Return the gotten item to idle list
    End if
    Else if the current ant is unloaded
        Check whether to load the whole heap or the dissimilar item
                Load the whole heap or the dissimilar item
        If the dissimilar item is taken away
      Return the gotten item to idle list
        End if
    End else if
    NumberOfIteration++
End While
Wait for ending of each task and merge heaps
//Here we have found clusters' centers which will be the input of FCM
Generate clusters with FCM
```

值得注意的是，在上述算法中，分组的中心作为 FCM 的输入，进而生成优化的分组。因此，分组的中心在寻找更优的分组之前已经找到。这些中心在本文中也具有相当重要的作用。

3.3.4 算法测试及结果

并行和串行算法的对比测试采用了 Java 实现。在实现中，使用了线程池及一些并发开发工具，基于 Carrot2 的组件实现，主要用于文档聚类。 串行算法和并行算法的测试采用了来自 Yahoo!以 *"data mining"* 为关键字的搜索结果，且文档数量相同。以下为测试文档的片断：

```
<searchresult>
  <query>data mining</query>
  <document id="0">
    <title>Data mining - Wikipedia, the free encyclopedia</title>
    <snippet>Article about ..... data for patterns.</snippet>
    <url>http://en.wikipedia.org/wiki/Data_mining</url>
  </document>
  .....
</searchresult>
```

其中，**query** 为查询关键字，**document** 为文档内容，**id** 为其编号，**title** 为编号，**snippet** 为文档内容的片断，**url** 为该文档的链接。

以下为测试环境：

硬件环境：

CPU：Xeon MP 2 CPU 1.5 GHz；

RAM：2048 MB；

软件环境：

OS：Windows Server 2003；

Java Virtual Machine：Sun JDK 1.6.0；

JVM switches：-server -Xms100M -Xmx200M；

Packages：commons-logging-1.1.jar，lucene-core-2.0.0.jar，carrot2-util-tokenizer.jar，carrot2-util-common.jar，carrot2-filter-case-normalizer.jar，dom4j-1.6.1.jar，carrot2-stemmer-stempelator.jar，carrot2-input-yahooapi.jar，carrot2-filter-fuzzyants.jar *etc.*

经过不同数量及内容的文档对比测试，结果如图 3-3 及表 3-1 所示。在表 3-1 中，Δt=串行算法的运行时间－并行算法的运行时间。从二者中可看出：

（1）并行算法在文档数量较小（50-200）的情况下，在性能方便表现较串行算法稍差一些。究其原因，因为并行算法需要建立线程池等资源需要耗费时间。这些时间使得并行算法表现反而更差一些；

（2）当文档数量较大（>200）时，并行算法则表现出其优势。此时，并行算法建立线程池等资源所消耗的时间已经得到弥补，且运算速度也明显领先于串行算法。

本文中所提并行算法与串行算法相比，具有更有高的效率、更好的鲁棒性，更适合大的数据集聚类处理。

表 3-1 串行算法与并行算法测试结果对比

文档数量	串行算法（ms）	并行算法（4 只蚂蚁）（ms）	Δt
50	2157	3734	1577
100	3735	7688	3953
200	8266	12421	4155
400	29297	24750	-4547
800	51875	45797	-6078

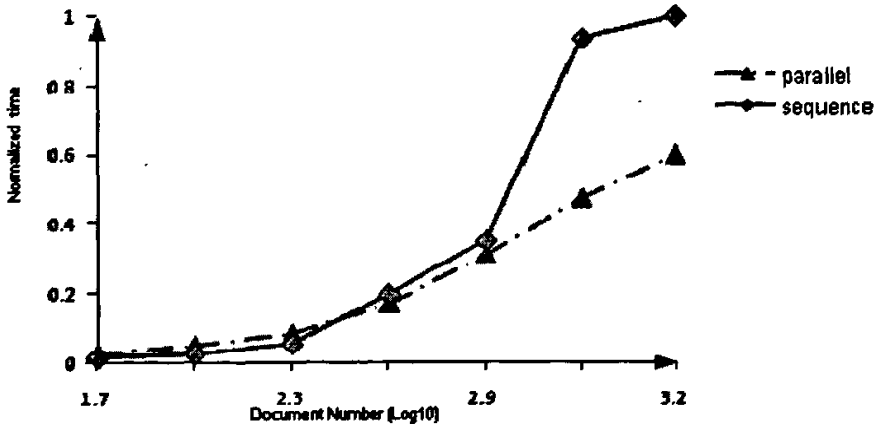


图 3-3 并行与串行算法的对比

3.3.5 算法应用

在本文中，聚类算法具有相当重要的作用。在文本中，居于并行模糊蚂蚁的聚类算法主要用于以下几个方面：

1. 对用户的查询结果进行非监督模式的分类（聚类），这有利于用户的查询结果整体的把握，更有力于找到需要的文件（信息）；
2. 知识挖掘。在用户文件（信息）进入系统之后，系统会自动对数据进行分析，然后对数据聚类，再根据一定的规则，提取各个类别中的核心信息，即是用户的知识点所在；
3. 用户个人兴趣的提取。用户个人兴趣的挖掘也是通过聚类算法对用户的行为、用户查询的关键字及用户的个人知识进行处理，然后将处理结果再经过一些算法加工，即可得到用户的个人兴趣

3.4 信息获取模型

3.4.1 布尔模型

布尔模型（Boolean Model）是基于集合论和布尔代数的一种简单检索模型。由于集合的定义是非常直观的，布尔模型提供了一个信息检索系统用户容易掌握的框架。查询串通常以语义精确的布尔表达式的方式输入，如：“电子科技大学 AND 电子工程学院 AND 信息工程”。

但布尔模型存在着一些缺陷：

（1）它的检索策略是基于二元判定标准（binary decision criterion），对于检索来说一篇文档只有相关和不相关两种状态，缺乏文档排序（rank）的概念，限制了检索功能；

（2）虽然布尔表达式具有精确的语义，但常常很难将用户的信息需求转换为布尔表达式，实际上大多数检索用户发现在把他们所需的查询信息转换为布尔时并不容易；

（3）不能够在查询传中说明关键词的相对重要性，即在布尔模型中，查询传中的关键词是被同等看待的。

瑕不掩瑜，虽然布尔模型有一些缺点，但是它的优点也是比较明显的。主要表现在：它实现起来比较容易，而且计算的代价相对较小；虽然在查询中布尔操作符对初学者来说比较麻烦，但较其他模型中的查询语言更容易理解；那些明确

知道自己想要查找什么信息的用户使用该模型能更精确的找到自己需要的信息。在本系统中，布尔表达式的支持是查询语句最基本的功能之一。

但由于布尔模型的主要缺陷--完全匹配会导致太多或者太少的结果文档被返回。众所周知，索引术语的权重从根本上提高了检索系统的功能，从而导致了向量空间模型（Vector Space Model）的产生。

3.4.2 向量空间模型

在本文中，采用向量空间模型[32]（Vector Space Model，VSM）表示文件信息特征。在 VSM 中，将文件信息表示为一组词元（term）$(t_1,t_2,...,t_n)$。对于每一词元t_i，根据其在文件信息中的重要程度赋予一定的权重w_i。因此，所有文件都可以用词元特征向量$(w_1,w_2,...,w_n)$表示。

而在词元特征向量中，词元的权重有多种方式可以衡量。TF-IDF 是比较常用的方式的之一。其中，词元权重有两个因子决定：词元 j 出现在文档 i 中的次数（词频tf_{ij}）；词元 j 在所有文档中出现的次数（df_j）。准确地说，词元 j 在文档 i 中的权重可表示如下：

$$w_{ij} = tf_{ij} \times idf_j = f_{ij} \times \log N / df_j \tag{3-21}$$

其中，idf 表示逆向文档频率（Inverse document frequency）。

经过多年的经验积累，比较常用的计算词元权重的式子为：

$$w_{ij} = \frac{tf_{ij} \times \log(N/df_j + 0.01)}{\sqrt{\sum_{i \in d}\left[tf_{ij} \times \log(N/df_j + 0.01)\right]^2}} \tag{3-22}$$

其中，N 是所有文件的数量；分母为归一化因子。

在 VSM 中，采用相似度函数计算两个特征向量之间相似性。常用的方法有欧几里德距离、曼哈坦距离和夹角余弦函数。在本文中采用夹角余弦函数。若在计算时遇到用于比较的两个特征向量长度不一样，可采用添零补齐的方式使两者长度一致。夹角余弦函数如下：

$$similarity(\overline{X},\overline{Y}) = \frac{\sum_i X_i * Y_i}{\sqrt{(\sum_i X_i^2) * (\sum_i Y_i^2)}} \tag{3-23}$$

其中，$similarity(\overline{X},\overline{Y})$表示向量$\overline{X}$与$\overline{Y}$的相似度，$X_i$与$Y_i$表示$\overline{X}$与$\overline{Y}$对应的特征词的权值。$\overline{X}$与$\overline{Y}$越相似，$similarity(\overline{X},\overline{Y})$值越大，反之则越小。

3.4.3 概率论模型

信息获取概率论模型的基础是概率排序规则：如果文档按照与查询的概率相关性的大小排序，那么排在前面的文档时最有可能被获取的文档。

信息获取的主要任务就是计算文档和查询之间的相关性。一个查询有来自一个固定的关键词空间的关键词组成，一个文档来自同一个关键词空间的关键词集合和组成，即 Doc=<$term_1$，$term_2$，…，$term_n$>。如果文档满足下式，则该文档将被获取。

$$P(\mathrm{Re}l \mid Doc) \geq P(Notrel \mid Doc) \tag{3-24}$$

其中，$P(\mathrm{Re}l \mid Doc)$表示文档 Doc 与查询有关的条件概率，$P(Notrel \mid Doc)$表示文档 Doc 与查询不相关的条件概率。

根据被贝叶斯规则，上式可改写为：

$$\frac{P(Doc \mid \mathrm{Re}l)P(\mathrm{Re}l)}{P(Doc \mid Notrel)P(Notrel)} \geq 1 \tag{3-25}$$

所获取的文档可以利用上市的左端进行排序。利用概率计算的主要问题是怎么计算这些概率。这可以通过计算相关性已知的文档样本来得到。如果查询串由一个关键词组成，或者组成查询串的关键词之间相互独立，那么每个与文档中的词条匹配的词条的相关性可通过下式表示：

$$w = \frac{r/(R-r)}{(n-r)/[N-n-(R-r)]} \tag{3-26}$$

其中，N是样本集合中文档的数量，n是样本集合众包含该词条的文档的数量，R是与查询相关的文档树木，r是与查询相关而且包含该词条的文档数目。

这样查询q就可以表示为<w_1，w_2，…，w_n>，其中w_i由上式计算得到。文档d表示为<x_1，x_2，…x_t>，其中，如果关键词存在于文档中，则对应的x_t=1，反之，x_t=0。这样可以定义查询串q与文档d之间的相似度函数，如下式所示：

$$similarity(q,d) = \sum_{k=1}^{t} x_k w_k \tag{3-27}$$

概率论模型的效率明显优于布尔模型，但是比向量模型略差。概率论模型有以下特点：

（1） 与向量模型一样，关键词之间是假设相互独立的；

（2） 在没有获得样本文档之前，即没有相关的文档之前，无法顾及词条的

相关性。

由于该模型适合于超文本系统，因此在超文本信息成为信息获取主流信息的情况下，该模型的应用越来越广泛了。

3.4.4 模糊集合模型

使用关键词组表示文档和查询的方式只能部分表达文档和查询的真实语义内容。因此，文档和查询词元之间的匹配是近似的。所以此匹配过程可以将查询词元定义为一个模糊集（Fuzzy Set），同时每个文档在该集合中有一定的隶属度。以此方式作为多个模糊集合模型的基础已经发展了较长的时间。由此可看出，模糊集合模型是建立在模糊数学的基础之上的。关于模糊集合的基本性质等在前文已经作了介绍，在此不再赘述。

对于文档集合$\{d_1, d_2, \cdots d_n\}$，词元集合$\{t_1, t_2, \cdots t_m\}$：

（1） 定义词元之间的关系：建立一部相关性词典，定义所有词元和词元之间的相关性。t_i和t_l的相关性c_{il}可以通过下式计算（下式实际计算的是共现频率，$c_{il} \in [0,1]$）：

$$c_{il} = \frac{n_{il}}{n_i + n_l - n_{il}} \tag{3-28}$$

（2） 定义词元和文档之间的关系：对于一个t_i建立一个模糊集合A_i，其元素是所有文档集合，其中每个文档d_j对该模糊集A_i的隶属度通过下式计算（d_j中和t_i越相关的词元越多，则函数值越大）：

$$\mu_{ij} = 1 - \prod_{t_i \in d_j} (1 - c_{il}) \tag{3-29}$$

（3） 最后，定义查询q和文档d_j之间的关系，查询q对应一个模糊集合，求每个文档d_j的隶属度。

