华夏英才基金学术文库

搜 索 引 擎

— 原理、技术与系统

Search Engine: Principle, Technology and Systems

李晓明 闫宏飞 王继民 著

by Li Xiaoming, Yan Hongfei and Wang Jimin

科 学 出 版 社

2004

内 容 简 介

本书比较系统地介绍了互联网搜索引擎的工作原理、实现技术及其系统构建方案。全书分三篇共 13 章内容，从基本工作原理概述开始，到一个小型简单搜索引擎实现的具体细节，进而详细讨论了大规模分布式搜索引擎系统的设计要点及其关键技术；最后面向主题和个性化的 Web 信息服务，阐述了中文网页自动分类等技术及其应用。本书层次分明，由浅入深；既有深入的理论分析，也有大量的实验数据，具有学习和实用双重意义。

本书可作为高等院校计算机科学与技术、信息管理与信息系统、电子商务等专业的研究生或高年级本科生的教学参考书和技术资料，对广大从事网络技术、Web 站点的管理、数字图书馆、Web 挖掘等研究和应用开发的科技人员也有很大的参考价值。

前　言

随着互联网的不断发展和日益普及，网上的信息量在爆炸性增长，在 2004 年 4 月，全球 Web 页面的数目已经超过 40 亿，中国的网页数估计也超过了 3 亿。目前人们从网上获得信息的主要工具是浏览器，而通过浏览器得到信息通常有三种方式。第一，直接向浏览器输入一个关心的网址（URL），例如 http://net.pku.edu.cn，浏览器返回所请求的网页，根据该网页内容及其包含的超链文字（anchor text）的引导，获得自己需要的内容；第二，登录到某个知名门户网站，例如 http://www.yahoo.com，根据该网站提供的分类目录和相关链接，逐步“冲浪”浏览，寻找自己感兴趣的东西；第三，登录到某个搜索引擎网站，例如 http://e.pku.edu.cn，输入代表自己所关心信息的关键词或者短语，依据返回的相关信息列表、摘要和超链接引导，试探寻找自己需要的内容。

这三种方式各有特点，各有自己最适合的应用场合。第一种方式的应用是最有针对性的，例如要了解北京大学计算机系网络与分布式系统实验室在做些什么工作，从某个渠道得知该实验室的网址为 http://net.pku.edu.cn，于是直接用它驱动浏览器就是最有效的方式。第二种方式的应用类似于读报，用户不一定有明确的目的，只是想看看网上有什么有意思的消息；当然这其中也可能是关心某种主题，例如体育比赛，家庭生活等等。第三种方式适用于用户大致上知道自己要关心的内容，例如“国有股减持”，但不清楚哪里能够找到相关信息（即不知道哪些 URL 能给出这样的信息）；在这种场合，搜索引擎能够为用户提供一个相关内容的网址及其摘要的列表，由用户一个个试探看是否为自己需要的。现在的搜索引擎技术已经能做到在多数情况下满足用户的这种需要。CNNIC 的信息统计指出，目前搜索引擎已经成为继电子邮件之后人们用得最多的网上信息服务系统。

同时，随着网上信息资源规模的增长，尤其是其内容总体和我们社会的演化发生着越来越密切的联系，研究网上存在的海量信息逐渐成为许多学科关注的一个方向。为此，不少研究人员也有采样搜集特定内容、一定数量网页的需要。

本书以我们设计、实现并维护运行北大“天网”搜索引擎的经验，介绍大规模搜索引擎的工作原理和实现技术。我们要向读者揭示，为什么向搜索引擎输入一个关键词或者短语，就能够在秒钟内得到那么多相关的文档及其摘要，而点击其中的链接就能够被引导到文档的全文，且其中相当一部分可能正是用户需要的。

我们按照上、中、下三篇展开相关的内容。上篇讲搜索引擎的基本工作原理，要解决的是为什么搜索引擎能提供如此信息查找服务的问题，以及它在功能上有什么本质的局限性。这一篇的内容包括网页的搜集过程，网页信息的提取、组织方式和索引结构，查询提交和响应的过程以及结果产生，等

等。这其中，虽然我们假定读者熟悉 URL，HTML，HTTP，CGI，MIME 等基本概念，但在上下文中也给予了必要的介绍，力图保持行文的流畅性。这一部分内容对于需要构建小规模搜索引擎的研究人员会有直接的参考价值。

中篇讨论和大规模实用搜索引擎有关的技术问题。所谓大规模在这里指至少维护超过 1 千万的网页信息，提供相关的查询服务。所涉及的内容包括并行分布处理技术的应用，数据局部性的开发，缓存技术的应用，以及搜集的网页在提供服务之前的预处理问题和高效倒排文件的建立技术等等。这一部分的讨论有比较强的计算机系统结构的风格，我们向读者展示计算机系统结构课程中的那些概念是如何生动地体现在一个实际应用系统中的。这一部分的内容对构建大规模数字图书馆的技术人员也应该有帮助。

下篇介绍挑战性更强一些的内容。一般地讲，前面所述可以称为是“通用搜索引擎”，为最广泛的人群提供信息查询服务是它的基本宗旨。这意味着它的应用模式必须尽量简单，即关键词或查询短语的提交和匹配响应。尽管这已经可以解决许多问题了，但对有些重要的信息需求依然显得力不从心。例如，一个人可能会关心最近半年来网上出现了哪些关于他（她）的信息，一个企业可能要关心它做了一次大规模促销活动后一个月内网上有什么反响，一个政府机构可能会关心在一项政策法规颁布后的网上舆论。面向主题和个性化的信息查询服务就是我们试图描述的一种基本途径。这一部分内容更多的和网上中文信息处理技术有关。更准确地讲，我们要介绍网络与并行分布处理技术与中文处理技术的结合，从而实现大规模、高性能、高质量、有针对性地网上信息查询服务。这一部分内容反过来可能对从事中文信息处理的研究人员有启发作用。

本书的内容是集体智慧的结晶，主要概括了北大计算机系网络与分布式系统实验室自 1996 年以来的研究成果。其中许多段落直接来自同学的博士和硕士论文，他们是雷鸣、赵江华、冯是聪、单松巍、谢正茂、彭波、张志刚、龚笔宏、孟涛、咎红英，等等。署名作者的主要工作是将这些内容系统化，使其表述的风格统一。我们特别感谢陈葆珏教授，是她在北京大学计算机系开创了搜索引擎这一研究方向，从而使我们能在其后发扬光大，还要感谢刘建国和王建勇，是他们分别带领攻关队伍，实现了天网 1.0 和天网 2.0 版本。感谢黄蕊为本书进行的文字校对。最后，我们感谢国家“九五”攻关计划，“973”计划和“985”计划的支持，是它们的不断支持使我们得以将天网不断推上新的台阶，实现“让天网和中国网上信息资源规模同步成长”的理想。

作者

2004 年 5 月于北大燕园

目 录

图 示

表 格

第一章 引论

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。

我们注意到，上述全文检索系统一般工作在一个规模相对有限、内容相对稳定的馆藏（collection）上，被检索的对象通常是经过认真筛选和预先处理的（例如人工提取出了“作者”,“标题”等元数据，形成了很好的“摘要”等），并且系统需要同时响应的查询数量通常都不会太大（例如每秒钟 10 个左右）。

1994 年左右，万维网（World Wide Web，简记为 WWW 或 Web）出现。它的开放性（openness）和其上信息广泛的可访问性（accessibility）极大地鼓励了人们创作的积极性。作为一个信息源，Web 和上述全文检索系统的工作对象相比，具有许多不同的特征，它们给信息检索领域带来了新的发展机遇和技术挑战。

规模大。在短短的 10 年左右时间，人类至少生产了 40 亿网页[Google,2004]，而人类有文字上万年以来产生了大约 1 亿本书;中国网上到 2004 年初大致有了约 3 亿网页[天网,2004]，而中华民族有史以来出版的书籍大约不过 275 万种。尽管书籍的容量和质量是一般网页不可比的，但在对应的时间背景上考察其文字的总体数量，我们不能不为人类在 Web 上创造文字的激情惊叹！

内容不稳定。除了不断有新的网页出现外，旧的网页会因为各种原因被删除（有研究指出 50%网页的平均生命周期大约为 50 天[Cho and Garcia-Molina,2000, Cho,2002]）;

从原则上讲，读者数和作者数在同一个量级，形式和内容的随意性很强，权威性相对也不高，也不太可能进行人工筛选和预处理。

与生俱来的数字化、网络化。传统载体上的信息，人们目前正忙于将它们数字化、上网（花费极高），而网络信息天生如此。这个特性是一把双刃剑：一方面便于我们搜集和处理，另一方面也会使我们感到太多，蜂拥而至，鱼目混珠。

而作为要在 Web 上提供服务的信息查询系统，如搜索引擎和数字图书馆，通常要具备同时对付大量访问的能力（例如每秒钟 1000 个查询），而且响应时间还要足够的快（例如 1 秒钟）。

本书旨在介绍构建这类搜索引擎的有关技术。传统的 IR 是其基础，同时也充分讨论了由上述 Web 信息的特征所带来的新问题及其解决方案。

第一节 搜索引擎的概念

如上所述，本书的主要内容是介绍搜索引擎的工作原理和实现技术。搜索引擎，在本书指的是一种在 Web 上应用的软件系统，它以一定的策略在 Web 上搜集和发现信息，在对信息进行处理和组织后，为用户提供 Web 信息查询服务。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，然后很快返回一个可能和用户输入内容相关的信息列表（常常会是很长一个列表，例如包含 1 万个条目）。这个列表中的每一条目代表一篇网页，至少有 3 个元素：

标题：以某种方式得到的网页内容的标题。最简单的方式就是从网页的<TITLE></TITLE>标签中提取的内容。（尽管在一些情况下并不真正反映网页的内容）。本书第七章会介绍其他形成“标题”的方法。

URL：该网页对应的“访问地址”。有经验的 Web 用户常常可以通过这个元素对网页内容的权威性进行判断，例如 http://www.people.com 上面的内容通常就比 http://notresponsible.net（某个假想的个人网站）上的要更权威些（不排除后者上的内容更有趣些）。

摘要：以某种方式得到的网页内容的摘要。最简单的一种方式就是将网页内容的头若干字节（例如 512）截取下来作为摘要。本书第七章会介绍形成“摘要”的其他方法。

通过浏览这些元素，用户对相应的网页是否真正包含他所需的信息进行判断。比较肯定的话则可以点击上述 URL，从而得到该网页的全文。图 1-1 是 2003 年 8 月 20 日在天网搜索引擎（http://e.pku.edu.cn）上的一个例子，用户提交了查询词“伊拉克战争”，系统返回一个相关信息列表。列表的每一条目所含内容比上述要丰富些，但核心还是那三个元素。如果用户主要是想从军事角度关心伊拉克战争，第一条目可能就是很好的选择，不仅摘要看起来军事味道要浓一些，而且从 URL（http://mil.eastday.com）上能看到提供信息的大概是一个专门的军事题材网站。如果用户主要是想关心伊拉克战争对全球经济的影响，则后面的条目可能会更相关些。

这个例子提示了我们一个重要的情况，即搜索引擎提供信息查询服务的时候，它面对的只是查询词。而有不同背景的人可能提交相同的查询词，关心的是和这个查询词相关的不同方面的信息，但搜索引擎通常是不知道用户背景的，因此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“最可能被关心”的信息排在列表的前面。这也就是对搜索引擎的根本要求。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。

作为对搜索引擎工作原理的基本了解，这里有两个问题需要首先澄清。第一，当用户提交查询的时候，搜索引擎并不是即刻在 Web 上“搜索”一通，发现那些相关的网页，形成列表呈现给用户；而是事先已“搜集”了一批网页，以某种方式存放在系统中，此时的搜索只是在系统内部进行而已。第二，当用户感到返回

图 1-1 2003 年 8 月 20 日在天网上检索“伊拉克战争”的结果

结果列表中的某一项很可能是他需要的，从而点击 URL，获得网页全文的时候，他此时访问的则是网页的原始出处。于是，从理论上讲搜索引擎并不保证用户在返回结果列表上看到的标题和摘要内容与他点击 URL 所看到的内容一致（上面那个“伊拉克战争”的例子就是如此！），甚至不保证那个网页还存在。这也是搜索引擎和传统信息检索系统的一个重要区别。这种区别源于前述 Web 信息的基本特征。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。

第二节 搜索引擎的发展历史

早在 Web 出现之前，互联网上就已经存在许多旨在让人们共享的信息资源

了。那些资源当时主要存在于各种允许匿名访问的 FTP 站点（anonymous ftp），内容以学术技术报告、研究性软件居多，它们以计算机文件的形式存在，文字材料的编码通常是 PostScript 或者纯文本（那时还没有 HTML）。

为了便于人们在分散的 FTP 资源中找到所需的东西，1990 年加拿大麦吉尔大学（University of McGill）计算机学院的师生开发了一个软件，Archie。它通过定期搜集并分析 FTP 系统中存在的文件名信息，提供查找分布在各个 FTP 主机中文件的服务。Archie 能在只知道文件名的前提下，为用户找到这个文件所在的 FTP 服务器的地址。Archie 实际上是一个大型的数据库，再加上与这个大型数据库相关联的一套检索方法。该数据库中包括大量可通过 FTP 下载的文件资源的有关信息，包括这些资源的文件名、文件长度、存放该文件的计算机名及目录名等。尽管所提供服务的信息资源对象（非 HTML 文件）和本书所讨论搜索引擎的信息资源对象（HTML 网页）不一样，但基本工作方式是相同的（自动搜集分布在广域网上的信息，建立索引，提供检索服务），因此人们公认 Archie 为现代搜索引擎的鼻祖。

值得一提的是，即使是在 10 多年后的今天，以 FTP 文件为对象的信息检索服务技术依然在发展，尤其是在用户使用界面上充分采用了 Web 风格。北大天网文件检索系统就是一个例子（见 http://bingle.pku.edu.cn）。不过鉴于本书写作定位的关系，后面将主要讨论网页搜索引擎的相关问题。

以 Web 网页为对象的搜索引擎和以 FTP 文件为对象的检索系统一个基本的不同点在于搜集信息的过程。前者是利用 HTML 文档之间的链接关系，在 Web 上一个网页、一个网页的“爬取”（crawl），将那些网页“抓”（fetch）到本地后进行分析；后者则是根据已有的关于 FTP 站点地址的知识（例如得到了一个站点地址列表），对那些站点进行访问，获得其文件目录信息，并不真正将那些文件下载到系统上来。因此，如何在 Web 上“爬取”，就是搜索引擎要解决的一个基本问题。在这方面，1993 年 Matthew Gray 开发了 World Wide Web Wanderer，它是世界上第一个利用 HTML 网页之间的链接关系来监测 Web 发展规模的“机器人”（robot）程序。刚开始时它只用来统计互联网上的服务器数量，后来则发展为能够通过它检索网站域名。鉴于其在 Web 上沿超链“爬行”的工作方式，这种程序有时也称为“蜘蛛”（spider）。因此，在文献中 crawler, spider, robot 一般都指的是相同的事物，即在 Web 上依照网页之间的超链关系一个个抓取网页的程序，通常也称为“搜集”。在搜索引擎系统中，也称为网页搜集子系统。

现代搜索引擎的思路源于 Wanderer，不少人在 Matthew Grey 工作的基础上对它的蜘蛛程序做了改进。1994 年 7 月，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，成为第一个现代意义的搜索引擎。在那之后，随着 Web 上信息的爆炸性增长，搜索引擎的应用价值也越来越高，不断有更新、更强的搜索引擎系统推出（下一节会有介绍）。这其中，特

别引人注目的是 Google（http://www.google.com），虽然是个姗姗来迟者（1998 年才推出），但由于其采用了独特的 PageRank 技术，使它很快后来居上，成为当前全球最受欢迎的搜索引擎（作者 2003 年初访问印度，就听到总统阿卜杜勒·卡拉姆讲他经常用 Google 在网上查找信息！）。

图 1-2 2003 年 8 月 20 日在搜狐上检索“伊拉克战争”的结果

在中国，据我们所知，对搜索引擎的研究起源于“中国教育科研网”（CERNET）一期工程中的子项目，北京大学计算机系的项目组在陈葆珏教授的主持下于 1997 年 10 月在 CERNET 上推出了天网搜索 1.0 版本。该系统在这几年里不断发展，目前已成为中国最大的公益性搜索引擎（http://e.pku.edu.cn）。在这之后，几位在美国留学的华人学者回国创业，成立了百度公司，于 2000 年推出了“百度”商业搜索引擎（http://www.baidu.com），并一直处于国内搜索引擎的领先地位。我们看到慧聪公司也在中国推出了一个大规模搜索引擎（http://www.zhongsou.com），用起来感觉也不错，但往后发展如何，还有待时间的考验。

当我们谈及搜索引擎的时候，不应该忽略另外一个几乎是同期发展出来的事物：基于目录的信息服务网站。1994 年 4 月，斯坦福（Stanford）大学的两名博士生，David Filo 和杨致远（Gerry Yang）共同创办了 Yahoo!门户网站，并成功地使网络信息搜索的概念深入人心。1996 年中国出现了类似的网站，“搜狐”，

（http://www.sohu.com）。在许多场合，也称 Yahoo!之类的门户网站提供的信息查找功能为搜索引擎。但从技术上讲，这样的门户中提供的搜索服务和前述搜索引擎是很不同的。这样的门户依赖的是人工整理的网站分类目录，一方面，用户可以直接沿着目录导航，定位到他所关心的信息；另一方面，用户也可以提交查询词，让系统将他直接引导到和该查询词最匹配的网站。图 1-2 就是我们在搜狐上查询“伊拉克战争”的结果。和图 1-1 相比，不难看到其风格是很不相同的。在需要区别的场合，我们可以分别称“自动搜索引擎”和“目录搜索引擎”，或者“网页搜索引擎”和“网站搜索引擎”。一般来讲，前者的信息搜索会更全面些，后者则会准确些。在没有特殊说明的情况下，本书中所讨论的“搜索引擎”不包括 Yahoo!和搜狐这样的搜索方式。

随着网上信息越来越多，单纯靠人工整理网站目录取得较高精度查询结果的优势逐渐退化——对海量的信息进行高质量的人工分类已经不太现实。目前有两个发展方向。一是利用文本自动分类技术，在搜索引擎上提供对每篇网页的自动分类，这方面最先看到的例子是 Google 的“网页分类”选项，但它分类的对象只是英文网页。在中文方面，文本自动分类的研究工作有很多，但我们知道的第一个在网上提供较大规模网页自动分类服务的是北大网络实验室冯是聪和龚笔宏等人的工作[冯是聪,2003]，他们于 2002 年 10 月在天网搜索上挂接了一个 300 万网页的分类目录。另一个发展方向是将自动网页爬取和一定的人工分类目录相结合，希望形成一个既有高信息覆盖率，也有高查询准确性的服务。

互联网上信息量在不断增加，信息的种类也在不断增加。例如除了我们前面提到的网页和文件，还有新闻组，论坛，专业数据库等。同时上网的人数也在不断增加，网民的成分也在发生变化。一个搜索引擎要覆盖所有的网上信息查找需求已出现困难，因此各种主题搜索引擎，个性化搜索引擎，问答式搜索引擎等纷纷兴起。这些搜索引擎虽然还没有实现如通用搜索引擎那样的大规模应用，但随着互联网的发展，我们相信它们的生命力会越来越旺盛。另外，即使通用搜索引擎的运行现在也开始出现分工协作，有了专业的搜索引擎技术和搜索数据库服务提供商。例如美国的 Inktomi，它本身并不是直接面向用户的搜索引擎，但向包括 Overture（原 GoTo）、LookSmart、MSN、HotBot 等在内的其他搜索引擎提供全文网页搜集服务。从这个意义上说，它是搜索引擎数据的来源。

搜索引擎出现虽然只有 10 年左右的历史，但在Web上已经有了确定不移的地位。据CNNIC统计，它已经成为继电子邮件之后的第二大Web应用。虽然它的基本工作原理已经相当稳定，但在其质量、性能和服务方式等方面的提高空间依然很大，研究成果层出不穷，是每年WWW学术年会[1]的重要论题之一。

[1] International WWW Conference Committee, 网址 http://www.iw3c2.org.

第三节 一些著名的搜索引擎

为了让感兴趣的读者有目的的试一试，我们整理了一些当前主流的搜索引擎，包括网址，首页面图片及其介绍。在这些搜索引擎中，排在最前面的几个搜索引擎提供多语言的支持，可以满足不同母语读者的需求。

主流搜索引擎的选定参考了[Sullivan,2004]，主流搜索引擎是指非常有名，或者被广泛使用的搜索引擎。为使读者有感性认识特别加入了每个网站的相关页面。

Google, http://www.google.com

四次荣获Searchenginewatch[Searchenginewatch,2004]读者选举出的“最杰出搜索引擎”称号的Google作为在网络上搜索页面的首选是无愧于这个称号的。它基于搜集器[2]的服务既保证了能够覆盖广泛的网页，同时在查询效果上也表现得极其优秀。

为了方便的检索到所需网页，Google 提供几种可供选择的方法。利用 Google 首页搜索框上面的标签，可以容易的检索网络上的网页，图像，网上论坛，新闻和 Open Directory 提供的经过人工整理后的网页目录。

Google 还因为提供许多其它特性而闻名，例如网页快照，保证您在存有网页的服务器暂时出现故障时仍可浏览该网页的内容，或者可以浏览到不是最新版的该网页的内容；拼写检查，如果您查询词包含错误的拼写，它会提示正确的查询

[2] 自动搜索引擎的搜集子系统

词；股票行情查询；街区地图查询等特殊功能。更多的特性可以查看 Google 的帮助大全。此外，Google 工具条因为提供了方便存取 Google 和它的特性而为其赢得了一定的声誉。

Google 除了提供无需付费的排序结果，还有自己的竞价排名程序。与其他提供此项服务的公司一样，依据点击才有花费，竞价排名程序在 Google 的返回结果中放置广告。Google 还提供自己的无需付费的排序结果给其它一些搜索引擎。

Google 最初起源于斯坦福大学的 BackRub 项目，当时是由学生 Larry Page 和 Sergey Brin 主要负责。到了 1998 年，BackRub 更名为 Google，并且走出校园成为一个公司。

AllTheWeb, http://www.alltheweb.com

作为一个优秀的基于搜集器的搜索引擎，AllTheWeb 提供广泛的网络覆盖与显著的相关性。除了提供网页查询，AllTheWeb 还提供新闻，图像，视频和音频的检索。AllTheWeb 于 1999 年 5 月推出，先是由 FAST 运作；2003 年 4 月 Overture 收购了 AllTheWeb；后来 Yahoo!买下了 Overture，现在的 AllTheWeb 由 Yahoo!运作。

Ask Jeeves, http://www.askjeeves.com

Ask Jeeves 最初获得名声是在 1998 和 1999 年。作为自然语言搜索引擎，能够让用户通过输入问题来得到查询结果，并且所得到的结果看起来好像是对的。

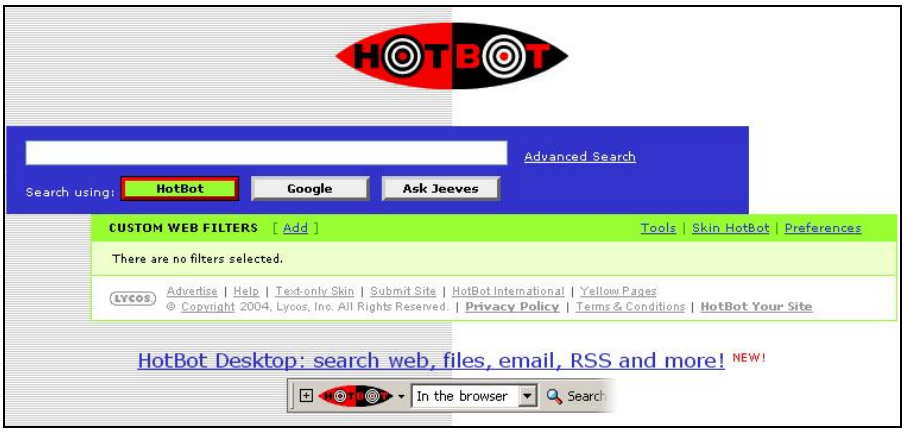
事实上，技术并不是 Ask Jeeves 运行很好的原因。在幕后，公司曾经指定 100 个编辑人员监视查询日志。然后这 100 个人上网查找与最常用查询词最相关的网页链接。目前，Ask Jeeves 仍然在使用人来参与结果的查找，但是现在编辑只有 10 个人左右。尽管如此，通过人的参与提供答案仍然是一个卖点，尤其对于那些新接触网络的人，他们会想使用 Ask Jeeves。对于通常的查询，人工选择的匹配结果让人感觉非常的相关。如果显示出来，这些结果出现在查询结果页面的最上端。除了人工参与外，Ask Jeeves 还利用基于搜集器的技术提供查询结果给用户。这些结果来自它所拥有的 Teoma 搜索引擎。

HotBot, http://www.hotbot.com

HotBot提供便于访问三个搜索引擎（HotBot, Google, Ask Jeeves）的入口，但是不同于元搜索引擎[3]，它不能将各搜索引擎的返回结果综合显示。

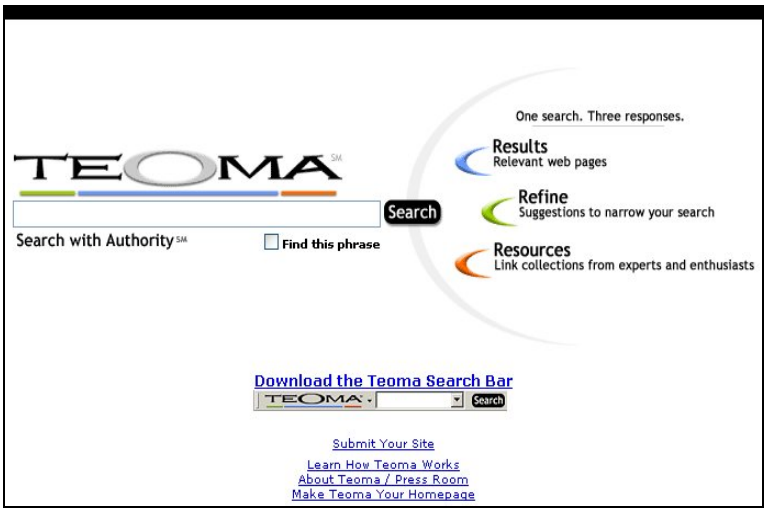
HotBot 在 1996 年初次登场，因为其庞大的由 Inktomi 提供的基于搜集器的检索页面和质量，而成为搜索者喜欢的引擎。特别是它的不同寻常的颜色和接口，还为它赢得了有经验的网民的注意。

[3] 元搜索引擎又称集合型搜索引擎，是将多个独立的搜索引擎集合在一起形成的检索工具，即搜索引擎之搜索引擎。



1999 年，HotBot 因为采用 Direct Hit 的 clickthrough 结果作为排序列表获得了恶名。Direct Hit 当年出现时是一个很热的搜索引擎。不幸的是，Direct Hit 的结果与同期登场的 Google 不能相比。HotBot 的声望开始下降。

Teoma, http://www.teoma.com



Teoma 是基于搜集器的搜索引擎，2001 年 9 月被 Ask Jeeves 收购。它索引的网页比同样基于搜集器的竞争对手 Google 的少。然而对于通常的查询检索，索引网页多少并不会产生很大的分别，自从 2000 年 Teoma 出现，就因为它很好的网页相关性赢得了称赞。一些人喜欢 Teoma 的“相关检索”特性，您先输入一个简单词语搜索，然后，Teoma 会为您提供其它相关搜索词作为参考。“专家推荐资源”部分也是 Teoma 的一个特色，指导用户去访问不同主题的链接。

Lycos, http://www.lycos.com

Lycos 是一个资格最老的搜索引擎，1994 年开始提供服务。在 1999 年 4 月它停止了自己基于搜集器的结果，取而代之的是利用 LookSmart 人工整理的常用查询分类结果和其它基于搜集器的搜索引擎，如：Yahoo!，Inktomi 等搜集器提供的结果。那么用户为什么不直接使用其他的搜索引擎而还要使用 Lycos 呢？你也许是喜欢 Lycos 提供的一些特性。

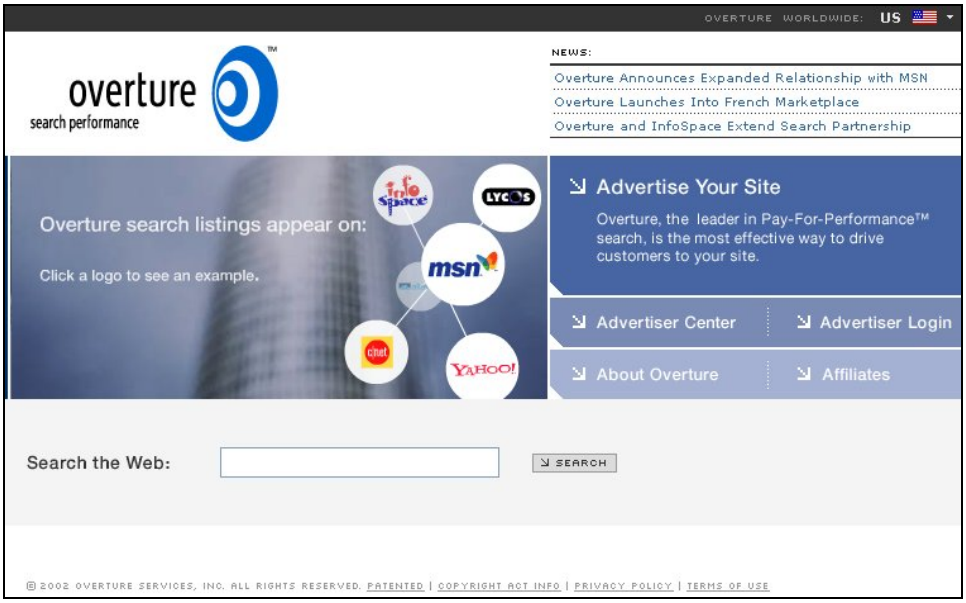
在搜索框的下方 Lycos 会建议其他的与用户检索主题相关的查询词，也许正是用户想看和感觉更确切的查询词。在这之下，就是 Lycos 提供的与其他搜索引擎一样的既相关又广泛覆盖的结果。

Lycos 属于 Terra Lycos 公司，它是在 2000 年 10 月由 Lycos 合并了 Terra 网络公司后形成的公司。Terra Lycos 公司还有 HotBot 搜索引擎。

WiseNut, http://www.wisenut.com

与 Teoma 类似，WiseNut 是基于搜集器的搜索引擎，在 2001 年出现的时候吸引了大家的注意力。WiseNut 的结果也有很好的相关性，并且有很大的数据库，几乎像 Google、AllTheWeb 和 Inktomi 一样大。然而，WiseNut 的数据库更新很慢，查询结果经常是几个月前的内容。LookSmart 在 2002 年 4 月并购了 WiseNut。

Overture, http://www.overture.com



最初叫 GoTo，2001 年更名为 Overture。Overture 是一个非常流行的竞价排名搜索引擎，它提供广告给许多搜索引擎排在检索结果的上方。Overture 在 2003 年 3 月购买了 AllTheWeb，2003 年 4 月又收购了 AltaVista。Yahoo 在 2003 年 10 月购买了 Overture。

Vivisimo, http://www.vivisimo.com

Vivisimo 于 2000 年 6 月由卡耐基-梅隆大学（CMU）推出，作为不同于基于搜集器的元搜索引擎，有自己的独到之处。它把其他搜索引擎的返回结果利用自动聚类的办法来满足不同类型客户的需要。在搜索引擎上，任何人搜索同一个词的结果都是一样。这样明显不能满足访问者。科学家搜索“星球”，可能是希望了解星球的知识，但普通人可能是想找“星球大战”电影，但搜索引擎所给的都是一样的结果。如何满足这些不同类型的访问者，需要对搜索结果进行个性化处理。搜索结果排序从单一化到个性化，Vivisimo 已经迈出了一步。

Baidu（百度）, http://www.baidu.com

百度于 2000 年推出，是目前在中国最成功的一个商业搜索引擎，主要提供中文信息检索，并且为门户站点提供搜索结果服务。搜索范围涵盖了中国内地、香港、台湾、澳门、新加坡等华语地区以及北美、欧洲的部分站点。拥有的中文信息总量达到 1 亿 2 千万网页以上，并且还在以每天几十万页的速度快速增长。

Tianwang（天网）, http://e.pku.edu.cn

于 1997 年 10 月开始提供服务，是中国最早的搜索引擎。它由北京大学网络与分布式系统实验室开发并维护运行，搜集了中国范围内大量的网络信息资源，尤其较全面地覆盖了中国教育科研网（CERNET）内的资源。天网目前索引的信

息资源除已经超过 3 亿的网页外，还包括 2000 多万各种非网页类型的文件，是目前世界上最大的中文搜索引擎之一。在系统功能上，天网除提供通常的关键词和短语检索外，还有自动网页分类目录。本书所介绍的技术内容主要就是以天网为背景展开的。

上篇　Web 搜索引擎基本原理和技术

上篇的主要目的是向读者介绍典型 Web 搜索引擎的基本工作原理，并通过一个实例具体展示该工作原理中各个环节的一种实现方法，以期使读者很快从技术上对搜索引擎系统的全貌有一个透彻的了解。

我们首先指出，所谓“搜索引擎”，说到底是一个计算机应用软件系统，或者说是一个网络应用软件系统。从网络用户的角度看，它根据用户提交的类自然语言查询词或者短语，返回一系列很可能与该查询相关的网页信息，供用户进一步判断和选取。为了有效地做到这一点，它大致上被分成三个功能模块，或者三个子系统；即网页搜集，预处理和查询服务。第二章详细分析了这三个部分的主要功能和其中需要关注的种种问题。应该指出，在实践中这三个部分是相对独立的，它们的工作形成了搜索引擎工作的三个阶段，通常分别由人工启动。同时我们注意到，在早期的搜索引擎中，系统处理的网页数量少，预处理部分的工作比较简单，只是涉及到汉语的分词（英文还没有这个问题）和建索引，因此也有将分词合并到网页搜集过程中，将建索引归到查询服务子系统中，从而整个系统看起来只有两个模块的安排[1]。

在介绍了基本原理后，第三、四、五章分别就上述三个阶段中的技术要求给出了一种实现方案。为了便于读者对搜索引擎技术在短时间内有一个全面实在的掌握，这三章的基本写作风格是提出问题，给出解决思路，然后是对应程序实现要点的注释和讲解。如果要掌握每一个细节（例如需要开发一个搜索引擎），要求读者对 C++程序设计语言比较熟悉。然而，从了解现代搜索引擎技术原理的需求来看，中篇和下篇并不很依赖这三章的内容，因此对 C++程序设计语言不熟的读者可以跳过这三章，直接阅读中篇和下篇，或者不一定要一句句探究那些程序段落的逻辑。程序代码可以在[TSE,2004]下载。

对于希望动手构建搜索引擎的读者来说，掌握了这一篇的内容，直接用我们提供的实例代码，应该能够很快（例如一周）构建出一个可用的小型通用搜索引擎。同时我们指出，一个实用的大规模搜索引擎还有许多其它重要问题要解决，集中在效率和质量两个方面，我们安排在中篇讨论。

[1] 1997 年 10 月我们在CERNET上发布的天网 1.0 版本就是这种结构，每抓来一个网页就立即在内存分词，然后将得到的结果存入数据库中，供建索引程序直接使用。

第二章 Web 搜索引擎工作原理和体系结构

本章介绍搜索引擎的基本工作原理和它作为一种网络应用软件的体系结构。在后面的三章中，我们将以一个实际的例子，具体展开在这些原理基础上实现的一种方案。通过这几章学习，读者将得到一个可实际运行搜索引擎的实现细节。

第一节 基本要求

如在第一章第二节所述，搜索引擎是一个网络应用软件系统，如图 2-1 所示，对它有如下基本要求。

能够接受用户通过浏览器提交的查询词或者短语，记作 q，例如“非典”，“伊拉克战争”，“床前明月光”等等。

在一个**可以接受的时间**内返回一个和该用户查询**匹配**的网页信息**列表**，记作 L。上一章讲过，这个列表的每一条目至少包含三个元素（标题，网址链接，摘要）。