每个模型都有其优缺点。模糊集合模型的优点在于克服原始布尔模型不能部分匹配的缺点；但它通常在模糊集研究领域涉及，在IR领域不流行且缺乏大规模语料上的实验证实其有效性。

3.5 基于 Double-Array Trie 的中文分词算法

3.5.1 意义

汉语的中文信息处理主要是“用计算机对汉语的音、形、义进行处理。”。同时，“词是最小的能够独立活动的有意义的语言成分”。然而，汉语文本中词与词之间却没有明确的分隔标记，而是连续的汉字串。例如，英文句子“I am a student”，用中文则为：“我是一个学生”。计算机可以很简单通过空格知道 student 是一个单词，但是不能很容易明白“学”、“生”两个字合起来才表示一个词。把中文的汉字序列切分成有意义的词，就是中文分词，有些人也称为切词。“我是一个学生”，分词的结果是：“我 是 一个 学生”。显而易见，自动识别词边界，将汉字串切分为正确的词串的汉语分词问题无疑是实现中文信息处理的各项任务的首要问题。只有正确的识别汉语语句中的词语，中文信息处理才有坚实的基础。

在本课题中，中文分词是整个项目的基础；中文分词的准确性及速度是整个个性化搜索平台性能的重要保障之一。在平台中，多处需要使用中文分词技术。在基础功能—搜索中，中文分词是用户的输入能够得到正确的理解是关键；同时，为了完成搜索功能，需要对用户的文件内容完成索引工作。而搜索过程是由文件 Spider 完成，它需要对文件信息作分析，从而提取特征项。而特征项的提取是以中文分词为基础的；在用户知识挖掘中，需要对用户的文档等文件中的知识做正确的理解，没有正确的分词，是难以到达的，即正确的分词是用户知识挖掘的重要前提；而在用户兴趣的提取中，用户的兴趣是以关键字及其相关数据联合表示的，则关键字需要准确的表达用户的兴趣所在。由于关键字主要来源于中文信息，所以对中文信息的正确理解、关键字的正确提取是核心之一，而完成该任务的前提和基础即是中外分词。

而为了完成上述任务，平台的底层算法，如：并行模糊蚂蚁聚类算法等都是以中文分词为基础。

综上所述，中文分词是本课题的重要基础之一，是整个系统的关键技术之一。它是否能快速正确的完成自动分词，对整个个性化搜索平台的成败有着重要的意义。如果没有正确的分词，整个平台便成“沙滩上的高楼”。

同时在文本分类、自然语言理解、机器翻译、文本校对汉字的简体/繁体转换、信息检索和信息摘录、拼音输入中同音字辨识等中文信息处理系统同样都首先需要分词作为其最基本的模块[17]。

3.5.2 常用分词算法

中文分词技术属于自然语言处理技术范畴，对于一句话，人可以通过自己的知识来明白哪些是词，哪些不是词，但如何让计算机也能理解？其处理过程就是分词算法。

现有的分词算法可分为三大类：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。

1．基于字符串匹配的分词方法 [2]

这种方法又叫做机械分词方法，它是按照一定的策略将待分析的汉字串与一个“充分大的”机器词典中的词条进行配，若在词典中找到某个字符串，则匹配成功（识别出一个词）。按照扫描方向的不同，串匹配分词方法可以分为正向匹配和逆向匹配；按照不同长度优先匹配的情况，可以分为最大（最长）匹配和最小（最短）匹配；按照是否与词性标注过程相结合，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：

（1）正向最大匹配法（MMB，由左到右的方向）

目的是把最长的词给切分出来。它首先假定最大词长度为 L ，从句首取长度为 L 的字串进行匹配，如果匹配成功则认为此字串为一个词，再从它的下一个字开始继续该过程；如果匹配不成功则去掉此字串的最后一个字进行匹配，直至匹配成功或子句为空；

（2）逆向最大匹配法（RMMB，由右到左的方向）

逆向最大匹配法(RMMB) 的算法与 MMB 的算法相同，只是切分方向为从右向左，而且如果匹配不成功去掉的是第一个汉字，相当于先将被切分字串逐字反转形成新的字串再进行正向最大匹配；

（3）最少切分（使每一句中切出的词数最小）

还可以将上述各种方法相互组合，例如，可以将正向最大匹配方法和逆向最大匹配方法结合起来构成双向匹配法。由于汉语单字成词的特点，正向最小匹配和逆向最小匹配一般很少使用。一般说来，逆向匹配的切分精度略高于正向匹配，遇到的歧义现象也较少。统计结果表明，单纯使用正向最大匹配的错误率为 1/169，单纯使用逆向最大匹配的错误率为 1/245。但这种精度还远远不能满足实际的需要。实际使用的分词系统，都是把机械分词作为一种初分手段，还需通过利用各种其它的语言信息来进一步提高切分的准确率。

一种方法是改进扫描方式，称为特征扫描或标志切分，优先在待分析字符串

中识别和切分出一些带有明显特征的词，以这些词作为断点，可将原字符串分为较小的串再来进行机械分词，从而减少匹配的错误率。另一种方法是将分词和词类标注结合起来，利用丰富的词类信息对分词决策提供帮助，并且在标注过程中又反过来对分词结果进行检验、调整，从而极大地提高切分的准确率。

对于机械分词方法，可以建立一个一般的模型，表示为 ASM(d，a，m)，即 Automatic Segmentation Model。其中，

d：匹配方向，+表示正向，-表示逆向；

a：每次匹配失败后增加或减少字串长度（字符数），+为增字，-为减字；

m：最大或最小匹配标志，+为最大匹配，-为最小匹配。

例如，ASM(+，-，+)就是正向减字最大匹配法（Maximum Match based approach，MM），ASM(-，-，+)就是逆向减字最大匹配法(简记为 RMM 方法)，等等。对于现代汉语来说，只有 m=+是实用的方法。

2、基于理解的分词方法

这种分词方法是通过让计算机模拟人对句子的理解，达到识别词的效果。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。它通常包括三个部分：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。

3．基于统计的分词方法

从形式上看，词是稳定的字的组合，因此在上下文中，相邻的字同时出现的次数越多，就越有可能构成一个词。因此字与字相邻共现的频率或概率能够较好的反映成词的可信度。可以对语料中相邻共现的各个字的组合的频度进行统计，计算它们的互现信息。计算汉字 X 和 Y 的互现信息公式为：

$$M(X,Y)=\log\frac{P(X,Y)}{P(X)P(Y)} \tag{3-29}$$

其中，$P(X,Y)$ 是 X、Y 的相邻共现概率，$P(X)$、$P(Y)$ 是 X、Y 在语料中出现的概率。互现信息体体现了汉字之间结合的紧密程度。当紧密程度高于某一个阈值时，便可认为此字组可能构成了一个词。这种方法只需对语料中的字频进行统计，不需要切分词典，因而又叫做无词典分词法或统计取词方法。但这种方法也有一

定的局限性，会经常抽出一些共现频度高、但并不是词的常用字组，例如“这一”、“之一”、“有的”、“我的”、“许多的”等，并且对常用词的识别精度差，时空开销大。实际应用的统计分词系统都要使用一部基本的分词词典（常用词词典）进行串匹配分词，同时使用统计方法识别一些新的词，即将串频统计和串匹配结合起来，既发挥匹配分词切分速度快、效率高的特点，又利用了无词典分词结合上下文识别生词、自动消除歧义的优点。

词汇切分算法最重要的指标是准确，在兼顾准确性的情况下也要考虑时间复杂度。在本文中，采用了基于字符串匹配的分词方法，即以汉语词典为基础对中文语句通过匹配进行切分。不论是哪种基于词典的分词方法，分词词典的查询速度是匹配算法效率的直接决定因素，因而建立高效快速的分词词典机制势在必行。在后文中，将首先分析几种常见的词典查询算法。然后再介绍一种新的基于Double-Array Trie 的最大前缀中文分词法。它具有快速、分词精度高等特点。

3.5.3 典型词典查询算法

在文献[19]中指出了三种典型的分词词典机制—整词二分法、TRIE 索引树法、逐字二分法及改进机制。下边对它们进行简单的介绍：

1．基于整词二分的词典机制

如图 3-4 所示，该机制的词典结构分为词典正文、词索引表、首字散列表等三级。词典正文是以词为单位的有序表，词索引表是指向词典正文中每个词的指针表。通过首字散列表的哈希定位和词索引表很容易确定指定词在词典正文中的可能位置范围，进而在词典正文中通过整词二分进行定位。

2．基于 Trie 索引树的词典机制

Trie 索引树是一种以树的多重链表形式表示的键树。如图 3-6 所示，基于 Trie 索引树的分词词典机制由首字散列表和 Trie 索引树结点两部分组成。Trie 索引树的优点是在对被切分语句的一次扫描过程中，不需预知待查询词的长度，沿着树链逐字匹配即可；缺点是它的构造和维护比较复杂，而且都是单词树枝(一条树枝仅代表一个词)，浪费了一定的空间。

3．基于逐字二分法的查询机制

这种词典机制是前两种机制的一种改进方案。逐字二分与整词二分的词典结构完全一样，只是查询过程有所区别:逐字二分吸收了 Trie 索引树的查询优势，即采用的是“逐字匹配”，而不是整词二分的“全词匹配”，这就一定程度地提高了

匹配的效率。但由于采用的仍是整词二分的词典结构，使效率的提高受到很大的局限。

4. 双字哈希的词典查询机制

此词典查询机制是根据汉语中双字词语较多的特点，词典采用前两字逐个哈希索引、剩余字串有序排列的结构，查询过程采用逐字匹配的方法，在不提升已有典型词典机制维护复杂度的情况下，提高了中文分词的速度(虽然它的速度相对于 Trie 索引树的优势并不明显，但比后者的构建和维护要简单得多)，一定程度上提高的查询性能。

另外在以上几种查询机制的基础上，还有诸如 PATRICIA tree 的汉语词典查询机制、基于四字哈希的查询机制等，在一定程度上作了改进，有所提高。

3.5.4 基于 Double-Array Trie 的中文分词算法

Trie 树是搜索树的一种，本质是一个确定的有限状态自动机（DFA），每个节点代表自动机的一个状态，根据变量的不同，进行状态转移，当到达结束状态或者无法转移的时候，即完成查询。如图 3-5 所示。

用 Trie 树搜索一个关键码的时间与关键码自身及其长度有关，最快是O(1)，即在第一层即可判断是否搜索到，最坏的情况是O(n)，n 为 Trie 树的层数。由于很多时候 Trie 树的大多数结点分支很少，因此 Trie 树结构空间浪费比较多。

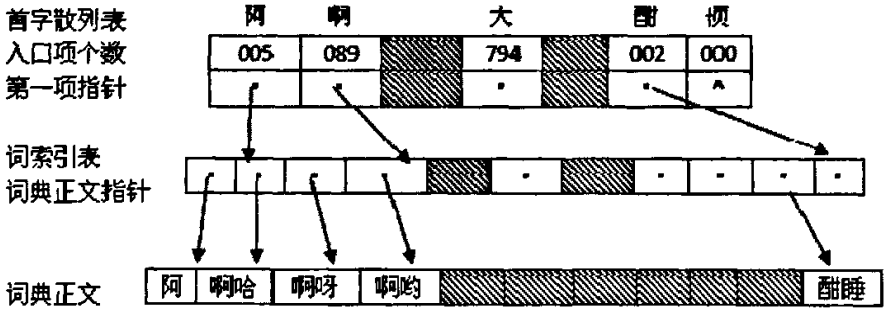


图 3-4 基于整词二分的分词词典机制

为了让 Trie 实用的实现算法减少空间占用的同时保证查询效率，Johnson S.C 提出了一种用 4 个线性数组表示 DFA 的方法，且由 Aho，A.V 进一步提出了用 3 个线性数组表示 Trie 树的方式。并在此基础上，文献[21][22]做出了进一步改进，用 2 个线性数组来进行 Trie 树的表示，即双数组 Trie（Double-Array Trie）[23]

双数组 Trie（Double-Array Trie）的本质就是将 Trie 树结构简化为两个线性数

组，由两个整数数组构成，一个是base[]，另一个是check[]，如图3-7所示。base数组中的每一个元素相当于Trie树的一个节点，其值作为状态转移的基值，而check数组的值相当于校验值，用于检查该状态是否存在。对于从状态s到状态t的一个转移，必须满足如下两个条件（如图3-8所示）：

1. $base[s]+c=t$

2. $check[t]=s$

其中c是输入变量。

如果$base[i]$，$check[i]$均为 0，表示该位置为空。如果$base[i]$为负值，表示该状态为可结束的状态（即找到词语）。

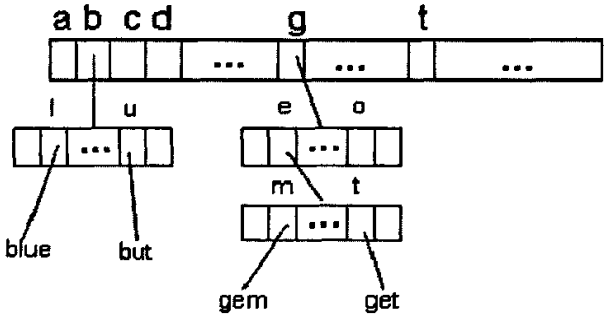


图 3-5 Trie 索引树结构

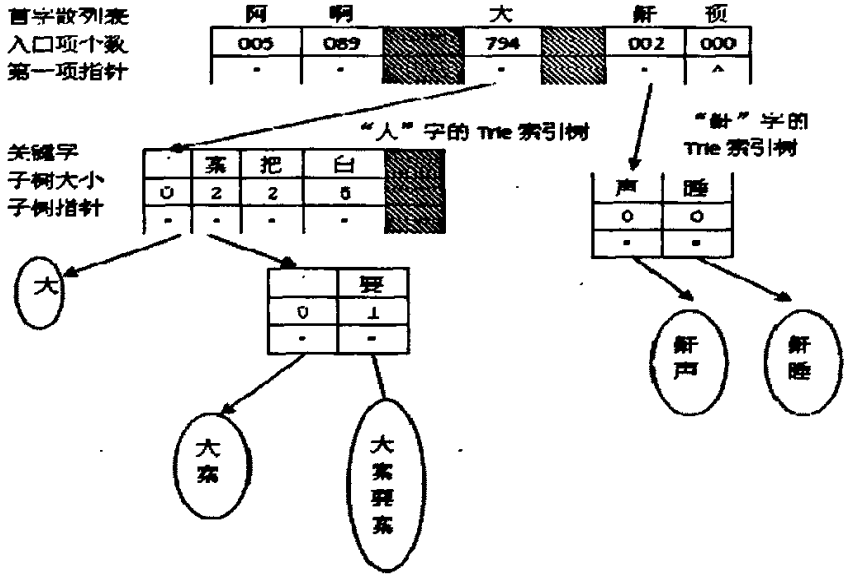


图 3-6 基于 Trie 索引树的分词词典机制

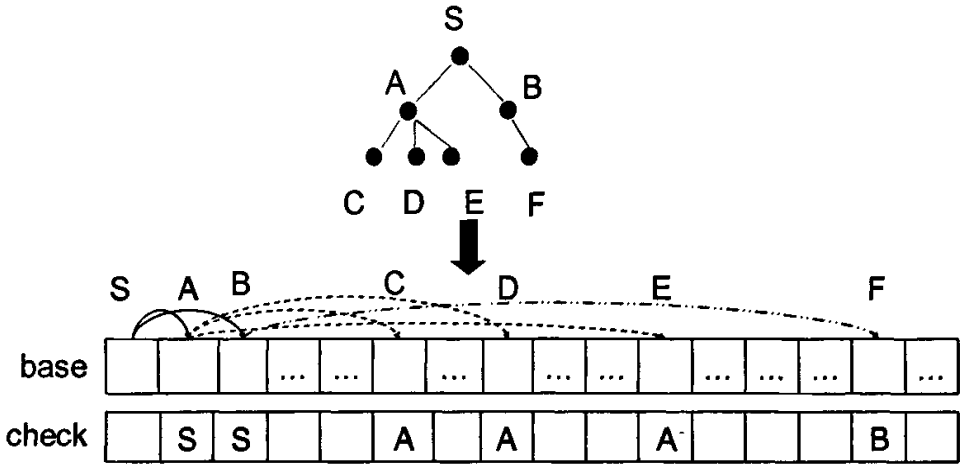


图 3-7 双数组 Trie 索引树结构

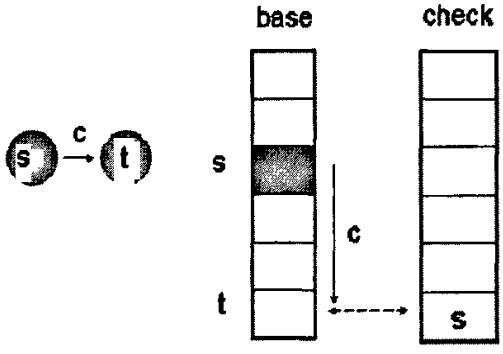


图 3-8 双数组 Trie 索引树状态转移

在汉字词典查询中，也可采用同样的方法构造数组。先对 6763 个常用汉字根据其相应内码的顺序而赋予从 1－6763 的序列码，放入*base*[1]-*base*[6763]中。在本文中，汉字均采用 GBK 编码，因而汉字的序列码使用与 GBK 编码相对应的方式，采用生成 Hash 值得方式得到。Hash 值的生成式为：

$$gbkHashCode = (gbkWord[0]-129)*190 + (gbkWord[1]-64) - gbkWord[1]/128$$

其中，*gbkHashCode*为 Hash 值，*gbkWord*为汉字的 GBK 编码。

对于每一个汉字，确定一个 base 值，使得对于所有以该汉字开头的词，在双数组中都能放下。例如，现在要确定“娃”字的 base 值，假设以“娃”开头的词的第二个字序列码依次为 a_1，a_2，a_3 ..... a_n，我们必须找到一个值 i，使得

$base[i+a_1]$，$check[i+a_1]$，$base[i+a_2]$，$check[i+a_2]$，……$base[i+a_n]$，$check[i+a_n]$均为0。一旦找到了这个i，“娃”的base值就确定为i。对于第二个字，第三个字也是类似。

在将词语放入数组种之后，然后还须遍历一遍词表，修改base值。因为我们用负的base值表示该位置为词语。如果状态i对应某一个词，而且$base[i]=0$，那么令$base[i]=-1*i$，如果$base[i]$的值不为0，那么令$base[i]=-1*base[i]$。

双数组构造完成以后，查询起来极为方便。待查词有几个字，就将汉字分别转换为对应的序列码，然后作几次加法，即可查到相应的词语，无须折半查找。查询步骤如下：

1．读入输入变量c；

2．t = base[s] + c

if check [t] = = s then

nextState = t

else fail

endif

3．若base[t]不为负，重复步骤1。否则， t为一个可结束状态

由于汉语中常用词的平均长度不到3个字，因此双数组查询算法的效率是极高的。

在本文中，分词是基于词典查询，且词典的组织采用了上述的Double-Array Trie，但查询过程与上述过程有所不同。构建过程如下：

1．读入词典；

2．设置最长词语长度$WordLength_{max}$；

3．根据汉字的内码生成Hash值；

4．按照上文中构建Double-Array Trie的方法构建汉语词典搜索树。

最长前缀分词过程如图3-9所示。

下面我们将举例说明（在此例中，采用简化方式，序列码并未严格按照上文中的算法生成）：

假定词表中只有“啊，阿根廷，阿胶，阿拉伯，阿拉伯人，埃及”这几个词，则我们首先对词表中所有出现的10个汉字进行编码：啊-1，阿-2，唉-3，根-4，胶-5，拉-6，及-7，廷-8，伯-9，人-10。按上文的方法建立的Double Array如表3-2所示：

若输入文本为：“阿拉伯人”，则根据词典内容，设置$WordLength_{max}$为4。首先

读取 4 个字符，首先找到“阿”序列码 2 及 base[2]=4，然后根据 c=“拉”的序列码 6，base[2]+c = 10 且 check[10] = 4，则“阿拉”为一状态，可以继续；再根据 c=“伯”的序列码 9 及 base[10]=4，则 base[10]+c=13 且 check[13]=10，则“阿拉”为一状态。同时 base[13]<0，则“阿拉伯”为可结束状态，将 *currentWordCount* = 3 加入 resultList 中，然后继续；根据 c=“人”的序列码 10 及|base[13]|=4，则 |base[13]|+10=14，且 check[14]=13，则“阿拉伯人”为一状态。同时 base[14]<0，则“阿拉伯人”为可结束状态，将 *currentWordCount* = 4 加入 resultList 中。返回 resultList 中的最大值 4。则“阿拉伯人”为一词语。此时分词结束。

表 3-2 按照算法所生成的 Double-Array

下标	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Base	-1	4	4	0	0	0	0	4	-9	4	-11	-12	-4	-14
Check	0	0	0	0	0	0	0	2	2	2	3	8	10	13
词缀	啊	阿	埃					阿根	阿胶	阿拉	埃及	阿根廷	阿拉伯	阿拉伯人

3.5.5 算法测试及结果

基于 Double-Array Trie 的中文分词测试同样采用了 Java 实现由第四章可看出，本算法的实现是以 Lucene 的分词模块为基础。同时在测试中，我们也对比了本算法与 Lucene 自带的几种分词算法。在算法的测试中，采用文献[2]的内容作为分词内容。值得注意的是，分词内容并非是原书内容，而是做了部分修改。因为修改的内容对分词不会造成影响，所以在此忽略修改的具体细节。以下为分词内容的片断：

……浩如烟海的互联网信息推动了搜索引擎的普及和应用，从而也促进了搜索引擎技术的蓬勃发展……表可见一斑……针对不同的应用需求，搜索引擎可能就会有一种新的、更加广阔的发展空间。我们拭目以待，同时也在不断追求…… 国互联网信息的历史内容，还能开展各种研究工作。语言学，经济学，社会学，新闻学…… [2]

以下为测试环境：

硬件环境：

CPU：Mobile AMD Sempron 3000+(1.79GHz)；

RAM：768 MB；

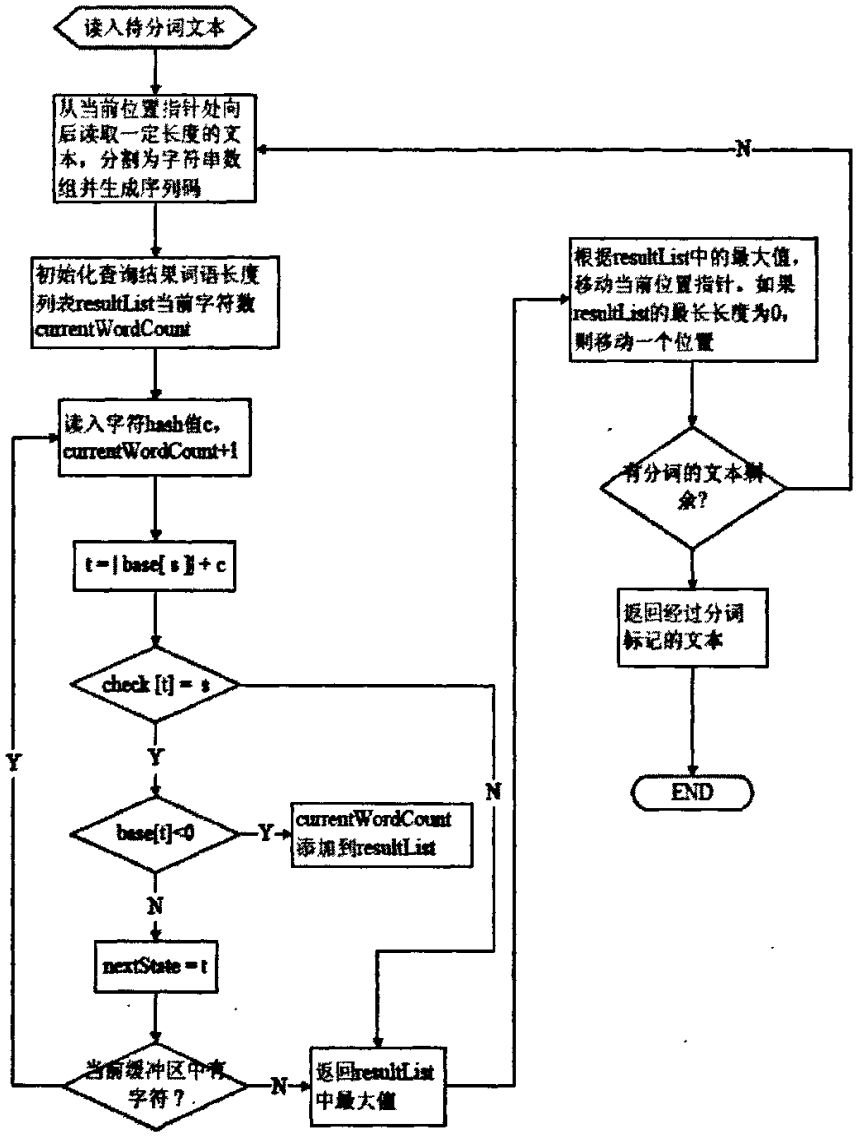


图 3-9 最长前缀分词过程

软件环境：

OS：	Windows XP SP2；
Java Virtual Machine：	Sun JDK 1.6.0，BEA JRockit 5.0，BEA JRockit 6.0；
JVM switches：	（-server）-Xms100M -Xmx200M；
Packages：	lucene-core-2.0.0.jar，commons-logging-1.1.jar，*etc.*

经过不同的 JDK 及不同的分词算法对比测试之后，结果如表 3-3、表 3-4 所示。其中文档字数为 1228514 个（*其中包括英文字符*），词典词数为 195830。从表 3-3 中可看出：

（1）尽管在不同的 JDK 与不同的 JVM Switch 下的分词速度不同且以 Sun jdk 1.6 与“-server -Xms100M -Xmx200M”的组合速度最快，但每种组合都有较快的速度；

（2）不论是 Sun 还是 BEA 的 JDK，1.6 版本的性能明显优于 1.5 版本。

表 3-3 不同分词算法对比测试结果

算法	分词结果
Whitespace	[我们拭目以待]
Chinese	[我][们] [拭] [目] [以] [待]
CJK	[我们][们拭] [拭目] [目以] [以待]
Standard	[我][们][拭][目][以][待]
本文算法	[我们][拭目][以待]

表 3-4 基于 Double-Array Trie 的中文分词算法对比测试结果

JDK	Switch(es)	运行时间（ms）	速度（字/秒）
Sun jdk 1.6	-Xms100M -Xmx200M	3656	336027
Sun jdk 1.6	-server -Xms100M -Xmx200M	2766	444148
BEA jrockit 5.0	-server -Xms100M -Xmx200M	9109	134868
BEA jrockit 5.0	-Xms100M -Xmx200M	8765	140161
BEA jrockit 6.0	-Xms100M -Xmx200M	6438	190822
BEA jrockit 6.0	-server -Xms100M -Xmx200M	6641	184989

而从表 3-3 中可明显看出，本文中的基于 Double-Array Trie 的中文分词算法在

准确度明显高于 Lucene 自带的算法。结合表 3-4 则知：本文的分词算法无论是在分词准确度还是在性能方面都有较出色的表现。

3.6 用户模型

个性化是本文的一个重要方面。本课题拟为用户提供个性化的知识管理、个性化的服务推荐等方面的个性化内容。这些服务都必须以掌握用户兴趣为前提。

一些信息过滤系统中，多以关键词来构造用户模型，那么包含这些词的信息就认为是满足用户兴趣的信息，即为相关信息。使用这种用户兴趣表示方法，常常会造成不正确的匹配。因为这些关键词往往无法清晰地表达兴趣，一个词可能有多种含义（多义现象），相同的概念又能用不同的词来表述（同义现象）。

本文提出一种以用户知识库中最能代表用户兴趣的知识作为用户兴趣表示的方法。因为在 U-CLASS 中，用户的知识主要是以用户所存储的文件来表示。所以，本文是通过聚类的方式找到最能代表用户兴趣的一个或者多个文档的特征作为用户兴趣的表示。

在文献[29][30]中已经在文档与用户兴趣的关系进行讨论，本文的算法与它们有所不同，主要体现在：

1. 应用环境不同：[29][30]主要用于 Web 环境，本文主要用于用户知识管理；前者主要是 HTML 格式，后者存在多个形式；
2. 聚类的算法不同：在本文采用基于并行模糊蚂蚁的聚类算法，这有 别于[29]中所采用的算法；
3. 用户兴趣生成有所不同：本文除了对用户知识进行处理之外，还为对用户行为等进行处理，进而生成用户兴趣。

本文的用户兴趣模型包括了个人兴趣模型和公共兴趣模型。前者是只针对个人，仅仅影响个人服务，如：个性化搜索结果、个性化推荐等；后者是为所有用户服务，它来源于所有用户的知识及行为等（不侵犯用户个人隐私），可作为个人用户兴趣模型的初始值，也可作为公共资源推荐的依据之一。本文将在后面两节做详细讨论。

3.6.1 个人兴趣模型

建立用户兴趣比较直接的做法是提取文件信息中具有代表性的特征[31]。用户兴趣是多方面的，可以根据用户文件信息中最具有代表性的词汇作为。该方法需要

一个训练的过程，首先从预定义的主题词（Concept）表中选取词来描述训练文档，为每个词创建一个分类器，新文档将被每个分类器处理，对该文档有意义的词就赋予该文档。这样用户兴趣可以表示为一个主题词向量$\bar{u}=(w_1,w_2,\cdots,w_n)$，其中$w_i$表示第$i$个主题词出现的权重。向量的维数$n$一般是固定的，这样就保证了文档和用户兴趣之间相似性计算的精度。

不过，预先定义好主题词表需要做大量的工作，而且其覆盖的范围也有限，更简单的做法就是直接利用从文档中抽取的词来表达用户兴趣。该方法不局限于预定义的主题词表，向量的维数一般是不固定的，当然也可以指定一个固定的大小。

在本文中，采用的方式与上述常用方式有所不同，是上述方法与数据聚类及用户行为相结合进而提取用户兴趣的一种新的方法。它更能适应U-CLASS在线存储，并具有更好的性能。下面将先介绍文件信息表示方式，然后再介绍这种新的方法。

3.6.1.1 数据结构

在采用主题词向量$\bar{u}=(w_1,w_2,\cdots,w_n)$来表示用户兴趣的方法中，主题词向量并不能清晰地表达用户兴趣，从而影响正确的匹配。在本文中，采用矩阵方式表达用户兴趣。如下式所示，其中P表示用户的用户兴趣；$\bar{u}_i$表示第i个兴趣；n为$\bar{u}_i$的维数，表示第i个兴趣的主题词的个数；$w_{ij}(i\in[1,m],j\in[1,n])$表示第$i$个兴趣中第$j$个主题词的权重。从上述可看知，用户可有$m$个兴趣。

$$P=\begin{bmatrix}\bar{u}_1\\ \bar{u}_2\\ \vdots\\ \bar{u}_m\end{bmatrix} \tag{3-30}$$

$$\bar{u}_i=(w_{i1},w_{i2},\cdots,w_{in}),\quad \|\bar{u}_i\|=1,\quad i\in[1,m]$$

3.6.1.2 建立

用户个人兴趣的建立相对简单，当用户文件数量较多且高于文件数量门限$\eta_{threshold}$时，系统才采用前文所述的并行模糊蚂蚁进行聚类处理。从简约和效率角度考虑，仅从中提取顶层聚类结果的中心作为用户个人兴趣的初始值。具体算法如下：

输入：经过预处理的多个用户文件信息

输出：用户个人兴趣的矩阵P

过程：

1. 并行模糊蚂蚁聚类算法对文件信息进行聚类，但仅仅做顶层处理，同时提取聚类中心；
2. 从上述结果中提取文件信息聚类中心 $\vec{u}_i$（$\|\vec{u}_i\|=1$，$i\in[1,m]$）；
3. 将 $\vec{u}_i$ 合并成为矩阵 $P=\begin{bmatrix}\vec{u}_1\\ \vec{u}_2\\ \vdots\\ \vdots\end{bmatrix}$。

3.6.1.3 更新

在用户兴趣模型建立以后，可以允许用户主动更新，也可以通过跟踪用户的行为进行动态更新。在本文中，主要采用后者，即根据用户的不同动作产生不同的更新；同时用户的搜索关键词也是用户兴趣模型动态更新的重要数据来源。下面将依次对上述两种方式进行说明。

i. 用户行为

用户的动作包括上载文件、下载文件、在线浏览、发送给朋友（通过邮件方式）、共享给朋友（通过 U-CLASS 平台共享）等。这些动作体现用户不同的兴趣，因而具有不同的意义，如表 3-5 所示。

表 3-5 用户不同的行为所对应的意义

用户行为	重要性
下载文件	♥♥♥
在线浏览	♥♥
发送给朋友	♥
共享给朋友	♥

在表 3-5 中，“♥”表示用户行为的重要性，数量越多就越重要。在更新过程中，重要性会转换为一定量的数值。假定用户 *user* 当前的动作为 *action*，其对应的重要性值为 b_{action}；用户动所针对的文件为 *file*，其重要性为 $b(file)$，其特征向量为 $\vec{d}_{file}$；用户查询的关键词组成的向量为 $\vec{q}$；α 是学习率，为一较小常量；c 为一常量；$\gamma_{threshold}$ 为用户兴趣与文件信息或查询关键词的相似度门限，低于门限则表示相似度太低。用户行为对系统的行为分为两类：

1. 动作源于搜索结果中的文件

此情况可根据以下式子对用户兴趣 P 及文件的重要性进行调整：

$$\begin{aligned}
&\bar{u}_{i_{\max}} = \max_i(similarity(\bar{u}_i, \bar{q})) \\
&\forall\ \bar{u}_{i_{\max}} >= \gamma_{threshold} \Rightarrow \bar{u}_{i_{\max}} \leftarrow normalize(\bar{u}_{i_{\max}} + \alpha\bar{q} + c) \\
&\forall\ \bar{u}_{i_{\max}} < \gamma_{threshold} \Rightarrow \text{add } \bar{q} \text{ to } P \text{ as a new row} \\