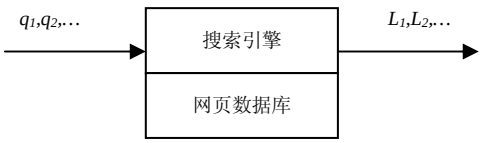


图 2-1 搜索引擎示意图

这里有几个问题需要注意，它们对应上面黑体的文字：

“可以接受的时间”，也就是响应时间。对于在 Web 上面向广大用户提供服务的软件来说，这个时间不能太长，通常也就在“秒”这个量级。这是衡量搜索引擎可用性的一个基本指标，也是和传统信息检索系统的一个差别。更进一步的，这样的响应时间要求不仅要能满足单个用户查询，而且要能在系统设计负载的情况下满足所有的用户。也就是说，系统应该在额定吞吐率的情况下保证秒级响应时间。这其中详细的分析将在中篇第八章展开。

“匹配”，指的是网页中以某种形式包含有 q 的内容，其中最简单、最常见

的形式就是 q 在其中直接出现。不过后面我们会看到，如果一个搜索引擎就是以百分之百满足这种简单的包含关系为目标，即使实现了也并不就达到了最好的效果。

“列表”，这蕴含着一种“序”（rank）。在绝大多数情况下，L 是相当长的，例如超过 1 万个条目（这是和图书馆全文检索系统的又一个不同，那里返回的列表通常较短，例如几十个条目）。这不仅是由于 Web 上的信息量大，也由于搜索引擎的查询方式简单。简单，意味着抽象；抽象，意味着有更多的具体事物可能是它的体现。对于一个长长的列表，很少有用户有耐心都审视一遍（不仅是因为长，还因为大多数使用搜索引擎的用户通常都是“找到为止”，而不是“不全部找到不罢休”，加上这个列表中和一个用户关心的其实只占很少的比例）。有分析统计表明，用户平均察看返回结果不超过 2 页[Baldi, et al.,2003],[Wang, et al.,2001],[单松巍,2003]。

现代大规模高质量搜索引擎一般采用如图 2-2 所示的称之为三段式的工作流程，即：网页搜集、预处理和查询服务。

图 2-2 搜索引擎三段式工作流程

第二节 网页搜集

搜索引擎这样一个软件系统应该是何种工作方式？如果说软件系统是工作在某个数据集合上的程序的话，这个软件系统操作的数据不仅包括内容不可预测的用户查询，还要包括在数量上动态变化的海量网页，并且这些网页不会主动送到系统来，而是需要由系统去抓取。

首先，我们考虑抓取的时机：事先还是即时。我们都有经验，在网络比较畅通的情况下，从网上下载一篇网页大约需要 1 秒钟左右，因此如果在用户查询的时候即时去网上抓来成千上万的网页，一个个分析处理，和用户的查询匹配，不可能满足搜索引擎的响应时间要求。不仅如此，这样做的系统效益也不高（会重复抓取太多的网页）；面对大量的用户查询，不可能想象每来一个查询，系统就到网上“搜索”一次。

因此我们看到，大规模搜索引擎服务的基础应该是一批预先搜集好的网页

（直接或者间接[1]）。这一批网页如何维护？可以有两种基本的考虑。

定期搜集，每次搜集替换上一次的内容，我们称之为“批量搜集”。由于每次都是重新来一次，对于大规模搜索引擎来说，每次搜集的时间通常会花几周。而由于这样做开销较大，通常两次搜集的间隔时间也不会很短（例如早期天网的版本大约每 3 个月来一次，Google 在一段时间曾是每隔 28 天来一次）。这样做的好处是系统实现比较简单，主要缺点是“时新性”（freshness）不高，还有重复搜集所带来的额外带宽的消耗。

增量搜集，开始时搜集一批，往后只是（1）搜集新出现的网页，（2）搜集那些在上次搜集后有过改变的网页，（3）发现自从上次搜集后已经不再存在了的网页，并从库中删除。由于除新闻网站外，许多网页的内容变化并不是很经常的（有研究指出 50% 网页的平均生命周期大约为 50 天[Cho and Garcia-Molina,2000]，[Cho,2002]），这样做每次搜集的网页量不会很大（例如我们在 2003 年初估计中国每天有 30-50 万变化了的网页），于是可以经常启动搜集过程（例如每天）。30 万网页，一台 PC 机，在一般的网络条件下，半天也就搜集完了。这样的系统表现出来的信息时新性就会比较高，主要缺点是系统实现比较复杂；这种复杂还不仅在于搜集过程，而是还在于下面要谈到的建索引的过程。

上面讲的是系统网页数据库维护的基本策略。在这两种极端的情况之间也可能有一些折中的方案，J. Cho 博士在这方面做过深入的研究[Cho and Garcia-Molina,2000]，[Cho,2002]，根据一种网页变化模型和系统所含内容时新性的定义，提出了相应优化的网页搜集策略。其中一个有趣的结论是：在系统搜集能力一定的情况下，若有两类网页（例如“商业”和“教育”），它们的更新周期差别很大（例如“商业”类网页平均更新周期是“天”，而“教育”类网页平均更新周期是“月”），则系统应该将注意力放在更新慢的网页上[Cho and Garcia-Molina,2000]，以使系统整体的时新性达到比较高的取值。

在具体搜集过程中，如何抓取一篇篇的网页，也可以有不同的考虑。最常见的一种是所谓“爬取”：将 Web 上的网页集合看成是一个有向图，搜集过程从给定起始 URL 集合 *S*（或者说“种子”）开始，沿着网页中的链接，按照先深、先宽、或者某种别的策略遍历，不停的从 *S* 中移除 URL，下载相应的网页，解析出网页中的超链接 URL，看是否已经被访问过，将未访问过的那些 URL 加入集合 *S*。整个过程可以形象地想象为一个蜘蛛（spider）在蜘蛛网（Web）上爬行（crawl）。后面我们会看到，真正的系统其实是多个“蜘蛛”同时在爬。

这种方式的好处除了概念很漂亮，一般实现起来也不困难外，还有很重要的一条是容易通过一定的策略，使搜集到的网页相对比较“重要”。前面提过，任何

[1] 所谓“间接”，指的是提供搜索服务的系统可能利用别人已经事先抓好的数据，元搜索引擎就是如此。

搜索引擎是不可能将Web上的网页搜集完全的，通常都是在其他条件的限制下决定搜集过程的结束（例如磁盘满，或者搜集时间已经太长了）。因此就有一个尽量使搜到的网页比较重要的问题，这对于那些并不追求很大的数量覆盖率的搜索引擎特别重要。研究表明[Najork and Wiener,2001]，按照先宽搜索方式得到的网页集合要比先深搜索得到的集合重要（这里当然有一个重要性的指标问题）。这种方式的一个困难是要从每一篇网页中提取出所含的URL。由于HTML的灵活性，其中出现URL的方式各种各样，将这个环节做得彻底不容易（例如我们现在还没有很好的简单办法从JavaScript脚本中提取URL）。同时，由于Web的“蝴蝶结”形状[Broder, et al.,2000]，这种方式搜集到的网页不大会超过所有目标网页数量[2]的 2/3。

另外一种可能的方式是在第一次全面网页搜集后，系统维护相应的 URL 集合 *S*，往后的搜集直接基于这个集合。每搜到一个网页，如果它发生变化并含有新的 URL，则将它们对应的网页也抓回来，并将这些新 URL 也放到集合 *S* 中；如果 *S* 中某个 url 对应的网页不存在了，则将它从 *S* 中删除。这种方式也可以看成是一种极端的先宽搜索，即第一层是一个很大的集合，往下最多只延伸一层。

还有一种方法是让网站拥有者主动向搜索引擎提交它们的网址（为了宣传自己，通常会有这种积极性），系统在一定时间内（2 天到数月不等）定向向那些网站派出“蜘蛛”程序，扫描该网站的所有网页并将有关信息存入数据库中。大型商业搜索引擎一般都提供这种功能。

第三节　预处理

得到海量的原始网页集合，距离面向网络用户的检索服务之间还有相当的距离。宏观地看，服务子系统是一个程序。采用 Wirth 关于“程序 = 算法+数据结构”的观点来考察这个程序，一个合适的数据结构是查询子系统工作的核心和关键。这里只是指出：现行最有效的数据结构是“倒排文件”（inverted file）；倒排文件是用文档中所含关键词作为索引，文档作为索引目标的一种结构（类似于普通书籍中，索引是关键词，书的页面是索引目标）。我们在第八章中有进一步分析。下面讨论从网页集合形成这样的倒排文件过程中的几个主要问题，即我们所说的“预处理”。主要包括四个方面，关键词的提取，“镜像网页”（网页的内容完全相同，未加任何修改）或“转载网页”（near-replicas，主题内容基本相同但可能有一些额外的编辑信息等，转载网页也称为“近似镜像网页”）的消除，链接分析和网页重要程度的计算。

1. 关键词的提取

[2] 所谓“目标网页”指的是搜索引擎设计覆盖的网页范围。例如Google是全球，天网是全中国。

随便取一篇网页的源文件（例如通过浏览器的“查看源文件”功能），我们可以看到其中的情况纷乱繁杂。除了我们从浏览器中能够正常看到的文字内容外，还有大量的HTML标记。根据天网统计，网页文档源文件的大小（字节量）通常大约是其中内容大小的 4 倍（例如http://net.pku.edu.cn就是如此！）。另外，由于HTML文档产生来源的多样性，许多网页在内容上比较随意，不仅文字不讲究规范、完整，而且还可能包含许多和主要内容无关的信息（例如广告，导航条，版权说明等）。这些情况既给有效的信息查询带来了挑战，也带来了一些新的机遇，在后面的章节将会有进一步的论述。这里我们只是指出，为了支持后面的查询服务，需要从网页源文件中提取出能够代表它的内容的一些特征。从人们现在的认识和实践来看，所含的关键词即为这种特征最好的代表。于是，作为预处理阶段的一个基本任务，就是要提取出网页源文件的内容部分所含的关键词。对于中文来说，就是要根据一个词典 Σ，用一个所谓“切词软件”，从网页文字中切出 Σ 所含的词语来。在那之后，一篇网页主要就由一组词来近似代表了，$p = \{t_1, t_2, \ldots, t_n\}$。一般来讲，我们可能得到很多词，同一个词可能在一篇网页中多次出现。从效果(effectiveness)和效率（efficiency）考虑，不应该让所有的词都出现在网页的表示中，要去掉诸如“的”，“在”等没有内容指示意义的词，称为“停用词”(stop word)。这样，对一篇网页来说，有效的词语数量大约在 200 个左右。

2. 重复或转载网页的消除

与生俱来的数字化和网络化给网页的复制以及转载和修改再发表带来了便利，因此我们看到 Web 上的信息存在大量的重复现象。天网在 2003 年的一次大规模统计分析表明，网页的重复率平均大约为 4。也就是说，当你通过一个 URL 在网上看到一篇网页的时候，平均还有另外 3 个不同的 URL 也给出相同或者基本相似的内容。这种现象对于广大的网民来说是有正面意义的，因为有了更多的信息访问机会。但对于搜索引擎来说，则主要是负面的；它不仅在搜集网页时要消耗机器时间和网络带宽资源，而且如果在查询结果中出现，无意义地消耗了计算机显示屏资源，也会引来用户的抱怨，“这么多重复的，给我一个就够了”。因此，消除内容重复或主题内容重复的网页是预处理阶段的一个重要任务。第七章对此有详细的分析论述。

3. 链接分析

前面提到，大量的 HTML 标记既给网页的预处理造成了一些麻烦，也带来了一些新的机遇。从信息检索的角度讲，如果系统面对的仅仅是内容的文字，我们能依据的就是“共有词汇假设”（shared bag of words），即内容所包含的关键词集合，最多加上词频（term frequency 或 tf、TF）和词在文档集合中出现的文档频率（document frequency 或 df、DF）之类的统计量。而 TF 和 DF 这样的频率信

息能在一定程度上指示词语在一篇文档中的相对重要性或者和某些内容的相关性，这是有意义的。有了 HTML 标记后，情况还可能进一步改善，例如在同一篇文档中，<H1>和</H1>之间的信息很可能就比在<H4>和</H4>之间的信息更重要。特别地，HTML 文档中所含的指向其他文档的链接信息是人们近几年来特别关注的对象，认为它们不仅给出了网页之间的关系，而且还对判断网页的内容有很重要的作用。例如“北大学报”这几个字在北京大学学报社会科学版的主页上是没有的，因此一个仅靠内容文字分析的搜索引擎就不可能返回该主页作为结果。但是北京大学主页上是用“北大学报（社）”作为链接信息指向了北京大学学报社会科学版的主页。因此在很好利用链接信息的搜索引擎中应该能返回北京大学学报社会科学版的主页。

4. 网页重要程度的计算

搜索引擎返回给用户的，是一个和用户查询相关的结果列表。列表中条目的顺序是很重要的一个问题。由于面对各种各样的用户，加之查询的自然语言风格，对同样的q_0返回相同的列表肯定是不能使所有提交q_0的用户都满意的（或者都达到最高的满意度）。因此搜索引擎实际上追求的是一种统计意义上的满意。人们认为Google目前比天网好，是因为在多数情况下前者返回的内容要更符合用户的需要，而不是所有情况下都如此。如何对查询结果进行排序有很多因素需要考虑，后面将有深入的讨论。这里只是概要解释在预处理阶段可能形成的所谓“重要性”因素。顾名思义，既然是在预处理阶段形成的，就是和用户查询无关的。如何讲一篇网页比另外一篇网页重要？人们参照科技文献重要性的评估方式，核心想法就是“被引用多的就是重要的”。“引用”这个概念恰好可以通过HTML超链在网页之间体现得非常好，作为Google创立核心技术的PageRank就是这种思路的成功体现[Page, et al.,1998]。除此以外，人们还注意到网页和文献的不同特点，即一些网页主要是大量对外的链接，其本身基本没有一个明确的主题内容，而另外有些网页则被大量的其他网页链接。从某种意义上讲，这形成了一种对偶的关系，这种关系使得人们可以在网页上建立另外一种重要性指标[Kleinberg,1998]。这些指标有的可以在预处理阶段计算，有的则要在查询阶段计算，但都是作为在查询服务阶段最终形成结果排序的部分参数。

第四节 查询服务

如上述，从一个原始网页集合 S 开始，预处理过程得到的是对 S 的一个子集的元素的某种内部表示，这种表示构成了查询服务的直接基础。对每个元素来说，这种表示至少包含如下几个方面：

- 原始网页文档
- URL 和标题
- 编号
- 所含的重要关键词的集合（以及它们在文档中出现的位置信息）
- 其他一些指标（例如重要程度，分类代码等）

而系统关键词总体的集合和文档的编号一起构成了一个倒排文件结构，使得一旦得到一个关键词输入，系统能迅速给出相关文档编号的集合输出。

然而，如同我们在第一章提到的，用户通过搜索引擎看到的不是一个“集合”，而是一个“列表”。如何从集合生成一个列表，是服务子系统的主要工作。从搜索引擎系统功能划分的角度，有时候将倒排文件的生成也作为服务子系统的一部分功能，但我们这里将它划分到预处理阶段中觉得更方便些。换句话讲，服务子系统是在服务进行的过程中涉及的相关软件程序，而为这些软件程序事先准备数据的程序都算在预处理子系统中。下面来看对服务子系统的要求和其工作原理，主要有三个方面。

1. 查询方式和匹配

查询方式指的是系统允许用户提交查询的形式。考虑到各种用户的不同背景和不同的信息需求，不可能有一种普适的方式。一般认为，对于普通网络用户来说，最自然的方式就是“要什么就输入什么”。但这是一种相当模糊的说法。例如用户输入“北京大学”，可能是他想了解北京大学目前有些什么信息向外发布，想看看今年的招生政策（于是希望看的是北大网站上的内容），也可能是他想了解外界目前对北京大学有些什么评价（于是希望看到的是其他权威网站上关于北大的消息）。这是两种相当不同的需求。在其他一些情况下，用户可能关心的是间接信息，例如“喜马拉雅山的高度”，8848 米应该是他需要的，但不可能包含在这短语中。而用户输入“惊起一滩鸥鹭”则很可能是想知道该词的作者是谁，或者希望能提醒前面几句是什么。尽管如此，用一个词或者短语来直接表达信息需求，希望网页中含有该词或者该短语中的词，依然是主流的搜索引擎查询模式。这不仅是因为它的确代表了大多数的情况，还因为它比较容易实现。这样，一般来讲，系统面对的是查询短语。就英文来说，它是一个词的序列；就中文来说，它是包含若干个词的一段文字。一般地，我们用q_0表示用户提交的原始查询，例如，q_0 = “网络与分布式系统实验室”。它首先需要被“切词”（segment）或称“分词”，即把它分成一个词的序列。如上例，则为“网络 与 分布式 系统 实验室”（注意，不同的分词软件可能得出不同的结果，这里用的是北大计算语言所的在线分词软件）。然后需要删除那些没有查询意义或者几乎在每篇文档中都会出现的词（例如“的”），在本例中即为“与”。最后形成一个用于参加匹配的查询词表，$q = \{t_1, t_2, \ldots, t_m\}$，在本例中就是$q$ = {网络，分布式，系统，实验室}。前面讲过，

倒排文件就是用词来作为索引的一个数据结构，显然，q中的词必须是包含在倒排文件词表中才有意义。有了这样的q，它的每一个元素都对应倒排文件中的一个倒排表（文档编号的集合），记作$L(t_i)$，它们的交集即为对应查询的结果文档集合，从而实现了查询和文档的匹配。上述过程的基本假设是：用户是希望网页包含所输入查询文字的。

2. 结果排序

上面，我们了解了得到和用户查询相关的文档集合的过程。这个集合的元素需要以一定的形式通过计算机显示屏呈现给用户。就目前的技术情况看，列表是最常见的形式（但人们也在探求新的形式，如Vivisimo 引擎将结果页面以类别的形式呈现）。给定一个查询结果集合，$R=\{r_1, r_2, \ldots, r_n\}$，所谓列表，就是按照某种评价方式，确定出R中元素的一个顺序，让这些元素以这种顺序呈现出来。笼统地讲，r_i和q的相关性（relevance）是形成这种顺序的基本因素。但是，有效地定义相关性本身是很困难的，从原理上讲它不仅和查询词有关，而且还和用户的背景，以及用户的查询历史有关。不同需求的用户可能输入同一个查询，同一个用户在不同的时间输入的相同的查询可能是针对不同的信息需求。为了形成一个合适的顺序，在搜索引擎出现的早期人们采用了传统信息检索领域很成熟的基于词汇出现频度的方法。大致上讲就是一篇文档中包含的查询（q）中的那些词越多，则该文档就应该排在越前面；再精细一些的考虑则是若一个词在越多的文档中有出现，则该词用于区分文档相关性的作用就越小。这样一种思路不仅有一定直觉上的道理，而且在倒排文件数据结构上很容易实现。因为，当我们通过前述关键词的提取过程，形成一篇文档的关键词集合，$p = \{t_1, t_2, \ldots, t_n\}$的时候，很容易同时得到每一个$t_i$在该文档中出现的次数，即词频，而倒排文件中每个倒排表的长度则对应着一个词所涉及的文档的篇数，即文档频率。然而，由于网页编写的自发性、随意性较强，仅仅针对词的出现来决定文档的顺序，在Web上做信息检索表现出明显的缺点，需要有其他技术的补充。这方面最重要的成果就是前面提到过的PageRank。通过在预处理阶段为每篇网页形成一个独立于查询词（也就和网页内容无关）的重要性指标，将它和查询过程中形成的相关性指标结合形成一个最终的排序，是目前搜索引擎给出查询结果排序的主要方法。

3. 文档摘要

搜索引擎给出的结果是一个有序的条目列表，每一个条目有三个基本的元素：标题，网址和摘要。其中的摘要需要从网页正文中生成。一般来讲，从一篇文字中生成一个恰当的摘要是自然语言理解领域的一个重要课题，人们已经做了多年的工作并取得了一些成果。但相关的技术用到网络搜索引擎来有两个基本困难。一是网页的写作通常不规范，文字比较随意，因此从语言理解的角度难以做

好；二是复杂的语言理解算法耗时太多，不适应搜索引擎要高效处理海量网页信息的需求。我们做过统计，即使是分词这一项工作（文本理解的基础），在高档微机上每秒钟也只能完成 10 篇左右网页的处理。因此搜索引擎在生成摘要时要简便许多，基本上可以归纳为两种方式，一是静态方式，即独立于查询，按照某种规则，事先在预处理阶段从网页内容提取出一些文字，例如截取网页正文的开头 512 个字节（对应 256 个汉字），或者将每一个段落的第一个句子拼起来，等等。这样形成的摘要存放在查询子系统中，一旦相关文档被选中与查询项匹配，就读出返回给用户。显然，这种方式对查询子系统来说是最轻松的，不需要做另外的处理工作。但这种方式的一个最大的缺点是摘要和查询无关。一篇网页有可能是多个不同查询的结果，例如当用户分别查询“北大计算机网络”和“北大分布式系统”，我们实验室的主页 http://net.pku.edu.cn 在两种情况下应该都作为结果返回。当用户输入某个查询，他一般是希望摘要中能够突出显示和查询直接对应的文字，希望摘要中出现和他关心的文字相关的句子。因此，我们有了“动态摘要”方式，即在响应查询的时候，根据查询词在文档中的位置，提取出周围的文字来，在显示时将查询词标亮。这是目前大多数搜索引擎采用的方式。为了保证查询的效率，需要在预处理阶段分词的时候记住每个关键词在文档中出现的位置。

除上述外，查询服务返回的内容还有一些细节的支持。例如，对应一个查询往往会有成千上万的结果，返回给用户的内容通常都是按页组织的，一般每页显示 10 个结果。统计表明[Wang, et al.,2001]，网络用户一般没有耐心一页页看下去，平均翻页数小于 2。这告诉我们将第一页的内容组织好非常重要。如果希望用户多用搜索引擎，就要让第一页的内容尽量有吸引力。

第五节 体系结构

在上述工作原理的基础上，作为一个网络应用软件，我们可以勾画出搜索引擎的体系结构，如图 2-3 所示，其中的大部分模块和前面的原理描述有直接的对应。这里需要特别讨论的是还没有专门提及的“控制器”模块。

网页的搜集，如果只是为了做些简单的实验，不过上万篇网页的话，许多矛盾都不会出现，可以用最简单的工具（例如wget[3]）完成。但如果是为了向大规模搜索引擎稳定地提供网页数据，通常需要每天搜集上百万网页，而且是持续进行，情况则要复杂许多，核心是要综合解决效率、质量和“礼貌”的问题。这就是“控制器”的作用。

[3] http://www.gnu.org/software/wget/wget.html

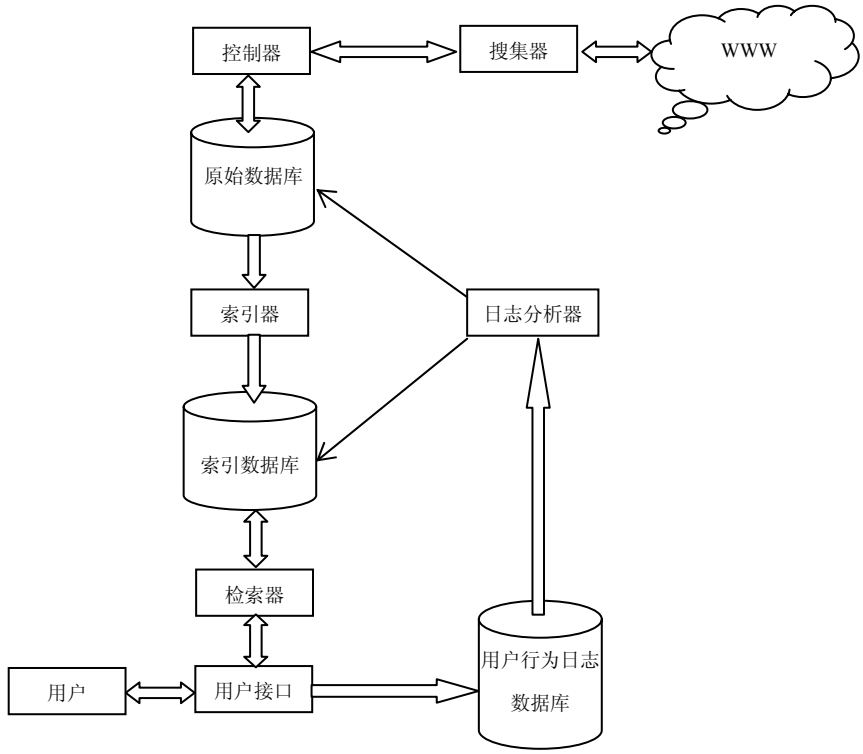


图 2-3 搜索引擎的体系结构

所谓效率，在这里就是如何利用尽量少的资源（计算机设备、网络带宽、时间）来完成预定的网页搜集量。在批量搜集的场合，我们通常考虑半个月左右能搜集到的网页，自然是越多越好。由于网页之间存在的独立性，利用许多台计算机同时来做这项工作是一个吸引人的想法，在第六章有专门的论述。这里需要指出三点：第一，即使是用一台计算机来搜集网页，也应该注意并发性的开发和利用。由于从网上抓取一篇网页通常需要秒量级的等待网络通信时间，同时启动多个抓取进程/线程，或者利用操作系统提供的异步通信机制，让多个网络通信时间重叠起来，让网络通信时间和存放网页的磁盘访问时间重叠起来是很有意义的。同时启动抓取进程的数量取决于硬件条件和搜集软件的设计，一般情况下可以上百个，做得好也可能上千个（即上千个进程也不会造成 CPU 成为瓶颈）。

在效率问题上应该指出的第二点是并不是设备越多越好。在用若干台计算机形成一个机群的安排下，它们共同分享出口网络带宽，随着设备量的增加，这个网络带宽（或者是周围的某个环境带宽）很快就成为瓶颈。经验表明实际上用不了超过 10 台计算机。人们曾经有过分布式搜集的想法，即让多台设备分布在网络上的不同位置，从而克服上述带宽瓶颈问题；理论上是对的。但对于维护千万到

亿量级的搜索引擎来说似乎还用不上，具体实现起来的麻烦会超过可能带来的好处（也许 Google 那样的针对多个国家用户的巨型搜索引擎需要用这种技术）。影响搜集效率的第三点发生在网络的另一端，即服务器方，它可能来不及提供所需的网页。这除了有些 Web 服务器所处的网络条件比较差，或者有太多其他人访问外，搜索引擎太频繁对它们发出网页请求也是一个重要原因。落实到技术上，就是要有一个访问策略或者 URL 规划，不要让搜集器启动的抓取进程都集中在少数几个网站上。

将搜集活动的关注过分集中在几个网站上，或者在一小段时间里从一个网站抓取太多的网页还可能引起其他的严重后果，即所谓“礼貌”问题。一般来讲，网站的管理人员都很愿意让自己的网页被搜索引擎索引，从而有可能得到更多的访问流量；但这只是问题的一方面。问题的另一方面是网站绝不希望由于搜索引擎的“密集”抓取活动阻碍了普通用户通过浏览器的访问，使那些用户得到这个网站访问起来很困难的印象，从而不再光顾。不加控制的网页抓取，给网站造成的现象有时候和制造拒绝服务（Denial of Servide, DoS）攻击的黑客造成的现象一样。因此，管理良好的网站常常会有一个监视器运行，监视是否有来源于单个 IP 地址的过分密集的访问。一旦出现这种情况，要么会通告该 IP 地址的拥有者注意行为，或者会干脆屏蔽来自它的访问，更有甚者，还可能在网上公布该 IP 地址作为黑名单。因此，适当地规划网页的抓取，限制单位时间内对一个网站抓取网页的数量（例如每天不超过 2 万个，或者至少每隔 30 秒才对同一个网站发出下一个网页请求，等等），是大规模搜索引擎必须要认真对待的问题。总之，搜索引擎需要和网站“和睦相处”，它们是相互依存的。

所谓质量问题，指的是在有限的时间，搜集有限的网页，希望它们尽量是比较“重要”的网页，或者说不要漏掉那些很重要的网页。哪些网页是比较重要的？也是仁者见仁，智者见智的，不可能有一个统一认可的标准。如果让重要性和流行度等同起来，即越多人看过的网页越重要，至少是直觉上有一定道理的。这样，我们可以考虑一个网站从主页开始向下，按照链接的深度将网页组织成一层层的，上层中的网页统计上会比下层的网页重要些。这样一种认识通过 PageRank 得到了加强，即较靠近主页的网页通常 PageRank 值较高。这样，首先得到尽量多的主页，然后从主页开始的先宽搜索就应该是一个较好的策略。闫宏飞论文[闫宏飞,2002]是支持这种观点的一个工作。

网页搜集过程中还有一个基本的问题是要保证每个网页不被重复抓取。由于一篇网页可能被多篇网页链接，在 spider 爬取过程中就可能多次得到该网页的 url。于是如果不加检查和控制，网页就会被多次抓取。遇到循环链接的情况，还会使爬取器陷死。解决这个问题的有效方法是使用两个表，unvisited_table 和 visited_table。前者包含尚未访问的 url，后者记录已访问的 url。系统首先将要搜集的种子 url 放入 unvisited_table，然后 spider 从其中获取要搜集网页的 url，搜

集过的网页 url 放入 visited_table 中，新解析出的并且不在 visited_table 中的 url 加入 unvisited_table。此方法简单明了，适合在单个节点上实现。但是当搜集子系统涉及到多个节点的时候，如何避免各个节点之间的重复工作就复杂了，还要考虑网络的通信量、负载平衡、以及单个节点性能瓶颈等问题。这些问题的细节将在第六章讨论。

第三章 Web 信息的搜集

从第二章的学习中，我们知道搜索引擎一般采用三段式的工作流程，即：网页搜集，预处理和查询服务。为了便于读者对搜索引擎技术很快有一个全面实在的掌握，从本章开始的连续三章，我们讲解一个完整的搜索引擎 TSE（Tiny Search Engine）的实现，编程语言采用 C++，代码可以在[TSE,2004]下载。TSE 包括三段式工作流程，分别对应本章的 Web 信息的搜集，第四章搜集信息的预处理和第五章的信息查询服务。

作为三章内容的一个起始，在本章第一节中简单介绍了互联网上的 HTTP 协议和 TSE 系统框架。然后主要内容集中在 TSE 的 Web 信息搜集部分。在后续的章节叙述中，文档也是指网页，我们不加以区分。

第一节 引言

一、 超文本传输协议

超文本传输协议（Hypertext Transfer Protocol, HTTP）[1]是Web的基础协议。为了本章的完整，首先对HTTP进行简要的介绍，然后重点讲解如何实现Web信息的收集。

HTTP 是一个简单的协议。客户进程建立一条同服务器进程的 TCP 连接，然后发出请求并读取服务器进程的应答。服务器进程关闭连接表示本次响应结束。服务器进程返回的内容包含两个部分，一个“应答头”（response header），一个“应答体”（response body），后者通常是一个 HTML 文件，我们称之为“网页”。

在 Linux（或 window）环境下，有一个简单的方法可以让我们来感受一下 HTTP 协议的工作情况，即运行一个 Telnet 的客户程序与一个 HTTP 服务器程序通信。

下面是获取北京大学主页的例子（注意，下面显示的从服务器得到的内容不包括上述“应答头”）。

[1] 关于HTTP协议的详细介绍可以参看RFC2068 Hypertext Transfer Protocol -- HTTP/1.1 [RFCs,2004]。此外，在很多书中都有专门的章节进行介绍，如：《TCP-IP详解卷三：TCP事务协议，HTTP，NNTP和UNIX域协议》的第 13 章HTTP：超文本传送协议；《UNIX技术大全—Internet卷》的第 21 章超文本传输协议简介。

```
[webg@BigPc ]$ telnet www.pku.cn 80        // 连接到服务器的 80 号端口
Trying 162.105.129.12...                   // 由 Telnet 客户输出
Connected to rock.pku.cn (162.105.129.12). // 由 Telnet 客户输出
Escape character is '^]'.                  // 由 Telnet 客户输出
GET /                                      // 我们只输入了这一行
<html>                                     // Web 服务器输出的第一行
<head>
<title>北京大学</title>
……                                         // 这里省略了很多行输出
</body>
</html>
Connection closed by foreign host.         // 由 Telnet 客户输出
```

我们只输入了 GET /，服务器却返回了很多字节。这样，从该 Web 服务器的根目录下取得了它的主页。Telnet 的客户进程输出的最后一行信息表示服务器进程在输出最后一行后关闭了 TCP 连接。

一个完整的 HTML 文档以<HTML>开始，以</HTML>结束。大部分的 HTML 命令都像这样成对出现。HTML 文档含有以<HEAD>开始、以</HEAD>结束的首部和以<BODY>开始、以</BODY>结束的主体部分。标题通常由客户程序显示在窗口的顶部。关于 HTML 规范的详细介绍可以参看[W3C,1999]。

在接下去的几节中，将通过一个小的搜索引擎系统 TSE（运行在 Red Hat Linux 8.0 以上的系统中）[TSE,2004]的循序渐进实现，一边讲原理技术，一边讲代码，描述 Web 信息搜集的过程，本处的 Web 信息搜集主要指网页信息。

网页搜集子系统，就是第一章第二节和第二章第五节中讲到的 spider，可以用 C/C++、Perl、Java，Python 等语言来编写，可以运行在 Intel, Sparc, Mac 等平台上的 Unix 或 Window 系统下。网页“爬取器”（gatherer），指网页搜集子系统中根据 URL 完成一篇网页抓取的进程或者线程，通常一个 spider 会同时启动多个 gatherer 并行工作。Spider 设计是否合理将直接影响它访问 Web 的效率，影响搜集数据的质量，另外，在设计 spider 时还必须考虑它对网络和被访问站点的影响，因为 spider 一般都运行在速度快、带宽高的主机上，如果它快速访问一个速度比较慢的目标站点，就有可能会导致该站点出现拥塞甚至宕机。Spider 还应遵守一些协议（例如：robot 限制协议[Wong,1997]），尊重被访问站点管理员确定的内容保护策略。

二、 一个小型搜索引擎系统

TSE 是一个适合教学用的搜索引擎，设计的目标之一是让它足够小，便于任何一个对搜索引擎感兴趣的人都可以利用自己有限的硬件资源（例如自己的台式机）搭建；让它尽量简单，让具有一般程序设计基础的爱好者可以全部理解；让它的功能相对完整，能够反映一个大规模搜索引擎的主要成分。

首先从 TSE 的外部表现形式来看它所能完成的工作，然后给出 TSE 系统结构图。搜索引擎是通过浏览器界面展现给用户的，图 3-1 是 TSE 的用户界面。

图 3-1 TSE 搜索引擎界面

在查询输入框输入查询短语并回车 (或者点击 “搜索”按钮) ，即可得到相关资料。查询时关键词之间不需要使用“and”，因为 TSE 会在关键词之间自动添加“and”。TSE 提供符合您查询条件的全部网页。

例如：想查北大校庆，只需在图 3-1 的搜索框中输入“北大校庆”，然后回车。图 3-2 是 TSE 的查询返回结果。为方便解释起见，我们用 A，B，C 标识其中的几个部分。

1) “A”表示统计栏，包括用户输入的查询词，有关查询结果和搜索时间（一般搜索响应时间不超过 1 秒钟）的统计数字；
2) “B”表示一条查询结果， 包括该网页网址、网页摘要（在摘要信息中，您的原始查询字词，都用红色字体表示，以便阅读）。
3) “C”表示网页快照。通过链接访问网页失效时，可以访问 TSE 的缓存网页；或者网络拥塞的时候，可以通过访问缓存网页避免直接访问该网

页。

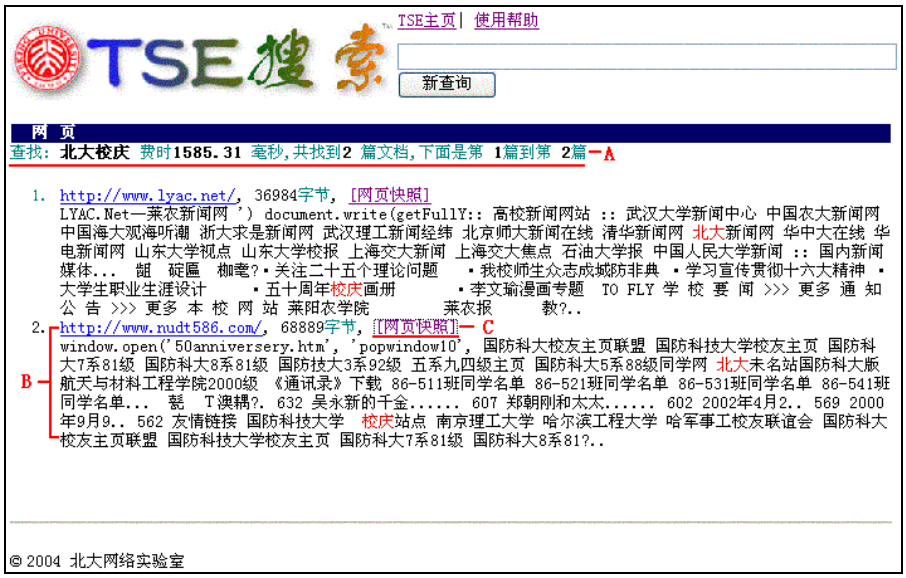


图 3-2 TSE 查询结果页面

图 3-3 TSE 网页快照页面

点击图 3-2 中的“网页快照”链接，得到图 3-3 所示的 TSE 网页快照页面。

图中最上部分标明此网页来自 TSE 的网页快照。用户输入的查询短语如果被系统分成多个关键词，用不同颜色表示，并增加链接便于点击相应关键词直接到达正文中该关键词出现的位置。正文部分取自网页原文的缓存，其中包含用户查询关键词的文字加亮显示。

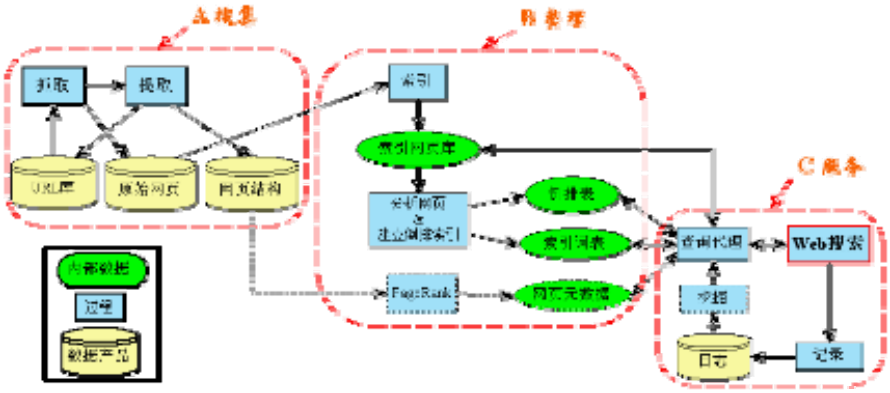


图 3-4 TSE 系统结构

图 3-1，3-2，3-3 展示了 TSE 的查询服务功能，为了完成上述功能，需要网页搜集和预处理两个部分的支持。图 3-4 所示为 TSE 系统结构，对应于搜索引擎三段式工作流程，是图中左侧的 A 表示搜集部分，中间的 B 表示整理（即预处理）部分和右侧的 C 表示服务部分。其中黄色圆柱形图表示数据产品，按照统一并且简单易懂的格式存储，除本系统使用外，可以提供给其他科研机构使用；椭圆形绿色图表示系统流程中的内部数据，由于与系统中使用的数据结构结合紧密，不适合作为数据产品提供给其他研究机构；矩形蓝色表示系统流程的程序部分（过程），是数据产品与内部数据之间的桥梁。系统起始于 A 搜集，结束于 C 服务，整个流程可以重复进行，从而达到系统的更新。图 3-4 中的各个数据产品，内部数据和过程在后续章节相应部分细致讲解。在 TSE 中不包括 PageRank 的计算和日志挖掘，这两个过程主要是对查询结果的排序产生作用，在实际应用中的搜索引擎是必不可少的，但是对于讲解搜索引擎的工作过程不是必需的。

本章后续内容讲解搜集部分（对应图 3-4 的 A）；第四章讲解预处理部分（对应图中的 B）；第五章讲解服务部分（对应图中的 C）。

第二节 网页搜集

在 TSE 中网页搜集对应图 3-4 的左侧部分，我们将它细化为图 3-5 所示。网页搜集的过程是从 URL 库（初始时包含用户指定的起始种子 URL 集合，可以是

1 个或多个）获得输入，解析 URL 中标明的 Web 服务器地址、建立连接、发送请求和接收数据，将获得的网页数据存储在原始网页库，并从其中提取出链接信息放入网页结构库，同时将待抓取的 URL 放入 URL 库，保证整个过程的递归进行，直到 URL 库为空。

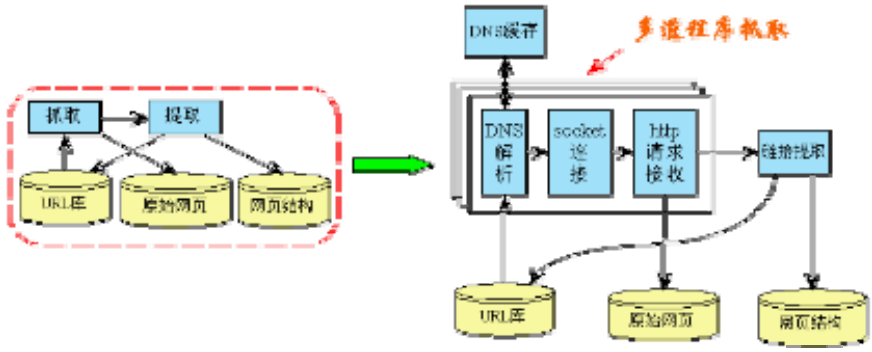


图 3-5 Web 信息的搜集

搜索引擎为了提供检索服务，需要保存网页原文。网页搜集子系统不但要能够获取以.html, .htm, .txt 结尾的 URL 对应的网页（在本章后面的小节对于搜集信息类型会有更详细的阐述），还应该能够获取不是以.html 结尾的 URL，比如.pdf，.doc，因为.pdf，.doc 等文件可以通过转换程序生成为.html 或者.txt 文件，同样为搜索引擎提供检索服务。作为搜索引擎的起始流程，搜集的网页要按照一定的格式存储，便于后续组织和提供服务。

传统应用的实现在系统结构定下来后，重点是确定其中的数据结构。而面向对象方法具有更好的特性，我们是采用 C++面向对象的方法结合源代码讲述。针对系统实现的描述，依据重要名词定义为类的原则，重点讲解 URL 类和 Page 类。本章后续部分是把 TSE 中的搜集部分分解开来讲解，包括：定义 URL 类，定义 Page 类，与服务器建立连接，构造请求消息体并发送给服务器，获取服务器返回的网页元信息（或者称为网页头信息）和获取网页信息（或者称为网页体信息）。

一、 定义 URL 类和 Page 类

1. 定义 URL 类

TSE 程序首先必须要做的一件事情是根据一个给定的 URL，组成消息体，发送给该 URL 指向的服务器。为此，定义 Url 类。关于 URL 的详细介绍可以参看

RFC2396 Uniform Resource Identifiers (URI): Generic Syntax[RFCs,2004]。

下面是 URL 类的定义，对应文件 Url.h。

```
enum url_scheme{
        SCHEME_HTTP,
        SCHEME_FTP,
        SCHEME_INVALID
};

class CUrl
{
public:
        string      m_sUrl;                    // URL 字串
        enum url_scheme m_eScheme;             // 协议名
        string      m_sHost;                   // 主机名
        int    m_nPort;                        // 端口号
       string m_sPath;                         // 请求资源
public:
        CUrl();
        ~CUrl();
        bool ParseUrl( string strUrl );
private:
        void ParseScheme ( const char *url );
};
```

URL 可以是 HTTP，FTP 等协议开始的字符串，TSE 主要是针对 HTTP 协议，为了不失一般性，在 url_scheme 中定义了 SCHEME_HTTP，SCHEME_FTP，SCHEME_INVALID，分别对应 HTTP 协议，FTP 协议和其他协议。一个 URL 由 6 个部分组成：

<scheme>://<net_loc>/<path>;<params>?<query>#<fragment>

除了 scheme 部分，其他部分可以不在 URL 中同时出现。

Scheme 表示协议名称，对应于 URL 类中的 m_eScheme.

Net_loc 表示网络位置，包括主机名和端口号，对应于 URL 类中的 m_sHost 和 m_nPort.

下面四个部分对应于 URL 类中的 m_sPath.

Path　　表示 URL 路径.
Params　表示对象参数.
Query　　表示查询信息，也经常记为 request.
Fragment 表示片断标识.

为了程序的简化，URL 类的实现主要是解析出 net_loc 部分，用于组成消息体，发送给服务器。

其中 void CUrl::ParseUrlEx(const char *url, char *protocol, int lprotocol, char *host, int lhost, char *request, int lrequest, int *port) 执行具体的字符串匹配找出协议名，主机名，请求信息和端口号，找到后赋给 Url 类的成员变量保存。在 URL 类中还有些是 TSE 抓取过程中的细节函数，如 char* CUrl::GetIpByHost(const char *host)，CUrl::IsValidHost(const char *host)，bool CUrl::IsVisitedUrl(const char *url)等，此处就不一一介绍了。读者可以通过阅读 TSE 的代码获得这部分的具体实现。

2. 定义 Page 类

有了 URL，搜集系统就可以按照 URL 标识抓取其所对应的网页，网页信息保存在 Page 类中。

下面是 Page 类的定义，对应文件 Page.h。

```
class CPage
{
public:
        string m_sUrl;

        // 网页头信息
        string m_sHeader;
        int m_nLenHeader;

        int m_nStatusCode;
        int m_nContentLength;
        string m_sLocation;
        bool m_bConnectionState;        // 如果连接关闭，是 false，否则为 true
        string m_sContentEncoding;
        string m_sContentType;
        string m_sCharset;
```

```
    string m_sTransferEncoding;

    // 网页体信息
    string m_sContent;
    int m_nLenContent;
    string m_sContentNoTags;

    // link, in a lash-up state
    string m_sContentLinkInfo;

    // 为搜索引擎准备的链接，in a lash-up state
    string m_sLinkInfo4SE;
    int m_nLenLinkInfo4SE;

    // 为历史存档准备的链接， in a lash-up state
    string m_sLinkInfo4History;
    int m_nLenLinkInfo4History;


    //为搜索准备的链接，in a good state
    RefLink4SE m_RefLink4SE[MAX_URL_REFERENCES];
    int m_nRefLink4SENum;

    //为历史存档准备的链接, in a good state
    RefLink4History m_RefLink4History[MAX_URL_REFERENCES/2];
    int m_nRefLink4HistoryNum;

    map<string,string> m_mapLink4SE;
    vector<string > m_vecLink4History;

    enum page_type m_eType;                          // 网页类型
public:
    CPage();
    CPage::CPage(string strUrl, string strLocation, char* header, char* body,
            int nLenBody);
    ~CPage();
```

```
        void ParseHeaderInfo(string header);    // 解析网页头信息
        bool ParseHyperLinks();                 // 从网页体中解析链接信息

        bool NormalizeUrl(string& strUrl);
        bool IsFilterLink(string plink);
private:
        // 解析网页头信息
        void GetStatusCode(string header);
        void GetContentLength(string header);
        void GetConnectionState(string header);
        void GetLocation(string header);
        void GetCharset(string header);
        void GetContentEncoding(string header);
        void GetContentType(string header);
        void GetTransferEncoding(string header);

        // 从网页体中解析链接信息
        bool GetContentLinkInfo();
        bool GetLinkInfo4SE();
        bool GetLinkInfo4History();
        bool FindRefLink4SE();
        bool FindRefLink4History();
};
```

Page 类的具体实现请参看 Page.cpp 文件。一个网页是以 URL 作为标识的，所以 Page 类的第一个成员变量是 m_sUrl。Page 类主要完成两个任务：解析网页头信息和提取链接信息。

解析网页头信息包括获得状态码 m_nStatusCode，网页体长度 m_nContentLength（内容字节数），转向信息 m_sLocation，连接状态（如果没有关闭，下次请求同一个网站可以重新利用已经建立好的 socket，节约资源），网页体编码 m_sContentEncoding（如果是 gzip 编码，要解压缩，然后提取链接信息。现在门户网站的首页有增大趋势，为了加快传输速度，通常采用 gzip 编码压缩后传输），网页类型 m_sContentType，网页体字符集 m_sCharset，和传输编码方式 m_sTransferEncoding。

提取链接信息，是从获得的网页体中，根据 HTML 的规定，提取出链接信息

和相应的链接描述信息，形成网页结构库，扩充 URL 库。在 TSE 中对于网页内容的链接提取区分了为搜索引擎提取和为历史网页存档提取两种。因为对于通常意义下的搜索引擎而言，图片链接，网页格式链接是没有用处的，如果不区分开，会增加程序运行的负担，增加存储空间。而且区分开后，可以单独保存下来，便于以后单独搜集这些信息。

二、 与服务器建立连接

已经从 URL 中获得了服务器的主机名，要能够从服务器上获取网页内容，还需要客户端进程与服务端进程建立连接。UDP 和 TCP 的通信采用 Socket 方法实现，Socket 为进程间通信提供了端点。通信由消息组成，消息是在一个进程的 Socket 与另一个进程 Socket 之间传送的。如图 3-6 所示，一个进程要能够接收消息，它的 Socket 必须绑定到一个本地端口和本地地址上。发送到指定 Internet 地址和端口上的消息，只能被绑定到该地址和端口的 Socket 所属进程接收。

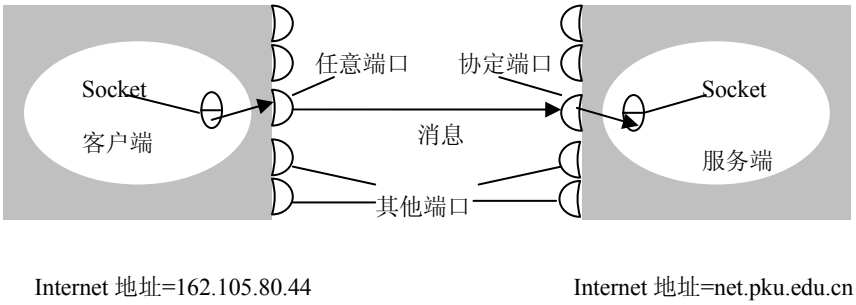


图 3-6 Sockets 和端口

连接的建立过程是异步的，如图 3-7 所示，一方在监听建立连接的请求，一方将发起建立连接的请求。连接一旦被接受，操作系统（例如 UNIX）自动创建新的 socket 使之与客户端连接成通信的通道，这样服务端就可以在原来的 Socket 上继续监听其他客户的请求了。连接建立后，双方进程可以通过建立好的连接进行读写操作。

与服务器建立连接函数 CreateSocket 在 Http.h 中定义，Http.cpp 中实现。在与服务器建立连接的时候，使用了非阻塞连接。超过定时，就放弃。具体代码可以参看 Http.h 中定义的 int nonb_connect(int, struct sockaddr*, int)函数。

考虑到此函数是为 Web 信息的搜集服务设计的，特别增加 DNS 缓存和 IP 范围控制功能。

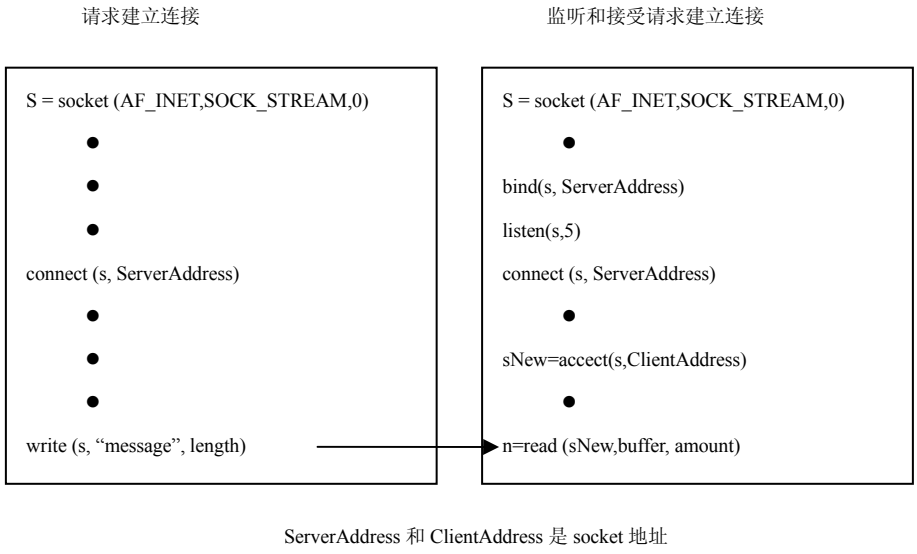


图 3-7 通过 Socket 建立连接

增加 DNS 缓存的原因：

1) URL 数以亿计，而主机数以百万计。如果没有 DNS 缓存，搜集子系统每次搜集新的 URL，即使刚在上一次的任务中解析过该主机名，也要重新进行域名解析（除了用 IP 地址直接表示的主机，如：162.105.80.44）。为了避免频繁的查询 DNS 服务器，造成类似于拒绝服务攻击的副作用，要建立 DNS 缓存。
2) 针对小规模的网页搜集（如百万量级）， DNS 缓存只需要建立在内存中就够了，一是因为内存占用不多，二是因为域名解析有时效性，即使把这些解析出的信息保存到外存上，过几天也失去了意义。如果是为亿量级服务的 Web 信息搜集模块，要考虑单独建立为搜集子系统服务的 DNS 模块，这样能够既加快网页信息的获取，又进一步降低对 DNS 服务器的压力。

增加 IP 范围控制的原因：

1) 有些站点不希望搜集程序搜走自己的资源。比如很多建设在大学范围内的论文检索系统，通常只是对学校 IP 范围内开放的。如果搜集程序处于允许检索 IP 范围内，搜集了这些资源，然后提供服务，就有可能使非授权用户看到了论文信息。
2) 针对特定信息的搜索，比如：校园网搜索，新闻网站搜索。
3) 网络资费方式也会对搜集策略产生影响。有的网络基础设施提供商可能

会规定在对不同 IP 范围信息的访问采用不同的资费政策，因此从运行成本考虑，也可能需要对 IP 范围进行控制。

三、 发送请求和接收数据

1. 构造请求消息体并发送给服务端

搜集端与服务端建立好连接后，前者按照 HTTP 的要求构造消息体发送给服务端。这段实现代码包含在函数 int CHttp::Fetch(string strUrl, char **fileBuf, char **fileHeadBuf, char **location, int* nPSock)中。

HttpFetch 函数中部分代码参考了 http://fetch.sourceforge.net 中的 int http_fetch(const char *url_tmp, char **fileBuf)函数，但是它不能够返回请求 URL 的网页头信息，不能确定网页是否已经重定向，所以有针对性地做了改动。另外由于该函数是用 C 写成的，为了保持原代码的完整，对于 URL 的解析需要单独作一次，而不是采用前面已经给出的 URL 类来实现（URL 类在 TSE 的其他部分会经常用到）。

代码实现在文件 Http.cpp 中。在 TSE 中消息体的组装中采用 HTTP 1.1 协议，抓取程序不主动关闭 Socket 通道，这样当 Web 服务器支持持续连接，后续该网站的网页请求就可以利用已经建立的 Socket 通道进行，可以节省时间和网络带宽。如果服务器不支持持续连接，使用已经建立的 Socket 通道会报错，此时需要重新建立连接。

2. 获取网页头信息和体信息

服务端接受搜集端发送的请求消息后，先返回一个 HTTP 头信息（为方便起见，后面我们称“网页头信息”），其中包含文件类型，大小，最后修改时间等内容；接着是两个“\r\n”，表现为一个空行；然后返回 HTTP 体信息，其中包含网页的全文内容。获取上述头信息和体信息的实现代码同样包含在函数 int CHttp::Fetch(string strUrl, char **fileBuf, char **fileHeadBuf, char **location, int* nPSock)中，在文件 Http.cpp 中。

网页头信息获取后，进行解析，根据返回码，判断 Web 服务器是否针对该请求转向，如果转向，应该重新组装消息体发送请求；根据传输类型，网页体的大小，申请内存空间准备接收，如果超出预定接收大小，放弃该网页；根据网页类型，判断是否获取该网页。如果满足获取条件，继续进行网页体信息的获取。

注意网页头信息中给定的网页体大小有可能错误，所以读取网页体信息应该是处在一个循环体中，直到不能读到新的字节为止。因此根据需求增减内存空间，是格外需要注意的地方。在读取网页信息时，存在服务器长时间不响应的情况，

为了加快搜集效率，要设定超时机制，超时后放弃该网页。接收的数据如果超出预定接收大小，放弃该网页。

四、 网页信息存储的天网格式

将获取网页信息保存在磁盘中，需要按照规定的格式保存，便于后续处理和提供服务。在 TSE 中，采用天网格式存储获得的网页。下面介绍天网格式存储方案。注意这种方案只是顺序保存网页信息，没有索引文件。在第四章中我们会讲到，如何组织这些保存下来的原始网页信息，建立索引文件，支持单个网页信息的检索。

原始网页信息的存储格式应当设计为适合长期保存并易于处理，可以作为终端产品提供给用户使用。考虑到终端产品使用的便利性，要求原始网页库的存储格式具备简单性的特点。

由于存储介质都是有寿命的，所以应当考虑当存储介质损坏时数据的可恢复性。例如：磁盘的某个扇区损坏，导致部分数据不能读出，如果剩下的数据仍然可以使用，就能将损失降到最少。对于海量数据来说，在存储和传输的过程中，由于硬件和软件问题导致数据错误是不可避免的。因此，原始网页的存储格式还应当具备隔离错误的特点。

1. 天网存储格式定义

根据以上考虑，天网存储格式定义如下：

1) 一个原始网页库（RAW_DB）由若干记录组成，每个记录（RECORD）包含一个网页的原始数据，记录的存放是顺序追加的，记录之间没有分隔符；
2) 一个记录由头部（HEAD）、数据（DATA）和空行（BLANK_LINE）组成，顺序是：头部 + 空行 + 数据 + 空行；
3) 一个头部由若干属性组成，每个属性（PROPERTY）是一个非空的行，头部内部不允许出现空行；
4) 一个属性包含属性名（NAME）和属性值（VALUE），并由冒号“:”隔开，顺序是：属性名 + 冒号 + 属性值；
5) 头部的第一个属性必须是版本属性，属性名为 version，例如：version: 1.0，该属性表明记录的版本号；
6) 头部的最后一个属性必须是数据长度属性，属性名为 length，例如：length: 1800，该属性值必须是数据（DATA）的长度（字节数），不包括空行的长度；
7) 为简化起见，属性名必须是小写的字符串。

注：一个空行（BLANK_LINE）仅由一个换行符（line feed，LF，即 C 语言中的“\n”）组成，在 UNIX 系统的显示中表现为一个空行，所以称为空行。Microsoft Windows 系统和 UNIX 系统在空行机制上有所区别：在 Windows 系统下，纯文本显示中的一个空行由一个回车符（carriage return，CR）和一个换行符组成（即 C 语言中的“\r\n”）；而在 UNIX 系统中一个空行仅由一个换行符组成。

2. 当前存储格式版本描述

存储格式允许有多个版本，以满足将来进行扩展的需要。

当前存储格式的版本属性为 1.0。一个记录的存储格式如下（// 后为注释）：

```
version: 1.0                              // 版本号
url: http://www.pku.edu.cn/               // URL
origin: http://www.somewhere.cn/          // 原来的 URL
date: Tue, 15 Apr 2003 08:13:06 GMT       // 抓取时间
ip: 162.105.129.12                        // IP 地址
unzip-length: 30233                  // 如果数据经过压缩，则需有此属性
length: 18133                             // 数据长度
                                          // 空行
XXXXXXXX                                  // 以下为数据
XXXXXXXX
….
XXXXXXXX                                  // 数据结束
                                          // 最后再插入一个空行
```

各属性说明：

version 属性为版本号，以下说明适用版本号为 1.0 的情况。

url 指该网页的 URL，如果因为网页头信息中包含 Location 字段而产生网页转向时，该 URL 为最后实际抓取的 URL 地址。该属性是必需的。

origin 指该网页的原始 URL。该属性仅在 HTTP HEAD 中包含 Location 字段而产生网页转向时存在，指向最原始的 URL。

date 属性为该网页的保存时间，保存格式为 RFC822 所制定的格式。该属性是必需的。

ip 属性为该网页所在服务器的 IP 地址。

unzip-length 属性仅在数据经过压缩时存在，记录数据未压缩时的原始长度。

length 属性记录数据长度。

若存在其它未加说明的属性，应用程序可以简单地忽略。

关于数据是否压缩的问题：天网格式并不指定数据是否必须经过压缩。但是压缩的数据必须包含 unzip-length 属性而未压缩的数据不能包含该属性。该属性同时也是解压缩所必需的。如果数据经过压缩，还应附带说明压缩算法，必要时附带压缩函数库及源代码。

3. 数据的可恢复性分析

假设由于数据遭到破坏，只得到其中一个残存的片段。则可按以下步骤找出该残存片段中所有完整的记录：

1) 特定字符串“version:”，除非没有一个完整的记录，该字符串肯定能找到。记录该字符串的位置 POS。
2) 找到该字符串后，判断其后的数据是否满足存储格式 2)、3)、4)、6)、7）条件。如果任何一个条件不满足，返回 1，从记录的位置 POS 开始继续查找下一个特定字符串“version”。
3) 当满足条件 2 时，假定这是一个正确的记录，则下一个记录也必定是一个正确的记录。检查该记录满足天网存储格式 2)、3)、4)、5)、6)、7）条件，如果任何一个条件不满足，说明原先的假定错误，返回 1，从记录的位置 POS 开始，继续查找下一个特定字符串“version”。如果条件都满足，则继续检查下一个记录是否正确。
4) 如果连续 3 个记录都是正确的，则认为 1）所找到的“version”是一个正确的记录的开始，可以依此提取出全部正确的原始网页。

由于原始网页是随机的，而存储格式是严格的，因此经过上述方法得到的记录为错误记录的可能性极小，是完全可以接受的。

4. 其它问题

在实际应用天网存储格式时，应该注意下面两个方面。

1) 文件打开模式。用标准 IO 库打开文件时有两种模式：文本（text）模式和二进制（binary）模式。在 UNIX 下这两种模式并没有区别，但在 Windows 下如果用文本模式打开，“\r\n”会被当成一个字符，而天网格式中的“length”域表示的是实际字节数，这就可能会引起错误。因此，在用标准 IO 读取文件时，为了兼容性最好用二进制模式打开，例如 FILE *fp=fopen("filename","rb")。
2) FTP 传输。FTP 传输也有两种传输模式：文本（text）模式和二进制（binary）模式。传输原始网页库文件时，应以二进制模式进行传输。如果以文本模式传输，可能会出现“\r\n”被替换为“\n”或“\n”被替换为“\r\n”

的现象，导致数据错误。

第三节 多道搜集程序并行工作

搜集端程序相当于客户端，在 HTTP/1.0 中，即使客户端希望在同一次会话中从同一服务器传输更多的 HTML 页面，该 TCP 连接也会被终止，每一个新的请求需要建立另一个 TCP 连接，这造成了 HTTP 服务器的负担。在 HTTP 1.1 版中，提供对持续 TCP 连接的支持，可以参看 RFC2068 Hypertext Transfer Protocol -- HTTP/1.1[RFCs,2004]。这样改变一下可节省 Web 服务器资源，而且可以节省网络可使用的带宽。此外，由于避免了每次请求都重新建立连接的开支，使用一个持续的连接比 HTTP /1.0 的实现具有更高的操作效率。在 TSE 中使用的是 HTTP/1.1 的请求方式，注意不是简单的更换 HTTP/1.0 这个字串为 HTTP/1.1，而是需要保留上次已经建立的连接，如果该连接没有失效，则本次继续使用。

通常情况下局域网的延迟（latency）在 1-10ms，带宽（bandwidth）为 10-1000Mbps，Internet 的延迟在 100-500ms，带宽为 0.010-2 Mbps。所以针对搜索引擎应用的搜集程序通常是在同一个局域网内的多台机器，每个机器多个进程并发的工作。这样一方面可以利用局域网的高带宽，低延时，各节点充分交流数据，另一方面采用多进程并发方式降低 Internet 高延迟的副作用。因为局域网的利用率几乎接近 1，而 Internet 因为有路由和拥塞控制等额外要求利用率不高，所以局域网与 Internet 的连接是至关重要的。因此需要同时启动多个 gatherer 并发的创建多个 TCP 连接，并发的下载网页。这种方式加快了 Web 信息的搜集，但是要避免多个 gatherer 重复的收集网页（在下一小节中具体介绍解决方法），还要避免由于同一时间内与同一服务器连接过多而给服务器端造成的严重性能问题。

究竟应该有多少个节点并行搜集网页，每个节点启动多少个 gatherer？下面逐步分析得到。先给出理论值，然后给出经验值，最后给出单节点的搜集效率。

计算理论值：

天网 2002 年 1 月统计纯文本网页平均大小为 13KB[Yan and Li,2002]（下文中不特殊说明的网页均指纯文本网页）。

在连接速率为 100Mbps 快速以太网络上（以太网的 MTU（Maximum Transfer Unix）是 1500 字节），假设线路的最大利用率是 100%，则最多允许每秒传输（1.0e+8b/s）/（1500B×8b/B）≈ 8333 个数据帧，也即每秒传输 8333 个网页。

如果局域网与 Internet 的连接为 100Mbs，Internet 带宽利用率低于 50%，则每秒传输的网页数目平均不到 4000 个。

如果搜集系统由 n 个节点组成，单个节点启动的 gatherer 数目应该低于

4000/n。

经验值：

在实际的分布式并行工作的搜集节点中，还要考虑 CPU 和磁盘的使用率问题，通常 CPU 使用率不应该超过 50%，磁盘的使用率不应该超过 80%，否则机器会响应很慢，影响程序的正常运行。在天网的实际系统中局域网是 100Mbps 的以太网，假设局域网与 Internet 的连接为 100Mbps（这个数据目前不能得到，是我们的估计），启动的 gatherer 数目少于 1000 个。这个数目的 gatherer 对于上亿级的天网搜索引擎[天网,2004]是足够的。

单节点的搜集效率：

以太网数据帧的物理特性是其长度必须在 46~1500 字节之间。[Stevens,1996]说明在一个网络往返时间 RTT 为 200ms 的广域网中，服务器处理时间 SPT 为 100ms，那么 TCP 上的事务时间就大约 500ms（2×RTT+SPT）。网页的发送是分成一系列帧进行的，则发送 1 个网页的最少时间是(13KB/1500B) × 500ms ≈4s。如果系统处于满负荷运转，单个节点启动 100 个 gatherer 程序，则每个节点每天应该能够搜集（24×60×60s/4s）× 100pages = 2,160,000 个网页。考虑到 gatherer 实际运行中可能存在超时，搜集的网页失效等原因，每个节点的搜集效率小于 2,160,000 个网页/天。

一、 多线程并发工作

在 TSE 中启动多个 gatherer 是用多线程方式实现的。

代码如下，在文件 Crawl.cpp 中。

```
// 创建多个搜集线程.
Pthread_t tids = (pthread_t*)malloc(NUM_WORKERS * sizeof(pthread_t));
if( tids == NULL)
      cerr << “malloc error” << endl;

for(unsigned int i=0; i< NUM_WORKERS; i++){
     if (pthread_create(&tids[i], NULL, start, this) )
          cerr << “create threads error” << endl;
}
```

对应 start 的工作代码实现功能是多个搜集线程并发的从待抓取的 URL 队列

中取出任务，然后去抓取 URL 对应的网页。请参考 start 过程。

二、 控制对一个站点并发搜集线程的数目

提供 Web 服务的机器，能够处理的未完成的 TCP 连接数有一个上限，未完成的 TCP 连接请求放在一个预备队列，这个队列有一个预先定义的数量限制，它规定在任何时刻能够处理的请求数目。当一个服务器的预备队列达到它定义的限制时，任何新的连接请求都会被忽略，直到队列可用为止。

多道搜集程序并行的工作，如果没有控制，势必造成对于搜集站点的类似于 DoS 攻击的副作用，也就是塞满了 Web 服务器的预备队列，导致后续请求被忽略。

因此在 TSE 的 Crawl.cpp 中增加一段控制代码。

其中 NUM_WORKERS_ON_A_SITE，定义在文件 Tse.h 中，为 const unsigned int NUM_WORKERS = 2，表示一个站点同一时刻最多有 2 个 gatherer 在搜集它的网页。代码的实现是根据待放入 URL 的主机部分，判断待访问队列中是否已经有足够数目的包含该主机的 URL，如果是则等待，直到有空缺才插入待访问队列。空缺的产生是因为 gatherer 程序下载了该 URL，并从待访问队列删除了该 URL。

第四节 如何避免网页的重复搜集

重复搜集，是指物理上存在的一个网页，在没有更新的前提下，被搜集程序重复访问。造成重复搜集的原因，一方面是搜集程序没有清楚记录已经访问过的 URL，另一方面是由于域名与 IP 多重对应关系造成的。下面分情况介绍解决方法。

一、 记录未访问、已访问 URL 和网页内容摘要信息

gatherer 从一个事先制定好的 URL 列表出发（可以理解为初始的未访问 URL 列表），这个列表中的 URL 通常是从以往访问记录中提取出来的，特别是一些热门站点和“What′s New”网页，此外，很多搜索引擎还接受用户提交的 URL。Gatherer 访问了一个网页后，会对它进行分析，提取出新的 URL，将之加入到待访问列表中，如此递归地访问 Web。

完成搜集网页的工作，即使是只有一个 gatherer，也需要解决如何避免重复搜集网页的问题。因此定义两个表，“未访问表”和“已访问表”。“未访问表”中存储准备取入待访问队列的 URL，“已访问表”中存储已经请求过网页的 URL。

这样当 gatherer 因为访问新的网页解析出新的 URL 后，根据“未访问表”，“已访问表”，就可以判断哪些工作已经完成，从而避免重复搜集。

在 TSE 中，除了存储上述“已访问表”和“未访问表”的摘要信息，还存储了已经获取网页内容的摘要信息。存储网页内容的摘要信息的原因是 Web 上存在大量的复制网页，它们是 url 不同，内容完全一样。

在 TSE 中，对访问过的 URL，未访问过的 URL 和获得的网页内容分别作 MD5 摘要（算法可以参看 RFC1321）[RFCs,2004]，获得其唯一标识，建立 3 个集合。新解析出的 URL，首先根据已经访问过的 URL 的 MD5 集合判断是否已经抓取过，如果没有则放入未访问 URL 库，否则放弃；查找的时候可以做到 O(1) 的时间复杂度。

二、 域名与 IP 的对应问题

记录未访问和已访问 URL 信息，可以保证搜集的网页中所有的 URL 都不同。但是域名与 IP 的对应存在复杂的关系，导致即使 URL 不同，也可能指向的物理网页是相同的，这就导致重复搜集。为了解决这个问题，需要分析清楚域名与 IP 的对应关系。

域名与 IP 的对应关系存在 4 种情况：一对一，一对多，多对一，多对多。一对一不会造成重复搜集，后三种情况都有可能造成重复搜集。

后三种情况出现的原因：

1) 可能是虚拟主机。例如：www.pku.edu.cn ，www.gh.pku.edu.cn, www.acgs.pku.edu.cn, caspu.pku.edu.cn 都映射到同一个 IP 162.105.129.12 的情况，但这是采用虚拟主机技术完成的，由于 4 个站点的内容互不重复，认为是 4 个 Web 服务器。

2) 可能是 DNS 轮转。多域名的情况在商业站点中较常见，因为商业站点的单位时间访问量大，需要复制服务器内容，通过 DNS 轮转达到负载均衡，满足用户需求。

例如：

一个域名对应多个 IP。

```
[webg@BigPc ]$ ping ent.sina.com.cn
PING    upiter.sina.com.cn (202.112.8.5) 56(84) bytes of data.

[webg@BigPc ]$ ping ent.sina.com.cn
PING    upiter.sina.com.cn (202.112.8.2) 56(84) bytes of data.

[webg@BigPc ]$ ping ent.sina.com.cn
```

PING upiter.sina.com.cn (202.112.8.3) 56(84) bytes of data.

多个域名对应一个 IP。

[webg@BigPc]$ ping cul.sina.com.cn

PING upiter.sina.com.cn (202.112.8.3) 56(84) bytes of data.

[webg@BigPc]$ ping ent.sina.com.cn

PING upiter.sina.com.cn (202.112.8.3) 56(84) bytes of data.