&b(file) \leftarrow b(file) + \alpha b_{action} + c
\end{aligned} \tag{3-31}$$

2. 动作源于浏览列表中的文件

此情况可根据以下式子对用户兴趣 P 及文件的重要性进行调整：

$$\begin{aligned}
&\bar{u}_{i_{\max}} = \max_i(similarity(\bar{u}_i, \bar{d}_{file})) \\
&\forall\ \bar{u}_{i_{\max}} >= \gamma_{threshold} \Rightarrow \bar{u}_{i_{\max}} \leftarrow normalize(\bar{u}_{i_{\max}} + \alpha\bar{d}_{file} + c) \\
&\forall\ \bar{u}_{i_{\max}} < \gamma_{threshold} \Rightarrow \text{add } \bar{d}_{file} \text{ to } P \text{ as a new row} \\
&b(file) \leftarrow b(file) + \alpha b_{action} + c
\end{aligned} \tag{3-32}$$

从上述可知，当用户的动作源于浏览文件时，文件的重要性会作相应的调整。在本系统中，文件的重要性会影响文件的排序，重要性也大，搜索结果则可能相应排在前面。

ii. 搜索关键词

用户在搜索的时候，一般都需要关键词进行查询。在上节中，已经对当用户对搜索结果中文件产生动作的情况已经作了描述。此节主要针对用户进行了搜索，但是没有搜到任何结果或者搜索到结果但用户没有任何动作的情况。此情况可根据以下式子对用户兴趣 P 进行调整：

$$\begin{aligned}
&\bar{u}_{i_{\max}} = \max_i(similarity(\bar{u}_i, \bar{q})) \\
&\forall\ \bar{u}_{i_{\max}} >= \gamma_{threshold} \Rightarrow \bar{u}_{i_{\max}} \leftarrow normalize(\bar{u}_{i_{\max}} + \alpha\bar{q} + c) \\
&\forall\ \bar{u}_{i_{\max}} < \gamma_{threshold} \Rightarrow \text{add } \bar{q} \text{ to } P \text{ as a new row}
\end{aligned} \tag{3-33}$$

3.6.2 公共兴趣模型

公共兴趣模型主要所有用户的共同兴趣的表示，可用作个性化推荐和用户个人兴趣的初始化值。考虑到实时性、动态性及用户数量，本文采用一种 Leader 层次算法[30]，算法描述如下：

输入：用户个人兴趣 P 构成的集合及公共兴趣矩阵 P_{common}

输出：更新之后的公共兴趣矩阵 P'_{common}

过程：

（1）将 $P_i(i \in [1, UserNumber])$ 及 P_{common} 的行向量全部添加到用户兴趣向量集 U 中，并初始化聚类 $C=\{\}$。若 P_{common} 为空，则不处理 P_{common}；

（2）对 U 中的每个用户兴趣向量 $\vec{u}$ 寻找聚类 c，使 $\vec{u}$ 和 c 的质心距离(相似性)最短且记为 $d_{\min}$。如果 $d_{\min}$ 小于距离门限 $d_{threshold}$，则将 $\vec{u}$ 加入 c，否则将 $\{\vec{u}\}$ 加入 C。然后质心可以用平均值来计算。设有聚类 $A=\{\vec{u}_1, \vec{u}_1, \cdots, \vec{u}_k\}$，则 A 的质心 m 所对应的向量为：

$$\vec{u}(m)=\frac{\vec{u}_1+\vec{u}_2+\cdots+\vec{u}_k}{k} \tag{3-34}$$

上述聚类算法计算相对简单，可实现动态聚类。如当有一个新的用户个人兴趣向量 $\vec{u}_{new}$ 加入到聚类 A 时，则可用下式对公共用户兴趣进行更新：

$$\vec{u}'(m) \leftarrow \frac{1}{k+1}(k \cdot \vec{u}(m)+\vec{u}_{new}) \tag{3-34}$$

与一般聚类算法相比，这样计算聚类可以节省大量运算时间，从而达到实现动态聚类的目的，提高了聚类具有相似兴趣的用户的准确性，进而形成公共兴趣。

（3）经过步骤（2）处理之后，$C=\{\vec{u}_1, \vec{u}_2, \cdots\}$，将 $\vec{u}_i$ 合并成为矩阵 $P'_{common}=\begin{bmatrix}\vec{u}_1\\ \vec{u}_2\\ \vdots\\ \vdots\end{bmatrix}$。

第四章　系统设计及实现

在对本文的重点—三个重点算法及基础理论做了分析和研究之后，本章将对算法及系统做设计及实现。主要采用自顶向下的方式，从系统架构出发完整地设计整个系统，最后完成系统的编码等实现工作。

4.1 整体架构

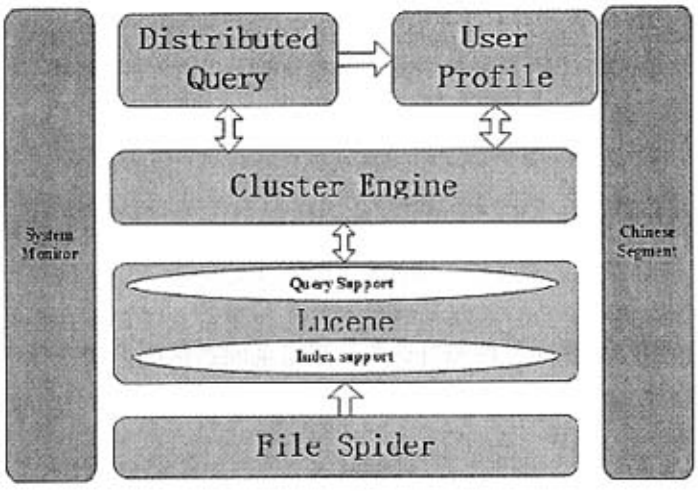


图 4-1 个性化搜索引擎整体架构

如图 4-1 所示，基于模糊信息处理的个性化分布式文件搜索引擎是由各个子模块组成，具有强内聚性、弱耦合性的特点。各个模块具有一定的独立性和完整性，能够独立完成某些任务的同时能在耦合性很小的情况下，能与其它模块共同协作完成系统级任务。

4.1.1 系统特点

1. 由于系统是由纯 Java 编写，并且配合 U-CLASS 独有的安全加密中间件，因此系统具有良好的安全性和稳定性。这是系统正常运行的前提条件和重要条件；

2. 因 U-CLASS 是一个开放的平台，具有设计良好的接口。因而作为 U-CLASS

的核心模块之一，本系统也需要具有良好的开放性、易用性。在系统设计的过程中，在接口设计方面作了一定的工作，使得接口能够完成功能的同时，也具有良好的易用性；

3. 由于 U-CLASS 中，存储的文件量及节点数量不定。而且随着用户的增多，文件量和节点会逐渐增大。这就要求系统需要具有良好伸缩性、负载平衡及容错性。而系统中的分布式查询支持、Lucene 及相关组件组成的集群结构使得本系统在上述要求中具有良好的表现。关于 Lucene 集群，请参考文献[27] [28]；

4. 一个设计良好的系统在易测试性方面必须有出色的表现。本系统以“设计到接口”为标准、以模块为粒度，因而不论是单元测试、还是模块测试都能方便进行。

4.1.2 系统模块分析

从图 4-1 所示，系统由中文分词等模块组成，下面将简要介绍主要模块：

1. 中文分词（Chinese Segment）模块，负责中文文本的分词，是整个系统的基础，贯穿了整个系统，对整个系统的正常运行和性能有着重要的作用；

2. 系统监控（System Monitor）模块负责监控整个系统运行的健康状况，并负责动态调整系统运行的参数。如：系统并行处理的文件数、系统当前正在处理文件数等；

3. 文件 Spider（File Spider）是系统数据的主要来源之一，负责处理系统存储的文件信息的抓取，是用户知识管理的基础。系统存储的各类文件的各类信息，包括文档文件的内容、多媒体文件的元数据等都是由它负责分析、处理，然后将分析得到的初步数据提交给 Lucene，然后 Lucene 将对信息进行索引及后续工作；

4. Lucene 是整个系统的核心模块之一，它负责对文件信息进行加工、索引，同时对文件信息的搜索也有它作为核心支持。由于 Lucene 的重要作用，本文将在下一节对其作详细的介绍；

5. 聚类引擎（Cluster Engine）是本系统中的另外一个核心模块，它负责对用户知识的分类、用户查询结果的分类，它的处理结果将是用户兴趣采取模块的数据来源之一。本文也将在后文作详细介绍；

6. 用户兴趣（User Profile）模块是用户兴趣采集、用户个性化服务的核心。它负责从用户知识、用户查询、用户个人信息中获取用户兴趣数据，然后经过一定处理之后，提取用户兴趣，为个性化服务提供基础数据；

7. 分布式查询（Distributed Query）模块为本系统的重要功能之一。它使得不同节点的数据能够融合起来，使系统用户所能得到的知识得到大幅提高，即分布式查询是整个系统分布式构架的基础；

8. 个性化推荐（Personalized Recommendation）模块（未在图中表示）是本系统个性化服务的一个重要方面，它向用户推荐与用户兴趣相关的公共资源及共享资源，从而不需要用户的额外工作的情况下，即可获得丰富的知识。

4.1.3 知识流图

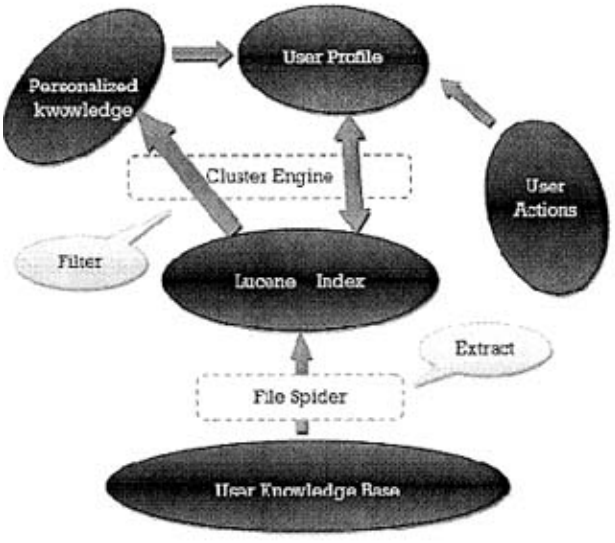


图 4-2 个性化搜索引擎知识流图

由图 4-2 可看出用户知识（User Knowledge Base）在整个系统中的流向。首先文件 Spider 从用户的知识库(即用户在 U-CLASS 中所存储的文件)中抽取(Extract)知识，经过初步分析之后，传递给 Lucene 作进一步处理并建立索引（Index），进而作为用户知识库在搜索平台中的存储形式。

在用户知识完成转换之后，用户即可从处理之后的知识库中搜索所需要的知识。当用户查询时，查询结果会被聚类引擎结合用户兴趣进行过滤（Filter），然后展示给用户的是个性化处理之后的知识（Personalized Knowledge）。

值得注意的是，用户兴趣（User Profile）的数据来源于聚类引擎的处理结果、用户查询关键字以及用户的行为（User Actions）。详情将在个人兴趣和文本聚类的章节进行介绍。

4.2 Lucene 框架分析

Apache Lucene[24]是一个高性能、纯 Java 的全文检索引擎，而且免费、开源。它几乎适合于任何需要全文检索的应用，尤其是跨平台的应用。虽然 Lucene 不是一个完整的全文索引应用，但是由于它由纯 JAVA 实现，它可以方便的嵌入到各种应用中实现针对应用的全文索引/检索功能。

本文利用 Lucene 作为搜索的核心组件。索引的建立、查询等都由 Lucene 完成。它是整个个性化搜索引擎的基础，其重要性在图 4-1 也可看出。

4.2.1 基本原理

在 Lucene 发展的早期，作者将其发布在自己的网站 www.lucene.com 之上，后来将其发布在 SourceForge，2001 年年底成为 Apache 基金会 Jakarta 的一个子项目，并于 2005 年脱离 Jakarta 而成为 Apache 的顶级子项目。

由于良好的性能等原因，Lucene 已经有很多 Java 项目都使用了 Lucene 作为其后台的全文索引引擎，比较著名的有：

- Jive：WEB 论坛系统；
- Eyebrows：邮件列表 HTML 归档/浏览/查询系统，本文的主要参考文档“The Lucene search engine: Powerful, flexible, and free”作者就是 EyeBrows 系统的主要开发者之一，而 EyeBrows 已经成为目前 APACHE 项目的主要邮件列表归档系统。
- Cocoon:基于 XML 的 发布框架，全文检索部分使用了 Lucene
- Eclipse:基于 Java 的开放开发平台，帮助部分的全文索引使用了 Lucene

在国内使用 Lucene 比较著名的有：

- 搜房引擎：http://search.soufun.com/
- WebLucene: http://sourceforge.net/projects/weblucene/

对于中文用户来说，最关心的问题是其是否支持中文的全文检索。但通过后面对于 Lucene 的结构的介绍，你会了解到由于 Lucene 良好架构设计，对中文的支持只需对其语言词法分析接口进行扩展就能实现对中文检索的支持。在目前 Lucene 发布的版本中，中文已经得到了良好的支持。但在中文分词方面功能较弱，在本文中采用了 Double-Array Trie 分词方法较好的解决了这个问题，使得 Lucene 在个性化文件搜索中能够出色地完成其工作。

4.2.2 实现机制

Lucene 的 API 接口设计的比较通用，输入输出结构都很像数据库的表->记录->字段。所以很多传统的应用的文件、数据库等都可以比较方便的映射到 Lucene 的存储结构/接口中。从总体上来看，可以把 Lucene 当成一个支持全文索引的数据库系统。试比较一下 Lucene 和数据库，表 4-1 可以看出，全文检索并不等于 like "%keyword%"。而且在很多方面，前者表现较后者优秀很多。

表 4-1 Lucene 与数据库结构对比

Lucene	数据库
索引数据源：doc(field1，field2...) doc(field1，field2...) \ indexer / _____ | Lucene Index| _____ / searcher \ 结果输出： Hits (doc (field1， field2) doc (field1...))	索引数据源：record(field1，field2...) record(field1..) \ SQL: insert/ _____ |DB Index | _____ / SQL: select \ 结果输出： results (record (field1，field2..) record (field1...))