3) 可能是一个站点有多个域名对应，例如：www.pku.edu.cn 和 www.pku.cn 等价，net.cs.pku.edu.cn 和 net.pku.cn 等价。

如何解决由于后三种情况造成的重复搜集呢？

注意不能简单的根据 IP 地址来判断是否为同一个站点，因为有虚拟主机的情况。要解决重复收集网页，就要找出那些指向同一物理位置 URL 的多个域名和 IP。这是一个逐渐累积的过程。首先要累积到一定数量的域名和 IP，比如 100 万个，然后把这些域名和 IP 对应的首页和首页链接出的最开始的几个页面抓取回来。如果比较结果一样，则应该归为一组。以后搜集的时候可以只选择其中的一个进行搜集。选择的时候应该优先选择有域名的，有的网站对于直接用 IP 访问是被禁止的，例如 www.163.com 对应 IP 地址为：202.108.42.73 ，202.108.42.91，202.108.42.64 ，202.108.42.63 ，202.108.42.71 ，202.108.42.72 。但是直接用 http://202.108.42.73/ 访问是被拒绝的。

在 TSE 中没有实现此功能，有兴趣的读者可以自己练习实现。

第五节 如何首先搜集重要的网页

Web 上的信息具有异质性和动态性，由于受时间和存储空间的限制，即使是最大的搜索引擎也不可能将全球所有的网页全部搜集过来[Lawrence and Giles,1998]，一个好的搜集策略是优先搜集重要的网页，以便能够在最短的时间内把最重要的网页抓取过来。在此要求下，一方面要采用分布并行的体系结构来协同工作，一方面要优先搜集重要的网页。

对于网页重要程度的评定，要依据搜集信息所针对的不同应用而定。从而信息的搜集可以采用不同的策略。对于信息量相对较小的应用，如为发现专业信息而设计的主题 Web 信息搜集系统，可以依据定制的关键词，优先搜集网页中包含或部分包含这些关键词的网页，通过提高该网页 URL 及包含的 URL 的权值来达到目的。对于为处理海量数据而设计的可扩展 Web 信息搜集系统，如何评定一个

网页的“重要度”，目前还是一个值得研究探讨的问题。

根据搜集经验，体现网页重要度的特征有：

1) 网页的入度大，表明被其他网页引用的次数多；
2) 某网页的父网页入度大；
3) 网页的镜像度高，说明网页内容比较热门，从而显得重要；
4) 网页的目录深度小，易于用户浏览到。

这里定义“URL 目录深度”为：网页 URL 中除去域名部分的目录层次，即 URL=schema://host/localpath 中的 localpath 部分。如：URL 为 http://www.pku.edu.cn，则目录深度为 0；如果是 http://www.pku.edu.cn/cs，则目录深度为 1。

这样的特征并非臆断，而是从长期从事搜索引擎工作中得来的，从天网搜索引擎多年的工作及用户行为日志中，可以反映出这种一般性规律，这样的例子如：重要的学术论文网页，因为经常被引用，就表现为入度大；如果被重要的网页引用或多次被其他站点镜像，也可被认为有价值、重要；如网页 URL 目录深度浅，说明位于网站“浅层”，通常是被编辑网页的人认为重要而放在易于访问到的地方，网站的主页或各板块的首页一般被经常浏览而显得重要。

需要说明的是，URL 目录深度小的网页并非总是重要的，目录深度大的网页也并非全不重要，有些学术论文的网页 URL 就有很长的目录深度。多数重要度高的网页会同时具有上述 4 个特征，即上述表示重要度特征的因素并非独立无关的。

网页的权重可以形式化表示为

$$weight(p) = f(indegree(p), indegree(father\_p), mirror(p), directorydepth(p))$$

其中 weight(p)表示网页 p 的权重，indegree(p)表示网页 p 的入度函数，indegree(father p)表示网页 p 的父网页的入度函数，mirror(p)表示网页 p 的镜像度函数，directorydepth(p)表示网页 p 的目录深度函数。

如果能够综合利用或部分利用上述特征，可以认为是重要的。但是如何确定每一个特征量的影响因子却很困难。如果定义的不好，反倒会影响重要网页的发现。是否有简单的方法来确定重要的网页呢？不妨先来分析一下网页的分布情况。

整个 Web 就象一个深不见底的海洋，如图 3-8 所示。把这个海洋分成两个层次，表层和底层[Bergman,2000]。表层包含的主要是“静态网页”（不通过提交查询信息即可获得的页面），底层包含的主要是“动态网页”（需要通过提交查询信息获得含有内容的网页）。目前搜索引擎的工作主要集中在表层工作。在表层中重要网页的分布或者更接近于海面，或者更接近于底层。对于网页的搜集工作，就象一条捕鱼的船行驶在海面上，目的是撒网捕捉尽可能多而且重要的网页。对于

重要网页的获取，人为的策略干预难免会有疏漏，因此不妨考虑如何尽可能多的获得包含重要 URL 的网页问题。实际搜集网页经验说明，网站的首页是飘浮在海面上的，网站数目远小于网页数，并且重要的网页也必然是从这些网站首页链接过去的，因此搜集工作应当优先获得尽可能多的网站首页。由此不难想象宽度优先搜集是尽快获得重要网页最好的办法。采用宽度优先搜集最直接有效的方法就是根据网页 URL 的目录深度确定优先级，这是一个既客观又容易获得的信息。

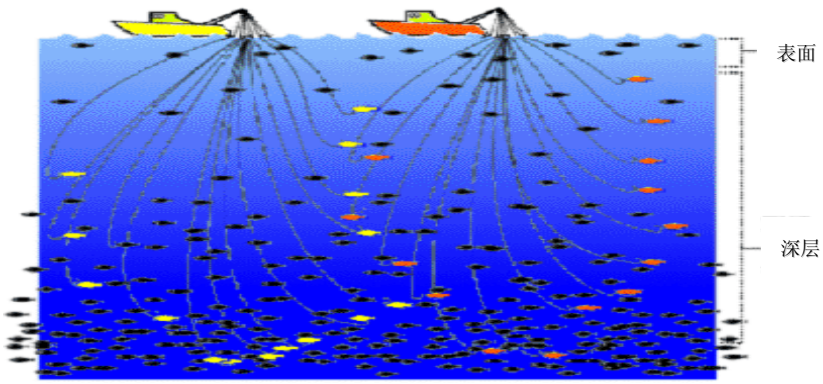


图 3-8 Web 象个海洋

搜索引擎开始工作时，既不知道要搜的网页入度大小（即不知道要访问的网页 URL 被哪些其他网页指向），也不知道网页内容是什么，所以对于表征网页重要性的第 1、2、3 项特征在搜集工作开始时无法确定。这些因素只能在获得网页或几乎所有的 Web 链接结构之后才能够知道。只有特征 4 是不需要知道网页的内容（没有抓取网页之前）就可以确定某个 URL 是否符合“重要”的标准，而且网页 URL 目录深度的计算就是对字符串的处理，统计结果表明一般的 URL 长度都小于 256 个字符，这使得 URL 目录深度的判别易于实现。所以对于搜集策略的确定，特征 4 是最值得考虑的指导因素。

如前面所述，特征 4 具有局限性，所以首先探讨它对于决定网页重要度到底有多大的影响，采取什么样的措施能够尽量避免这种局限性带来的片面搜集，从而对搜集策略做出有效的调整。

由于搜集网页是由多台机器分布式协同工作的，所以搜集策略的制定是在系统的调度层就开始了，从系统的角度考虑，天网 2.0 搜集部分[2]由 18 台+2 台机器冗余构成，采用了两阶段映射模型[Yan, et al.,2001a],[Yan, et al.,2001b]确保分配不

[2] 2000 年，天网搜索引擎升级为 2.0 版，系统结构由集中式改为分布式，支持的数据由百万量级上升为千万量级，乃至目前（2004 年）的亿量级。

同的URL给spider抓取，使不同的机器搜集回来的网页无重复。

对负责搜集的单个节点而言，根据网页 URL 的目录深度和链接关系设定权值，以决定网页重要度，并优先搜集权值大的网页。实现类似于宽度优先搜索的启发式搜集策略。这种搜集策略在实际系统运行中证明有很好的效果。以下主要阐述单个节点上的搜集策略。

以目录深度评估网页重要度，辅以下述的修正方法，在天网实际系统中应用，实际测试表明，可以达到比较好的搜集效果。

1) URL 权值的设定：根据 URL 的目录深度来定，深度是多少，权值就减少多少，权值最小为零。
2) 设定 URL 初始权值为 10（此值设定的越小，从未访问 URL 集合中排序输出就越快。但是也不能太小，否则 URL 的权值意义就不大了，导致搜集策略不明显）；
3) URL 中出现字符”/”，”?”，或”&” 1 次，则权值减 1，出现”search”，”proxy”，或”gate” 1 次，则权值减 2；最多减到零。（包含”?”，或”&”的 URL 是带参数的形式，需要经过被请求方程序服务获得网页，不是搜索引擎系统侧重的静态网页，因此权值相应降低。包含”search”，”proxy”，或”gate”，说明该网页极大可能是搜索引擎中检索的结果页面，代理页面，因此要降低权值）。
4) 选择未访问 URL 的策略。因为权值小不一定说明不重要，所以有必要给一定的机会搜集权值小的未访问 URL。选择未访问 URL 的策略可以采用轮流的方法进行，一次按照权值排序取，一次随机取；或者 N 次按照权值排序取，M 次随机取（N≥1，M≥1）。N，M 的选择可以根据系统实际运行情况获得。

这种方法可以在搜到国内网页总量 1/10 的时候，已经覆盖了国内绝大多数站点。保证“重点突出、覆盖全面”。在具体实现中采用两次轮流充满待访问 URL 队列，可以保证不漏掉其他重要网页，做到兼顾。

第六节 搜集信息的类型

搜集信息类型取决于最后搜索引擎提供服务需要哪些信息，只要把最后提供服务的信息搜集回来，搜集子系统的目的就达到了。

在TSE中不仅限于HTML文本和普通文本，还可以搜集pdf，doc等资源，因为

这些资源可以利用免费得到的转化工具如pdftotext[3]，wvware[4]转化为文本格式。根据MIME规范[RFC2045, et al.]，TSE搜集的网页类型包括：“text/html”，“text/plain”，“text/xml”，“application/msword”，“application/pdf”，“application/postscript”，“application/vnd.ms-execl”，“text/rtf”，“application/vnd.ms-powerpoint”。

在 TSE 搜集网页文件类型内，还具有过滤 URL 功能，该函数在文件 Page.cpp 中实现。

```
/*
 * 过滤掉无用的链接
 * 如果要过滤，返回 ture; 否则返回 false
 */
bool CPage::IsFilterLink(string plink)
{
        if( plink.empty() ) return true;
        if( plink.size() > URL_LEN ) return true;

        const char *filter_str[]={
                "gate","search","library","data/scop","uhtbin","staff/staff",
                "enter","userid","pstmail?","pst?","find?","ccc?",
                "fwd?","tcon?","&amp","Counter?","forum","cgisirsi",
                "+","{","}","proxy","login","mailto:","javascript:"
              };
        int filter_str_num = 25;
        string link = plink;
        CStrFun::Str2Lower( link, link.length() );

        for(int i=0; i<filter_str_num; i++){
                if(link.find(filter_str[i]) != string::npos)
                return true;
        }

        return false;
}
```

[3] http://www.gnu.org/directory/text/misc/xpdf.html
[4] http://sourceforge.net/projects/wvware

为加快搜集过程，在搜集过程中，如果遇到 URL 对应的网页内容超过 5MB，gatherer 放弃该页面。这一过程在获取了网页头信息后（如第二节第三部分所述），根据其中指示的网页大小属性值，可以达到目的。

第七节 本章小结

本章首先介绍了超文本传输协议，为搜集工作提供必要的背景知识；接着给出我们的搜索引擎教学程序 TSE 的系统结构，结合 TSE 的搜集端的源程序进行讲解，分析了一个综合性搜索引擎搜集端所应具备的绝大部分功能。

本章内容是搜索引擎三个步骤（搜集、预处理、服务）的首要环节。对于本章内容的理解，是深入理解整个搜索引擎流程的基础，同时考虑到预处理和服务模块，会更清楚为什么在搜集这个环节需要保存原始网页库和网页结构库。

第四章 对搜集信息的预处理

在第三章中，结合 TSE 搜索的 C++源程序代码介绍了搜索引擎三段式工作流程中的网页搜集部分。本章开始讲述三段式流程中的网页预处理部分。

各节内容安排如下：首先给出信息预处理的系统结构，继而介绍建立索引网页库的算法，接着介绍中文切词技术，然后讲解网页的分析，最后讲如何生成用于查询的网页倒排索引文件。

第一节 信息预处理的系统结构

在 TSE 中网页预处理对应第三章图 3-4 的中间部分，我们将它单独取出来，去掉 TSE 中没有实现的 PageRank 部分后，如图 4-1 所示。

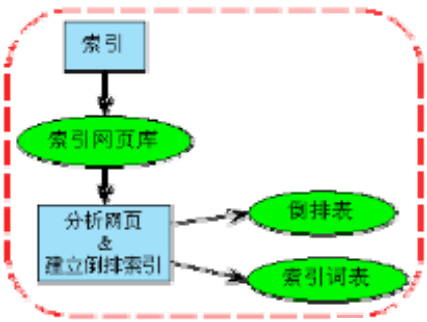


图 4-1 网页预处理系统结构

经过 TSE 的 Web 信息搜集，保存下来的网页信息是按照天网格式顺序存储的。按照天网格式保存网页信息，容错性好，即使有数据损坏，也是局部性的，不会导致扩散或者其他数据无法存取。缺点是不能够按照网页 url，随机存取其所指向的网页。因此网页预处理的第一步就是为原始网页建立索引，实现图 4-1 中索引网页库，有了索引就可以为搜索引擎提供网页快照功能（在本章第二节中讲解）；接下来针对索引网页库进行网页切分，将每一篇网页转化为一组词的集合（在本章第三节中讲解）；最后将网页到索引词的映射转化为索引词到网页的映射，形成倒排文件（包括倒排表和索引词表），同时将网页中包含的不重复的索引词汇聚成索引词表（在本章第四节中讲解）。

网页预处理处于搜索引擎三段式工作流程的中间，所产生的数据都是中间数据，如果不提供必要的应用程序接口，难以作为数据产品提供给其他程序使用。

第二节 索引网页库

使用 TSE 的 Web 信息搜集程序，获得的网页信息是原始网页库，取其中一个文件 Tianwang.raw.2559638448，来进行下面的讲解，它包括 12,933 个网页，容量为 146MB，文件名 Tianwang 表示存储格式符合天网格式要求，扩展名一 raw

```
version: 1.0                                    // 记录头部信息（HEAD）
url: http://net.pku.cn/~cnds/
origin: http://net.pku.cn/~cnds
date: Tue, 16 Sep 2003 14:13:20 GMT
ip: 162.105.203.25
length: 11683

HTTP/1.1 200 OK                                 // 网页头信息（header）
Date: Tue, 16 Sep 2003 14:19:15 GMT
Server: Apache/2.0.40 (Red Hat Linux)
Last-Modified: Tue, 16 Sep 2003 13:18:19 GMT
ETag: "10f7a5-2c8e-375a5cc0"
Accept-Ranges: bytes
Content-Length: 11406
Connection: close
Content-Type: text/html; charset=GB2312

XXXXXXXXX                                       // 网页体信息（body）
....
XXXXXXXXX
```

图 4-2 原始网页库中的记录格式

是表示未经过处理的原始网页，扩展名二 2559638448 是 TSE 搜集程序中线程号为 2559638448 搜集器搜集到的原始网页信息。注意：在 32 位文件系统中，每个文件不能超过 2GB，负责存储网页信息的线程要有检查机制，超过后应该存储到

新创建的文件中。天网格式存储网页支持网页压缩存储，在实际的面向海量数据的搜索引擎中是很必要的，因为网页信息通常可以压缩到原来所占空间的 1/3 以下，节省大量磁盘空间开销。

```
a.  初始化，文档编号 id = 0
b.  for each ri in R do
    begin
        b1.  读取ri的记录头部和数据read(ri)，
                得到URL，ri在R中偏移位置offset和网页内容content
        b2.  记录索引信息write(R, ri )
            b2.1  记录 id，offset 和网页内容摘要 digest(content)到网页索引文件
            b2.2  记录 URL 摘要 digest(URL)和 id 到 URL 索引文件
            b2.3  id 加 1
    end
```

图 4-3 索引网页库算法

在天网格式中一个原始网页库由若干记录组成，每个记录包括记录头部信息（HEAD）和数据（DATA），每个数据由网页头信息（header），网页内容信息（content）组成，如图 4-2 所示，其中具体内容的含义请参照第三章第二节网页信息存储的天网格式。索引网页库的任务就是完成给定一个 URL，在原始网页库中定位到该 URL 所指向的记录。

如果不对网页库建立索引信息，可以通过顺序查找的方法完成 URL 到指定记录的过程，但是会消耗大量的 I/O，数据量增大的时候不能够满足搜索引擎的快速响应要求，所以需要创建索引。对原始网页集 R，索引网页库算法描述如图 4-3 所示。

索引网页库算法在程序 DocIndex.cpp 中实现。搜索引擎面对的是海量数据，选择数据结构的时候需要考虑两个因素：紧凑的数据格式和高效的检索能力。在 TSE 中，对网页内容和 URL 的摘要算法可以采用 MD5 算法，产生 16 个字节的唯一标识。为了便于查看，把 16 字节的标识转化为可以用 ASCII 码表示的 32 字节唯一标识。

网页索引文件 Doc.idx 如表 4-1 所示，以 ISAM（索引顺序访问模式）存储。这种结构可以保证数据的紧凑性和 O(1)的检索能力。为节省空间，索引文件中的每一行记录不保存文档的长度，因为文档长度可以通过后续文档起始位置偏移和当前文档起始位置偏移的差获得。为了保证对最大文档号数据读取的一致性，在

最后一行增加“哨兵”，它不对应实际的文档数据，其中文档摘要为空，表示文档索引的结束。

表 4-1 网页索引文件

文档编号	偏移位置	网页内容摘要
0	0	bc9ce846d7987c4534f53d423380ba70
1	76760	4f47a3cad91f7d35f4bb6b2a638420e5
2	141624	d0194330085 38f65329ae8e39b86026c
……		
12931	210383421	e109b8aa7833faeca510d7619a214544
12932	210411107	

表 4-2 URL 索引文件

URL 摘要	文档编号
5c36868a9c5117eadbda747cbdb0725f	0
3272e136dd90263ee306a835c6c70d77	1
6b8601bb3bb9ab80f868d549b5c5a5f3	2
……	
7bf8caeae9e9b69ebcdc2b1a09dfdb43	12931

URL 索引文件 Url.idx 如表 4-2 所示，以 ISAM 存储，包含了 URL 的摘要和文档编号。为了能够快速的对给定 url 找到对应的文档编号，将表 4-2 按照 URL 摘要排序，得到文件 Url.idx.sort。然后根据二分查找算法在 Url.idx.sort 中查找到对应的文档编号。采用二分查找算法获得网页快照的实现在 Snapshot.cpp 文件中。

第三节 中文自动分词

对索引网页库信息进行预处理包括网页分析和建立倒排文件索引两个部分。中文自动分词是网页分析的前提。文档由被称作特征项的索引词（词或者字）组成，网页分析是将一个文档表示为特征项的过程。在提取特征项时，中文又面临了与英文处理不同的问题。中文信息和英文信息有一个明显的差别：英语单词之间用空格分隔；而在中文文本中，词与词之间没有天然的分隔符，中文词汇大多是由两个或两个以上的汉字组成的，并且语句是连续书写的。这就要求在对中文文本进行自动分析前，先将整句切割成小的词汇单元，即中文分词（或中文切词）。

用具体例子来说明，就是如何把“我的笔记本”这样连续书写的语句切分为“我”，“的”和“笔记本”三个词汇单元。

在检索和文档分类系统中，自动切词系统的速度直接影响整个系统的效率。中文信息的检索主要有两种：基于字的检索和基于词的检索。基于单字的检索系统建立单字索引。在检索时得到每个单字的索引，而后加以适当的逻辑运算，得到检索结果。而基于词汇的检索系统对词汇建立索引，检索词汇时一次命中。TSE系统采用基于词的索引，具有较快的速度和较高的准确性，同时能够减少索引信息对磁盘空间的占用。

本节从介绍切词软件的原理开始，讲到真正切词软件的实现。此切词软件中使用的基本词典包括 108,782 个词条及其对应词频。词典和切词源程序可以在[ChSeg,2003]获得，运行在 Red Hat Linux 8.0 以上的系统中。

1. 算法介绍

自动分词的基本方法有：基于字符串匹配的分词方法和基于统计的分词方法。更多关于语言学的知识可以参看北大计算语言研究所网站 http://icl.pku.edu.cn 内的相关内容，本节以下部分内容直接引自该网站提供的资料。

1) 基于字符串匹配的分词方法

这种方法又称为机械分词方法，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）。

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，可以分为最大或最长匹配，和最小或最短匹配；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：

- 正向最大匹配；
- 逆向最大匹配；
- 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（这可能是因为汉语的中心语靠后的特点）。

对于机械分词方法，可以建立一个一般的模型，形式地表示为 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：

$M(X,Y)=\log\frac{P(X,Y)}{P(X)P(Y)}$，其中 P(X,Y)是汉字 X、Y 的相邻共现概率，P(X)、P(Y)分别是 X、Y 在语料中出现的概率。互现信息体现了汉字之间结合关系的紧密程度。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组频度进行统计，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性。下面具体介绍正向减字最大匹配法。

2. 正向减字最大匹配法

正向减字最大匹配法切分的过程是从自然语言的中文语句中提取出设定的长度字串，与词典比较，如果在词典中，就算一个有意义的词串，并用分隔符分隔输出，否则缩短字串，在词典中重新查找（词典是预先定义好的）。

算法要求为：

输入：中文词典，待切分的文本 d，d 中有若干被标点符号分割的句子 s1，设定的最大词长 MaxLen。

输出：每个句子 s1 被切为若干长度不超过 MaxLen 的字符串，并用分隔符分开，记为 s2，所有 s2 的连接构成 d 切分之后的文本。

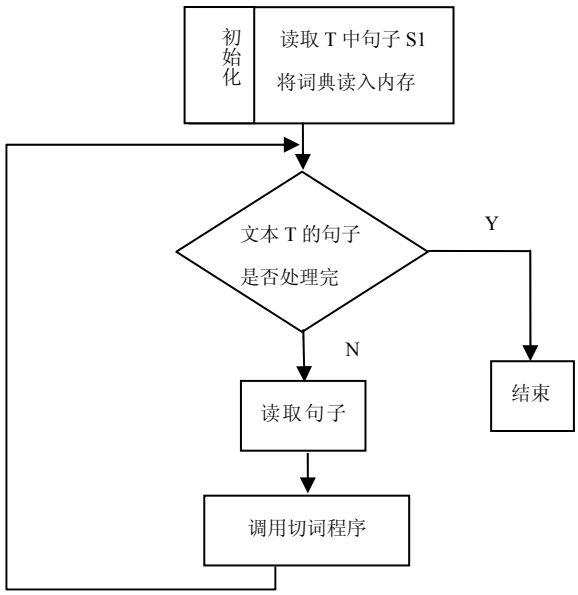


图 4-4 正向减字最大匹配算法流程

该算法的思想是：事先将网页预处理成每行是一个句子的纯文本格式。从 d 中逐句提取，对于每个句子 s1 从左向右以 MaxLen 为界选出候选字串 w，如果 w 在词典中，处理下一个长为 MaxLen 的候选字段；否则，将 w 最右边一个字去掉，继续与词典比较；s1 切分完之后，构成词的字符串或者此时 w 已经为单字，用分隔符隔开输出给 s2。从 s1 中减去 w，继续处理后续的字串。s1 处理结束，取 T 中的下一个句子赋给 s1，重复前述步骤，直到整篇文本 d 都切分完毕。

考虑到词典的条目较多，而且每切一个词都要比较，采用 STL 中的 map 容器[Josuttis,1999]作为存储词典的数据结构。map 容器采用的数据结构是“红黑树”，“红黑树”是一种较常用的查找效率较高的平衡二叉搜索树[Cormen, et al.,2001]。在实际应用中可以采用 hash 表数据结构存储获得更快速的查找。

算法的主程序流程如图 4-4 所示，切词程序 CutWord 如图 4-5 所示。其中 MaxLen 是一个经验值，通常设为 8 个字节（即 4 个汉字），MaxLen 过小，长词会被切断；过长，又会导致切分效率低。

为进行算法分析，先给出算法的伪码实现。

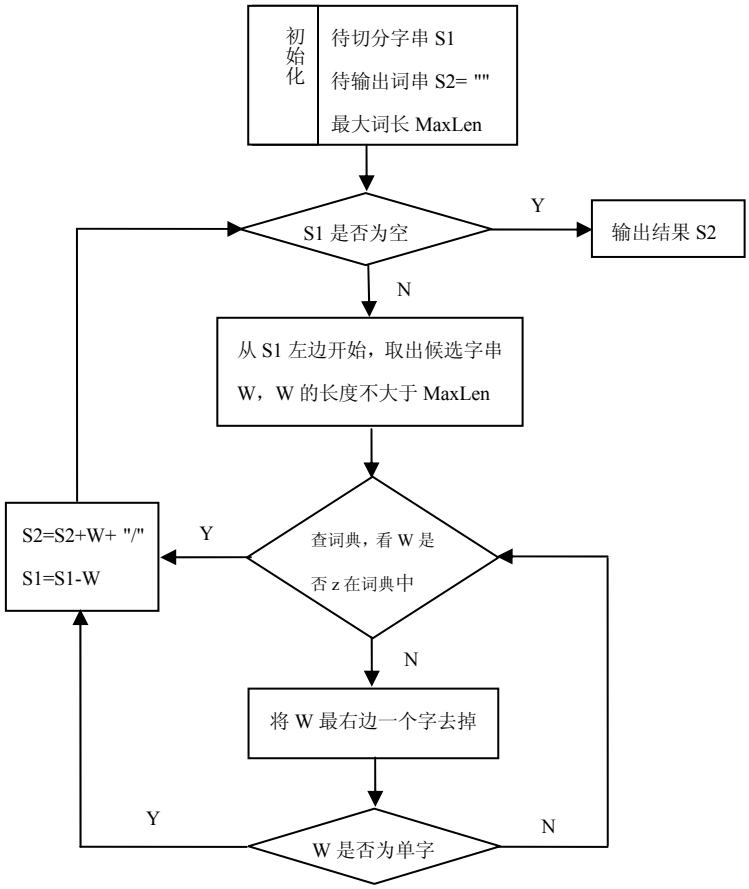


图 4-5 切词算法流程

```
void SegmentAFile (T)                          // 对文件进行分词处理的函数
1       while( getASentenceFromFile (T, s1) )  // 循环读入文件中的每一行
2           S2 = CutWord (s1 )                 // 调用句子分词处理函数
3           OutputSentence ( s2 )              // 将分词结果写入目标文件

string CutWord ( s1 )
4       Preprocess ( s1 )                      // 跳过非汉字部分字符串
5       While (s1 != “”)                       // 如果输入不为空
6           W = s1.substr ( 0, MaxLen )        // 取等于最大词长的候选词
```

```
7              While( length（W）> 1 )
8                  If( FindInRBTree (W)= false)   //如果不是词并且不是单字
9                      then W = W – 1             // 将 W 中最右边一个字去掉
10             s2 =   W + “/”                     // 将找到的词用分隔符隔开
11             s1= s1 – W                         // 去掉找到的词，继续分析
12      return s2
```

算法分析

设文本文件含有句子的数目为 m，句子的平均长度为 k，词典的条目为 n。实际中 m 和 k 远远小于 n，整个算法复杂度中起决定作用的步骤在于 n 相关的语句。

行 1，O(m)

行 2，O($k\log n$) //通过对第 4 到 12 行子程序 CutWord 的复杂度分析得知

行 3，O(1)

行 4，O(1)

行 5，O(k)

行 6，O(1)

行 7，O(1)

行 8，O($\log n$)

行 9，O(1)

行 10，O(1)

行 11，O(1)

行 12，O(1)

整个算法的时间复杂度为 O($mk\log n$)。

上面是对效率的分析，考虑到切词的效果，本算法还可采用[Cormen, et al.,2001]中讲述的“回溯”思想作改进。即除了上述从左到右切分一遍句子，还从右到左切分一遍，对于两遍切分结果不同的字符串，用回溯法重新处理。例如“学历史知识”顺向扫描的结果是：“学历/ 史/ 知识/”，通过查词典知道“史”不在词典中，于是进行回溯，将“学历”的尾字“历”取出与后面的“史”组成“历史”，再查词典，看“学”，“历史”是否在词典中，如果在，就将分词结果调整为：“学/ 历史/ 知识/”。

第四节 分析网页和建立倒排文件

为网页建立全文索引是网页预处理的核心部分，包括分析网页和建立倒排文件。二者是顺序进行，先分析网页，后建立倒排文件（也称为反向索引），如图 4-6 所示。

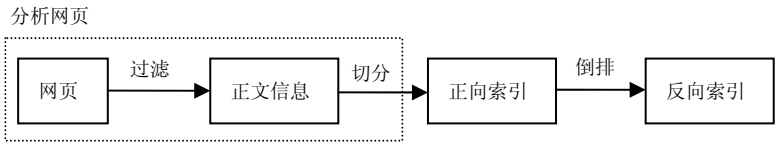


图 4-6 分析网页与建立倒排文件流程

1. 分析网页

分析网页过程如图 4-6 所示，它包括提取正文信息（指过滤网页标签，scripts，css，java，embeddedobjects，comments 等信息）和把正文信息切分为索引词两个阶段。形成的结果是文档号到索引词的对应关系表。每条记录中包括文档编号，索引词编号，索引词在文档中的位置信息，“索引词载体信息”（在本书中网页与文档是不加以区分的，但是网页中包含普通文档所没有的 HTML 标签信息，这些信息标识了文档中索引词的字体和大小写等信息，或称载体信息，是搜索引擎的服务阶段提供更好的结果排序所需要的）。在 TSE 中只保存了文档编号和索引词信息。

过滤网页中非正文信息的算法如图 4-7 所示。

```
a.  初始化，指针 d 和 s 指向文档内容 C 起始地址
b.  for each *s in C do
    begin
        b1.  如果*s 是字符'<'，s 向后移动直到发现字符'>'，s 加 1
        b2.  否则*s 赋值给*d，s 和 d 分别加 1
    end
```

图 4-7 过滤网页中非正文信息算法

得到网页正文信息，调用切词模块，获得正向索引，格式如图 4-8 所示。每一个网页由两行信息组成，第一行是文档编号，第二行是使用切分模块将文档正

文信息划分成索引词后的集合。

0

中国/ 联通/ 郑州/ 分公司/ ……

1

四平/ 风采/ 吉林/ 信息港/ ……

……

12726

逍遥/ 阁/ 建/ 站/ 日期/ ……

图 4-8 正向索引表记录格式

2. 建立倒排文件

搜索引擎面临大量的用户检索需求（几十~几千点击/秒），要求搜索引擎在检索程序的设计上要高效，尽可能的将大运算量的工作在索引建立时完成，使检索时的运算尽量的少。一般的数据库系统不能快速响应如此大量的用户请求，在搜索引擎中通常采用倒排索引技术。

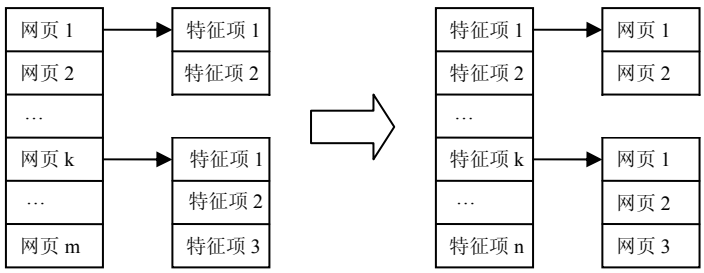


图 4-9 由正向索引建立反向索引

在图 4-6 中，创建倒排索引包括建立正向索引和反向索引。分析完网页后，得到以网页编号为主键的正向索引表，如图 4-9 a)所示（对应 TSE 中的正向索引如图 4-8 所示）。当反向索引建立完成后，得到图 4-9 b)。这是一个表的重组的过程，为了加快速度全过程需要在内存中完成。在小数据量时，有足够的内存保证

该创建过程可以一次完成。数据规模增大后，可以采用分组索引，然后再归并索引的策略。该策略是，建立索引的模块根据当时运行系统所在的计算机的内存大小，将索引分为 k 组，使得每组运算所需内存都小于系统能够提供的最大使用内存的大小。按照倒排索引的生成算法，生成 k 组倒排索引。然后将这 k 组索引归并，即将相同索引词对应的数据合并到一起，就得到了以索引词为主键的最终的倒排文件索引，即反向索引。具体程序实现可以参考 CrtForwardIdx.cpp 和 CrtInvertedIdx.cpp 的代码。

真正的搜索引擎，倒排文件很大，无法直接调入内存，通常在内存中存储以索引词和倒排表项偏移位置组合的 ISAM 信息。实现方法同样可以采用本章第二节介绍索引网页库的方法。

第五节 本章小结

本章结合 TSE 讲解了索引网页库算法，中文切词算法，分析网页和建立倒排文件索引的方法。

本章内容作为搜索引擎运行三个阶段（网页搜集、预处理和查询服务）的中间环节，具有举足轻重的地位。对于本章内容的理解，可以加深对于整个搜索引擎流程的理解，同时考虑到网页搜集和查询服务模块，也会更清楚为什么这个环节需要考虑中文切词、分析网页和建立倒排索引等问题。

第五章 信息查询服务

经过第三章和第四章的学习，现在进入搜索引擎三段式工作流程的最后一个环节——查询服务。查询服务包括接受用户输入的查询短语、检索、获得相应的匹配结果并显示给用户。此时我们已经有了索引网页库和倒排文件，需要做的就是通过查询代理实现索引数据与用户查询的互通。

本章首先给出查询服务系统结构，然后给出检索的形式化定义，最后结合 TSE 给出查询服务的具体实现。

第一节 查询服务的系统结构

在 TSE 中信息查询对应第三章图 3-4 的右侧部分，我们将它单独列出来，去掉 TSE 中没有实现的日志挖掘部分后，得到信息查询服务的系统结构，如图 5-1 所示。

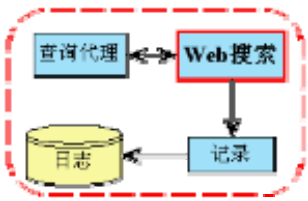


图 5-1 信息查询的系统结构

经过 TSE 的 Web 信息预处理，传递到服务阶段的数据包括索引网页库和倒排文件，倒排文件中包括倒排表和索引词表。查询代理接受用户输入的查询短语，切分后，从索引词表和倒排文件中检索获得包含查询短语的文档并返回给用户。

因为内存与外存（磁盘）的响应时间差距很大，在实际使用的搜索引擎中，为了提高响应时间，索引词表是驻留在内存中的，用户近期查询过的结果信息也是缓存在内存中的。如果内存足够大，所有倒排表项也可以驻留在内存中。只有这样，才能保证在大数据量和大访问量（例如每秒 1000 个查询）的情况下，搜索引擎在秒级内得到响应。

第二节 检索的定义

首先，定义系统的元检索——单个词汇的检索方式。

设 $D=\{t_1,t_2,\Lambda\ ,t_{|D|}\}$ 为系统的特征项词典；集合 $P=\{p_1,p_2,\Lambda\ ,p_{|P|}\}$ 为系统当前保存的网页集合；系统的索引可表示为集合：$R=\{<t,p>|\,r(t,p)>0,(t,p)\in D\times P\}$，其中 $r(t,p)$ 是相关度函数，表示词汇 t 和网页 p 的相关度，如果 t 是网页 p 的一个特征项，那么它就使用相关度算法给出相应的正值，如果不是，相关度为 0；搜索引擎系统 S 为一个三元组 $S=\{<t,p,r>|\,(t,p,r)\in D\times P\times R\}$。则有公式（5-1）：

$$WP(t)=\psi(t,S)=\{p\,|\,(t,p)\in R,p\in P\} \qquad (5\text{-}1)$$

其中，WP 代表检索词汇 t 的相关网页集合，ψ 函数是系统 S 的元检索函数。

显然，用户不可能总是进行词汇级的检索。大部分的用户输入的检索是词组或自然语句。如何从检索中提取出关键词，在各种信息检索系统中实现是不同的，有些系统直接从检索输入中提取关键词；有些系统提取关键词后，再根据一些规则对关键词进行扩充．在本模型中，统一的将它们定义为关键词提取函数：$g(q,D)=\{t_1,t_2,\Lambda\ ,t_m\}$，即通过函数 g 获得检索输入 q 的相关关键词集合．由公式（5-1），关键词 t_i 的相关网页集合为 $WP(t_i)$，则最终检索输入 q 的对应结果为：

$$WP=f(WP(t_1),WP(t_2),\Lambda\ ,WP(t_m))=f(q,D,P,R)=f(q,S) \qquad (5\text{-}2)$$