Document： 一个需要进行索引的"单元" 一个 Document 由多个字段组成	Record： 记录，包含多个字段
Field： 字段	Field： 字段
Hits： 查询结果集，由匹配的 Document 组成	RecordSet： 查询结果集，由多个 Record 组成

通常比较厚的书籍后面常常附关键词索引表（比如：北京：12， 34 页，上海：3，77 页……），它能够帮助读者比较快地找到相关内容的页码。而数据库索引能够大大提高查询的速度原理也是一样，相像一下通过书后面的索引查找的速度要

比一页一页地翻内容快多少。而索引之所以效率高，另外一个原因是它是排好序的。对于检索系统来说核心是一个排序问题。

表 4-2 Lucene 全文查询与数据库的模糊查询对比

	Lucene 全文索引引擎	数据库
索引	将数据源中的数据都通过全文索引一一建立反向索引	对于 LIKE 查询来说，数据传统的索引是根本用不上的。数据需要逐个便利记录进行正则表达式的方式进行模糊匹配，比有索引的搜索速度要有多个数量级的下降。
匹配效果	通过词元(term)进行匹配，通过语言分析接口的实现，可以实现对中文等非英语的支持	较差
匹配度	有匹配度算法，将匹配程度（相似度）比较高的结果排在前面	没有匹配程度的控制。比如：有记录中 net 出现 5 词和出现 1 次的，结果是一样的
结果输出	通过算法，将最匹配度最高的头 100 条结果输出，结果集是缓存的方式而小批量读取的	返回所有的结果集，在匹配条目非常多的时候（比如上万条）需要大量的内存存放这些临时结果集。
可定制性	通过不同的语言分析接口实现，可以方便的定制出符合应用需要的索引规则（包括多语言支持）	没有接口或接口复杂，无法定制
结论	高负载的模糊查询应用，需要负责的模糊查询的规则，索引的资料量比较大	使用率低，模糊匹配规则简单或者需要模糊查询的资料量少

由于数据库索引不是为全文索引设计的，因此，使用 like “%keyword%”时，数据库索引是不起作用的，在使用 like 查询时，搜索过程又变成类似于一页页翻书的遍历过程了，所以对于含有模糊查询的数据库服务来说，LIKE 对性能的危害是极大的。如果是需要对多个关键词进行模糊匹配：like”%keyword1%” and like “%keyword2%” ...其效率也就可想而知了。

表 4-3 Lucene 与数据库结构对比

	Lucene	其他开源全文检索系统
增量索引和批量索引	可以进行增量的索引(Append)，可以对于大量数据进行批量索引，并且接口设计用于优化批量索引和小批量的增量索引	很多系统只支持批量的索引，有时数据源有一点增加也需要重建索引
数据源	Lucene 没有定义具体的数据源，而是一个文档的结构，因此可以非常灵活的适应各种应用(只要前端有合适的转换器把数据源转换成相应结构)	很多系统只针对网页，缺乏其他格式文档的灵活性
索引内容抓取	Lucene 的文档是由多个字段组成的，甚至可以控制那些字段需要进行索引，那些字段不需要索引，近一步索引的字段也分为需要分词和不需要分词的类型： 需要进行分词的索引，比如：标题，文章内容字段；不需要进行分词的索引，比如：作者/日期字段	缺乏通用性，往往将文档整个索引了
语言分析	通过语言分析器的不同扩展实现，可以去 Stop words 及亚洲语言支持	缺乏通用接口实现
查询分析	通过查询分析接口的实现，可以定制自己的查询语法规则	
并发访问	能够支持多用户的使用	

所以建立一个高效检索系统的关键是建立一个类似于科技索引一样的反向索引机制，将数据源（比如多篇文章）排序顺序存储的同时，有另外一个排好序的关键词列表，用于存储关键词==>文章映射关系，利用这样的映射关系索引：[关键词==>出现关键词的文章编号，出现次数（甚至包括位置：起始偏移量，结束偏移量），出现频率]，检索过程就是把模糊查询变成多个可以利用索引的精确查询的逻辑组合的过程。从而大大提高了多关键词查询的效率，所以，全文检索问题归结到最后是一个排序问题。

由此可以看出模糊查询相对数据库的精确查询是一个非常不确定的问题，这也是大部分数据库对全文检索支持有限的原因。Lucene 最重要的特征是通过特殊的索引结构实现了传统数据库不擅长的全文索引机制，并提供了扩展接口，以方便针对不同应用的定制。

可以通过表 4-2 可看出二者的差别。

4.2.3 框架创新

大部分的搜索（数据库）引擎都是用 B 树结构来维护索引，索引的更新会导致大量的 IO 操作，Lucene 在实现中，对此稍微有所改进：不是维护一个索引文件，而是在扩展索引的时候不断创建新的索引文件，然后定期的把这些新的小索引文件合并到原先的大索引中（针对不同的更新策略，批次的大小可以调整），这样在不影响检索的效率的前提下，提高了索引的效率。Lucene 和其他一些全文检索系统/应用的比较，如表 4-3 所示。

4.3 基于 Double-Array Trie 的中文分词算法

4.3.1 模块结构

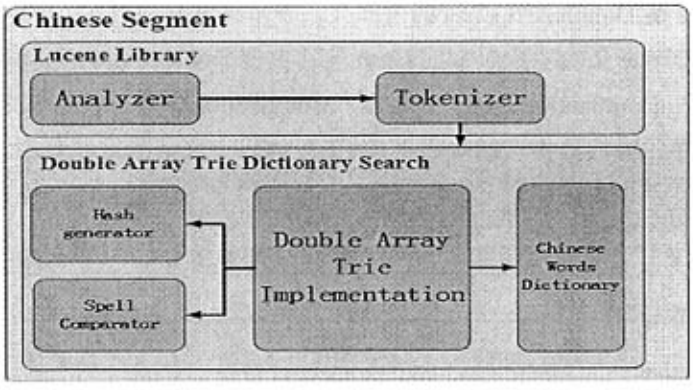


图 4-3 中文分词模块结构

如图 4-1 所示，中文分词模块是整个系统的基础。同时如图 4-3 所示，中文分词模块是在 Lucene Library 的基础之上进行了扩展，并使用了 Double-Array Trie 词典查询算法，从而实现基于 Double-Array Trie 的中文分词算法。在图 4-3 中，Analyzer 通过分析输入文本，从而构建 Token Stream。它表示了从文本中抽取索引

元（Index Term）的策略，通常实现是构建一个 Tokenizer，然后将字符流转换成为 Token。而在一个 Analyzer 中可以包含一个或者多个 Filter。在本文中，DoubleArrayAnalyzer 继承了 Lucene Library 中的 Analyzer，覆盖了 tokenStream 方法。在 tokenStream 方法中，DoubleArrayAnalyzer 使用了 DoubleArrayTokenizer 和 StopWordsFilter，前者负责分词处理，后者负责对输入文本中的停用词（Stop words）进行过滤。而 DoubleArrayTokenizer 继承了 Tokenizer，覆盖 next 方法，并在其中实现分词算法。

4.3.2 主要类分析

图 4-6、4-7 展示了中文分词模块各个主要类的 UML 结构及关联。其中：

1. DoubleArrayTrie 是词典查询算法的主要实现类，它负责构建词典（build）、保存查询树（save）、读取查询树（load）、完全匹配查询（search）及最长前缀查询（commonPrefixSearch）等，如表 4-4 所示；

2. DoubleArrayTrieFactory 是 DoubleArrayTrie 的工厂，根据不同的策略负责实例化 DoubleArrayTrie。目前因为 DoubleArrayTrie 在系统中仅需要一个实例，所以仅需要一个静态实例并 Lazily 加载就可以了。而 buildChineseDoubleArrayTrie 仅仅是对 DoubleArrayTrie 的 load 方法的一个简单封装；

3. Node 是 DoubleArrayTrie 的内部类，表示搜索树节点；

4. GBKCode 负责生成汉字的 Hash 值，方便 Double-Array Trie 的生成；

5. PinYinComparator 是一个比较器（Comparator），用于汉字按照汉语拼音的顺序进行排序，而汉语拼音的顺序正式 GBK 编码中汉字的排序顺序。

表 4-4：DoubleArrayTrie 中的主要方法

方法名称	返回类型	可见性	是否静态	注释
DoubleArrayTrie		public	✖	构造函数
load	void	public	✖	读取查询树
build	int	public	✖	构造查询树
search	int	public	✖	完全匹配查询
commonPrefixSearch	int	public	✖	最长前缀查询
save	void	public	✖	保存查询树
encodeChineseCharacter	int	private	✓	取汉字的 Hash 值

4.3.3 类交互

由图 4-4 可看出中文分词模块各个主要类的交互。当 Lucene Core 需要对输入文本进行分析时，它向 DoubleArrayTrieAnalyzer 发送"analyze text"的消息。后者收到消息之后，便向 DoubleArrayTrieTokenizer 发送分词消息。DoubleArrayTrieTokenizer 会按照分词算法向 DoubleArrayTrie 传递词查询消息，之后 DoubleArrayTrie 会将查询结果回传给 DoubleArrayTrieTokenizer。在 DoubleArrayTrieTokenizer 处理完输入文本之后，会将经过分词处理的结果发送给 DoubleArrayTrieAnalyzer。DoubleArrayTrieAnalyzer 然后再进行去除停用词（Stop words）等后续处理。

4.4 基于并行模糊蚂蚁的文本聚类算法

4.4.1 模糊蚂蚁

模糊蚂蚁是真实蚂蚁的抽象，在算法中主要用于模糊推理和数据项的搬运。它的结构如下图 4-5 所示。

4.4.2 Board model

蚂蚁移动的板的模型主要用于数据项的放置及蚂蚁的移动。其结构如图 4-6 所示。

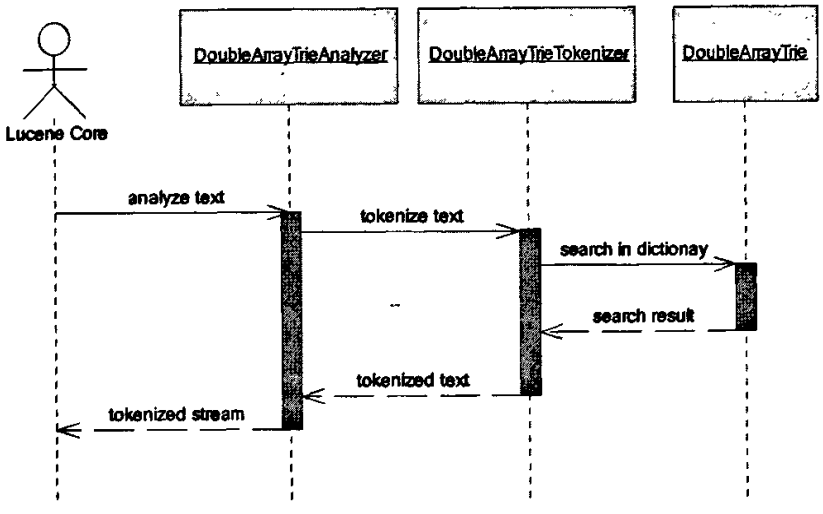


图 4-4 中文分词模块序列图

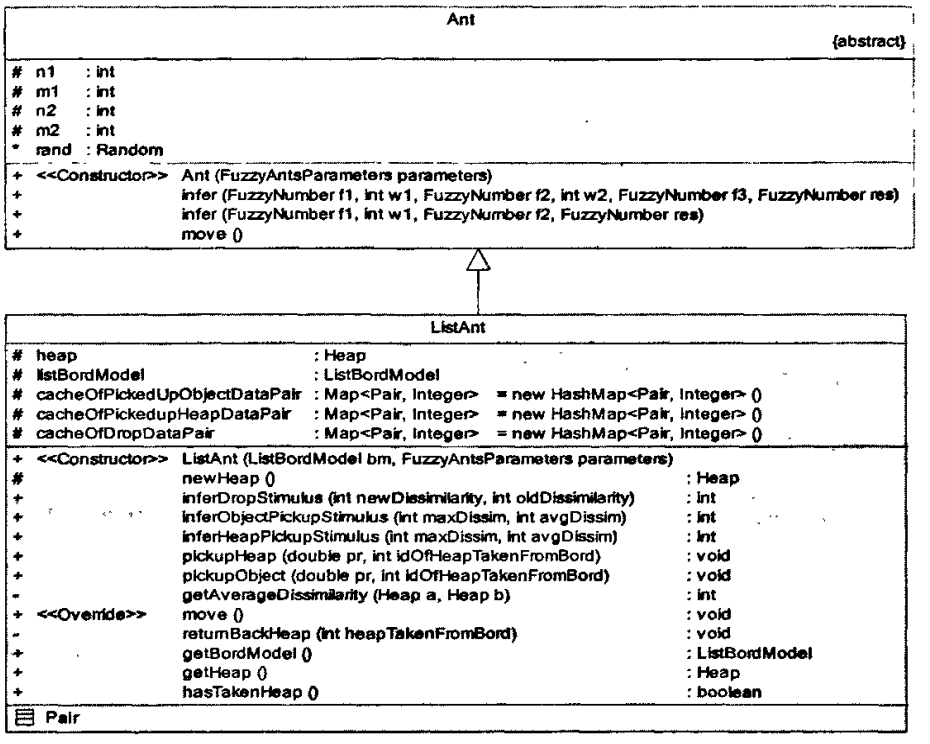


图 4-5 模糊蚂蚁结构

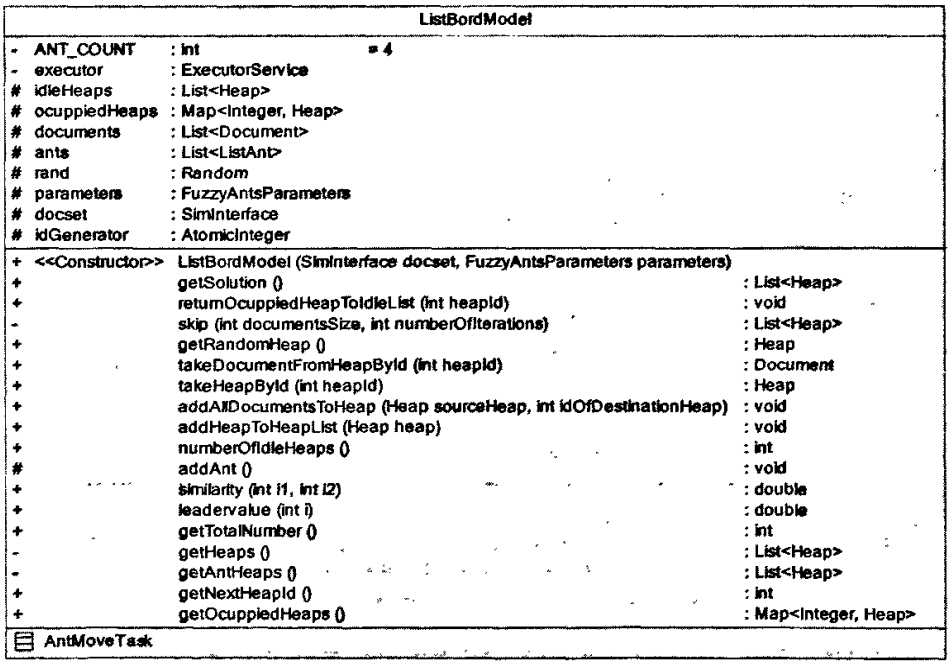


图 4-6 Board Model 结构

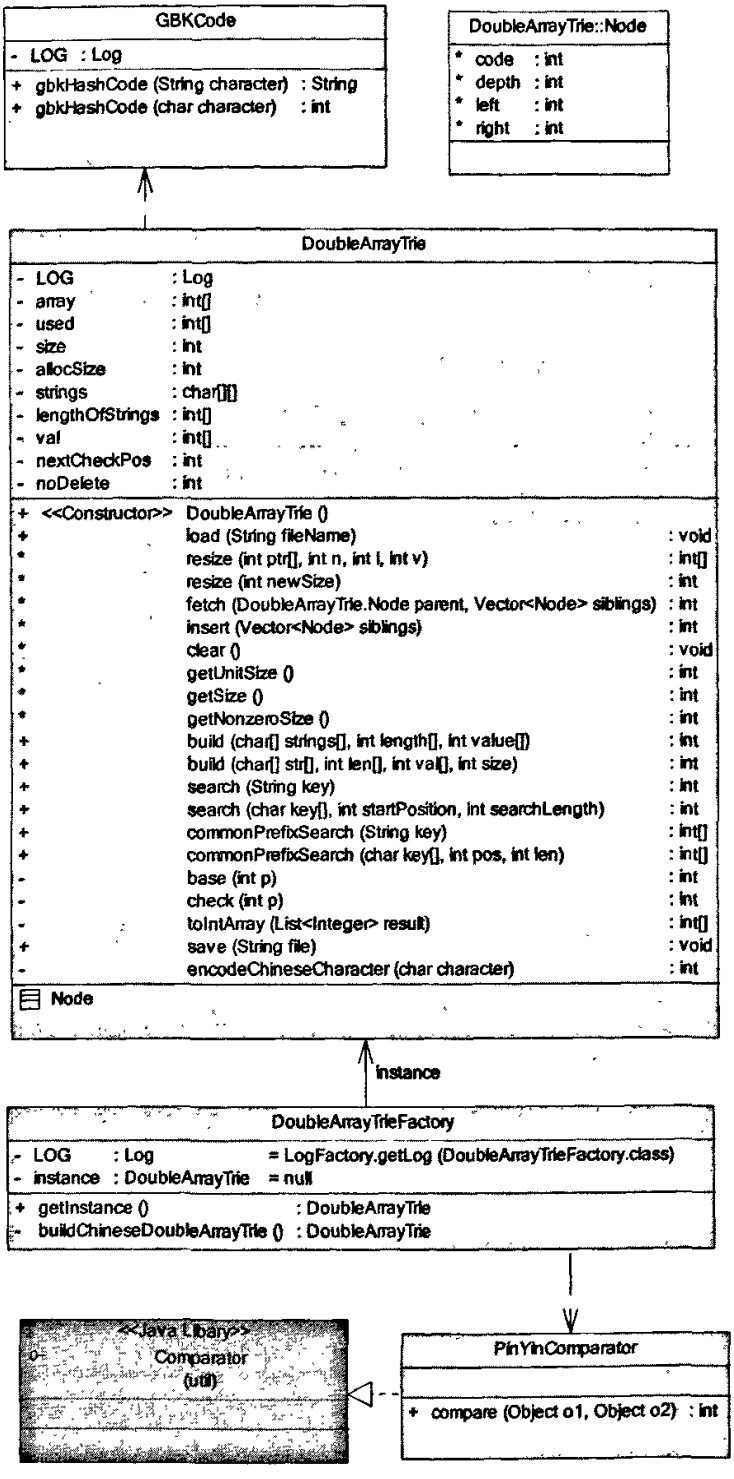


图 4-7 中文分词模块类图

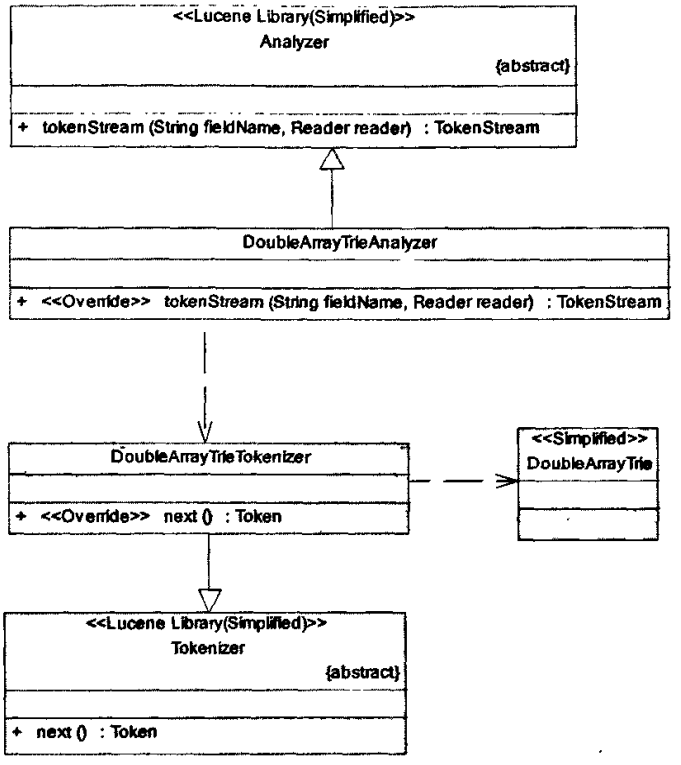


图 4-8 中文分词模块类图（续）

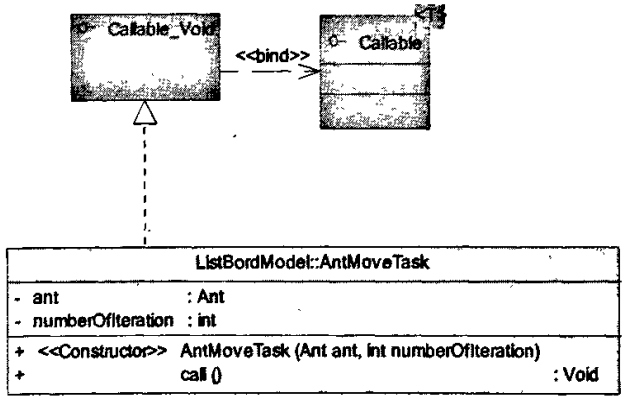


图 4-9 AntMoveTask 的结构

4.4.3 线程池与任务

由于多个蚂蚁是并行在板（Board）上行走，且每个蚂蚁执行一个任务，则需要对每个任务进行并行运行的控制。在本文中，采用 JDK 中的线程池[38]及相关工具对线程进行控制。下面是并行执行的执行流程：

（1）建立线程池；

（2）建立 *CompletionService*

（3）建立蚂蚁移动的任务；

（4）将任务提交给 *CompletionService*，并有线程池中的线程执行；

（5）等待线程结束，关闭线程池。

其中 AntMoveTask 的结构如图 4-9 所示。

4.4.4 Idle list 与 occupied list

在此算法中，idle list 和 occupied list 具有相当的重要性，对系统的性能影响较大。它们存放的数据会被并行访问，进而存在同步的问题。因而同步带来的性能问题是值得考虑的重要问题。

经过测试和比较之后，同时考虑二者存放的数据结构，本文选择了 Vector 与 ConcurrentHashMap[38]分别实现 idle list 和 occupied list，在时间和空间上取得了较好的均衡。

4.4.5 缓存（Cache）

由于在本算法中，在计算数据项之间的相似性时，计算量大，对某些数据项的计算上可能存在重复计算的情况。所以采用缓存是非常有必要的，可以大大提高系统性能。在缓存的实现上，也存在并行访问的问题。同时从简单和效率角度考虑，采用 key-value 的方式进行缓存。在实现上，仍然采用 ConcurrentHashMap。在对采用缓存和未采用进行对比之后发现，采用缓存能大大的提升性能。

4.5 基于 Lucene 的文本索引及查询设计及实现

4.5.1 索引结构设计及实现

下面对本系统中索引结构作简单、部分说明。在表 4-5 中，Store 表示是否存储该字段（Field）的原始值；Tokenize 是否对该字段进行分词；Position 表示是否存储分词之后的位置信息。值得注意的是，表 4-5 仅列出文件信息中的主要部分，并不是完整索引结构。

4.5.2 分布式支持