其中，函数 f 表示从 q 中提取的关键词的逻辑关系运算式，$f(q,S)$ 是系统检索的抽象表达式．由公式（5-1）和（5-2）可以得到，作为文档特征项的关键词在检索过程中起到了桥梁作用，关键词选择的好坏直接影响到检索结果的质量。

第三节 查询服务的实现

一、 结果集合的形成

根据用户输入的查询短语，产生结果集合，是检索倒排索引的过程。首先对用户输入的查询短语应用第四章中讲到的中文自动分词技术，获得查询q的向量表示，$q = \{t_1,t_2,\ldots,t_m\}$，然后执行检索算法，算法描述如图 5-2，实现代码在文件TSESearch.cpp中。这个算法是实际搜索引擎检索算法的简化，应用在TSE中。实际搜索引擎的倒排索引中记录了索引词的权重和位置信息，检索阶段应该一起读出，并加以综合考虑；并且为了在获得结果前读取尽量少的数据，查询q中的t_i按文档频率的倒数降序排列。在TSE中，文档权值由索引词频率决定，并且选取结果K是包含查询词的所有文档。

a. 初始化，结果集合 R=Φ，权值累加器 A=0

b. for each t_i in q do

begin

b1. 读取t_i的倒排项数据read(t_i)

b2. 逐项处理倒排项数据 merge(R,ti)

b2.1 得到t_i的文档集$\{d_{ti}\}$，权值w_{ti}

b2.2 执行布尔查询R=Boolean(R,$\{d_{ti}\}$)，一般布尔运算为AND

b2.3 使用w_{ti}更新结果集合R中元素对应的权值A

end

c. 根据累加器 A 从结果集合 R 中选取最大权值的 K 个结果 select(R,A,K)

图 5-2 基本检索算法

图 5-2 中b2.2 检索运算是算法中的重点，首先因为它的运算量很大。如果记$|P(t_i)|$为集合$P(t_i)$内所包含的网页数目为，那么，它要处理的网页数目就是：

$$NP(q) = \sum_{i=1}^{n} | P(t_i) |$$

其次，它处理的是集合运算。如果系统支持与和或运算，分别对应集合的交和并运算。例如集合 A 和 B，做运算时需要将集合内的元素两两比较，这样的复杂度为$|A|\times|B|$，即平方复杂度。显然，无法接受这么慢的速度，需要采用更加高效的算法。对于理论上提到的集合，在计算机中用表的方式表达。如果将这个表

表示为有序表，集合运算就成了有序表的归并运算。有序表的归并运算的时间复杂度为 O(|A|+|B|)，这可以大大提高了检索运算的效率。

如何将每个特征项对应的网页表达成按照网页编码顺序组成的有序表？这就需要排序。根据理论上的证明，无论采用何种排序算法，排序的最低复杂度为 $O(n\log n)$。但是，在索引系统设计时，可以通过给予先搜集到的网页以小的网页编码的方法，使得索引项自动保持了顺序，避免这一步的运算。

通过以上的分析，可以得出从接收到用户的请求到得到相关的网页，相应的运算的复杂度为 O(NP(q))，即线性复杂度。

对检索得到相关网页集合，下面的工作就是计算每个网页和查询检索串 q 的相关度，它在图 5-2 所描述的基本检索算法的第 c 步完成。

二、 查询结果显示

1. 列表显示摘要结果

用户界面主要用于和用户交互，包括响应用户的查询检索和记录用户的行为。

用户界面主要负责和用户直接接触的事件，如图 5-1 所示，它包括：

1) 获取用户的查询请求，提交给查询代理；
2) 查询代理检索索引词表和倒排表，产生结果按照一定的输出格式显示给用户;
3) 记录日志，包括用户查询短语和查询时间等信息。

对于功能 1)，通过 HTML 语言的<FORM>来实现。用户在相应的检索表格中输入需要查询的短语，然后提交即可。对于一个已经提交的检索，服务器方启动一个 CGI 程序进行响应。该程序对用户提交的各个表项进行合法性检查，通过后形成一个结构数据传给查询代理。

对于功能 2)，主要用到动态网页生成技术和动态摘要算法。按照查询代理检索回来的数据，CGI程序根据决定输出风格的模版，将结果数据整合到模版中，形成结果页面，输出给用户。动态摘要算法如图 5-3 所示，在文件TSESearch.cpp中实现。用户查询q的向量表示为$q = \{t_1,t_2,\ldots t_m\}$。包含查询q的网页正文Cnt，并且Cnt已经过滤掉网页标签等格式信息。为了保证用户检索的所有t_i都被加亮显示，摘要算法是在一个循环中实现的。其中b2.1 是很必要的一步。以中文编码的网页为例，每个汉字占用 2 个字节空间，如果摘要算法不能够正确的判断一个汉字的起始位置，而是从汉字的一半位置开始，就会形成第三章图 3-2 中的部分摘要信息乱码。TSE通过定位完整汉字的起始位置，可以正确显示摘要信息，没有乱码。

a. 初始化，保留 Cnt 开始字节数 resCnt = 48，每个查寻词前后各保留字节数 resTerm = 256

摘要内容 AbsCnt = Cnt 开始的 resCnt 个字节

b. for each t_i in q do

begin

b1. 在Cnt中定位到t_i出现的位置idx

b2.1 如果 idx > resNum，从 idx-resNumb 处到 idx 之间处找到一个完整汉字的位置 cur_idx，AbsCnt += 从 cur_idx 开始的 2*resNum 字节。

B2.2 否则，AbsCnt += 从 idx 开始的 2*resNum 字节

b2. 增加片断中间的分隔符，AbsCnt += "…"

b3. 加亮显示AbsCnt中的t_i

end

图 5-3 动态摘要算法

Wed-Apr-14-16:09:06-2004 北大网络实验室

Wed-Apr-14-16:11:00-2004 印度歌曲

Wed-Apr-14-16:15:10-2004 北大图书馆

Wed-Apr-14-16:15:23-2004 眼泪的故事

……

图 5-4 用户查询日志的记录格式

对于功能 3)，查询代理接受用户输入的信息，记录到日志文件中。图 5-4 是 TSE 中的一段用户查询日志记录，包括用户查询时间和查询短语。

2. 网页快照

搜索引擎索引的网页不一定是当前互联网上最新的网页，因此存在已经消失的可能性。为保证用户能够继续访问相应信息，搜索引擎一般都提供网页快照功能。

TSE 中的网页快照在第三章给出了图示 3-3，在文件 Snapshot.cpp 中实现。索引网页库是用第四章中第二节的方法生成的。

第四节 本章小结

本章内容作为搜索引擎三个步骤（网页搜集，预处理和查询服务）的最后环节，负责把前两个阶段建立好的索引网页库、索引词表、倒排表提供给用户服务，这个交互的过程是通过查询代理完成的。查询代理接受用户的查询请求，在倒排索引中查找符合要求的文档返回，并且提供网页快照功能。

本书上篇第三、第四和第五章这三个连续篇章，以设计并实现一个小的搜索引擎 TSE 为目标，讲述了目前流行的搜索引擎的基本特征，使读者可以快速对搜索引擎技术的整体有一个具体的认识，为进一步理解本书的中篇和下篇内容打下基础。

对搜索引擎技术感兴趣的读者，除了参考我们的 TSE 开放代码外[TSE,2004]，还可以参考其他的开放代码的小搜索引擎程序，例如：Swish（网址 http://swish-e.org）； htDig（网址 http://sourceforge.net/projects/htdig/）；Clucene（网址 http://sourceforge.net/projects/clucene/）等。

中篇　对质量和性能的追求

上篇中，我们讨论了搜索引擎的基本工作原理，并通过一个实际的例子，阐释了这些原理在一个简单搜索引擎中实现的各个细节。同时，在上篇中我们也多次提到了性能问题和质量问题。尽管没有展开讨论，但其中的要素已经显示出来。事实上，这两个问题在搜索引擎的三个子系统中都有不同程度的体现。

中篇将围绕这些因素展开，具体来说，将讨论五方面的内容：

1. 一个并行搜集子系统的详细设计方案。尽管一个搜集子系统的硬件并行度不需要很高，但不做并行是达不到性能要求的。而能否在较短的时间内搜集到足够多的网页，也对搜索引擎系统后续的服务质量有直接影响。但只要是多个节点同时在网上搜集网页，就会引入新的技术问题。这将是第六章讨论的内容。
2. 在我们浏览网页，从中获取所需信息的同时，还会常常看见大量和我们所关心内容无关的“噪音”内容，如广告信息、版权信息等，有效的去除和网页主题内容无关的噪音内容，提取网页的元数据信息，如关键词、摘要、网页内容类别等，是 Web 信息处理的一项重要内容。在网页搜集的过程中，通过两个数据结构 unvisited_table 和 visited_table，我们可以完全避免对相同的 url 执行多次网页抓取过程。但这并不保证抓到系统中来的网页都是不同的。Web 上大量的网页镜像和转载现象使得内容真正“独立的”网页要比实际搜集到的网页少很多。将相似的网页识别出来，当查询发生时只返回一个代表，这既是提高查询服务效率的需要，也是提高查询服务质量的需要。这是第七章的内容。
3. 查询子系统的性能模型和并行查询子系统。能够同时响应的用户查询数量，是商业搜索引擎的一个基本指标。与门户网站一样，商业搜索引擎也是以日访问量作为基本的业绩指标，但和门户网站不同的是，每一个访问在搜索引擎上引发的过程要复杂许多。因此，设计一个能够承载每秒几十个，每天上百万人访问的查询子系统是一个非平凡的工作。高档的硬件设备有可能降低设计的压力，但第八章立足于普通的 PC 机群，给出了一种性能评价模型。天网的实现和测试表明，该模型在系统设计之前，作为一种先验的性能估计手段是可用的。
4. 查询局部性的发现与利用。从第八章的分析中我们会看到，查询性能的一个主要瓶颈是磁盘操作。第九章通过分析天网用户查询序列的特点，发现在查

询内容上存在很强的时间局部性，即在任何一个不长的时间段内，许多用户关心的内容是相同或者相似的。这个结论和文献中的一些结果吻合。这种现象提示我们，通过在内存中设计查询缓存和热点击缓存，有可能明显减少系统的磁盘操作，从而提高查询的效率。天网的实践也验证了这种认识和理论。

5. 相关排序与质量评估。从根本上讲，搜索引擎的服务质量体现在查询结果序列上。首先我们需要有足够多的结果来参加排序，这由搜集足够多的网页和良好的词典设计等技术来保证；然后是这些结果的排序问题。一般来讲，影响结果排序有两类因素，静态因素和动态因素。静态因素指的是和查询无关的因素，例如网页内容的重要性、新颖性、权威性等，可以在预处理阶段形成；动态因素指的是和查询相关的因素，例如和查询词的匹配程度，查询词在网页中出现的“份量”等[1]。它们是第十章的讨论内容。

[1] 例如，在同一篇网页中出现在<H1>和</H1>之间的词很可能要比出现在<H4>和</H4>之间的词更有份量。

第六章 可扩展搜集子系统

天网域名 e.pku.edu.cn, 寓意心有灵犀一点通，就是鼠标一点（e.）天网就在用户的疑问与答案之间建起了通途。天网 1.0 采用集中式系统结构，搜索量为百万级。为了扩大信息搜集规模，让系统承载能力和中国网页规模增长的速度同步，在 1.0 基础上推出了以中国全部 Web 信息为对象的天网 2.0，为此我们重新设计了系统结构，修改了实现方法，设计了可扩展性的 Web 信息搜集系统。本章将详细介绍此系统的设计和实现。

天网主要包括搜集子系统、索引子系统、检索子系统三个组成部分。可扩展 Web 信息搜集子系统是天网核心部分之一，由 N 个独立自主的集中式系统和协调模块组合而成。本章先介绍天网的总体系统结构，再介绍集中式搜集系统的设计与实现，接着重点介绍可扩展搜集系统。然后通过模拟实验与实际环境的实验，分析说明可扩展分布式系统结构如何达到设计目标。

第一节 天网系统概述和集中式搜集系统结构

一、 天网系统结构

1997 年发布的天网 1.0 版采用单机搜集、单机服务的系统结构（我们习惯称之为集中式结构）不适应 Web 上信息规模的迅猛发展，为此我们从 1999 年开始花了大约一年的时间设计和实现了天网 2.0 版的分布式并行系统结构。

系统分布的核心是数据的分布。对搜集部分而言，实际是将 URL 分布在执行搜集任务的机器之间，保证它们搜集的 URL 不会重复。对查询部分，则是将索引数据分布在执行检索任务的机器之间。天网 2.0 系统概貌如图 6-1 所示。为了突出搜集和查询部分的并行化问题，其中略去了搜索引擎三段工作流程中的预处理部分（尽管在大数据量时它也是需要并行的！）。

搜集节点之间相互协调，分配 URL，保证一个 Web 主机的全部网页只能存在于一个搜集节点上。每个索引节点对应搜集节点搜集的网页，查询代理节点通过多播向所有索引节点发送查询命令，等待搜集到全部索引节点返回的检索结果后，对所有结果依据相关度排序，并缓存一定数量的结果，最

后向用户返回结果的首页。用户的后续查询（翻页），将会在缓存命中，不必再次启动后面的网络查询，这将大大减少查询的响应时间，降低后台查询系统的负载，从而提高查询系统的性能。

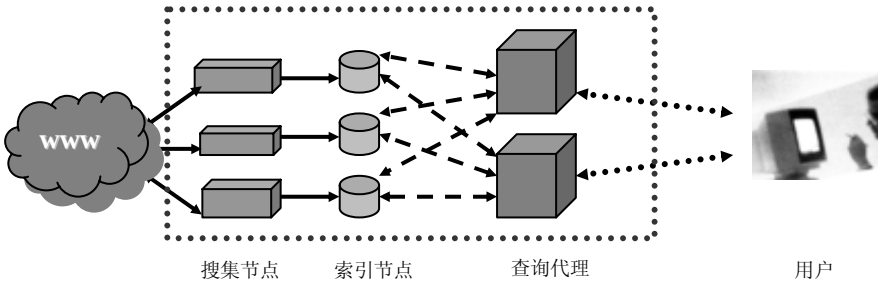


图 6-1 天网系统概貌

第二章的图 2-3 示出了天网的详细系统结构。可以看出，从功能模块上划分，天网系统由搜集子系统、索引子系统、检索子系统和日志挖掘子系统四个子系统构成。搜集子系统包括主控、搜集器和原始数据库；索引子系统包括索引器和索引数据库；检索子系统包括检索器和用户接口。日志挖掘子系统包括用户行为日志数据库和日志分析器。主控除了按照启发式算法优先选择重要的 URL 并分派给各个机器人外，还完成站点过滤、实现 robot 协议和域名解析并缓存等功能。搜集器按照 HTTP 协议负责从 Web 上抓取网页，为提高网页搜集速度，通常可以启动上百个搜集器同时工作。搜集器同时对搜集回来的网页内容进行分析处理，包括调用切词软件以提取关键词和摘要、提取 URL 超链、记录网页的元信息（如作者、修改日期、长度等），并将这些内容存入原始数据库。索引器将原始数据库的内容重新组织，建立索引数据库，以提高检索效率。用户接口在截取用户的查询请求后，将它转发给检索器，检索器根据查询项和索引数据库的内容，找到匹配的网页后，进行相关度计算并排序，然后通过用户接口返回给用户。另外，用户接口程序还将用户行为信息（包括用户查询项、用户点击的 URL、用户翻页情况等）记录到日志数据库。日志分析器用于跟踪用户行为，以提高搜索引擎的服务质量，如可以学习新词来动态更新词典内容。

二、 集中式搜集系统

搜集系统包括主控模块、搜集器和原始数据库。主控模块是其中的控制模块，它主要负责：

1) 与网页抓取与分析进程的交互：发送配置信息，发送 URL，接收分

析结果。

2) 与原始数据库的数据交互。
3) 访问控制：智能导向，robots 协议，主机访问频度，IP 地址等的控制。
4) 与外部系统的接口。

1. 系统设计目标

1) 主控与网页抓取与分析进程的分布。在系统设计中，必须采用分布式技术将任务分布到多台机器上并行的处理。海量网页独立的分布在网络上，对并行访问提供了充分的可能性和合理性。同时，分布并行还会节省网络带宽资源。
2) 可定制性。系统可以让用户依据自己对信息的兴趣，配置用于引导系统搜集的导向词，以及搜集的范围。
3) 良好的开放性。尽可能使用和遵循现有的标准和协议，加强与其它系统交换信息的能力，包括支持 HARVEST 系统的 SOIF 信息格式和信息内容，以及 robot 协议。
4) 一定的扩展性。系统能在 CERNET 的网络环境下有效的运行，不需改动或改动很少就能适应不同的需要。
5) 一定的智能化。为了提高整个系统的查全率、查准率及搜索速度，搜索的内部算法应具备一定的智能化。

2. 系统结构和主要设计思想

集中式系统结构主要体现在主控的系统结构上。主控系统结构如图 6-2 所示。可以看出主控系统由 6 个进程组成，这些进程的功能如下：

“主进程”负责产生其他五个进程，接受“抓取分析进程”的连接并交互。。

“结果插入进程”通过 PIPE 接收主进程得到的访问结果，通过各种检查后存入数据库已访问网页列表。其中新的 URL 放入 NewUrlCache 或者新的未处理 URL 列表（当 NewUrlCache 满时）中，等待“新 URL 处理进程”进行处理。

“robots存取分析进程”得到URL选取进程发出的SIGUSR1 信号后，检查主机表中需要处理的表项进行存取分析。

“URL 过期检查进程”定期检查数据库中已经访问的网页，如过时，则将该主页的 URL 放入未访问 URL 列表中。

“未访问 URL 选取进程”从未访问 URL 列表中选取合格的未访问的 URL 放入缓存中。

“新 URL 处理进程”从 NewUrlCache 或新的未处理 URL 列表中，取出新 URL 进行处理。

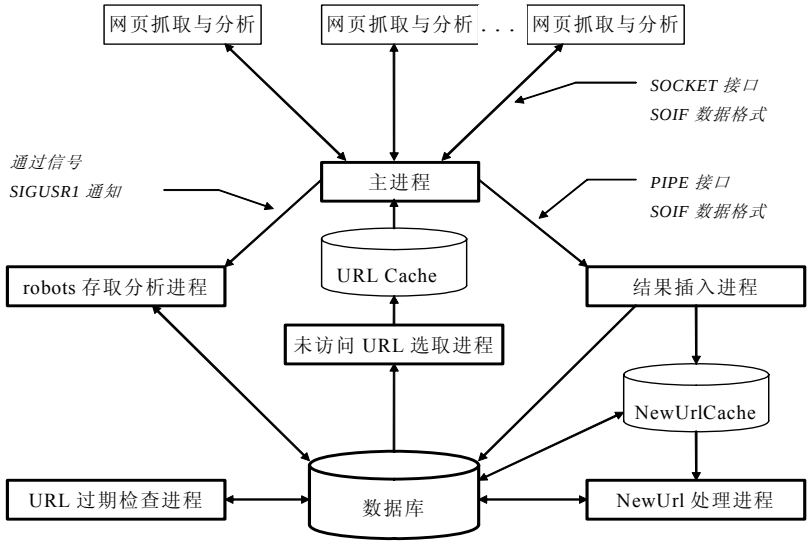


图 6-2 搜集系统的主控结构

表 6-1 Soif 数据描述

数据包格式	描　　述
@GETO{- }	存取分析进程发向主控，索取导向词。
@PUTO{- Orientation-words{n}: Sensitive-words{n}: }	主控响应存取分析进程，发回导向词。
@GETU{- }	网页抓取与分析进程发向主控，索取要进行存取分析的 URL。
@PUTU{URL Last-Modification-Time{n}: }	主控发向存取分析进程，响应 GETU，发送 URL。

@PUTR{URL Result{n}:→result code [Abstract Author Description File-Size Keywords Last-Modification-Time Title Type URL-References Time-to-live Weight Code-Type Retry-After Chars] }	存取分析进程向主控返回结果。 ---结果码； ---以下项可能没有： ---含有 anchor 的权值， 范围 1 – 1000 ---此文章的权值，范围 1– 1000 ---此文章的编码； ---文章的权值较高的单个中文字；
@GETS{- }	敏感内容监控进程发向主控，表明自己实时地需要敏感信息。
@PUTS{URL Sensitive-number{n}: }	当主控发现敏感信息时通过此数据包将信息发送给监控进程。URL 为敏感内容的地址，Sensitive-number 为目前为止发现的敏感文章的总数。

主进程与网页抓取与分析进程之间的交互是通过 SOCKET 来实现的，这样主控与存取分析进程可以运行于不同的机器上。它们之间是通过 SOIF 数据格式进行交互的。另外，敏感内容监控程序为了实时监控，也向主控发 GETS 的 SOIF 数据包，主控对此连接加以标识，每当出现敏感内容时就通过此连接发送 PUTS 的 SOIF 数据包。系统所用的 SOIF 数据包如表 6-1 所示。

主进程与结果插入进程之间是通过 PIPE 进行通信的，当主进程收到 PUTR 的数据包时便转发给结果插入进程，由结果插入进程处理结果。插入进程对新的 URL 不做处理，而是放入新 URL 的缓冲区，或当缓冲区满时放入数据库。

新的 URL 最终由新 URL 处理进程统一处理。重复的抛弃，未访问的插入数据库的未访问表中。

主进程与 robots 存取分析进程的通信：当主进程将新的主机插入到数据库中的主机表后，就向 robots 存取分析进程发 SIGUSR1 信号进行通知。Robots 存取分析进程在信号处理过程中置标记，随后在主机表中查询到相应的表项进行处理。

表 6-2 Soif 具体语法

SOIF	OBJECT SOIF | OBJECT
OBJECT	@TEMPLATE-TYPE{URL ATTRIBUTE-LIST}
ATTRIBUTE-LIST	ATTRIBUTE ATTRIBUTE-LIST | ATTRIBUTE
ATTRIBUTE	IDENTIFIER{VALUE-SIZE}DELIMITER VALUE
TEMPLATE-TYPE	Alpha-Numeric-String
IDENTIFIER	Alpha-Numeric-String
VALUE	Arbitrary-Data
VALUE-SIZE	Number
DELIMITER	:<tab>

主进程与 URL 选取进程的通信是通过共享内存来进行的。将共享内存组织成循环队列，主进程与选取进程通过 PV 操作来实现互斥区的操作。

可以看出，通过这种设计，系统的功能及模块的划分比较清晰，主控模块中多个进程并发工作极大地提高了搜集处理信息的速度。另外，系统在选取未访问 URL 时和处理新 URL 时使用了两个缓冲，未访问 URL 选取进程不必每次等待主进程取走选取结果而可以继续选取。结果插入进程也不必等所有新 URL 处理完才从主进程接收下一个结果进行处理。这样，进一步提高了进程间的并行程度。

管理员可以根据实际情况，将主控和网页抓取与分析进程进行分布和组合，也可以动态控制网页抓取与分析进程的数目。以达到分担服务器负载，提高并发度，加快信息搜集的目的。

在进行信息访问时，我们同时要遵循有关“Web robot”的各种约定：例如，不要在短时间内多次访问同一个服务器；获取“robots.txt”文件，不访问由它指定的目录等。为此我们在数据库中建立了一个主机表和一个禁止访问目录表，主机表记录了最近一次访问一个主机的时间，以及

“robots.txt”文件的访问信息（没有、超时或成功访问）。只有在当前时间与此主机上次访问时间之差大于规定的时间间隔，并且不在禁止目录表中时，才允许访问此主机上的 URL。

每个节点网页抓取与分析进程和主进程之间信息交换的格式采用了扩展的 SOIF 格式，如表 6-2 所示。

从 SOIF 的语法定义中我们可以看出，每一个对象“OBJECT”包含一个模板类型“TEMPLATE-TYPE”，一个 URL 和一串由字节数控制长度的属性和属性值的列表。由于是由长度分隔属性值，SOIF 属性值也可以是二进制的。SOIF 的格式易于分析，易于扩展，格式简单并且有足够的表达摘要对象的能力。它可以由软件自动生成，也可以手工建立。SOIF 有几个基本的属性，它们是:Abstract, Author, Description, Keyword, Title。

下面给出一个 SOIF 的例子：

```
@FILE{http://net.cs.pku.edu.cn
Abstract{57}: HomePage of Computer Networks Lab of CS Peking University
Author{11}:   LI Xiaoming
Title{12}:    Networks Lab
Keywords{21}:    networks computer lab
Type{4}:      HTML
File-Size{4}: 1248
}
```

在摘要信息表示中，常用的属性有：Abstract，Author，Description，File-Size，Full-Text，Keywords，Last-Modification-Time，Refresh-Rate，Time-to-Live，Title，Type，Update-Time，URL-References 等。

可以看出，SOIF 提供了通用的信息表示格式。不同的信息系统之间可以通过 SOIF 格式来交换信息，从而达到共享信息的目的。我们的系统支持所有的常用属性，可以通过 SOIF 与其它的系统交互，具有良好的开放性。

目前 Web 上信息量庞大，试图穷尽搜索比较困难。但是，完全有可能在搜集的过程中尽量优先搜集用户配置的感兴趣的信息，或者重要性较高的信息。这样尽管没有穷尽 Web 上的信息，但是已经搜集的信息对用户的利用价值也会很高。这一点是通过加权的启发式搜索算法来解决的。每个 URL 有自己的权值，反映了其文档内容的重要性，没有访问的 URL 有一个预测权值，访问后计算出其真正权值。访问时，就依据权值的大小来决定优先顺序。我们是通过以下几个参数来预测 URL 的权值的：父 URL 的权值，即本 URL 所在文档的权值；URL 的 Anchor 权值，这是出现在 HTML 文件中对

URL 的描述信息的权值；URL 目录长度权值，每个 URL 都有一个目录（位于主机名和端口号后），一般来说，目录短的 URL 重要性相对高些；被别的文档引用的次数；加入者的信息；URL 的域名的深度。另外，还有别的因素也会影响权值的计算，例如，如果用户定制的是优先搜集中文信息时，中文编码的 URL 权值就应高些。

还有用户也可以为系统配置导向词，当文档内容同用户配置的导向词相关度较大时，文档的权值就会较高。在预测权值时，URL 所在文档和此 URL Anchor 信息与导向词的相关程度会影响预测权值的高低。

另外，在 URL 发现过程中，还可以主动的预测可能有价值的 URL，例如，得到了“http://www.pku.edu.cn/a/”可以推测“http://www.pku.edu.cn/”是一个有价值的 URL。这样做进一步增加了信息发现的主动性。

在 URL 被加入数据库中时，结果插入进程要调用过滤函数对 URL 进行过滤，合格的 URL 才会被插入到数据库中。对 IP 地址的过滤，主控提供了一个配置文件 field.conf。用户可以利用此文件定义 IP 地址的范围。

过滤重复的 URL 也是很重要的，这样可以避免重复的访问和冗余信息。查重的时候，首先要规整 URL，排除“./”或者“../”或者 URL 中的编码，例如“foo/bar/../baz.html”与“foo/baz.html”是等价的。其次，还要注意 URL 的主机名部分可能是 IP 也可能是此 IP 地址的多个域名。

对一个过期 URL，进行重复访问更新时，我们可以利用 HTTP 中的条件取操作，即“If-Modified-Since”选项，如果文档没有改动则系统不需要重新获取该文档。需要注意的是，动态网页都不包括网页最后修改的时间，不能用这种方法进行更新检查。

另外，在系统进行信息发现过程中，可能对域名解析的需求比较大，往往有时在短时间内多次对一两个域名进行解析。这时，可将域名解析的结果存入本地缓存，不但会加快信息搜索的速度而且会减少解析域名带来的网络负载。

第二节　利用并行处理技术高效搜集网页的一种方案

Web 上成千上万的 WWW 服务器通过网页之间的链接构成海量信息，各个主机之间的联系或多或少，但都可以说是相对独立的。单处理机系统受限于 CPU 的处理能力、磁盘存储的容量，不可能具备处理这种海量信息的能力，更不必说跟上 Web 信息的飞速增长了。

采用并行处理技术成为一个自然的选择。高性能并行计算机系统的种类有很多：SMP, NUMA, MPP, 机群。比较起来，后者对我们的应用是最适合的。这不

仅是由于其价格较低，还由于我们网页收集应用的基本特征：操作单纯，进程之间的通信量不大，对磁盘容量和聚集 I/O 吞吐率要求高，需要有经济效能较高的系统可扩展性以适应未来信息量增长的要求。这些都使我们认为机群是最好的选择。

在网页搜集子系统中，最基本的任务是一篇网页的抓取，它大致上要完成如下功能：从任务池中取一个网页的地址 URL，通过 DNS 得到其 IP 地址，用该 IP 地址与 Web 服务器建立 TCP/IP 连接，发 HTTP 请求，等待接收 HTTP 应答，关闭 TCP/IP 连接，分析收到的网页，将其中包含的新链接加入任务池中，将网页存放到磁盘数据库中。

由于网页抓取任务的这些阶段内容在完成时间上的不均匀性和不确定性（例如取决于网络状态等等），系统的并行性可以很自然地体现在两个层次：一个节点内部多个进程的并行和节点之间的并行。我们看到，如上所述系统微观上表现为明显的任务并行特征，但为了减少通信量，设计中考虑了一种在节点间采用数据并行、节点内采用任务并行的基本策略。即首先是确定一个在节点间动态划分网页 URL 的算法，保证不同的节点收集的网页不会重复，而在节点内由抓取进程自由获取下一个任务。

作为一个并行系统的设计，我们追求如下目标：单台机器的搜集能力不应随着搜集机器数量的增加下降很多，即要在追求负载平衡的同时将系统的通信和管理开销降到最低。同时，从实际运行出发，我们还要考虑系统的动态配置问题，即要允许在运行过程中添加和删除节点机器（这是因为一次搜集时间比较长，在这期间有时难免出现机器故障等，需要修复然后再投入服务；而在修复期间系统应该继续运行）。下面的讨论将按照如下要点展开。

1) 在节点间划分 URL 的策略
2) 关于性能的讨论，包括负载平衡、通信开销和可扩展性等
3) 性能测试和评价
4) 系统的动态可配置性设计

一、 节点间 URL 的划分策略

为方便讨论起见，本节我们约定如下符号和术语。

URLs= $\{URL_1, URL_2, \ldots\}$，所要完成收集的网页地址集合。这是一个开放和动态变化的集合。所谓“开放”，指其中元素的个数是事先未知的，具体有哪些元素当然也事先未知。在本文讨论的意义下，URLs的大小至少在千万量级。所谓“动态变化”，指它在收集过程中随着新发现的地址增加。通常，一次搜集过程由某些

“种子”网页开始，沿着它们包含的超链，按照某种搜索策略（先宽，先深，等等）往下进行，直到没有新的地址发现，或者人为决定不要再进行了（例如磁盘已满）。

HOST(URL)，一个网页地址的域名部分（或称主机部分），通常对应某台Web服务器，例如URL = http://net.pku.edu.cn/pp_outline.htm，HOST(URL) = net.pku.edu.cn。于是我们可以看到在URL上有一个按照域名的自然划分（即url_1和url_2同属这个划分的一个“块”，当且仅当$HOST(URL_1)=HOST(URL_2)$），记为HOST(URLs)，即它的每一个元素是一台主机上所有网页的集合。在不至于引起混淆的情况下，我们也可以方便地将HOST(URLs)的元素看成是所涉及主机的域名。在本文讨论中国网页的意义下，HOST(URLs)的大小在10万量级（同样在不至于引起混淆的情况下，我们有时也用HOST(URLs)直接表示它的大小，于是符号HOST(URLs)的含义在本文是“重载的”，由上下文确定具体是哪一种。）。

抓取进程，负责根据一个 url 完成一篇网页的抓取、分析和存储。它不负责更新任务池，只是将得到的链接返回给下述协调进程。

让 n 表示并行收集系统的节点数，$[n] = \{0,1,2,\ldots,n-1\}$表示网页搜集节点的集合。经验告诉我们每个节点启动200个左右并发抓取进程是比较合适的，它们共享（争用）CPU、网卡和硬盘资源。从原理上讲，我们可以考虑从URLs到$200\times n$个进程之间的映射。但我们认为所带来的管理复杂性要比可能带来的好处大得多。于是我们考虑粗一些的粒度，将目标设定在URLs上形成一个“优化的”n-划分。而在节点内部的并发进程则以一种自由的方式来共同完成分到该节点的任务。

何为优化？就是要在负载平衡和降低开销之间求得一种最好的折中。由于每个任务的执行时间可能相差很大（受网页的大小、网络状态的波动等的影响），简单的按照任务个数的平均分配即使对于负载平衡也是没有意义的。

我们所采用方法的基本思路是对 HOST(URLs)进行随机分配，即建立一个从HOST(URLs)到$[n]$之间的映射。一旦一个HOST(URL)映射到了某一搜集节点，该搜集节点就要负责 HOST(URL)下面所有网页的收集。尽管不同的 Web 站点含有的网页数量会相差很大，但由于$HOST(URLs) >> n$，并且如果我们的分配函数足够随机，则可以认为分到各个节点上的网页个数比较均匀的可能性很大。

具体来说，每个节点维护两张表，visited_table, unvisited_table，分别代表到目前为止本节点已经抓过的网页和需要抓取的网页。宏观上看，整个系统有向量visited_ table[n], unvisited_ table[n]。每个节点除启动若干抓取进程外，还运行一个**协调进程**，controller。协调进程的工作是根据本节点抓取进程和从其他节点的协调进程来的信息，维护本节点的 visited_ table 和 unvisited_ table。下面是协调进程的工作算法如图 6-3，算法中从抓取回来的网页中解析出的超链接集合 links 包含的每一个 link 就是一个 URL。

```
for(;;)
begin
    a. 等待从其他节点传来的一个 URL，或者它所管辖的抓取进程返回的一个 URL 及相应网页。
    b. 若得到其他节点传来的一个URL
        b.1 看 URL 是否已经出现在 visited_table 中，若没有，则将 URL 放到 unvisited_table 中；
    c. 若得到从抓取进程返回的URL，则从URL对应的网页中解析出超链接links，
        c.1 从 unvisited_table 中分给该抓取进程一个新的 URL，并将返回 URL 放到 visited_table 中
        c.2 并对每一个超链接符号串 HOST(link)进行模 n 散列，得到某个整数 i；
        c.3 对每一个超链接 link 及其对应整数 i
            c.3.1 如果本节点编号为 i，执行 b.1 的动作
            c.3.2 否则，将 link 发给节点 i
end
```

图 6-3 协调进程工作算法

从它的工作内容来看，每个节点的 visited_table 和 unvisited_table 是此消彼长的。unvisited_table 就相当于一个“任务池”，在整个收集过程中它先是逐渐变大，然后逐渐变小，理论上讲，是可以最终为空的。

我们还看到，在不同节点上运行的协调程序是需要通信的：发送不该本节点负责的 URL，接收该本节点负责的 URL。而从理论上讲（只要不同的 url 对应不同的网页），这种两级并行的策略能保证网页不被重复抓取。

然而，互联网上复杂的现象使得这个理论的前提不成立。不同的域名可以对应相同的IP地址，例如net.pku.edu.cn和net.cs.pku.edu.cn都对应 162.105.203.25，从而沿着这两个不同域名的访问是等效的（这只需要在相应的DNS服务器上做适当映射就能实现）。这就意味着两个不同的URL可能对应相同的网页，于是该网页就可能被多次抓取。一种考虑是当收到一个URL时，首先得到HOST(URL)的IP地址，然后依据这个IP地址来发出HTTP请求。但我们看到，在某些情况下在HTTP请求中用IP地址（而不是用域名）会得不到所希望的HTTP应答。典型的情况就是大型门户网站为动态分配访问负载所采用的组技术：一个组的代表可以是某个“内部域名”，例如pagegrp1.sohu.com.cn，但它可以包含若干IP地址，这些IP也可能根据应用的需要分成几个子组，例如一个子组对应来自对www.sohu.com.cn的访问，另一个子组对应来自对news.sohu.com.cn的访问。当用域名引起的访问发生时，pagegrp1 服务器动态确定某个IP来提供服务，而不允许直接通过pagegrp1 内的任何IP来进行访问。于是，就出现了这样矛盾的要求：为了不重复抓取同一个网页，

需要在HTTP请求中用IP地址；但为了能够正常访问采用组技术的大型网站，在HTTP请求中不能用IP地址。我们目前的做法是照顾后者，即容忍了有些网页的重复抓取（然后通过一个“消重”过程去除冗余）[1]。