本文采用 RMI、WebService 与 Lucene 的分布式支持相结合的方式，并自定义了 RMI 的传输协议，加强了安全性。关于 Lucene 分布式支持，请参阅文献[39]；关于 RMI，请参考文献[37]；关于自定义 RMI 的传输协议和加密请参考文献[40][41]。

表 4-5 索引结构

Field 名称	意义	Store	Tokenize	Position
FILE_ID	文件 id	✓	×	×
FILE_RELATIVE_PATH	文件相对路径	✓	×	×
FILE_NAME	文件名	✓	✓	✓
FILE_EXTENTION_NAME	文件扩展名	✓	×	×
FILE_LAST_MODIFIED_DATE	文件最后修改时间	✓	×	×
FILE_CONTENT	文件内容	✓	✓	✓

4.6 基于多线程的文件 Spider 的设计及实现

4.6.1 模块结构

如图 4-10 所示，文件扫描器（File System Scanner）扫描用户存储的文件，并将文件打开，从中提取基本的文件信息，同时形成数据流并提交给内容抓取模块（File Content Crawler Module）；内容抓取模块中的各种格式的文件内容抓取器（Crawler）自动针对不同格式的文件进行内容或者元素据抓取；抓取之后同步提交给文件信息（阻塞）队列（File Information (Blocked) Queue）；在文件内容抓取的同时，内容处理模块（Content Processor Module）从文件信息（阻塞）队列中同步提取文件信息。值得注意的是，内容处理模块完成了索引等工作，在本文其它部分作了讨论，在此不再赘述。

从文件 Spider 的结构图可看出，这是一个 Producer-Consumer 模型的扩展，下

面本文将对 Producer-Consumer 模型作介绍，并分析它在本文中的实现概要。

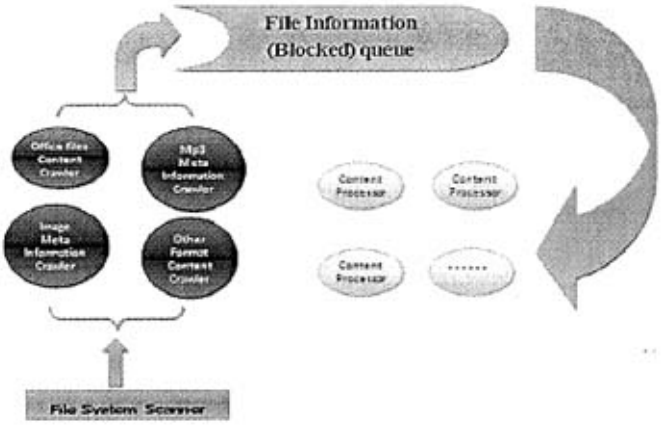


图 4-10　File Spider 结构

4.6.2　Producer-Consumer 模型

4.6.2.1　模型

如图 4-11 所示，生产者-消费者（Producer-Consumer）模型有生产者、消费者及信息队列构成。为了使结构更清晰，本文不对原始模型做解释，而是结合文件 Spider 作说明。

从图左边可以看出，有多个内容抓取器在并行的提取文件内容，它们即是生产者；然后文件内容被并行送往文件信息队列，由于该队列是阻塞式队列，生产者和消费者均可并行访问它（在不同的情况下，该队列有不同的实现方式）；在图的右边是内容处理器。它们负责从文件信息队列中并行获取文件信息，然后对它们进行处理。即文件内容处理器为消费者。

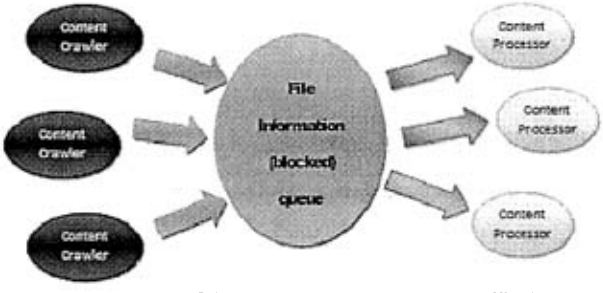


图 4-11　Producer-Consumer 模型

4.6.2.2 实现

由于本系统采用纯 Java 实现，则文件 Spider 也不例外。下面对模型的三部分分别作实现上的说明：

（1）内容抓取器：

内容抓取器和文件内容处理器均是多线程，每个处理任务都是由单独的线程完成。结构如图 4-12 所示。

（2）文件信息（阻塞）队列：

从上节可看出，文件信息队列是一个接口 *Puttable*，仅定义了方法 *put(Object item)* 和 *offer(Object item, long msecs)*。*put* 方法是直接放入队列，但可能等待直到队列可以接受为止；而 *offer* 方法和 *put* 基本功能相同，不同之处在于后者 *offer* 只等待 msecs 这段时间，否则放弃。

在文件 Spider 中，*Puttable* 接口由 *BoundedLinkedQueue*[34]（有限长链阻塞队列）实现。

（3）文件内容处理器：

与内容抓取器相似，结构如图 4-12 所示。

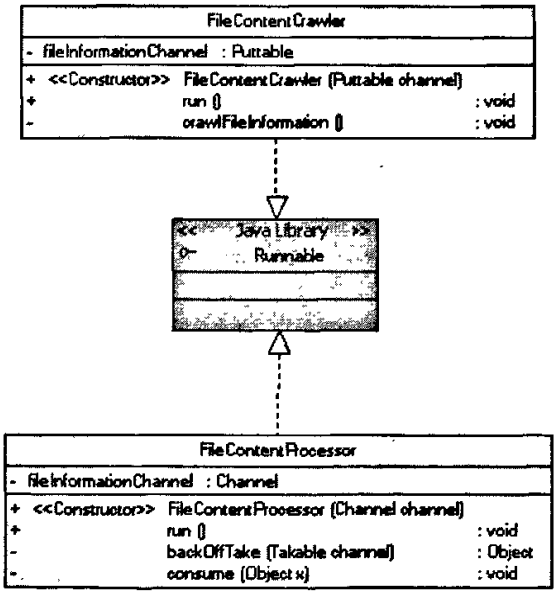


图 4-12 文件 Spider 部分结构

4.7 系统实时监控的设计

4.7.1 JMX 介绍

JMX（Java Management Extensions，即 Java 管理扩展）的前身是 JMAPI，是一个为应用程序、设备、系统等植入管理功能的框架[33]。它可以跨越一系列异构操作系统平台、系统体系结构和网络传输协议，灵活的开发无缝集成的系统、网络和服务管理应用。

JMX 是一个为应用程序植入管理功能的框架。JMX 是一套标准的代理和服务，实际上，用户可以在任何 Java 应用程序中使用这些代理和服务实现管理。

JMX 致力于解决分布式系统管理的问题，因此，能够适合于各种不同的环境是非常重要的。为了能够利用功能强大的 Java 计算环境解决这一的问题，Sun 公司扩充了 Java 基础类库，开发了专用的管理类库。

JMX 是一种应用编程接口，可扩充对象和方法的集合体，可以用于跨越一系列不同的异构操作系统平台、系统体系结构和网络传输协议，灵活的开发无缝集成的系统、网络和服务管理应用它提供了用户界面指导、Java 类和开发集成系统、网络及网络管理应用的规范。

管理对象是 JMX 应用程序的核心。JMX 结构包括：支持 Java 的 Web 浏览器用户接口，管理运行模块 ARM(Admin Runtime Module)和应用。这三个部件之间通过 RMI（Remote Method Invocation）进行通信。这里需要说明的是，RMI 是使得一个 Java 虚拟机（JVM）上运行的程序可以调用远程服务器上另一个 JVM 总的对象。

用户接口用来发布管理操作，这些操作可以间接的通过浏览器或通过单独的应用程序来激发。管理运行模块用来给应用提供实例化的管理对象。它包括 Agent 对象接口，通知接口和被管数据接口。应用指的是那些被管设备单元。

JMX 是一个完整的网络管理应用程序开发环境，它同时提供了：厂商需要收集的完整的特性清单，可生成资源清单表格，图形化的用户接口；访问 SNMP 的网络 API；主机间远程过程调用；数据库访问方法。

JMX 这一轻型的管理基础结构，价值在于对被管理资源的服务实现了抽象，提供了低层的基本类集合，开发人员在保证大多数的公共管理类的完整性和一致性的前提下，进行扩展以满足特定网络管理应用的需要。

JMX 注重于构造管理工具的软件框架，并尽量采用已成熟的技术。

JMX 可以用来管理网络、设备及应用程序等资源，其架构如图 4-13 所示。它具有以下优点：

1. 可以非常容易的使应用程序具有被管理的功能；
2. 提供具有高度伸缩性的架构；

   每个 JMX Agent 服务可以很容易的放入到 Agent 中，每个 JMX 的实现都提供几个核心的 Agent 服务，你也可以自己编写服务，服务可以很容易的部署，取消部署。
3. 主要提供接口，允许有不同的实现。

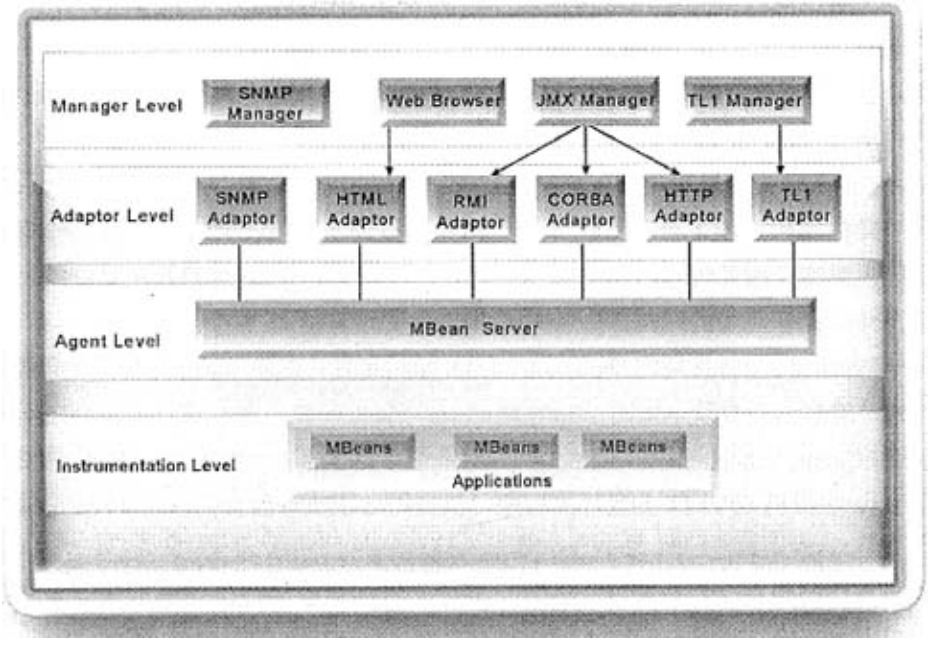


图 4-13　JMX 架构

4.7.2 系统实时监控

由于 JMX 的灵活性，要利用 JMX 建立实时监控组建进而监控系统健康状况，只需做些简单的工作即可。在此，只对文件监控作简要说明，详情略去。

1. 建立一个接口，展示需要接受监控的功能：

   在此，需要将当前文件 Spider 中正在处理的文件数及已经处理完成的文件数展示出来。首先建立接口，如图 4-14 所示；

2. 建立一个实现类，实现监控功能，如图 4-14 所示；
3. 将 MBean 注册到 MBeanServer，则 MBean 发布完成，JMX 的可访问的客户端都可访问当前监控的内容了。

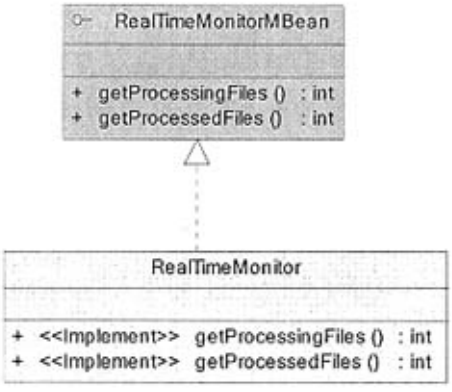


图 4-14　文件监控部分结构

4.8 接口设计

4.8.1 WebService 类型接口

WebService 的主要目标是跨平台的可互操作性。为了达到这一目标，WebService 完全基于 XML（可扩展标记语言）、XSD（XMLSchema）等独立于平台、独立于软件供应商的标准，是创建可互操作的、分布式应用程序的新平台。由此可以看出，在下列情况下，使用 WebService 则具有相当的优势：

（1） 跨防火墙的通信；

（2） 应用程序集成；

（3） B2B 的集成；

（4） 软件和数据重用；

但在下列情况下，使用 WebService 的理由则显得薄弱：

（1） 单机应用程序；

（2） 局域网的同构应用程序

在本文中，由于 U-CLASS 中，核心部分是基于 J2EE 的。但是在外围部分及其它功能组建可能运行于不同的平台，如：基于 Linux 平台的 PHP 环境等；同时

也存在与其它服务提供商进行业务集成等。即在 U-CLASS 中存在 WebService 具有优势的四种情况，所以在接口开发中，首先考虑 WebService。

而在个性化搜索系统中，需要跟 U-CLASS 对接，同时需要将搜索结果提供给 U-CLASS 的其它组件，因此也使用 WebService 作为接口开发的首选。

XFire[35]是一款开源的 Java Web Service 框架。它拥有一个轻量级的信息处理模块，通过 STAX 来与 SOAP 信息相结合。XFire 的简单易用的 API 使得面向服务的开发变得更加容易。XFire 支持绝大多数的 Web Service 标准，如：SOAP、WSDL、WS-I Basic Profile、 WS-Addressing、WS-Security 等。

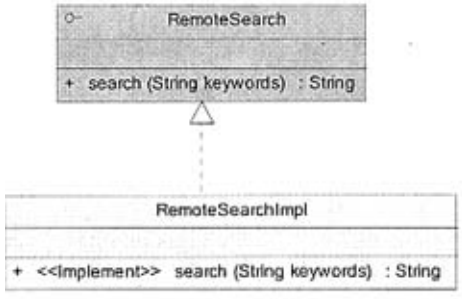


图 4-15 WebService 实现部分结构

Spring[36] 则是当下非常流行的 J2EE 框架，由于它的工厂非常强大且灵活，以 IoC 和 Bean 管理而闻名于 Java 界。但在此不对它作详细的介绍，请参阅文献[36]。Spring 作为 IoC 框架且使用了它的其它集成的组件，包括了多 XFire 的默认支持。

下面以一个简要阐述基于 WebService 的接口设计，详细配置方法请参考文献[36]的 XFire-Spring 集成的章节。

（1） 设计接口，如图 4-15 所示；

（2） 实现接口，如图 4-15 所示；

（3） 配置 XFire 的配置文件及 Spring 的配置文件；

（4） 部署到服务器上之后，则可通过 SearchService 调用该 WebSearvice

4.8.2 RMI 类型接口

RMI[37]是 Remote Method Invocation（远程方法调用）的缩写。它大大增强了 Java 开发分布式应用的能力。Java 作为一种风靡一时的网络开发语言，其巨大的威力就体现在它强大的开发分布式网络应用的能力上，而 RMI 就是开发百分之百纯 Java 的网络分布式应用系统的核心解决方案之一。其实它可以被看作是 RPC 的

Java 版本。但是传统 RPC 并不能很好地应用于分布式对象系统。而 Java RMI 则支持存储于不同地址空间的程序级对象之间彼此进行通信，实现远程对象之间的无缝远程调用。因为 RMI 是基于 Java 的分布式编程模型，所以使用 RMI 进行远程方法调用时，无须考虑方法底层的网络传输细节。但实现 RMI 调用的程序和被调用的方法，都必须是 Java 代码，即客户端和服务器端都必须通过纯 Java 实现。

一个正常工作的 RMI 系统由下面几个部分组成：

（1）远程服务的接口定义

（2）远程服务接口的具体实现

（3）桩（Stub）和框架（Skeleton）文件

（4）一个运行远程服务的服务器

（5）一个 RMI 命名服务，它允许客户端去发现这个远程服务

（6）类文件的提供者

（7）一个需要这个远程服务的客户端程序

与其它的远程调用方式相比，RMI 具有如下的优势：

（1） 面向对象；

（2） 可移动属性；

（3） 安全；

（4） 便于编写和使用；

（5） 可连接现有/原有的系统；

（6） 编写一次，到处运行；

（7） 分布式垃圾收集；

（8） 并行计算。

在本文中，RMI 的使用主要用于分布式查询的支持，基于安全性及性能等方面的考虑，具体的实现细节在此省略。关于 RMI 的实现方式请参看文献[37]。

4.9 界面设计及实现

如图 4-11 所示，左边为搜索面板，右边为搜索结果。此为界面的雏形，在进一步的设计中，会采用更加人性化的方式进行。


test 文件夹
按下面任意或所有条件进行搜索。
全部或部分文件名(O):
文件中的一个字或词组(W):
查找范围(L):
当前目录
查找类型(T):
全部类型
后退(B)
搜索(R)
数据仓库和...
成华区556...

图 4-16　搜索界面

第五章 系统测试

5.1 测试概念及方法

5.1.1 概念

软件测试方法之所以没能完全标准化和统一化，主要原因是因为软件产业产品到软件测试有各式各样的软件[42]。但也存在基本的常用概念和方法。下文提及的概念和方法几乎都是通用的。其中，以下几个概念都是软件测试中非常重要的概念：白箱测试、黑箱测试、灰箱测试、有效用例与无效用例、边界条件及等价类测试。下面将结合本文的系统，分别对上述几个概念作简要解释：

（1）白箱测试（White-box testing）是通过程序的源代码进行测试而不使用用户界面。这种类型的测试需要从代码句法发现内部代码在算法、溢出、路径、条件等中的缺点或者错误，进而加以修正。在本文中，由于系统采用 Java 开发，所以在白箱测试以单元测试的方式进行，且以 JUnit 作为测试的基础框架；

（2）黑箱测试（Black-box testing)是通过使用整个软件或软件功能来严格地测试，而并没有通过阅读程序的源代码或者清楚地了解该软件或软件功能的源代码程序具体的设计或实现。测试人员通过输入他们的数据然后看输出的结果从而了解软件怎样工作。通常测试人员在进行测试时不仅使用肯定出正确结果的输入数据，而且还会使用有挑战性的输入数据以及可能结果会出错的输入数据以便了解软件怎样处理各种类型的数据。在本文中，黑箱测试主要体现在功能测试方面。测试进行时，按照功能点对系统进行测试；

（3）灰箱测试或灰盒测试（Gray-box testing）。此方式介于白箱和黑箱测试之间。测试人员不但以黑箱的方式进行测试，同时测试人员还可能了解软件的结构，甚至可能熟悉设计或者源代码。因此测试人员可以有的放矢地进行某种确定的条件/功能的测试。这样做的意义在于：如果你知道产品内部的设计和对产品有透过用户界面的深入了解，测试人员就能够更有效和深入地从用户界面来测试它的各项性能；

（4）有效用例(Valid case)是指那些已知软件能正确地处理的测试用例。一般是指软件输入的测试用例。无效用例（Invalid case）是指那些事先就知道软

件不支持处理的测试用例；

（5）边界条件（Boundary Cases）：环绕边界值的测试。通常意味着最大值，最小值或者所设计软件能够处理的最长的字符串等；

（6）等价类（equivalent classes）是指如果有很多测试用例执行再多也不会找到新的缺陷。因为虽然输入和输出结果有所不同，但是它们都通过同样的软件的源代码路径。通常只要一个源代码程序的路径是用于处理一定数值范围内的所有数值，那么除了边界值以外，在边界值范围以内的所有数值一般都属于等价类。因为如果软件程序能正确处理一个值，也就意味着该程序能正确处理在这个范围内的除了边界值以外的其他任何有效输入值。

5.1.2 方法

本文所指的基本软件测试方法主要侧重于软件功能的黑箱测试方法：功能测试（Functionality Test）、用户界面(User interface 或 UI)测试、边界条件测试(Boundary Condition)、性能测试(Performance Test)、回归测试(Regression Test)、压力测试(Stress Test)、配置和安装测试(Configuration and Setup Test)及兼容性测试（Comparability Test）等。下面将对上述方法作简要分析解释：

（1）功能测试：验证测试软件功能能否正常按照它的设计工作。看运行软件时的期望行为是否符合原设计；

（2）用户界面测试：分析软件用户界面的设计是否合乎用户期望或要求。它常常包括菜单，对话框及对话框上所有按钮，文字，出错提示，帮助信息(Menu 和 Help content）等方面的测试；

（3）边界条件测试：指环绕边界值的测试。通常意味着测试软件各功能是否能正确处理最大值，最小值或者所设计软件能够处理的最长的字符串等；

（4）性能测试：指通常验证软件的性能在正常环境和系统条件下重复使用是否还能满足性能指标，或者执行同样任务时新版本不比旧版本慢。一般还检查系统内存在程序运行时会不会泄露（memory leak）；

（5）回归测试：根据修复好了的缺陷再重新进行的测试。目的在于验证以前出现过但已经修复好的缺陷不再重新出现。一般指对某已知修正的缺陷再次围绕它原来出现时的步骤重新测试。通常确定所需的再测试的范围时是比较困难的，特别当临近产品发布日期时。因为为了修正某缺陷时必需更改源代码，因而就有可能影响这部分源代码所控制的功能。所以在验证修好的缺陷时不仅要服从缺陷

原来出现时的步骤重新测试，而且还要测试有可能受影响的所有功能。因此应当鼓励对所有回归测试用例进行自动化；

（6）压力测试：它通常验证软件的性能在各种极端的环境和系统条件下是否还能正常工作。或者说是验证软件的性能在各种极端环境和系统条件下的承受能力；

（7）集成与兼容性测试：验证该功能能够如预期的那样与其他程序或者构件协调工作。兼容性经常意味着新旧版本之间的协调，也包括测试的产品与其它产品的兼容使用。比如用同样产品的新版本时不影响与用旧版本用户之间保存文件，格式，和其他数据等操作。

5.2 功能测试

5.2.1 白箱测试

白箱测试即为程序员测试。因为程序员知道被测试的软件如何（How）完成功能和完成什么样（What）的功能。在本文中，采用基于 JUnit 框架的单元测试方式完成白箱测试。下面将对 JUnit 作简要分析。

JUnit 本质上是一套测试框架，即开发者制定了一套规则。遵循这些规则而编写测试代码，如继承某个类，实现某个接口，就可以用 JUnit 进行自动测试了。

由于 JUnit 相对独立于所编写的代码,所以测试代码的编写可先于实现代码的编写。则 XP 中推崇的 Test First Design 的实现可由一下步骤完成：

（1） 基于 JUnit 的测试代码；

（2） 实现代码；

（3） 运行测试；

（4） 测试失败；

（5） 修改实现代码；

（6） 再运行测试，直到测试成功；

（7） 以后对代码的修改和优化，运行测试成功，则修改成功。

Java 下的 team 开发，采用 cvs(版本控制)、ant(项目管理)、junit(集成测试)相结合的模式时，通过对 ant 的配置，可以很简单地实现测试自动化。在本系统的开发过程中，采用的是 IDEA 集成开发环境与 cvs 相结合的方式进行开发。由于 IDEA 集成了对 cvs、ant、junit 等的支持，使得开发变得容易且愉快。

由于本系统涉及的代码较多，所有的白箱测试不能一一罗列。下面将以本文4.3节中的基于Double-Array Trie的中文分词算法中的DoubleArrayTrie（如图4-6所示）的单元测试为例，对基于JUnit的白箱测试做较为详细的分析：

（1） 在需求分析的基础上（如4.3节所述），确定DoubleArrayTrie需要完成的功能。然后在此基础上，从JUnit出发，编写测试代码DoubleArrayTrieTest（如图5-1所示）；

（2） 以（1）中的功能及测试要求，书写DoubleArrayTrie的代码（UML类图如图4-6所示）；

（3） 编译、运行测试代码，如果通过测试，则任务完成；否则需要修改代码，直到全部测试通过为止。基于IDEA的测试运行结果如图5-2，5-3所示。

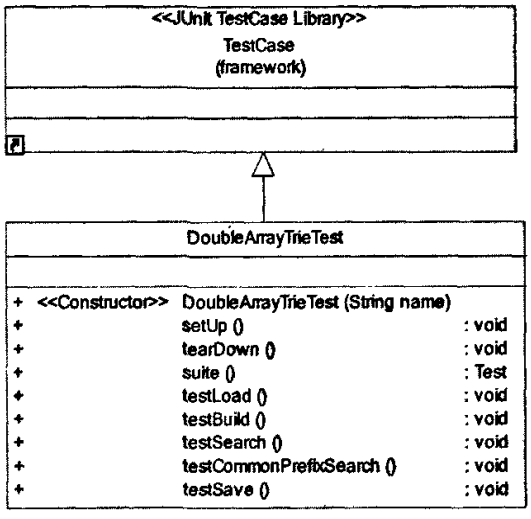


图5-1 DoubleArrayTrieTest结构

Done: 5 of 5 (0.015 s)

Output | Statistics

Test	Time elapsed	Usage Delta	Usage Before	Usage After	Results
Total:	0.0 s	17 Kb	1,059 Kb	1,093 Kb	P:5
testBuild	0.0 s	0 Kb	1,076 Kb	1,076 Kb	Passed
testCommonPrefixSearch	0.0 s	0 Kb	1,076 Kb	1,076 Kb	Passed
testLoad	0.0 s	17 Kb	1,059 Kb	1,076 Kb	Passed
testSave	0.0 s	0 Kb	1,093 Kb	1,093 Kb	Passed
testSearch	0.0 s	0 Kb	1,076 Kb	1,076 Kb	Passed

图5-2 IDEA测试运行结果

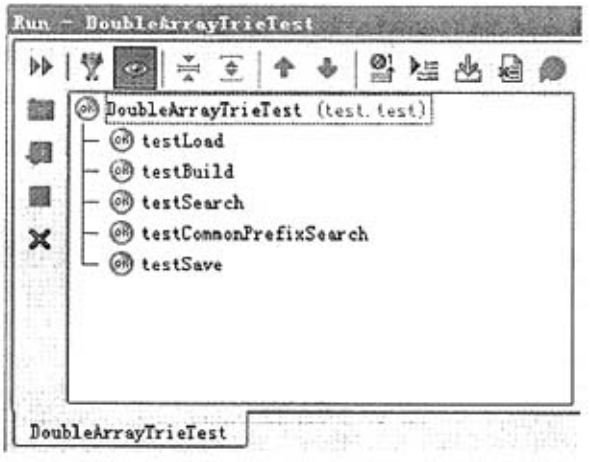


图 5-2 IDEA 测试运行结果（续）

5.2.2 黑箱测试

黑箱测试在本文中主要以功能测试为主。因为本系统涉及的功能较多，由于篇幅限制，不能列出所有功能。下文将给出“DoubleArrayTrie 中文词语查询”测试用例，从中可以看出本系统黑箱测试的过程。

表 5-1 DoubleArrayTrie 中文词语查询测试用例

用例描述	DoubleArrayTrie 中文词语查询				
用例目的	输入中文词语，则可查询该关键字是否存在于词典中				
前提条件	词典已经准备就绪				
步骤	输入/动作	期望输出/响应	实际情况	状态	执行时间
1	待查询中文词语	该查询词语是否存在于词典中	该查询词语是否存在词典中	通过	2007-3-8

5.3 压力测试

压力测试是测试软件系统在不同类型的环境中的负载能力。本系统是基于 JEE 的系统，压力测试在包括了并发搜索任务处理数量、并发处理的文件内容的数量、系统总共处理文件的数量等。下面本文将结合两种不同的压力测试工具分析本系统的压力测试方法。

(1) Microsoft Web Application Stress

它是为了实现模拟多浏览器对Web应用页面请求的压力测试工具，是由Web测试人员开发且在隐藏了Web服务器测试复杂度的基础上，使得它尽可能地方便使用。它对于搜集网站性能数据的用户是较为有用。Microsoft Web Application Stress秉承了Microsoft的一贯作风—简单易用。只需要做简单的设置即可开发测试，如图5-5所示。当然它提供的功能就相对简单了。测试结果（样例）如图5-4所示。

(2) Mercury LoadRunner

LoadRunner是一种预测系统行为和性能的工业标准级负载测试工具。通过以模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题。它能够对整个企业架构进行测试。通过使LoadRunner，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

LoadRunner是一种适用于各种体系架构的自动负载测试工具，它能预测系统行为并优化系统性能。LoadRunner的测试对象是整个企业的系统，它通过模拟实际用户的操作行为和实行实时性能监测，来帮助您更快的查找和发现问题。此外，LoadRunner能支持广范的协议和技术，可以为特殊环境提供特殊的解决方案。

LoadRunner的功能与Microsoft Web Application Stress相比则强大得多且复杂得多。鉴于复杂性及篇幅，本文不读其作详细分析。请参考LoadRunner相关文献。

```
Overview
--------------------------------------------------
Report name:                    2007-3-12 21:38:00
Run on:                         2007-3-12 21:38:00
Run length:                     00:00:10

Web Application Stress Tool Version:1.1.293.1

Notes
--------------------------------------------------
Sample Microsoft Web Application Stress Script

Number of test clients:         1

Number of hits:                 254
Requests per Second:            25.41

Socket Statistics
--------------------------------------------------
Socket Connects:                256
Total Bytes Sent (in KB):       52.09