这样，我们就得到如图 6-4 所示的一个系统运行示意图。其中的协调进程之间两两建立起连接，形成一个逻辑全互连关系，直接传递它们之间的交叉 URL。

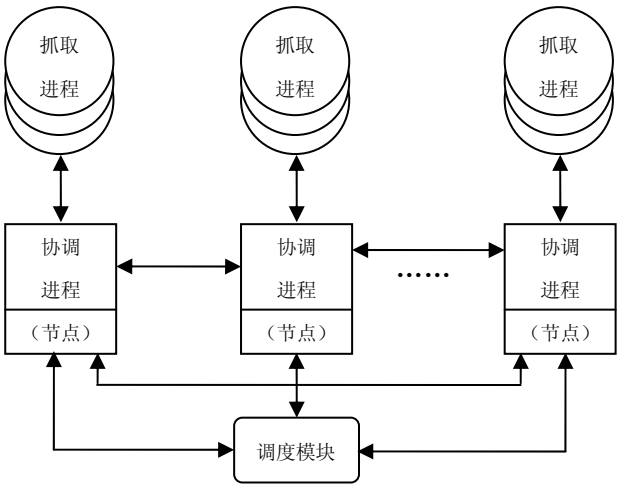


图 6-4 分布式 Web 搜集系统结构

在图 6-4 中调度模块（记为 WSR）有特别的意义，它维护系统内所有登记协调进程的信息，包括它们的 IP 地址和端口号。当任何一个协调进程的信息有所改变时，WSR 负责将更新的信息转送给其他协调进程，便于建立连接和变更连接。协调进程从 0 开始编号，直到 *n*-1，各自负责收集存储属于自己范围内的网页。每个节点上运行若干抓取进程，它们在协调进程的管理下工作。抓取进程负责接收从所属协调进程发送的 URL，抓取该 URL 指向的网页并传送给所属协调进程。各协调进程之间都建立有双向连接，可以全双工的工作。当任一协调进程发现自己的收集模块发回的网页中包含不属于自己的 URL 时，就将此 URL 传送给负责它的协调进程去处理。为减少通信量，各协调进程之间只传送 URL。

二、 关于性能的讨论

从实际应用的不同需求出发，海量网页搜集系统的性能可以有不同的定义，涉及四个主要参数，完成一批网页搜集所花的时间（*T*，小时），收到的网页数量

[1] 还有比较复杂的情况是“不同域名，相同IP，不同网页”，例如net.pku.edu.cn和i3s.pku.edu.cn。

（P），系统和Internet之间的带宽（B，Mbps），参与搜集的机器节点数（n）。如果按照平均每篇网页 15KB数据量计算，注意到上述参数的单位[2]，我们有最基本的

$$T \geq \frac{8\times 15\times 10^3 P}{B\times 3600}\times 10^{-6} = \frac{P}{3B}\times 10^{-4}$$

例如，P=10^8，B=100，T≥33.3 小时。当然，这是最理想的情况了。通常，如果B表示网络连接的额定带宽，$\hat{B}$ 表示实际达到的有效带宽，B>>$\hat{B}$。虽然有效带宽是随时间变化的，但在固定的环境下，同样一段长时间（例如一天）的平均有效带宽基本上是稳定的，因此讨论$\hat{B}$有意义。下面是关于性能的几种考虑：

1) 在给定硬件条件（节点数，网络有效带宽）下，给定时间内搜集不同网页的数量。上面提到过，在一段时间的平均有效带宽是有意义的。一次搜集过程通常要历经几天甚至几周，因此用有效带宽来比较系统设计是合理的。
2) 给定硬件条件（节点数，网络有效带宽），达到某给定网页搜集量所花的时间（越少越好）。网页搜集过程告诉我们，刚开始会比较快，unvisited_list 迅速增长，每一个抓回来的网页都带回来一些新的 URL。但随着过程的进行，进展会越来越慢，不仅新发现的 URL 少了，更重要的是新发现的 HOST(URL)少了，这导致搜集网页逐渐集中到几个大网站上，网站本身的吞吐能力限制了搜集速度，甚至引起比较多的 HTTP 应答失败。同时，由于网站的数量变少，负载平衡可能开始出现问题（HOST(URLs) >> n 不再成立）。这表现为随着搜集过程的进行，每天收到的网页越来越少。在实际中，人们可能先根据经验确立一个目标网页数量，达到后就停止。例如我们在 2003 年初估计中国的网页数量在 1 亿以上，但超过 2 亿的可能性不大，再考虑到以先宽方式搜集时得到的网页的重要性随时间快速递减[Najork and Wiener,2001]，于是搜集 1 亿左右就停止会是一种合理的考虑。
3) 在给定时间（T）内，完成搜集给定目标网页数量（P）所需的节点个数（n）。有许多因素使得给定 P，n 并不和 T 成线性反比。较小的 n 意味着许多好处：较小的节点之间通信开销，较少的外部通信资源（出口带宽）冲突，较好的负载平衡。因此，如果我们有一个搜集时间长度（例

[2] 一个字节是 8 位，一小时等于 3600 秒，带宽用Mbps为单位，等等。

如两周），确定一个目标网页数量（例如 1 亿），也许用 $n = 20$ 会提前 3、5 天，但用 $n = 5$ 可能也能满足要求。

无论哪一种标准，性能的瓶颈可能在不同的条件下表现在系统的不同部分。如前所述，抓取一个网页的任务包含有许多阶段，其中涉及系统的不同部件：处理器，磁盘，网络带宽。例如当节点数比较少时，处理器和磁盘会是个瓶颈，因为每个节点需要启动较多的抓取进程，否则达不到所需的抓取能力。当节点较多时（例如大于 10），网络带宽就成为主要矛盾。Internet Archive的Brewster告诉我们，他用 6 台机器，T1 连接，每次抓两个月，2×10^9左右网页，30TB数据量。30TB/60 = 0.5TB/天，需要 46Mps以上的平均有效带宽。[3]

三、 性能测试和评价

上述系统设计首先通过一个类似于“trace driven simulation”的方法进行了模拟。具体做法是在一个单节点网页搜集系统的正常运行过程中加入数据采集程序，产生并行算法需要使用的模拟数据。对于每个网页，采集了它的 URL 和所包含的 URL 链接，总共得到了 761,129 篇网页的信息，数据量大小为 507MB。以此作为我们并行算法模拟的输入，分别考察了节点数 n 为 2，4，8，16 四种情况。为对比起见，在每次运行多节点模拟时也同时运行单节点的模拟。下面是从几个方面对实验结果的评价。

1. 负载平衡分析

如前所述，我们为系统负载平衡采用的基本技术是利用散列函数在节点之间动态分配 HOST(URL)。由于这里要考虑的是一个过程，不能仅仅用过程结束后每个节点总共分得了多少 url 来评价是否平衡。因此，我们通过分析每个节点每小时搜集网页数来评估负载平衡的效果。具体来说，模拟程序以小时为数据采集的单位，考察了系统前 10 小时运行结果。

设系统中包含两个节点，在一个小时内节点一搜集的网页数为 2（即 n=2），节点二搜集的网页数为 3（由于我们只关心两行数据的差别，这里用一些小整数是无关紧要的），依此比例进行下去，取前十小时作为一组参照序列。规整化后如表 6-3 所示，其中 t 表示时间（小时），i 表示节点编号，中间的数据表示“节点 i 在 t 小时里得到的网页总数”。

如果四组实验数据好于参照序列，认为达到负载平衡要求。规整化四组实验

[3] 在本书即将脱稿之际，谢正茂为天网设计的一个新型搜集系统在千兆带宽上也达到了每天搜集 5000 万网页（超过 0.5TB数据量）的水平。

数据和参照序列数据，计算出的数据表示成图 6-5。在图 6-5 中三角形线表示参考数据方差，方形线表示节点数为 2 时的方差，钻石形线表示节点数为 4 时的方差，加号形线表示节点数为 8 时的方差，星号形线表示节点数为 16 时的方差。从图中可以看出任何一组实验数据的方差都小于参考方差。而且随着节点数的增加，平衡性越来越好。在节点数为 16 时，几乎和 x 轴重合。说明各个节点搜集的网页数基本相等，因此并行搜集子系统负载平衡达到预期目标。

表 6-3 参照序列，假设节点数为 2

i \ t	1	2	3	4	5	6	7	8	9	10
1	2	4	6	8	10	12	14	16	18	20
2	3	6	9	12	15	18	21	24	27	30

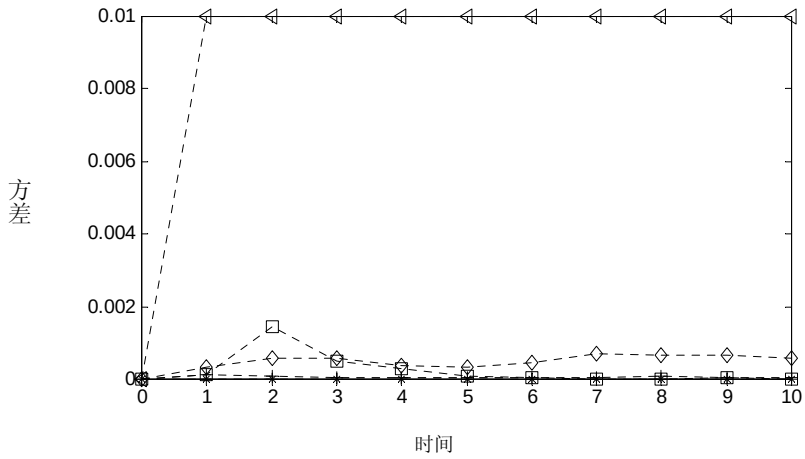


图 6-5 负载方差

2. 通信开销

实验中为保证环境一致性而采用集中解析域名方式，因此各个节点之间只有交叉 url 的通信量。各节点之间传递的只是 url，每个 url 长度设定最大为 128 字节。在今后的实际系统中，为提高域名解析的利用率，各节点已经解析出的主机名要互相传送,每个节点都维护一份当前系统已解析出的主机名与 IP 地址的对应表。各个节点中的表要保持一致，这也构成主机通信的一部分。每个主机名与 IP 对应关系的存储长度不会超过 72 个字节（用 64 个字节存储主机名，4 个字节存储 IP，4 个字节存储访问时间），因此节点之间通信量不大。另外，为了实现系统

的动态调度，需要在增减节点时，将现存节点中维护的相应表进行修改，为了保证表的一致性，需要在各个节点之间互相传递信息。这种情况并不会经常出现，因此不会过多的增加节点通信量。考虑在上述后两种情况下的节点通信中，一个副本要传送给多个节点，在实际系统的运行中我们采用组播技术。

3. 可扩展性

图 6-6 是模拟系统设计可扩展性的具体结果，我们考察了系统节点数 $n = 1$，2，4，8，16 等情形。其中，横轴表示目标搜集系统的运行时间，单位为小时；纵轴为累积搜集的网页数。图中分别显示了在 5 种不同的系统规模下前 10 小时的运行结果。

模拟过程是在一台机器上，通过多个进程的协调进行的，体现了本章第二节描述的方案和算法，以及如图 6-4 所示的系统模型。

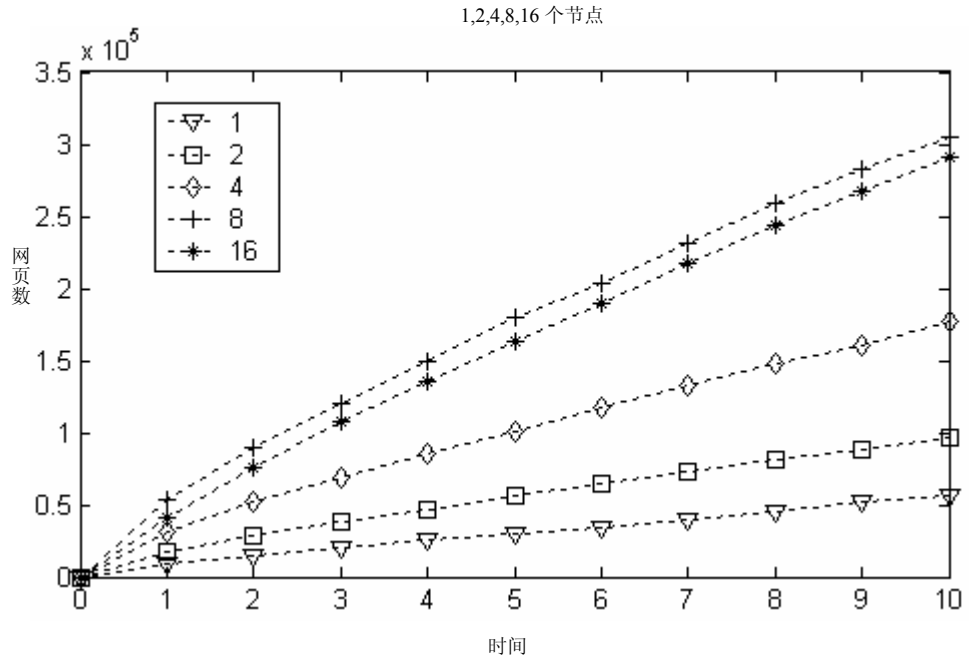


图 6-6 n 个节点并行搜集系统及集中式系统性能随时间的变化

将图 6-6 的内容综合起来，我们得到图 6-7。X 轴代表节点数，Y 轴表示 n 个节点协同工作搜集的网页数与单个节点在同样时间段里搜集网页数的比（称为加速比）。由图中可以看出加速比随着节点数的增加基本上是线性增加，因此并行系统具有良好的可扩展性。关于这里介绍的并行搜集子系统更多的细节见参考文献

[Yan, et al.,2001a]。

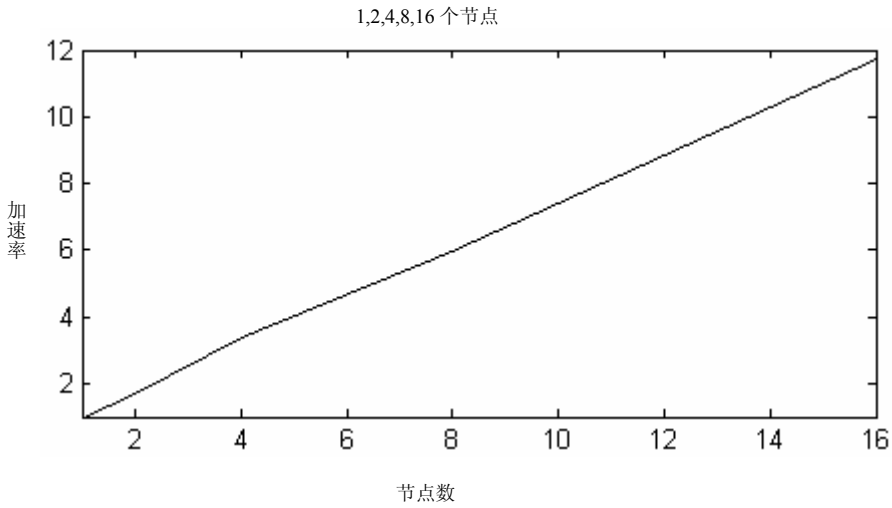


图 6-7 分布式系统效率

四、 系统的动态可配置性设计

并行系统中 WSR 模块的存在，使每个节点都能够保持当前系统中所有节点的最新信息，是系统动态可配置性的前提。在保证系统负载平衡的条件下，我们考虑三种方法保证系统具有动态调度性：

1) 采用散列函数动态调度 url。
2) 第二个方案是结合第一种方法，同时每个节点记录着一张 WWW 主机表，这张表在各个节点是相同的，其中每一条记录包含一个 WWW 主机及其所对应的一个节点。
3) 采用逻辑上二级映射的方法。首先用散列函数映射 URL 到一张逻辑表上，然后将这张表上的相应部分映射到各个节点。

对以上三种方法通过对节点数增减 1 的情况分析，可以知道动态调度性的好坏，设WWW主机数为m，系统初始节点数为n，n_i，n_j表示系统中任意两个节点，加 1 情况记为n->n+1，减 1 情况记为n->n-1。

第一种方法，系统初始，每个节点负责的主机数为m/n，散列函数对n取模，可以保证系统负载平衡。在n->n+1 情况下，散列函数对n+1 取模，此时可以保证系统负载平衡，但是由于模数改变使原先分配给节点n_i负责的URL可能分配给n_j负

责，从而导致重复搜集一些已经搜集过的网页；在*n*->*n*-1 情况下具有同样的缺陷。

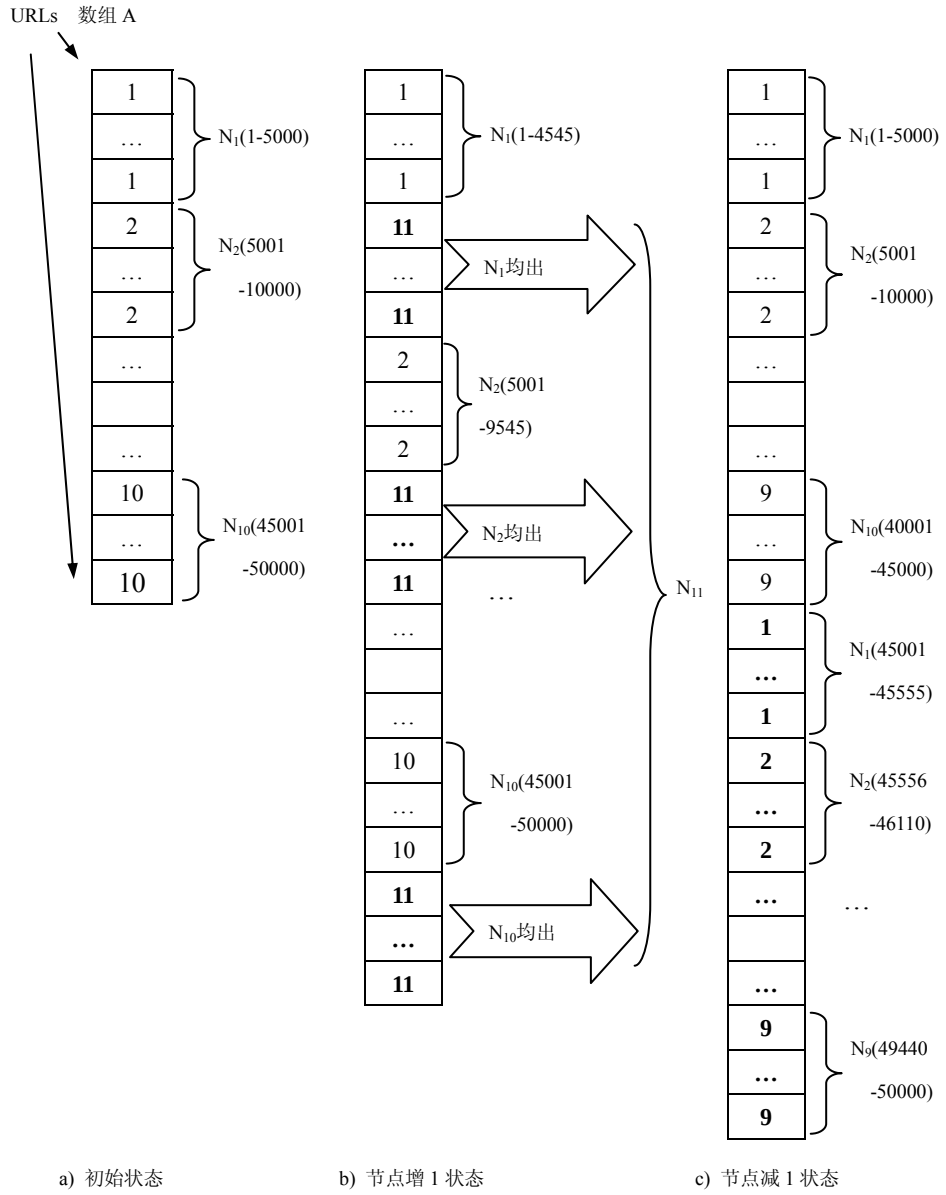


图 6-8 URL 两阶段映射

第二种方法，为了克服第一种方法存在的缺陷，每个节点需要附加存储两个表：主机表和本节点已经访问过的 url 表。每个节点保存的主机表相同，已经访

问过的 url 表各自不同。WWW 主机数目有限，后加进的节点没有足够的 WWW 主机去进行搜集，为达到负载平衡的要求，需要其它节点迁移一部分主机及其所附带的已访问过 url 的信息给新加入的节点。此时 url 经过散列函数处理后，需要额外采取一个步骤：在一个节点判断得到该 url 属于自己负责后，应先根据主机表判断此 url 对应主机是否已经被其他主机负责，如果没有，才应该自己处理；同理，在一个节点判断得到一个 url 属于另一个节点（不妨设为 A）负责后，应先根据主机表判断此 url 对应主机是否已经被其他主机负责，如果没有，才将该 url 传给 A。这种方法由于需要维护各节点之间主机表一致，节点数目变化时需要传送主机和附属 url 信息，较第一种方法增加了主机之间的通信量，我们决定将每个 url 采用 MD5 算法，用 16 个字节表示每个 url。

第三种方法，采用逻辑二级映射的方法。其中一级逻辑节点可以用数组形式存储(不妨设这个数组为 A)，每一个数组元素的下标即是它所代表的逻辑节点号，其中存储该逻辑节点对应的节点序号。举例说明，设 n=10 为 10 个物理节点，m=50,000 为逻辑节点的数目，系统初始如图 6-8 a)所示：其中 A[1]到 A[50,000]称为逻辑节点。URL 经过散列函数处理后首先平均映射到逻辑节点上，再经过第二次映射将第 1 到第 5,000 的逻辑节点映射到 1 节点，第 5,001 到第 10,000 的逻辑节点映射到 2 节点，以此类推将逻辑节点平均分配到 10 个节点上。

当节点数增 1，每个节点都要让出一部分控制的逻辑节点给新的节点，即对A做部分修改。如图 6-8 b)所示，n_{11}是由n_1，n_2，…，n_{10}匀出部分组成，其逻辑数组中对应的项被置为 11。当节点数减 1，如图 6-8 c)所示，减掉的节点要让出自己控制的主机，平均分配给其他节点，同样要修改逻辑数组中的值。

同第二种方法比较，这种方法多一次映射，同样每个节点需要保存自己访问过的 url，随主机迁移时需要转给新的节点。但是这种方法可以不必存储主机表，每个节点中只要保存一份同样的数组 A 就可以了；这样就减少了节点之间的通信量，并且系统配置改变后，不必像第二种方法一样在散列函数处理后作附加的判断步骤。

现在，由于第一种方法实现简单，在模拟系统中我们采用这种方法。第三种方法较前两种方法优越，是我们系统实现的选择，以保证系统具有良好的可动态配置性。关于这部分介绍的系统动态可配置性的更多细节见参考文献[Yan, et al.,2001b]。

第三节 本章小结

本章第一节以天网这样一个实际系统由集中式到分布式的过程为例，说明了基于 Web 信息急速膨胀的需要而产生的对搜索引擎技术发展的要求。

提出并设计了可扩展 Web 信息搜集系统结构，使之达到能够搜集数量不断增长的网页的要求。

在详细介绍可扩展 Web 搜集系统的主要设计思想时，还对比介绍了集中式搜集系统的设计与实现。它的实际应用就是天网系统。

目前此可扩展搜集系统结构已经实际应用于天网 2.0 系统，自 2001 年 6 月以来，共进行了多次大规模的 Web 信息搜集工作，采用多台搜索机器分布式并行工作。其中 2001 年 11 月 6 日 16 时至 2001 年 11 月 22 日 8 时，我们的实际运行系统用 12 台 PC 机连接在 100Mbps 速度线路上，对全国静态网页（不包括通过提交查询词动态生成的网页）进行了一次搜集。搜集过程从 http://net.cs.pku.edu.cn 站点开始，采用一种类似于先宽搜索的策略，直到没有进一步可以搜索的网页。共搜集到网页 47,707,998 个，涉及到 46,669 个 Web 站点，其中不重复的网页为 22,382,623 个，平均每个站点有 479.63 个不重复的网页。实际运行和我们的模拟评测结果一致，证明我们并行搜集系统设计高效可行，达到了预定目标。同时我们意识到分布式搜集系统的成功也带来了大量的后续工作：如并行分布式消除重复或转载网页，后台的分布式索引和面向客户的分布式检索，系统管理等。

搜索引擎作为网上信息搜集整理的最直接应用，虽然已经有多家实际系统，但是从分布式系统设计来看，Harvest 搜集系统有诸多地方考虑欠周没有最终实现；从早期的文献看，Google 搜集系统在 URL 分配上采用集中式，而不是分布式，从而会形成系统的瓶颈。从研究状况来看，有关天网的可扩展 Web 信息搜集系统结构设计的论文是较早在国际上发表的应用于搜索引擎实际系统中的阐述分布式系统结构的论文之一[Yan, et al.,2001a]，[Najork and Heydon,2001]。

分布式系统的动态可配置能力对于系统的适应性、可靠性都有重要的意义。在第二节中论述的两阶段映射模型提供了一种在系统成员变化时协调分布式搜集系统的方法。通过分析，可以采用优化的方案实现两阶段映射模型。当系统进入协调过程中，只需在各个节点间移动网页 URL 对应的 MD5 数据和未访问 URL 列表。每个主控应该维护自己已经搜集网页 URL 对应的 MD5 值。为使系统快速的从不稳定状态进入稳定状态，多播技术对于主控之间的信息更新很有帮助。

至此我们重点介绍了可扩展 Web 信息搜集系统的设计原理与实现方法，并通过一些实验和分析优化了系统的设计，尤其是针对系统动态可配置性的设计。

第七章 网页净化与消重

网页净化和消重是大规模搜索引擎系统预处理环节的重要组成部分。所谓网页净化（noise reduction）就是识别和清除网页内的噪音内容（如广告、版权信息等），并提取网页的主题以及和主题相关的内容；消重(replicas or near-replicas detection)是指去除所搜集网页集合中主题内容重复的网页。建索引一般是在消重后的网页集上进行的，这样就可以保证用户在查询时不会出现大量内容重复的网页。

本章第一节论述了一种 HTML 网页净化与元数据提取的方法，通过它我们可以从一个网页源文件中自动提取网页的一些主要元素，包括网页标识、网页类型、内容类别、标题、关键词、摘要、正文、相关链接等信息。第二节给出了五种转载网页的消重算法，通过这些方法可以消除绝大多数主题内容重复的转载网页。

第一节 网页净化与元数据提取

一、 引言

今天，当我们浏览 Web，从中获取所需信息的同时，还会常常看见大量和我们所关心内容无关的导航条、广告信息、版权信息以及调查问卷等，我们称之为**“噪音”**内容。有时候，我们可能从这些噪音内容中得到一些意外的惊喜；另一些时候，我们可能不喜欢这些东西消耗人类宝贵的注意力资源。同时，我们观察到噪音内容通常伴随着相关的超链。因此，噪音内容会导致相互链接的网页常常并无内容相关性。这样，网页内容的混乱不仅给 Web 上基于网页内容的研究工作带来困难，也给基于网页超链指向的研究工作带来困难。另外，随着 Web 上各种研究与应用的深入发展，仅仅是原始网页内容已经不能满足需求，还要求能够提供便于计算机处理的元数据信息，例如关键词、摘要、网页内容类别等。然而，现在 Web 上大部分网页仍然是普通 HTML 网页，并不包含必要的元数据。鉴于此，本节讨论一个网页表示模型建立和实现的方法，这一方面使我们能够自动从网页中提取相关的元数据，另一方面也去除了和网页主题内容无关的噪音内容，进而在原始 Web 上搭建一个噪音小、描述清晰、更易于处理和利用的网页信息平台。

在主题搜索领域，大量的广告、导航条等噪音内容会导致主题漂移（topic

drift)。这说明传统的主题搜索算法中以网页为粒度构造的 Web 图不够准确，必须深入到网页内部将处理单元的粒度缩小，才能提高内容分析的准确性。在[Chakrabarti, et al.,2001]中提出了一套解决方法，首先将网页表示为一棵 DOM 树结构并找到与主题一致性较高的子树，然后对这些子树作特别的处理，从而提高主题提炼的效果。

在 Web 信息检索领域，检索结果的相关性和检索的速度是评价一个 Web 检索系统的两个指标。如果不去除原始网页中的噪音内容，检索系统必然对噪音内容也建立索引，从而导致仅仅因为查询词在某张网页的噪音内容中出现，而把该网页作为结果返回，而网页的主题内容可能和这个查询词完全无关。可以看出，噪音内容不仅使索引结构的规模变大，而且还导致了检索准确性的下降。针对这个问题，[Lin and Ho,2002]中提出了一个去除网页中噪音内容的方法，该方法首先依据<table>标签构造网页的标签树，从而依据<table>标签将一张网页规划为相互嵌套的内容块；而后，对于使用同一个模板作出的网页集，找出在该网页集中多次出现的内容，作为冗余内容，而在该网页集中出现较少的内容块就是有效信息块。实验证明该方法是有效的，但该方法必须局限在基于同一个模板的网页集，而 Web 上的网页模板不计其数，该方法显然不够通用。实际上，任意一张网页，人是比较容易区别其中的噪音内容和主题内容的。这说明我们有可能追求自动识别一张网页中的主题内容和噪音内容而不需要依赖于一个网页集合；这样就可以使去除网页噪音内容的方法更加通用和独立。

在网页分类领域，由于噪音内容与主题无关，训练集中的噪音内容会导致各个类别的特征不够明显,而待分类网页中的噪音内容则会导致该网页类别不明确，因而影响了网页自动分类的效果。[Yang,1995]，[Li and Shi,2002]中提出了通过去掉网页中的噪音内容来提高网页分类质量的方法。

在网页信息提取领域，自动识别模式的方法必须要从整个网页中提取模式，而不是只针对主题内容提取。因此，在净化后的网页上作信息提取不仅可以排除噪音信息对信息提取的干扰，提高信息提取的准确性，而且可以使得网页中的结构简单化，提高信息提取的效率。

上述分析我们看到，噪音内容对基于网页的研究工作的影响是普遍而严重的，虽然各个领域采用的方法各不相同，但处理的目的都是为了去除网页中的噪音内容，得到真正的主题内容。

另一方面，随着 Web 上研究与应用的发展，单纯的网页内容已经不能满足需求，网页元数据得到越来越广泛的使用。在 Web 信息检索领域，单纯依赖关键词匹配的检索手段过于单一。内容类别、摘要等元数据信息的合理使用，不仅使用户可以从不同的角度进行查询，而且也使得查询的准确性得到提高。而主题搜索、个性化信息服务以及数字图书馆也都强烈的依赖资源的元数据信息。因此，准确且高效的提取必要的元数据是 Web 上各个研究领域面临的重要问题。

在元数据和主题内容的提取方法上，可以从信息提取领域的研究成果（特别是从 HTML 网页中提取语义信息）中得到很多启发。针对从 HTML 网页中提取语义信息，早期的方法是：针对某一类具体网页，人工提取该类网页的内容组织模式。然后，信息提取系统根据该模式从属于该类的网页中提取相应的内容[Hammer, et al.,1997]，[Ashish and Knoblock,1997]。对元数据和主题内容的提取可以采用同样的办法，但这些方法有一个共同的局限性，那就是需要人工提取内容组织模式，这对于内容组织风格繁多的 Web 来说显然是不适用的。因此，在[Wemble, et al.,1999]中提出了 5 条启发式规则，综合利用这 5 条规则系统可以自动地发现网页中各个主题信息块（chunk）的边界。[Yang and Zhang,2001]提出了一种基于视觉相似性来自动分析网页语义结构的方法，该方法首先比较 HTML 网页内容的视觉相似性，然后使用一个模式发现算法来确定这些视觉相似的内容最有可能的组织模式，最后按照该模式将内容重新组合。

通过对上述研究成果的分析，我们发现不同领域的工作存在两个共性：

1) 工作结果的共性。虽然各个领域所做的工作都是为了解决网页复杂化给本领域带来的问题，但各个领域的工作结果中有着共同的部分。譬如，各个领域都需要去除原始网页中的噪音内容，然后在净化后的网页上进行后续工作；很多领域都需要获取网页的元数据信息。即净化的网页和元数据是它们都需要的结果。
2) 工作过程的共性。在获得不同结果的过程中存在着共同的中间环节。譬如：网页分类、摘要的提取以及关键词的选取都需要对文档进行分词操作。而这些中间环节有时是整个工作中效率上的瓶颈。

这些共性启示我们有可能通过归纳不同应用需求中的通用元素，并作为一个模型一次性提取出来，从而对多种应用提供一个统一的支持。可以想象，这样做既便于提高所需信息的质量，又最大限度地避免重复工作带来的时间开销，从而在信息量和复杂性这两个相互制约的因素之间找到一个合理的折中点。

基于这样的思想，本节在参考了 Dublin Core[Dublincore,2003]和 EDA（Encoded Archival Description）[EAD,2002]后，提出了一个包含元数据和内容数据的网页表示模型（称为 HTML_DocView）。该模型包含这样几项信息元素：网页标识、网页类型、内容类别、标题、关键词、摘要、正文、相关链接。参考上述文献中提出的启发式规则,并结合我们自己对 HTML 网页的统计和观察，本节提出了一套更丰富的启发式规则。在这套启发式规则的基础上，借助传统信息检索领域的方法，结合 HTML 网页的特点，提出了从网页源文件生成其 HTML_DocView 模型的方法和算法。该方法与前述相关工作相比更为通用，不需要依赖网页模板以及同类的网页集。

本节的方法已被应用到几种具体的实验中；其中，在网页分类实验中，使用 DocView 模型对网页预处理后，分类效果得到普遍的提高。将 DocView 模型应用

于搜索引擎的消重过程中，其效果也是明显的[张志刚,2004]。

二、 DocView 模型

本节中提出的**DocView 模型**包括：网页标识、网页类型、内容类别、标题、关键词、摘要、正文、相关链接等要素。其中正文和相关链接要素属于网页的内容数据，而其他 6 项则属于网页的元数据。下面将对模型中的各个要素作详细描述。

网页标识是对 Web 上网页的唯一性标识，在 DocView 模型中使用网页的 URL 作为网页标识。

网页类型是根据网页内容的表现形式进行划分的，在本节中将网页分为三类：有主题网页（topic）、Hub 网页（hub）、图片网页（pic）。其中，有主题网页是指网页中通过文字描述了一件或多件事物，是有一定主题的；如一张具体的新闻网页就是典型的有主题网页。Hub 网页是指专门用来提供网页导向的网页，因而是超链聚集的网页；如门户网站的首页就是典型的 Hub 网页。图片网页是指网页的内容是通过图片的形式体现的，其中文字很少，仅仅是对图片的一个说明；如某个机构包含图片的人员介绍网页就是典型的图片网页。

将网页分为上述三个类型是因为三类网页在用途和处理方法上存在较大的差别。其中 Hub 网页与其它两类网页的区别在于网页在 Web 上发挥的作用不同，Hub 网页通常不会具体的讲述一件事物，而是提供关于相关信息的链接集。而图片网页与其它两类网页的区别在于处理的方法不同，由于图片网页的内容是通过图片表达的而不是通过文字，因而，传统信息处理领域的方法对图片网页是不够有效的。三类网页间的区别导致很多应用领域都会对它们作适当的区别。

内容类别是从语义上对网页的内容进行分类，它是计算机获取网页语义信息的一个直接手段，在 Web 上的研究领域中有着广泛的使用。它是通过特定的分类器对网页内容分类得到的，依赖于一定的分类体系，本书第十一章将有详细的介绍。