Bytes Sent Rate (in KB/s):      5.21
Total Bytes Recv (in KB):       3192.68
Bytes Recv Rate (in KB/s):      319.44

Socket Errors
--------------------------------------------------
Connect:                        0
Send:                           0
Recv:                           0
Timeouts:                       0

RDS Results
--------------------------------------------------
Successful Queries:             0
```

图5-4 Microsoft Web Application Stress压力测试结果

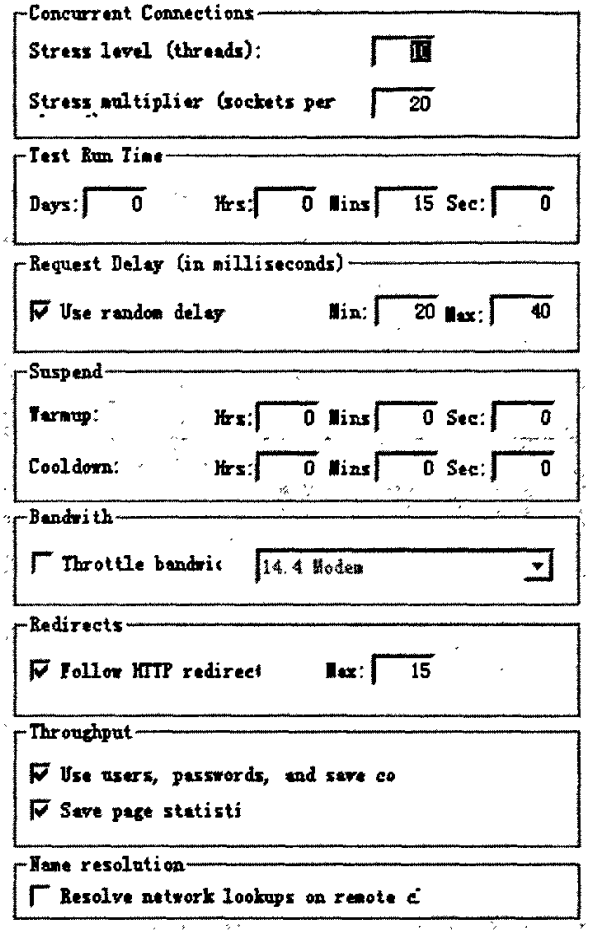


图 5-5 Microsoft Web Application Stress 设置界面

第六章 总结及展望

本文从需求分析、理论研究、系统设计及实现等方面做了较为详细的解释及说明，旨在给出一个清晰明了的基于模糊信息处理的个性化分布式文件搜索引擎。该搜索引擎是在挖掘用户的个人存储在 U-CLASS 的系统中的知识的基础之上，再结合用户的查询信息及用户行为等方面，使得用户能够在概览知识全局的同时，快速地定位自己需要的知识点。同时系统还能够自动挖掘用户个人兴趣、公共兴趣及知识自动分类等。

由于本文是学位论文，重在理论分析。因此，本文在系统需求分析、系统设计及系统实现等方面的内容就显得较为简略。同时，在理论分析方面，由于本文限于篇幅限制，在部分内容上作了省略。

本文所述的个性化搜索引擎是 U-CLASS 的一部分，功能较多。在本文完成之时，仅完成其中一部分。在本课题的后续工作中，将继续搜索引擎的工作。具体如下表所示：

工作名称	后续方向
需求分析	继续挖掘潜在的需求
并行模糊蚂蚁聚类算法	完善代码
中文分词	完善代码，提高性能
文件 Spider	完善代码
索引、查询	改进异构系统支持
多格式文件信息挖掘	添加更多文件格式支持
用户模型	改进模型，使用户兴趣识别更为精确
人性化界面	更为人性化界面的设计及实现

致 谢

研究生三年，毕业在即。

首先感谢我的导师张扬副教授。他为人和蔼可亲、理论基础深厚、治学态度严谨、工作热情饱满，都给我留下了深刻的影响，深深地影响了我；

其次感谢生我养我的父母；感谢重病的父亲，在病床上也不忘教育我注意身体健康、用心做人；感谢老母无尽的爱。虽然不认识几个字，但是任劳任怨，奉献的只有爱；

其次感谢我的爷爷奶奶。虽然仙逝多年，但是他们教育我的善良、纯朴、顶天立地的做人风格，永记心中；

其次感谢舅舅舅妈在七年中，在生活和做人方面的谆谆教诲；

然后感谢教研室各位兄弟姐妹。潘汉怀的博学、Conanson(高远)的严谨、肖治宇的沉稳、Bruce（严俊）的幽默洒脱、张莉的可爱豪气、张航的细心、郭世民的成熟等都深深地影响了我。感谢你们在研究生三年中在生活和学习中的帮助；

然后感谢各路江湖朋友。感谢马力遥等网优互联的同事们在课题和团队协作上的指导；感谢成都杰创科技的同事在技术方面的指导及提供的良好实习环境；

最后感谢在老父病重时，照顾老父的亲戚朋友们。

如果没有你们，就没有这篇文章的完成。再次感谢你们。祝张老师身体健康、工作顺利、家庭幸福；祝老父身体早日康复；祝老母身体健康；祝舅舅舅妈健康开心；祝教研室的兄弟姐妹前途光明、完美毕业；祝各路江湖朋友开心快乐。

参考文献

[1] 薛万新. 中文搜索引擎的现状与发展. 科技情报开发与经济, 2005 年第 15 卷第 3 期: 266-267.

[2] 李晓明, 闫宏飞, 王继民. 搜索引擎 — 原理、技术与系统. 北京:科学出版社, 2004.

[3] Page Lawrence, Brin Sergey, Motwani Rajeev, et al.. The PageRank Citation Ranking: Bringing Order to the Web.

[4] http://www.google.com.

[5] 搜索引擎发展史. http://www. se-express. com/about/about. htm. 2007/3/24.

[6] 中科院报告称三大搜索引擎仍不能满足用户需求. http://tech. sina. com. cn/i/2006-09-05/10031119910. shtml. 2007/3/24.

[7] 桌面盛宴：主流桌面搜索工具横向评测. http://tech. sina. com. cn/s/2005-03-23/0952558799. shtml. 2007/03/24.

[8] Benchmark Study of Desktop Search Tools. http://www.uwebc.org/decisiontools/. 2007/03/24.

[9] Laurianne McLaughlin. What's Next in Web Search?. IEEE Distributed Systems Online, no. 11,vol. 5,2004.

[10] 李士勇, 等. 蚁群算法及应用. 哈尔滨：哈尔滨工业大学出版社，2004.

[11] 李洪兴, 汪培庄. 模糊数学. 北京:国防工业出版社，1994.

[12] 彭祖赠, 孙韫玉. 模糊数学及其应用. 武汉:武汉大学出版社, 2002.

[13] 郑亚林等. Fuzzy 推理的 Mamdani 算法. 宝鸡文理学院学报, 第 21 卷, 第 3 期，2001 年 9 月:168-173.

[14] 戎月莉. 计算机模糊控制原理及应用. 北京:北京航空航天大学出版社, 1995:10-18.

[15] P. Kanade. Fuzzy ants as a clustering concept. M.S dissertation. University of South Florida, Tampa,FL.2004.

[16] S. Schockaert, M. De Cock, C. Cornelis, etc. Fuzzy Ant Based Clustering. Ant Colony Optimization and Swarm Intelligence,4th International Workshop (ANTS 2004),LNCS 3172:342-349.

[17] Valeri Rozin, Michael Margaliot. The Fuzzy Ant. IEEE International Conference on Fuzzy

Systems Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21[C] , 2006:1679-1686.

[18] 李东, 张湘辉. 汉语分词在中文软件中的广泛应用. 微软中国研究开发中心. http://www.microsoft.com/china/rdcenter/info/result/chinese.asp. 2007/03/24.

[19] 李江波, 周强, 陈祖舜. 汉语词典快速查询算法研究. 中文信息学报, 第 20 卷, 第 5 期, 2006:31-39.

[20] 王思, 张华平, 王斌. 双数组 Trie 树算法优化及其应用研究. 中文信息学报, 第 20 卷, 第 5 期, 2006:24-30.

[21] Aoe, J. An Efficient Digital Search Algorithm by Using a Double-Array Structure. IEEE Transactions on Software Engineering. Vol. 15, 9 (Sep 1989):1066-1077.

[22] JUN-ICHI AOE, KATSUSHI MORIMOTO. An Efficient Implementation of Trie Structures. Software-Practice and Experience.1992,22 (9):695-721.

[23] Theppitak Karoonboonyanan. An Implementation of Double-Array Trie. http://linux.thai.net/~thep/datrie/datrie.html.2006

[24] http://lucene.apache.org/.

[25] http://wiki.apache.org/lucene-java/PoweredBy.

[26] 车东. 在应用中加入全文检索功能——基于 Java 的全文索引引擎 Lucene 简介. http://www.chedong.com/tech/lucene.html. 2006

[27] Lucene Clustering.http://www.terracotta.org/confluence/display/labs/Lucene+Clustering. 2006/05/26

[28] Lucene in a cluster. http://bugs.sakaiproject.org/confluence/display/SEARCH/IndexClusterOperation. 2006/05/26

[29] 李健, 马力, 武波. 一种基于 Web 文本聚类的用户兴趣发现模型的研究. 现代电子技术, 第 23 期总第 190 期, 2004:10-14.

[30] 吴良杰, 刘红祥, 张立堃等. 个性化服务中网页推荐模型的研究. 计算机应用研究, 第 6 期, 2005:83-85.

[31] 曾春, 邢春晓, 周立柱. 基于内容过滤的个性化搜索算法. 软件学报, 第 14 卷, 第 5 期, 2003:999-1004.

[32] DIK L. LEE, HUEI CHUANG, KENT SEAMONS. Document Ranking and the Vector-Space Model. Software, IEEE Volume 14, Issue 2, Mar/Apr 1997:67-75.

[33] JMX. http://www.itisedu.com/phrase/200604261751455.html.2007/3/24.

[34] Puttable.
http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html.
2007/03/24.

[35] http://xfire.codehaus.org/

[36] http://www.springframework.org/

[37] http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

[38] Package concurrent discription.
http://java.sun.com/j2se/1.6.0/docs/api/java/util/concurrent/.

[39] Class RemoteSearchable discription.
http://lucene.apache.org/java/docs/api/org/apache/lucene/search/RemoteSearchable.html.

[40] Krishnan Viswanath.The New RMI. http://today.java.net/lpt/a/230. 2005/06/10.

[41] 创建自定义 RMI 套接字工厂.
http://www.iplab.cs.tsukuba.ac.jp/liuxj/jdk1.2/zh/docs/guide/rmi/rmisocketfactory.doc.html.

[42] 软件测试的基本概念和方法.
http://www.oldchild.net/jsjsj/spks/cps/rjcscjbgnhff.htm.

攻硕期间的研究成果

发表论文：

- 冯周，张扬. 基于并行模糊蚂蚁的聚类算法研究. 中国科技论文在线. 2007/04/02