标题、关键词和摘要是概括描述 Web 文档内容的重要的元数据，对于 Web 信息检索等领域的工作有非常重要的作用。

正文是原始网页中真正描述主题的部分，因此，在某些具体应用中用正文代替原始网页更为合理。

相关链接是指在本网页中指向与正文内容相关的网页的链接，而非广告等噪音链接。将正文和相关超链重新组合就得到了净化后的网页。

图 7-1 用 DocView 模型提取的网页要素，图 7-2 是将提取的网页正文和相关超链重新组合成的净化网页。

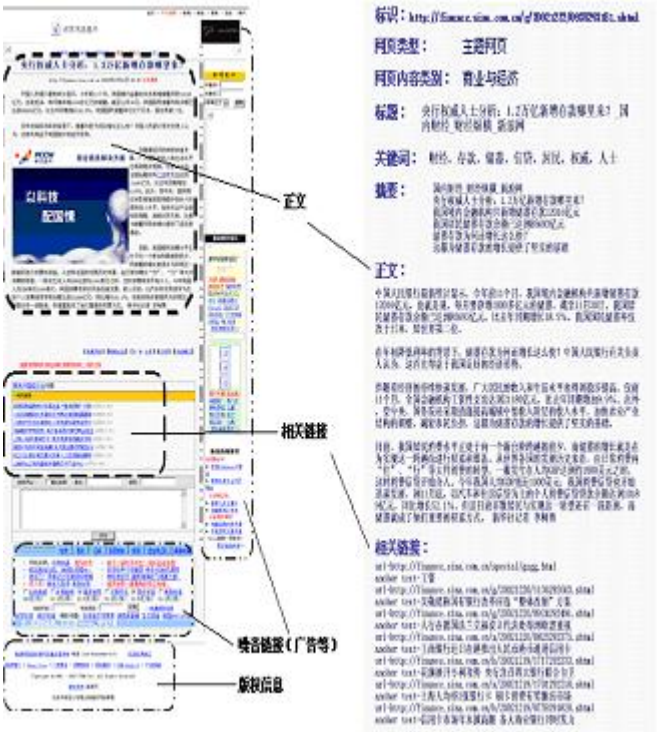


图 7-1 用 DocView 模型提取的网页要素

图 7-2 净化后的网页

三、 网页的表示

网页的表示是网页内容分析的基础，在网页内容分析过程中通常需要两个层次的表示，**抽象表示**和**量化表示**。**抽象表示**是以网页制作规范（如 HTML 规范）为依据和出发点，构造出能体现网页内容结构和内容重要性等信息的表示模型，其目的是充分利用网页制作规范，挖掘出网页中隐含的信息，为后续量化表示提供更多可利用信息。对于 HTML 网页，最常用的方法是构造网页的标签树。**量化表示**则是从计算机处理的角度出发，利用信息检索领域的技术和从网页中挖掘的隐含信息，生成计算机可以直接用于计算的表示模型，如向量空间模型等。下面对这两个层次的表示方法做详细描述。

1. 抽象表示

HTML 通过定义一套标签来刻画网页显示时的页面。因此，对于 HTML 网页最常用的抽象表示方法是构造网页的标签树。

依据标签的作用可以将 HTML 的标签分为三类：

规划网页布局的标签：在视觉上，网页是由若干提供内容信息的区域（我们

称之为**内容块**）组成的，而内容块是由特定的标签规划出的（称之为**容器标签**），而且容器标签是允许嵌套的。常用的容器标签有<table>、<tr>、<td>、<p>、 等。因此，依据容器标签可以将网页表示成树状结构，虽然该树状结构描述的是网页内容的布局结构，但布局信息中隐含着网页内部各部分内容的相关性信息。

描述显示特点的标签：在 HTML 标准中定义了一套标签来规范其包含的内容的显示方式（比如：字体变大、粗体、斜体），我们称之为**重要信息标签**。常用的重要信息标签有<b>、<I>、、<h1>、<h2>等十几种。这类标签中的内容通常是网页作者希望引起读者注意的，因此隐含着一定的内容重要性信息。

超链相关的标签：超链是 HTML 网页区别于传统文本的最明显的特点之一，表示着网页间的关系，因此整理出超链标签并作合理的分析可以挖掘出网页间的内容相关性信息。

目前，有很多构造标签树的工具（如：W3C HTML lexical analyzer[W3C,1997]和 HTML Tidy[HTMLTidy,2004]），它们各有特点，W3C HTML lexical analyzer 有很强的通用性，适合各种标识语言；HTML Tidy 则能够自动发现并修正标签的错误。由于内容分析需要在网页内部计算各个部分之间的相关性以及确定各部分内容的重要性，因此，用传统的顺序整理各种标签的方法构造出的标签树在用于内容分析时并不方便。适合内容分析的标签树强调**内容块**的概念，倾向于以内容块为单位的内容组织方式。另外，内容分析过程中经常会关心这样一些信息：标签树的规模（结点个数）、每个内容块包含的各种类型信息（如：文本、超链或图片）及其数量等。鉴于此，我们自行开发了更适合内容分析的标签树构造工具。

下面简要的描述标签树。给定一篇 HTML 网页，顺序整理出容器标签就得到了对应的标签树的框架。而后，整理每个内容块（对应标签树的一个结点）中的超链标签、图片标签和重要信息标签，并在标签树中对应的结点中记录下来。这样就构造了一棵基本的标签树。对上述基本标签树信息作适当的分析、整理就可以得到内容分析过程中需要的一些描述信息。譬如，依据内容块中词项数与图片数和超链数的比值可以为每个内容块设定一个类型，分为 topic、hub、pic 三种。如果内容块中词项数与图片数的比值小于某个阈值，该内容块就是 pic 类型，如果内容块中作为 anchor text 出现的词项数与该块中总词项数的比值小于某个阈值，该内容块就是 hub 类型，否则为 topic 类型。这样，标签树中每个结点都有类型和属性集两组描述性信息，以及超链集和重要标签集等数据信息。图 7-3 是一个标签树的图例，其中 link_list 表示该内容块中超链集合；weighty_tag_list 表示该内容块中重要标签集合。

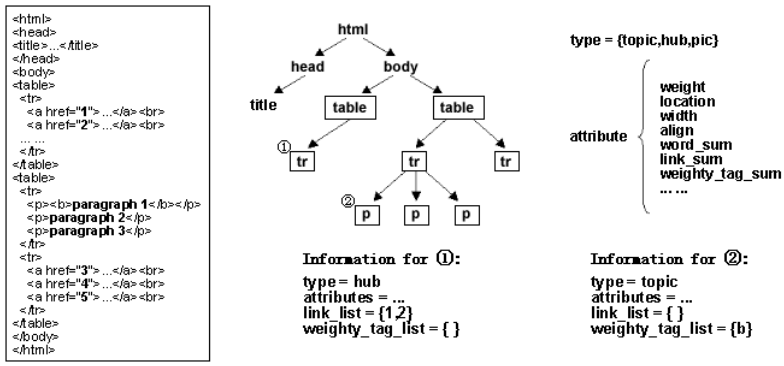


图 7-3 HTML Tree 结构

2. 量化表示

1) 合理利用网页的特点

在传统的文本处理领域中，一个文本被看作是一个特征项向量 $(w_1, w_2, ..., w_n)$，其中 w_i 是第 i 个特征项的权值，n 是特征项的总数。这样，每个文本就被映射到了向量空间中的一个点，因而向量空间中的点的距离就可以用来衡量其对应的文本的相似性。在量化方法上，对权值的计算，比较常用的是采用 TF*IDF 方法。

在量化方法上，我们可以充分的利用 HTML 网页中的重要信息标签信息以及 HTML 网页内容的布局结构。

为了体现重要信息标签中内容的重要性，通常的做法是对重要信息标签中的内容加权值。但重要信息标签中包含的并非都是重要内容，其中的噪音信息非常多，例如：“Tel”、“Fax”、“联系电话”、“传真”、“广告服务”、“前一页”等等。我们对此做了这样的统计，从 Web 上随机抓取的 20000 个网页中，包含在我们定义的重要信息标签中的内容有 9091 条，其中上述的噪音内容（共定义了 22 个）出现了 1200 条，也就是说，重要信息标签中的噪音信息至少占 13.2%。因此，简单的对重要信息标签中的内容加权是不合理的，整理噪音词集合并对重要信息标签中的内容进行过滤，对过滤后的真实重要内容加权值可以避免噪音扩大化。

由于网页中的标签结构是对页面布局的描述，我们不难得到这样的结论：如果某个内容块中存在真实重要信息，那么这个内容块的重要性也相对较高；如果一个内容块的重要性较高，那么这个内容块的外层嵌套块的重要性也相对较高。可以看出，导致网页中内容块重要性增加的是包含真实重要内容的重要信息标签。基于这个结论，我们给网页中每个内容块赋予一个权值，用来表示这个内容块的

重要性，并提出内容块权值的传递规则（我们称其为**权值传递规则**）。由于内容块与标签树中结点是一一对应的关系，以下对权值传递规则的描述统一使用标签树的结点而不使用内容块。

权值传递规则：

- 标签树中每个结点的初始权值为 1。
- 每个重要信息标签都有一个影响因子。如果标签树某个叶子结点中存在重要信息标签并且重要标签中的内容是真实重要内容，那么累加重要信息标签的影响因子，得到的和就是该叶子结点的影响因子。没有出现重要标签的叶子结点的影响因子为 1。
- 对于每一个叶子结点，如果影响因子为 λ 且 $\lambda > 1$，则该叶子结点的权值变为当前值的 λ 倍，它的父结点以及父结点下的其他子树中的结点均变为当前值的 $\sqrt{\lambda}$ 倍，然后以该父结点为变化源，按照上述规则再向外扩展一次。每一次扩展过程中，遇到父结点为<body>或父结点权值超过预定上限就结束整个权值传递过程。过程如图 7-4 所示。

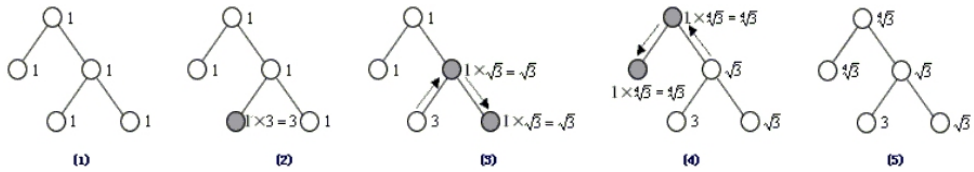


图 7-4 内容块权值传递过程

不难证明，“权值传递规则”有以下两个性质：

性质 7-1 对于初始的标签树，无论从哪个结点开始、以什么顺序执行“权值传递规则”，标签树最终的权值结果都是相同的。

性质 7-2 如果初始标签树中叶子结点影响因子的分布不同，那么标签树最终的权值结果一定是不同的。

其中，性质 7-1 是保证规则正确的基本条件，性质 7-2 则说明，“权值传递规则”可以保证：初始标签树中叶子结点影响因子的分布与最终标签树中权值结果是一一对应的。另外，可以证明，“权值传递规则”的两个性质与权值向上传递的层数是无关的。

2) 适合内容分析的 HTML 网页量化表示

内容分析过程中的处理对象是网页中的内容块，对于内容块的表示，特征向量方法同样是适用的。但在具体的量化方法上有所不同。一个最重要的区别在于，

内容分析过程中侧重的是一张网页内部各个内容块之间的相似度比较，而不是网页间的相似度比较。因此，在特征项权值的计算方法上，我们更侧重特征项在一张网页内部的重要性，而不是特征项在一个文档集合上基于统计的重要性。基于上述分析，我们使用公式（7-1）来计算特征项权值。

$$w_i = \frac{\sum_{j=1}^{BN} BWeight_j \times BTf_{ij}}{\sqrt{\sum_{i=1}^{n}\left(\sum_{j=1}^{BN} BWeight_j \times BTf_{ij}\right)^2}} \qquad (7\text{-}1)$$

其中，$Bweight_j$表示内容块j的权值，它的值由一个内容块中的重要标签来决定的；BN表示网页中内容块的总数；n 表示网页中不同关键词的总数；BTf_{ij}表示关键词i出现在内容块j中的词频。

3) 量化表示方法分析与改进

分析上述的特征项权值公式可以知道，如果网页中没有出现重要标签信息，则所有内容块的权值均为 1，那么公式（7-1）就变为完全依赖于词频的计算方法。统计发现，只有 19%的网页中有重要标签，这就是说，公式（7-1）对大多数网页而言，是简单使用词频来衡量特征项的重要性；但是，文档中很多高频词并不是真正重要的。基于上述分析可以知道，必须要对高频词做特殊的处理。

所谓“高频无关词”，是指虽然在文档中词频很高，但却没有主题描述能力和区别能力，例如：“中国”、“可以”。在基于词频的权值计算方法中，该类词的权值将会很大；另外，“高频无关词”很容易出现在重要标签中，因而对重要标签中的信息加权也使得这种词的权值很大。因此，在内容分析之前去掉“高频无关词”，既可以提高网页内容表示的准确性，又能减少网页向量中的维数，提高效率。

“高频无关词”最明显的特征是：在大量的文档中都以高频词的角色出现。基于这个特征，我们可以通过词频和文档频率确定某个文档集合的“高频无关词”集。如果使用网页集合并且该集合的规模足够大，那么就可以得到近似 Web 上的“高频无关词”集。实验数据表明，“高频无关词”与非“高频无关词”在作为高频词出现的文档频率上有很大的差别。因此，我们可以依据该跳变的位置确定高频无关词集。

四、 提取 DocView 模型要素的方法

对 Web 上的网页，我们根据其网页类型可以将它们分为三类：有主题网页、Hub 网页和图片网页。针对三类网页的信息提取算法各不相同，因此在对网页进行深入分析之前首先要判断网页的类型。为此，我们首先描述这三类网页的特征及判断方法，然后将对面向有主题网页的模型提取算法进行详细的讨论，最后简

要的介绍面向 Hub 网页和图片网页的算法。

1. 网页类型判断方法

在视觉上，大多数网页是容易区分类型的，因为三种类型的网页有着较为明显的视觉特征。在**有主题网页**中通过成段的文字描述了一件或多件事物，虽然也会有图片和超链，但这些图片和超链并不是网页的主体。**图片网页**中内容是通过图片体现的，而文字仅仅是对图片的一个说明，因而文字不多。**Hub 网页**通常不会描述一件事物，而是提供指向相关网页的超链，因此，Hub 网页中超链密集。

虽然视觉上判断网页的类型是比较容易的，但让计算机自动做到这点却不容易。下述的量化方法可以在绝大多数的情况下准确的识别网页的类型。网页都是有一定布局的，比如分左右两边或是中间和边缘。网页作者通常将重要的内容放在网页的中间部分，而边缘部分内容的重要性相对较低，这也是符合人的浏览习惯的。因此，依据网页中间区域的内容判断网页的类型是相对合理的，而网页中内容的位置信息在本节中构造的标签树中是通过内容块的属性记录下来的。本节前面提到，在构造标签树时，依据内容块中词项数与图片数的比值以及内容块中词项数与 anchor text 中词项数的比值将网页中的内容块分为 topic、hub 和 pic 三个类型，基于内容块的类型，我们可以使用网页中间区域 hub 内容块包含的词项数与网页中间区域词项数的比值来判断网页是否为 hub 类型。同理，使用网页中间区域 pic 内容块包含的词项数与网页中间区域词项数的比值可以判断网页是否为 pic 类型。实际效果表明，该方法判断网页的类型是较为准确的。

2. 有主题网页的信息提取算法

该算法以一组启发式规则为指导，首先提取出网页的正文信息，然后以正文信息为基础，提取 DocView 模型中其它的要素。过程如图 7-5 所示。下面按照各个要素的生成过程分别描述。

正文：

一篇有主题网页中的正文通常是用成段的文字来描述，中间通常不会加入大量的超链，而非正文信息通常是伴随着超链出现的。基于此，我们提出了正文选取的规则（称为**正文规则**）。

正文规则：有主题网页中，如果一个内容块是 topic 类型的，则该内容块中的内容为正文的一部分。

依据正文规则，深度优先遍历标签树并依次记录 topic 类型的内容块，就得到该网页的正文，也就是该网页的主题内容。

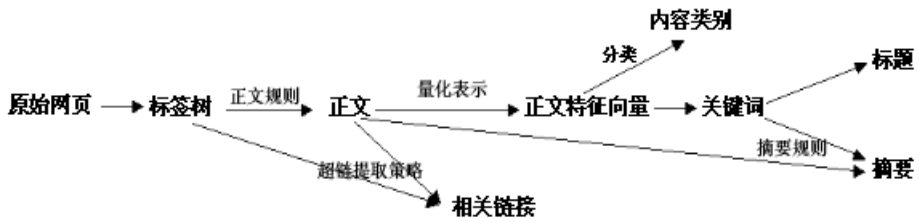


图 7-5 有主题网页 DocView 模型生成过程

关键词：

关键词选取的依据是特征项的权值，因而特征项权值的合理计算是正确提取关键词的保证。以标签树为基础，结合 HTML 网页的特点以及提出的量化方法，可以按照下述过程得到网页主题内容的特征向量。

```
1：for 标签树中的每个正文块CBi do
2：    if 该块中存在重要信息标签信息 then
3：        检查重要信息标签中的内容是否在噪音词集合中出现
4：        if 不在噪音词集合 then          // 为真实重要信息
5：            将重要信息标签的影响因子累加到该内容块的影响因子上
6：        end if
7：        if 该内容块的影响因子大于 1 then
8：            提出的权值传递策略在标签树中传递权值
9：        end if
10：   end if
11：end for
12：计算各个特征项的权值
```

图 7-6 计算网页特征项权值的算法

使用图 7-6 所示算法得到特征项向量后，我们可以用两种策略决定选取关键词的数量。

1) 绝对数量策略。首先定义好 DocView 模型中关键词的个数 α，严格选取权值最大的 α 个特征项作为该网页的关键词。
2) 相对数量策略。该策略中不需要规定要选取的关键词的个数，而是依据特征项权值的绝对大小。该策略首先定义了一个阈值 β，而后计算所有

特征项权值的算术平均值 avg，选取特征项中权值大于 avg * β 的作为该网页的关键词。这种策略虽然会导致每个网页中被选取关键词的数量不均，但它却可以更准确地提取关键词。

内容类别:

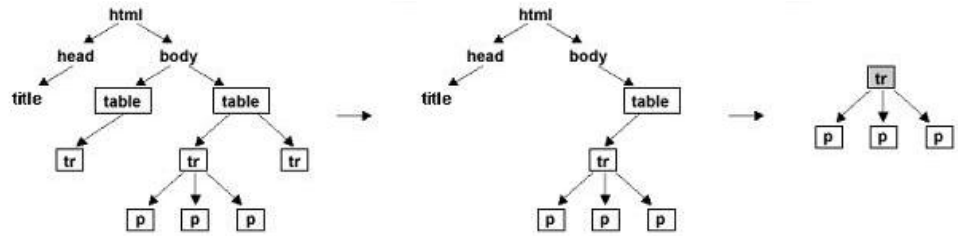
内容类别是通过对正文分类得到的。网页的量化表示是网页分类过程中必不可少的阶段，而在前面关键词提取过程中已经得到了正文的特征向量，于是直接使用正文特征向量进行分类可以节省网页量化过程的时间开销，这正是将共性需求的信息一次性提取的优势之一。仅对网页的正文分类有效的排除了噪音内容的干扰，从而提高了分类的准确性。我们使用的是北京大学网络实验室开发的分类器（详见第十一章）。

标题:

HTML 网页中，网页的标题由<title>标签标识。通过统计我们知道，94.12%的网页是有标题的，但在这些标题中，有很多是如下标题：“Untitled Document”、“New page”、“welcome”、 “欢迎访问”。这其中有的是网页制作工具为新创建的网页赋予的初始名称，有的是网页制作者较为常用的网页标题。它们是没有任何网页描述能力的，因而并不是合格的标题。针对没有标题或者使用上述无描述能力标题的网页，我们从关键词集合中选取权值最高的作为网页的新标题。

摘要:

摘要的提取基于这样的事实：文章都是按内容分段组织的；阅读者通常是根据一段文章中某几个子句来得到该段文章的大意，而这几个子句的选择通常是通过扫描某些关键词来定位的。因此，如果可以自动识别文章中不同的段落，那么基于上述得到的关键词，就可以得到能够模拟读者浏览文章过程的摘要提取算法。



a) 完整的标签树　　b) 正文块对应的标签树（正文树）　c) 确定的正文根结点及其各个段落

图 7-7 正文段落识别过程

识别文章段落:

HTML 网页中的结构信息是对网页版面的描述，这使得自动识别文章的不同

段落成为可能。在正文提取部分已经得到了网页的正文，在网页的标签树中，所有正文块也构成了一个树状结构，称之为正文树。在正文树中，首先找到所有叶子节点的最近共同祖先结点作为正文根结点。正文根结点的各个子结点对应的正文块就是正文的不同段落。段落识别过程如下图所示。图 7-7 中的<tr>内容块就是正文根结点，其下面三个<p>内容块就是三个段落。

基于段落的语句提取：

以正文的段落为单位，在各个段落中定位网页的关键词并累加关键词的权值作为关键词所在语句的权值；最后在每个段落中限量选取权值大的语句，就组成了网页的摘要。该方法得到的摘要不能保证摘要中的语句之间有上下文关系，但能作到简短的摘要能覆盖整个文章的内容。

相关超链：

在超链选取的过程中，我们基于这样一个假设：网页中的超链在网页排版时通常按照主题聚集，换言之，相同主题的超链在网页中的位置是相近的，通常放在一个最里层的内容块（该内容块中不再包含其他内容块）中或并列的几个最里层内容块中。这就意味着我们可以以内容块为单位对超链进行取舍。对于超链的选取，我们实验了两种策略。

1) 基于 anchor text 的超链选取策略

anchor text 是对超链所指向网页简短、概要的说明，在一定程度上体现了被指向的网页的内容。基于 anchor text 的超链选取方法是通过比较每个 hub 类型内容块中 anchor text 集合与正文的相似度来决定该块中链接的取舍。

```
1：计算网页正文对应的特征项向量 φ
2：for 网页中的每个叶子内容块CBi do
3：    if CBi 是 hub 块 then
4：        计算CBi对应的特征项向量φi
5：        计算φ与φi的相似度similarityi
6：        if similarityi > β then          // β 为相似度阈值
7：            保留CBi中的URL
8：        else 不保留CBi中的URL
9：        end if
10：   end if
11：end for
```

图 7-8 基于 anchor text 的超链选取算法

通过这一算法，我们可以对 hub 类型内容块中的超链进行取舍，而其它类型内容块中的超链通常是对正文中某些信息的详细说明，因而其它类型内容块中的超链通常是内容相关的。因此，整理上述算法保留的 Hub 类型内容块中的超链和其它类型内容块中的超链就构成了整篇网页的相关超链集。

2) 基于分类的超链选取策略

基于分类的超链选取方法是通过判断一个 Hub 类型内容块中某个超链（通常是第一个）指向的网页与本网页正文的类别是否相同来决定该块中所有链接的内容相关性。该方法可以有效的解决上述方法中 anchor text 信息过少的不足，而且实验结果证明，该方法确实比基于 anchor text 的方法准确，但需要动态的从 Web 上抓取并分类，因而时间开销很大。

3. Hub 网页和图片网页的提取算法

与有主题网页相比，Hub 网页和图片网页有着自己的特点。Hub 网页中，网页的主题就是提供超链，超链本身就是正文。因此，相关超链信息对 Hub 网页来说是没有意义的。图片网页中，网页的主题内容就是图片和描述图片的信息，而描述图片的信息很多情况下是以超链的形式出现的。因此，图片及其周围的信息（包括超链）都是网页的正文。在处理过程中基于如下的假设：在网页的布局上，重要的信息通常是放在中间，网页两边信息的重要性相对较弱。因此，对于 Hub 网页和图片网页，我们可以将网页中间区域的内容块作为网页的正文，而周围的内容块则通过与正文比较相似性来决定取舍。

五、 模型应用及实验研究

1. DocView 模型在网页自动分类中的应用及实验分析

网页与传统文本的一个重要区别是网页内容的随意性，这就导致网页内容中的噪音内容很多，因此，在网页分类过程的开始首先对网页作适当的净化，可以在一定程度上改进分类的准确性。将 DocView 模型中正文要素和相关超链要素重新组合就得到了净化的网页。在本实验中，我们以一个现有的分类器作为基准，提取基准分类器的训练集和测试集中网页的 DocView 模型，并用模型中正文要素和相关超链要素组合成的新网页替换原始网页，从而形成净化的训练集和测试集。然后，通过对净化后的训练集学习得到新的分类器，并用净化后的测试集进行测试。通过比较新分类器与基准分类器的测试结果，证明模型中正文要素和相关链接要素提取的正确性以及用它们代替原网页的合理性。

1) 实验数据集

实验中用北京大学网络实验室开发的分类器作为基准分类器，该分类器支

持的分类体系中共有 733 个类别，分为三层，其中顶层类有 12 个，该分类器的训练集和测试集共有 15570 个网页，这些网页就是按照上述分类体系组织的。

2) 如果 DocView 中正文要素和相关超链要素提取的正确性足够高，那么把 DocView 中正文要素和相关超链要素组合起来生成的新网页将有效的消除噪音内容，因此用新网页代替原网页后的新训练集会使得各个类别的主题更为明显，而新测试集中网页的类别也更为清晰。所以，基于新训练集和测试集的测试结果也会得到提高。

3) 实验评测标准

对分类效果的评价采用传统的查准率、查全率、F_1 值。

$$precision_i = \frac{\text{分类到类别 i 并且分类正确的文档个数}}{\text{实际分类到类别 i 的文档总数}}$$

$$recall_i = \frac{\text{分类到类别 i 并且分类正确的文档个数}}{\text{实际属于类别 i 的文档总数}}$$

$$F_{1i} = \frac{2 \times precision_i \times recall_i}{precision_i + recall_i}$$

4) 实验结果及其分析

图 7-9 是新旧分类器的性能比较，横轴表示不同的类别编号，对应上面式子中的下标 i（最后一项是关于所有 12 类的平均值），表 7-1 是图 7-9 中类别编号对应的类别含义。

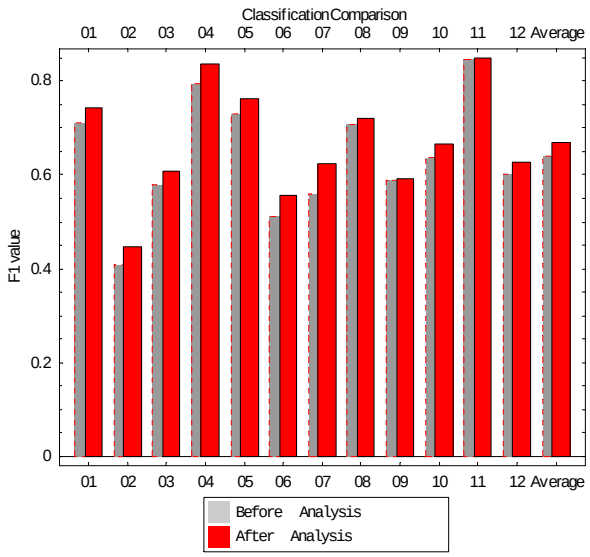


图 7-9 网页净化前后分类效果对比

通过图 7-9 我们看到，所有类别的分类结果均比原来有所改善。由于我们并没有对分类器的特征项提取算法和分类算法作任何改进，所以，图 7-4 所示的提高完全是从网页本身的改进得到的。因此可以得到两条结论：

- 净化过程中，没有出现明显的信息遗漏。
- 广告等噪音信息确实得到有效的去除。

表 7-1 类别编号对照表

类别号	01	02	03	04	05	06	07	08	09	10	11	12
类别名称	人文与艺术	新闻与媒体	商业与经济	娱乐与休闲	计算机与因特网	教育	各国风情	自然科学	政府与政治	社会科学	医疗与健康	社会与文化

由于人工选取训练集和测试集的网页时已经做到尽量选取正文信息多、噪音信息少的网页，因此，网页净化在实际应用中的效果要比该实验的结果更为明显。

2. DocView 模型在网页消重中的应用及实验分析

在本实验中，我们将DocView模型应用于预处理环节中的网页消重[1]。消重是指将搜集到的网页中的镜像或转载网页去掉的过程（镜像网页可以理解为一种特殊的转载网页），在消重后的网页集上建索引再提供服务可以保证用户查询时不会出现大量内容重复的网页。由于大量的转载网页并不是对原始网页的简单拷贝，而是将要转载的内容放在新的模板中再提供服务。因此模板中的内容就会干扰消重程序对转载网页的判断，从而导致错误消重。常见的错误消重有以下两种情况：

- 相同的内容，由于放在了不同的模板中导致应该被消掉但实际上被消重程序判断为非转载网页而保留。
- 不同的内容，由于放在了相同的模板中导致不应该被消掉但实际上被消重程序判断为转载网页而消掉。

从实际系统中也可以看出，模板因素是导致消重不够准确的一个主要原因。鉴于此，本实验中首先提取网页的 DocView 模型，然后将模型中的正文要素取代原始网页作为判断转载网页的依据。

1) 实验数据集

实验中使用三组网页：基准网页集、转载网页集、非转载网页集。

[1] 关于网页消重算法见本章第二节

基准网页集是用来模拟 Web 上被转载的原始网页，该集合中的每一个网页都存在一个或多个满足特定条件的转载网页在转载网页集中，也存在一个或多个满足特定条件的非转载网页在非转载网页集中。

转载网页集是用来模拟 Web 上的转载网页，该集合中存放基准网页集中每个网页对应的转载网页，这些转载网页满足如下条件：与对应的基准网页使用不同的模板。该集合用来测试上述情况一。

非转载网页集是用来模拟 Web 上使用相同模板但内容不同的网页，该集合中存放基准网页集中每个网页对应的非转载网页，这些非转载网页满足如下条件：与对应的基准网页使用相同模板而且内容属于同类。要求与基准网页内容属于同类是因为属于同类的网页容易被误消，所以使用同类的网页能更好的说明问题。该集合用来测试上述情况二。

理想情况下，转载网页集中的网页应该全部被消掉，而非转载网页集中的网页应该全部保留。因此，我们可以通过计算一个消重算法对转载网页集和非转载网页集消掉的网页个数来评价该消重算法。

我们人工为基准网页集、转载网页集、非转载网页集三个集合选取了网页，网页内容覆盖体育、娱乐、新闻、科技、社会、财经、教育、医学、IT、游戏等领域，目的是为了让实验网页的正文内容覆盖面足够广从而避免模板内容与网页正文内容有特殊关系。三个集合中共有 79 个网页，其中基准网页集有 26 个，转载网页集有 26 个，非转载网页集有 27 个。

2) 实验思想

由于消重的目的是消去主题内容相同的网页，而 DocView 模型中的正文要素正是去掉网页中模板内容后的主题内容，因此，在使用相同的消重算法的前提下，用 DocView 模型中的正文要素代替网页原文参与消重，消重准确性应该有所提高。

3) 实验结果及分析

表 7-2 中的数据是使用同一种基于关键词的消重算法但基于不同内容的消重结果。表中的数据是被消掉的网页数。

表 7-2 消重实验结果

消重结果	转载网页集（共 26 个）	非转载网页集（共 27 个）
基于网页原文的消重结果	1	0
基于净化网页的消重结果	20	0

从上表中的数据可以看到，在基于网页原文的消重方法中，由于模板因素的干扰，导致消重算法将模板不同的转载网页作为不同网页而没有消掉。而用

DocView 模型中的正文要素代替网页原文参与消重，可以很好的克服模板因素的干扰，从而提高消重准确性。

第二节 网页消重算法

如前所述，我们粗略地将内容完全相同的网页称作镜像网页，主题内容相同的网页称作转载网页。就消除主题内容重复的网页而言，我们完全可以把镜像网页看作转载网页的特例来处理。由此，所谓网页消重就是指去除网页集合中转载网页的过程。

国际上对转载文档消重算法的研究最初主要是针对大型文件系统的，后来又被拓展应用于数字化图书馆项目和搜索引擎系统。美国 Arizona 大学的研究人员采用计算文档的重叠程度的方法来发现一个大型文件系统中的相似文件[Manber,1994]。Stanford 大学的研究人员开发了 SCAM (Stanford Copy Analysis Mechanism)原型系统用于发现相似的数字化文档，后来在对其消重算法作了改进之后应用于 Google 搜索引擎系统。几乎所有的上述消重技术都基于这样一个基本思想：为每个文档计算出一组指纹（fingerprint），若两个文档拥有一定数量的相同指纹，则认为这两个文档的内容重叠性较高，也即二者是内容转载的。

由于上述系统的应用目标不同，它们计算文档指纹的方法以及在如何度量两个文档的相似程度方面有较大差别。文献[Shivakumar and Garca-Molina,1998]采用了一种对全文分段签名的算法。这种算法把一篇网页按一定的原则分成 N 段（如每 n 行作为一段），然后对每一段进行签名（即计算指纹），于是每一篇文档就可以用 N 个签名后的指纹来表示。对于两篇文档，当它们的 N 个签名中有 M 个相同时（m 是系统定义的阈值），则认为它们是互为转载的网页。该算法使用对<文档标识(DocID),段标识(ChunkID),指纹(Fingerprint)>三元组排序的方法避免了对所有网页作两两比较，使算法复杂度有所降低。但是该算法的空间复杂度和时间复杂度仍然是相当大的，若应用于海量的搜索引擎系统（通常包含上亿个 Web 页面），仍然难以取得理想的效果。我们结合基于关键词匹配的搜索引擎系统的特点，提出了 5 种网页消重方法，实验表明这些方法的效果很好，目前已被成功地应用于北大天网搜索引擎系统，用于消除转载网页[谢正茂,2003]。

一、 消重算法

1. 算法基础

当前比较成功的搜索引擎系统大多是基于关键词匹配和结合向量空间模型来完成用户的检索请求的。典型的系统包括 Google 和天网系统。通常这类系统在

对已抓取回来的网页进行分析时，要提取网页中出现的关键词和摘要信息，并以关键词作为网页的特征项。

天网系统在搜集并分析一篇网页时，提取并记录了网页中出现的关键词，同时根据公式赋予每个关键词一个权值，这些关键词的权值构成一个向量空间，可以用来表示该网页。另外，我们还从网页中提取了 512 个字节的有效文字（指用户实际访问该网页时能看到的文字，在 html 和其他格式的网页中，有一些用户看不到的文字，它们告诉浏览器该执行什么样的操作及如何显示网页，包括字体、颜色、排版等信息）作摘要。当用户查询时，摘要同网页的标题、URL 等信息一起显示给用户，供用户了解网页的内容，选择感兴趣的进行浏览。

2. 算法描述

考虑到基于关键词匹配的搜索引擎系统的特点，结合使用网页的向量空间模型，我们提出了 5 种网页消重算法，用于快速、有效地发现Web上的转载网页。下面逐一介绍这几种算法。在以下的描述中，用P_i表示第i个网页，其权值最高的前N个关键词构成的特征项集合为$T_i = \{t_1, t_2, ..., t_{in}\}$，其对应的特征向量为$W_i =< W_{i1}, W_{i2}, ..., W_{im} >$，其摘要用$Abstract(P_i)$表示，前$N$个关键词拼接成的字符串用$Concatenate(T_i)$表示，而先对前$N$个关键词按字母序排序后再拼接成的字符串用$Concatenate(sort(T_i))$表示。另外，用$MD5(X)$来表示字符串$X$的*MD5*散列值，用$Mirror(P_i, P_j)$表示$P_i$和$P_j$互为转载网页，用$A \Rightarrow B$表示“若A成立则B成立”。

算法 1：

$$(MD5(Abstract(P_i)) = MD5(Abstract(P_j))) \Rightarrow Mirror(P_i, P_j)$$

算法 2：

$$(MD5(Concatenate(T_i)) = MD5(Concatenate(T_j))) \Rightarrow Mirror(P_i, P_j)$$

算法 3：

$$\left.\begin{array}{l}(MD5(Concatenate(T_i)) = MD5(Concatenate(T_j))) \\ \left(\dfrac{|W_i - W_j|^2}{|W_i|^2 + |W_j|^2}\right) < \delta\end{array}\right\} \Rightarrow Mirror(P_i, P_j),$$

$$其中 0 \le \delta \le 1$$

算法 4：

$$\left.\begin{array}{l}(MD5(Concatenate(sort(T_i))) = MD5(Concatenate(sort(T_j)))) \\ \left(\dfrac{|W_i - W_j|^2}{|W_i|^2 + |W_j|^2}\right) < \delta\end{array}\right\} \Rightarrow Mirror(P_i, P_j),$$

$$其中 0 \le \delta \le 1$$

算法 5：

$$(MD5(Concatenate(sort(T_i))) = MD5(Concatenate(sort(T_j)))) \Rightarrow Mirror(P_i, P_j)$$

3. 算法分析

可以看出，我们设计的第 1 种算法采用了对网页摘要求 MD5 散列值的方法，当两个网页的散列值（占 16 个字节）相同时，就认为二者是互为转载的。由于 MD5 算法的严格性保证了当两个网页的摘要内容有一个字节不同时，其散列值就不同，这样就使得本来应作为转载处理的却没有被确定为转载网页。为此，我们又基于向量空间模型理论，提出了第 2 种和第 5 种算法。第 5 种算法表明，当两个网页的权值最高的前 *N* 个关键词集合相同时就认为二者是互为转载的网页。第 2 种算法比第 5 种要严格一些，它不仅要求两个转载网页的前 *N* 个关键词相同，其顺序也是一致的（按权值排序），因而第 2 种算法有可能会漏掉一些转载网页。

算法 2 和算法 5 都只是要求转载网页的前 N 个特征项相同，没有考虑到这些特征项所构成向量的夹角大小。算法 3 和算法算法 4 则分别在算法 2 和算法 5 的基础上分别考虑了两个网页特征向量的相似度。但向量相似度的计算并没有使用夹角余弦值来定义，因为它只度量了两个向量的夹角大小，而没有考虑向量的长度。我们认为只有当两个向量的夹角小，同时其长度相差也小时，二者才是相似的。针对这一点，我们又设计了判断两个向量相似度的方法，即算法 3 和 4 的第二个条件：

$$SIM = \left(\frac{|W_i - W_j|^2}{|W_i|^2 + |W_j|^2}\right)$$

可以看出，*SIM* 能够同时兼顾向量的夹角和长度两个因素。当两个网页内容毫不

相关时（即它们的关键词集合没有交集），W_i 与 W_j 垂直，*SIM* 的值为 1。当两个网页相同时，*SIM* 为 0。当两个网页相似而不相同时，*SIM* 的值介于 0 和 1 之间，于是 *SIM* 的值成为判断两个网页相似度的标准。另外，类似于算法 5 是对算法 2 的条件放松，算法 4 也是对算法 3 的放松。

后四种算法都对向量空间模型理论作了较大的简化。首先，我们只从网页中出现的所有关键词组成的M个特征向量提取了前 N个(N<M),这把理论模型的限制放松了。只所以可以这样做是因为：

1) 特征向量的前N个分量绝对值大，基本能确定特征向量的方向。取较少的关键词能减少算法的复杂度。尽管有可能降低其准确度，降低多少，后面的实验将对其作出评测。
2) 转载网页的制作人，对网页稍加改动变成相似网页时，不能改变其基本意思。而网页的基本意思一般通过其中出现的高频词来反映。后面的(M-N)个词出现的次数为 1 或 2，相对而言，这些词的出现是不稳定的，当使用这些词来判断相似网页时，反而会漏掉一大批相似网页。

其次，后 4 种算法都要求前N个关键词组成的集合要相同。这却把理论模型的限制加强了。这主要是由于对算法复杂度的考虑，判断两集合交集大小需要先求出它们的交集。求交集运算的复杂度较大，而把一百万网页两两求交集，其10^{12}量级的运算量是我们不敢提及的。我们只能考虑用 MD5 算法对集合签名，实际上就是对关键词序列签名，来表示集合的相同与不同。签名算法有极高的敏感性，作用对象稍有不同就会给结果带来很大的差异，并且不可能从签名差异的大小来判断原签名对象差异的大小。作这样的简化后，有可能出现这样的情况，位置在N附近的词在排序上出现的微小变动，如第N 个词与第N+1 个词位置互换了，本来是两篇相似度很高的文章，可能会被我们的算法漏掉。这对算法的影响到底有多大，我们仍需通过实验来评测。

二、 算法评测

1. 评价方法

我们为网页消重算法设计的评价指标包括算法复杂度、查全率和准确率三个方面，其中算法复杂度又包括时间复杂度和空间复杂度。在本节中，查全率是指消重算法所发现的转载网页占总网页的百分比，而准确率反映了算法所发现的转载网页中有多少是真正的转载网页。假设要处理的网页数为 *N*，后 4 种算法使用的关键词个数为 M，每个关键词的权值占 4 个字节，MD5 摘要占 16 个字节，则算法 1、2 和 5 的空间复杂度约为$(16\times N)$，算法 3 和 4 的空间复杂度约为

$(N \times M \times 4 + N \times 16)$。可以看出这 5 种算法的空间复杂度都很小。本节将重点考察算法的时间复杂度、查全率和准确率。时间复杂度和查全率将直接基于天网系统 2000 年 5 月份的数据并运行算法来得到。由于算法输出的转载网页太多，我们无法一一判断每个网页是否是真正的内容转载网页，因而对于准确率的计算我们采用了如下的估算方法：先将算法的输出结果（即算法求出的转载网页）分为 n 段，在每段中对镜像随机取 100 个采样，用人工确认的办法得到每 100 个采样的准确率，最后用这 n 个准确率的平均值来作为算法的准确率。

2. 实验结果

在 2000 年 5 月上旬，我们使用天网的搜集子系统从国内的 2 万多个 Web 站点上搜集了 1,182,899 个网页，对于每个网页我们都保存了其摘要、关键词及其权值等信息，依此作为我们的实验对象。这 5 种算法运行的机器是一台 PC 机，配有双 CPU，内存为 256MB，硬盘 36G，运行的操作系统为 Turbo Linux 6.0。

基于上述实验环境，我们分别使用上述 5 种算法来消除转载网页。当关键词个数 N 取 10、向量偏差度阈值 δ 取 0.01 时，得到了第一组实验结果如表 7-3 所示。其中“总和”是指只要某个网页被五种算法中的任一种检测为转载，它就被确定为转载。实际上这里的“总合”等价于算法 1 和算法 5 的“总和”，因为算法 2、3 和 4 发现的转载网页都能够由算法 5 所发现。 算法 1 与其它算法的差别较大，这是由于它们所选取的判断对象的不同造成的，前者选用了 512 字节的摘要，而后者用的是一组关键词。从表 7-3 可以看出，这五种算法的准确率都很高，而其中第 5 种具有最高的查全率，而准确率的损失又不是太大，我们认为效果是最好的。同时，也可以看出“总和”的查全率要比 5 种算法中的任意一种好得多，而其准确率也很高。这启示我们可以结合使用算法 1 和 5 来消除转载网页。

表 7-3 当 N=10、δ=0.01 时 5 种算法的查全率和准确率

算法编号	总网页数	转载数	查全率(%)	准确率(%)
算法 1	*1,182,899*	*199,763*	*16.89*	*98.11*
算法 2	*1,182,899*	*212,914*	*18.00*	*98*
算法 3	*1,182,899*	*210,475*	*17.79*	*98.03*
算法 4	*1,182,899*	*219,841*	*18.58*	*97*
算法 5	*1,182,899*	*225,919*	*19.10*	*97*
总和	*1,182,899*	*265,752*	*22.47*	*97.09*

由于算法 3 和 4 的效果受向量偏差度阈值 δ 取值的影响，我们令 δ=0.1，N=10 重复了上述实验，得到的结果如表 7-4 所示。可以看出 δ 的取值对算法的查全率

和准确率影响不大。

表 7-4 考察 δ 的取值对算法 3 和 4 的影响

算法编号	总网页数	转载数	查全率(%)	准确率(%)
算法 3	1,182,899	211,718	17.90	98
算法 4	1,182,899	222,151	18.78	97

关键词个数 N 的取值是影响后 4 种算法效果的一个关键因素，因而我们针对效果最好的第 5 种算法，通过取不同的 N 值，得到了查全率的变化曲线，如图 7-10 所示。从图中可以看出，当关键词取得极少时,镜像算法的查全率很高,这无疑是用极低的准确率作代价的。关键词慢慢增多，查全率迅速下降，当选取到 9 个或 10 个关键词时，这种下降趋势变得很平缓，同时准确率达到一个比较高的水平。当选取到二十个以上的关键词时，有些低频词被选取，一部分转载网页的差别被反映出来，于是查全率有较大下降。随着选取的关键词继续增多，有差别的转载网页全部被认为是不同网页，剩下的完全相同网页使查全率不再随选取关键词的多少而变化。我们得到的结论是，关键词取 10 个左右最恰当，在较高的准确率的基础上，获得了最大的查全率与最小的运算量（主要是指关键词排序和签名运算）。

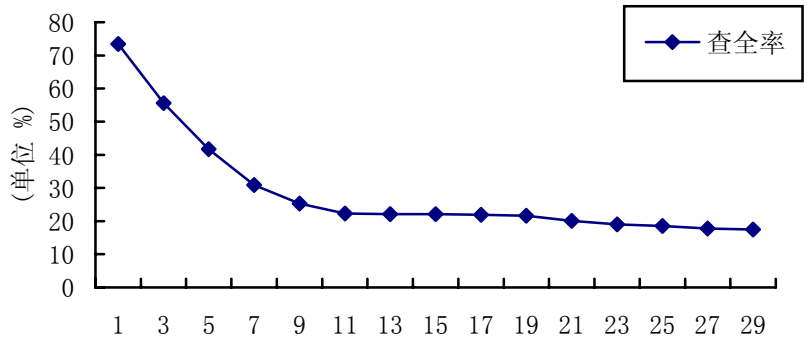


图 7-10 查全率随选取关键词个数的变化

3. 与现有算法的比较

为了与现有的算法进行比较，我们把文献[Shivakumar and Garca-Molina,1998]的实验结果（主要包括查全率和运行时间）列于表 7-5 中。我们的实验结果列于表 7-6 中。其中文献[Shivakumar and Garca-Molina,1998]的实验平台为带双 CPU

的 SUN UltraSPARC 工作站，内存为 256MB, 对换工作区大小为 1.4G，机器运行的操作系统为 SunOS 5.5.1。可以看出其运行环境是优于我们的。基于关键词的消重算法有一个优点，就是只用一次签名就可以很好的判断某网页是否为转载，这样对 N 个网页计算其网页转载的复杂度为 O(N)。通过比较表 7-5 和表 7-6 的数据，可以看出，算法 1 和 5 平均处理一个网页的时间总和不到“分段签名算法”的十分之一。我们的算法在时间上有明显的优势。另外，从这两个表可以看出，算法 1 和 5 总和的查全率要好于分段签名算法的，这表明消重的效果也略好于文献[Liu, et al.,2000]的分段签名算法。

表 7-5 分段签名算法的时间复杂度及性能

分段方法	网页总数	所用时间	查全率（%）
全文作一段	24,000,000	2760 分钟	17
每四行作一段	24,000,000	2940 分钟	22
每两行作一段	24,000,000	3240 分钟	22

当然我们的算法也有其局限性。如算法 1 要求事先知道各个网页的摘要信息，而后 4 种算法则要事先知道每个网页的关键词。但是目前绝大多数的搜索引擎是基于关键词匹配的，系统本身已为我们找出文章的摘要和关键词。而对于数字化图书馆项目而言，大多数数字化文档的摘要和关键词也是事先知道的，因而我们的算法仍然是适用的。

表 7-6 基于关键词的各算法的时间复杂度及性能 (N=10, δ =0.01)

算法编号	网页总数	所用时间	查全率（%）
算法 1	1,182,899	5 分钟	16.98
算法 5	1,182,899	4 分钟	19.10
总和	1,182,899	9 分钟	22.47

4. 小结

本节描述了 5 种网页消重算法，并使用天网系统对之进行了评测，体现了这些方法的优越性。我们的实验也进一步表明在 WWW 上存在大量的镜像和转载网页，如下为比较常见的几类：

1) 政府部门的法令、通告。例如，公安部颁布的《计算机信息网络国际联网安全保护管理办法》在国内的网站上有 120 个镜像。
2) 重大新闻、热点新闻。例如，《中欧 WTO 谈判取得进展但未达成协议》

有 75 个镜像。

3) 技术文档。如：TUCOWS WinSock utilitics 有一百个镜像，Micorsoft Internet Information Server 2.0 Manuls 有 90 个镜像。

4) 提供一定服务的网站和网页。例如，“黄金书屋”的镜像有 30 个，而列出了国内各所大学 URL 的网页有上千个。

除了采用本节的消重算法来提高搜索引擎系统的输出质量外，还可以研究以网页的镜像度为参考来判断网页的重要度，进而提高 Web 搜集系统的搜集质量和效率。同时，这些算法若应用于数字化图书馆项目，也将是行之有效的。

第八章　高性能检索子系统

以 Google 为代表的商业搜索引擎获得了很大成功。到 2004 年 4 月，Google 已经索引了全世界 42 亿个页面，每天接受上亿次查询请求。但是商业搜索引擎的核心技术属于商业机密，在激烈的竞争环境下不会公开。而在研究领域，因为受到条件的限制，对大规模通用搜索引擎系统的技术探讨也较少。在第二章我们介绍过，搜索引擎包括搜集子系统，预处理和服务子系统三大部分。有时候为方便起见，将建立索引和提供服务放在一起，称为检索子系统。搜集系统研究如何更快速抓取更多高质量网页的相关技术，检索系统研究如何进行网页文档索引，为用户提供高性能的检索服务。后者主要建立在信息检索领域的相关技术之上，同时根据 Web 自身的特点也发展了一些新技术。文献[Brin and Page,1998]是 Google 在斯坦福大学的原型系统的一个较全面技术介绍，其中重点介绍了其检索系统的设计与实现，验证了使用链接分析技术可以有效提高搜索引擎检索效果。和传统的信息检索系统相比，大规模搜索引擎的检索系统面临许多新的挑战。

本章以天网 Web 服务检索系统为基础，分析搜索引擎的检索系统设计与实现的基本技术。

本章内容安排如下：

第一节围绕检索效率和检索效果这两个最基本的指标，介绍了天网检索系统的软件集成框架结构和分布式硬件系统结构，并从索引创建和检索过程详细分析了高效检索系统的相关基本实现技术。

第二节讲述了一种倒排文件基本性能模型，给出了响应时间、吞吐率和系统内部参数之间的一种关系。并结合计算机的性能指标，对模型进行了精化。

第三节详细阐述了天网中所采用的混合索引技术。首先介绍了索引词选择对检索系统的影响，以及一般的索引词选择方法存在的优缺点，然后给出了一种基于倒排文件实现的混合索引方法，它可以有效提高在搜索引擎中进行短语查询的检索效率，同时不影响系统检索效果。

第四节阐述了倒排文件的缓存优化设计技术。通过对真实搜索引擎系统的倒排文件访问数据序列的特性分析和仿真试验，考虑不同的系统性能参数优化目标，得到了一种倒排文件缓存的优化设计方法

第一节 检索系统基本技术

一、 系统设计与结构

搜索引擎检索系统的设计围绕检索效率和检索效果这两个指标展开。对一个成功的搜索引擎来说，首先必须具有相当高的检索效率。由于通用搜索引擎是面向大众的，其信息需求的重要性参差不齐，绝大多数可以说是“随心所欲”的，其价值不值得等待很长的时间，因此一个响应迟缓的系统只能意味着较少的用户。按一般的习惯，搜索引擎对用户查询的响应时间应该不超过秒级，这相对于搜索引擎需要处理的海量网页数据而言是一个挑战。而如何提高搜索引擎检索效果，更是人们不断研究的课题，但它是要在保证检索效率的前提下才有意义。因此，信息检索领域有一种观点，认为搜索引擎的检索技术相对于最新的信息检索研究成果是一种倒退。如果仅从检索效果上看，确实如此，但由于搜索引擎面临的效率压力，使得在实现上往往需要在效率和效果之间折衷，而不一定采用效果最好的技术。同时，有统计表明在 Web 搜索环境下，用户普遍使用短查询、不做查询优化[Silverstein, et al.,1998],[Spink, et al.,2001],[Wang, et al.,2001]，这些特点也是搜索引擎提高检索效果面临的主要困难（因为用户向系统提供的信息太少）。但在另一方面，传统信息检索只从文本内容上计算文本和查询的相关程度，而 Web 环境下，除了网页文本数据，还有大量其它信息可以为这一相关性的计算提供辅助支持，比如网页内的 HTML 标记，URL，链接关系，Anchor text，网站目录数据等。如何有效利用这些信息是搜索引擎提高检索效果的一个重要途径。

天网的检索系统设计原则有两个：一是追求系统效率和可扩展性。二是力图通过一个集成的框架结构，能够有效地把各种有利于改善检索效果的技术集成起来，如图 8-1 所示。这样一个框架结构体现在三个方面：①文档表示。对一个网页文档可以有多种角度的表示方式，包括索引词、半结构化的元数据以及全局的网页属性。②用户信息需求的类型识别，以求能为不同类型的信息需求选择最佳的检索方式。③不同检索排序方式得到的结果的融合。

在图 8-1 中，方框表示检索系统，在服务点（SE ServicePoint）接受用户的查询请求（User InfoNeed）。用户请求经过检索代理(Retrieval Agent)分类，进行检索策略的选择，调用索引服务提供的相应检索机制来完成检索。通常，搜索引擎提供的最基本检索方式是基于关键词的布尔查询(Boolean OP)。但通常用户输入的查询为自然语言词语或者短语，并不是一个布尔表达式。一般情况下，搜索引擎默认用户的输入查询词之间为与(AND)关系。为了提高检索效果，有些搜索引擎也采取查询词扩展（query expansion）和相近(Proximity)计算技术，并用这些计算的结果来驱动后台的结果提取过程。Google 成功的使用了链接分析技术

[Kleinberg,1998],[Page, et al.,1998]，为每个网页赋予一个全局的权值（PageRank值）来表示网页的重要程度。网页的这种全局属性的检查在图 8-1 中由GlobalProperties 模块执行，除了 PageRank，还可以包括根据权威的网站目录数据、用户反馈或人工编辑等方式得到的网站权值。Meta 是元数据查询的执行模块，可以包括时间、文档格式、站点名称、分类类别等各种网页元数据，针对网页数据的信息提取技

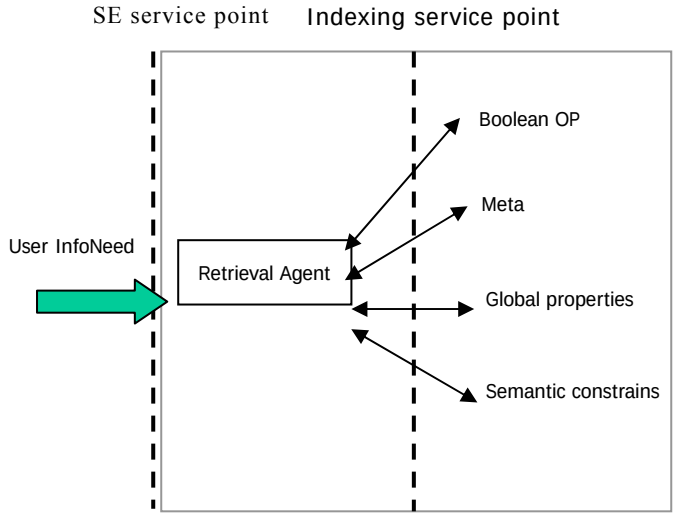


图 8-1 检索系统集成框架结构

术可以融合到这一模块中。天网在中文网页自动分类方面有一个研究小组，其网页文本自动分类技术已经应用在天网目录服务和检索中[冯是聪, et al.,2003]。在检索系统框架中，网页文档的分类类别起着重要的作用。利用 Meta 模块返回的网页类别信息，Retrieval Agent 可以进行类别聚合，把相同类别的网页集中显示给用户，这样一种方式可以更好的组织检索结果，改善检索效果。Semantic Constrains 模块是语义的约束检查模块，它建立在对网页文本中特定语义关系识别的自然语言处理技术之上，是实现回答自然语言问题的必要技术，第十二章将要介绍的“天网知名度”[Fame,2004]就是这样一种技术的成功应用。

天网检索系统的具体实现同样基于信息检索技术。

首先是排序算法和检索模型的选择。在图 8-1 的框架结构中，检索系统的相关性排序由多种因素综合决定。这其中，最基础的排序建立在信息检索的布尔模型和向量空间模型基础上。在 BooleanOP 模块中，首先执行布尔查询，得到的结果作为候选文档集合，然后按向量空间模型的相似度算法计算各个文档与查询的

相似度，结果作为排序的基础。最后由 RetrievalAgent 综合其它模块返回的信息，再进一步排序。典型的一种情况是，当查询词在 AnchorText 或者 Title 中出现时，把全局属性里的 PageRank 值与文档的相似度权值通过线性组合方式相加得到最后的排序权值。排序采用一种分级算法，分为三个级别：查询词的邻近关系运算结果；查询词出现的位置，包括 Title、AnchorText；相似度权值与其它的权值，如全局属性的 PageRank 权值。各种权值通过线性方式组合起来[Lei, et al.,2001]。

其次是索引的实现技术。天网检索系统采用倒排文件索引。对于大规模文档数据，倒排文件是经过大量实践检验的一种高效率的索引组织方式，能够很好的支持多种检索模型，提供高性能的检索。人们对倒排文件的组织和检索效率做了大量的研究工作，文献[Moffat and Zobel,1996],[Witten, et al.,1994],[Navarro, et al.,2000]对天网的索引实现有重要的影响。[Moffat and Zobel,1996],[Witten, et al.,1994]重点在倒排索引的压缩，[Navarro, et al.,2000]在倒排索引的随机访问技术，它们都被应用到天网的索引系统中。由于搜索引擎的责任是索引不断变化着的海量网络信息，倒排文件的组织还需要在检索效率和更新效率上进行折衷。一般倒排索引的索引项数据用链表方式分块存放有利于提高更新效率，但这会降低检索效率；反之，索引项数据连续存放有利于检索，而不利于更新。天网检索系统以检索效率为主要优化的目标，索引更新采用部分索引重建的方式。

整个检索系统采用分布式系统结构。搜索引擎的海量网页数据索引无法在单台机器上集中完成，分布式系统是解决数据规模和系统可扩展性问题的基本方法。天网检索系统的系统结构如图 8-2 所示。

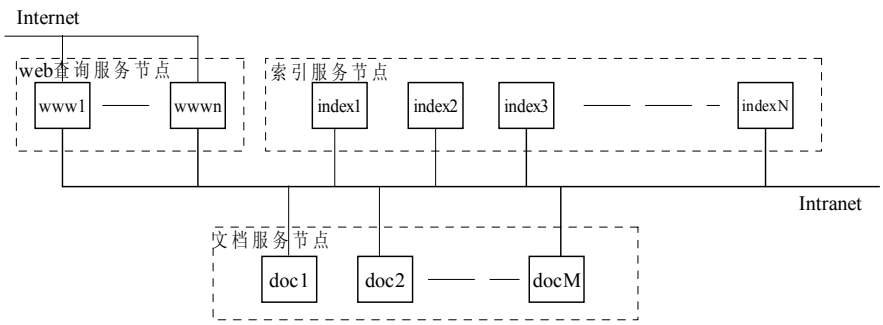


图 8-2 天网 WWW 检索分布式系统构架

现在运行的检索服务系统共使用 20 台 PC(PIII733/1GB)，其中一台为 WWW 查询服务器，其余 19 台为索引服务器，文档服务节点和 WWW 查询服务器使用同一机器。文献[Lu,1999]通过性能仿真实验对分布式信息检索系统的可扩展性进

行了深入研究。我们注意到，在学术界当人们谈及“分布式信息检索”时指的是研究不相交数据集在分布环境下的数据集选择和检索问题，和这里谈的搜索引擎检索的分布式系统结构不是一回事，但其提出的部分复制技术有很好的参考价值。文献[Tomasic and Garcia-Molina,1993]研究了倒排索引的物理组织对分布式查询的性能影响，认为最好的数据分布方式是“host index organization”,也就是每一个文档的全部索引项应该都分布在同一台处理机上。这种数据组织方式不仅可以最大限度降低节点间通讯开销，而且由于索引节点之间相互独立工作，整个查询系统有很好的容错性。天网的分布式检索系统设计就以此作为出发点[赵江华,2002]。

前面我们从总体上描述了天网检索系统的设计思想和系统结构，下面两部分分析天网的索引创建和检索实现技术。

二、 索引创建

对一个中文搜索引擎，索引创建不仅仅是一个高效的倒排算法，它还包含许多重要的方面：索引词的选择，中文分词、编码识别与转换、网页净化、强健的页面分析等。

1. 索引词选择

索引词的选择是检索系统实现的一个重要环节。现代搜索引擎普遍使用全文索引技术，即网页文档中所有词都参与索引。理想的索引词应该是表达文档内容的语义单位，即语言学里的词语，是那些专指义，而实际意义无法由组合成分相加得到的最小语言单位。但实际系统中中文文本必须通过自动分词程序的处理，分割成为独立的分词单位，再从分词结果中选择索引词。自动分词算法有两大类，普遍采用的方式是基于词典的分词方法，这一方法效率高，但分词精度受词典规模制约；另一种是基于统计语言模型的方法，可以发现一些新词。实际应用是两种方法的不同程度的组合。

除了中文分词外，对英文单词、数字、英文缩写词、特殊专有词的识别由一个词法分析器完成，通过不断完善词法规则，就可以支持如“C++，C＃，AT&T”这样的一些特殊词。英文单词统一转换为小写，但不作词根和词形变换。

2. 网页预处理

创建索引需要对网页进行分析，其中编码转换是一个重要步骤。Web 上的网页包含多种字符集和编码，搜索引擎的索引系统必须对它们转换，采用统一字符集和编码方案。UNICODE 是一种兼容性较好的字符集选择，而且可以使用不同的编码方案，比如 UTF8,UTF16 等。但由于程序移植和编程习惯上的困难，天网

目前仍然使用了 GBK 为系统的内部编码，这对于主要面向中文的搜索引擎已经足够了，但如果进一步考虑国际化，UNICODE 应该是更好的选择。常见的中文编码包括简体中文的 GBK，GB2312, ISO-2022-CN，GB18030，还有繁体中文的 BIG5，BIG5HKSCS 和 EUCTW。在 GNU 的 GLIBC 库中有对这些编码的转换支持，但是对于简体、繁体转换是简单的基于字－字映射，准确度不高。在网页分析中，识别网页的编码方式也比较繁琐。按 HTML 的规范[W3C,1999]，页面内容的编码依次由 Web 服务器返回的 HTTP 头信息中的 charset 字段；网页中 meta 标签里的 charset 属性以及每个网页元素的 charset 属性确定。但在实际情况中，网页存在许多编码设置错误的现象，尤其是那些由简繁体自动转换而生成的中文网页。这时需要有一个自动识别编码的模块，按统计的方法自动识别网页的正确编码。

此外，大量网页中存在不符合 HTML 规范的错误，这要求网页分析模块十分健壮。同时许多网页中存在大量无用的信息，比如广告、导航条等，这一现象在大型网站使用相同模板的网页中普遍存在。这些网页噪音对用户的信息检索没有意义，因此不应该被包含到索引范围内。

3. 索引创建算法

天网检索系统采用带位置信息的词级全文索引。系统采取了按站点划分网页数据的分布式方案，各个索引节点相互独立，索引创建过程在每个节点上独立进行。采用两趟的内存倒排创建算法，依次为每个小文档集倒排，最后执行多路归并，生成总的倒排文件。主要步骤如下：

1) 页面分析。按 HTML 语法规则分析网页标签结构、调用中文分词和英文词法分析器提取索引词。分析过程中记录每个索引词的文档频率 df 和在文档内的词频 tf，通过散列表转换为索引词编码，保存得到词典文件(lexicon file)，并保存页面分析的结果到临时文件。
2) 按统计得到的索引词的 tf 和 df 属性,可以估计出对应倒排项数据的长度，以此预申请整个文档集合倒排需要的内存空间。重新读取页面分析保存结果的临时文件，在内存中执行倒排，把结果保存到临时倒排文件中。
3) 对生成的多个临时倒排文件，执行多路归并，压缩编码，输出得到最终的倒排文件。

在索引创建过程中，页面分析，特别是中文分词为主要时间开销。算法的后两步相对很快。这样创建算法的优化集中在中文分词效率上，而没有采用效率更高的倒排创建算法[Heinz and Zobel,2003]。

三、 检索过程

天网分布式检索系统执行查询时，由 WWW 查询服务器通过多播把用户输入的查询串发送给每一个索引节点。各索引节点独立在本机上执行查询，再把检索结果中排序最前的 K（K=100）个结果返回给 WWW 查询服务器，在 K 值控制合理情况下，可以把返回结果数据包控制在一个以太网数据帧大小内，使系统具有很小的网络通信开销和延迟。WWW 查询服务器上的 RetrievalAgent 负责结果数据的收集、合并、重新排序，并访问文档服务器、提取摘要，格式化生成查询结果页面返回给查询用户。

文献[Wang, et al.,2001]通过分析搜索引擎用户查询日志，提出并在天网 WWW 查询服务器上实现了检索结果缓存。使用 LRU 缓存替换算法，缓存容量为 500 个检索结果时，缓存命中率能达到 60%左右，有效提高了系统整体性能。

在天网的分布式检索结构中，系统性能瓶颈最终在索引节点[赵江华,2002]。通过实验发现，索引节点的检索效率瓶颈在于磁盘系统的性能，检索算法中对倒排文件中查询词对应的倒排项数据读取是检索效率优化的重点。首先采用的是系统级优化措施。注意到操作系统中 I/O 系统的实现特点，采用 C 函数库提供的带缓冲的文件访问接口效率最差，操作系统提供的底层文件访问接口 read/write 效率较好，而使用内存文件映射或者直接设备访问可避免多次的内存拷贝问题，从而大大提高 I/O 访问的效率[Newman,2001]。其次再通过对倒排文件的组织优化，通过减少每次访问倒排项数据的数量和访问次数来提高检索效率。天网系统中采用了三种基本技术：索引压缩、随机访问的索引组织和重要索引词单独索引。

1. 索引压缩

倒排索引压缩可以减小倒排项数据长度。在检索过程中可减少内存和 I/O 带宽的使用，但同时要对压缩数据解码，增加了 CPU 时间耗用。实际系统中，I/O 是系统的瓶颈，而且 CPU 和 I/O 之间性能差距还在不断扩大，所以索引压缩技术作为一种有效提高检索效率的技术被普遍采用。

倒排索引压缩的方法基于“游程编码”，增量整数序列被变换为差分序列。组织倒排索引文件，可以把倒排项中的文档号和出现位置编号，都按递增序排列，这样可以通过“游程编码”变换，把大整数序列变换成较小的整数序列，再选取一种整数编码方案实现高效的倒排项数据压缩。

文献[Witten, et al.,1994]中给出了多种变长编码方法，其中 Golomb 编码是压缩效率较优的一种。文献[Williams and Zobel,1999]比较了多种编码方法的编码和解码效率。在搜索引擎应用中，检索效率是主要优化目标，而索引数据的空间占用相对并不重要。变长编码有解码慢的缺点，天网系统实际采用字节对齐的定长

编码 ByteCode。实验测试得到 ByteCode 和 Golomb 的平均压缩比率分别为 0.3359 和 0.2635，解码时间两者的比例为 1:6。

2. 随机访问的索引组织

文献[Navarro, et al.,2000]提出了对倒排索引的索引项建立二级索引，使得可以随机访问倒排项数据块。在一般情况下，这一技术可以减少倒排项数据的访问量，但同时可能增加 I/O 访问的次数。在采用这一技术时需要确定随机访问倒排项数据块的大小，在节省 I/O 带宽与 I/O 访问的次数的开销之间取得最好的折中。具体来说，小数据块访问会带来更多次系统调用，带来更多次的寻道时间消耗；大数据块访问读入冗余的数据，带来过多的数据传输时间消耗。根据磁盘访问性能分析的实验表明，使用较大的数据块系统性能较好，因此天网检索系统目前采用 32KB 为最小块单位。

在二级索引之外，倒排项数据还使用数据块自索引技术[Moffat and Zobel,1996]。选择 32KB 为二级索引块大小，每 32KB 的位置信息记录一个开始文档号。每块数据内部使用 512 字节作为自索引的段长，使用 ByteCode 压缩编码。自索引技术不减少 I/O 数据访问量和访问次数，但使得检索算法在处理倒排项数据时，可以跳过一些压缩的数据块，节省处理时间。

3. 重要索引词单独索引

对重要索引词单独索引，这样可以产生一个小的倒排索引文件，控制其大小能保存在内存中，如果有相当的查询在这个小索引文件中获得足够的返回结果，则查询结束；当检索得到的结果不足时，才去访问磁盘上的整个倒排文件。通过这一方式，系统可以节省大量磁盘的访问开销，大大提高效率。

这一技术有效应用的前提是小索引中查询得到的结果文档在整个倒排文件的查询结果集合中排序在最前面，否则会降低系统检索质量。这一点可由排序算法保证。被选择的重要索引词包括 Anchor text，Title 还有利用天网文档模型技术[Zhang, et al.,2004]提取的正文摘要中的词。

第二节　倒排文件性能模型

上一节，我们已经多次提到了倒排文件，可以说它是现代大规模搜索引擎工作的一个核心技术。虽然原理简单，但它灵活而高效，可以根据需要做不同的变通。本节结合检索系统的宏观需求和实现倒排文件的硬件参数，建立起倒排文件的一种性能模型，该模型对于在设计阶段估算倒排文件的运行效率有一定的指导意义。

一、 引言

评价一个大规模信息检索系统，有两个方面基本的考虑：效果（effectiveness）和效率（efficiency）[Frieder, et al.,1999]。“效果”常常也称为“质量”，指检索返回结果集合的准确性（或者相关性）和完整性，通常有两个指标：查准率和查全率，是第十章的内容，本章将不涉及。“效率”，我们在此也称“性能”，最重要的指标就是系统的查询响应时间（response time）和系统的查询吞吐率（throughput）。

响应时间是指从用户向系统提交查询到他开始看到结果的时间间隔。对于面向普通用户的网络查询系统来说，这个时间在“秒”量级是比较合理的；例如Web搜索引擎和数字图书馆，用户通常不会有耐心在一个查询上等待超过 10 秒钟还不见任何回应[1]。

吞吐率是指系统在单位时间（秒）里可以服务的最大用户查询数量。这里实际上有两个层次的含义。一是在单位时间里系统能够承受、不至于导致“拒绝服务”的查询的数量，二是在单位时间里能够接受、并满足服务质量的查询数量。前者往往和系统硬件与底层软件有关，是在构造实际的信息检索系统时必须考虑的；本节主要涉及后者，并假设在第一种含义上的“查询数量”是足够大的。对于第二种含义来说，所谓“满足服务质量”指的就是满足响应时间要求。例如，若一个系统设计的平均响应时间为 3 秒，它的“吞吐率为 20”意味着它每秒钟能接受 20 个查询，并在 3 秒钟左右都能给出查询结果。

显然，这两个指标既独立，也相关。前者是对单个用户查询表现出的性能，后者表示的是系统的整体性能，也称作系统的并发度。因此，对性能问题的讨论可以归结为系统在满足一定响应时间条件下的吞吐率。在 Web 环境下，好的信息资源总是有大量用户同时访问，吞吐率应该是需要考虑的一个重要问题。大型门户网站如此，搜索引擎、数字图书馆也如此。只是前者要简单些，由于主要提供的是 HTTP 服务，吞吐率基本就是在上述第一种含义上的；后两种应用不同，它们要涉及在服务器方比较复杂的操作，需要有更深层次的考虑。

倒排文件是大型信息检索中使用最广泛的文件索引方法。所谓“倒排”表示依据检索属性来列举相关文件，是计算机科学中基本的信息查询方法之一[Knuth,1973]，当前也在搜索引擎和数字图书馆中广为使用。但我们发现，尽管人们围绕倒排文件作了不少研究工作，不少也是旨在提高其性能[Moffat and Zobel,1992]，[Moffat and Zobel,1994]，[Witten, et al.,1994]，[Zobel, et al.,1992]，但尚未见到关于倒排文件（及其实现）性能的一般性讨论，从而使我们依然难以

[1] 不过在其他类型的搜索引擎上，例如问答式搜索引擎（Q&A系统），响应时间略长一些（例如 10 秒左右）似乎是可以接受的。部分原因是因为这样的搜索引擎往往要提供直接能用的结果，而不需要用户的后续参与。

回答下面的问题：给定一个倒排文件（规模，结构参数等）和它将在其上运行的计算机的基本性能（CPU，硬盘指标等），系统能够提供什么样的响应时间和吞吐率？如下我们将围绕这一问题的解决展开讨论。

二、 倒排文件的概念

这一节的内容在信息检索算法的教科书中有不同形式的介绍[Kowalski,1997]。为本章的完整起见，在此给一简要概述。

所谓倒排文件（inverted file），是描述一个词项[2]集合（TERMS）元素和一个文档集合（DOCS）元素对应关系的数据结构，记：

$$DOCS = \{d_1, d_2, ..., d_N\}, \qquad TERMS = \{t_1, t_2, ..., t_M\}$$

当我们以“文档”为出发点时，我们可以讲d_i中包含哪些t_j，或者某一个t_j在d_i文档中出现了多少次。而“倒排文件”直接给出的是一个t_j出现在哪些d_i中，进而还可以有它在某一个d_i中出现在哪些位置（含多少次）。用PL（t_j）表示t_j出现于其中的文档记录的集合，称为对应于t_j的倒排表（inverted list），下面是信息检索研究中常用的几个相关量。

N：文档集合的大小

M：词项集合的大小

$s_j = |PL(t_j)|$：词项t_j所涉及文档的个数

$DF(t_j) = \frac{s_j}{N}$：词项t_j的文档频率

$IDF(t_j) = -\log DF(t_j)$：倒置文档频率；其值越小表示出现频率越高。

$f_{i,j}$：第j个词项t_j在第i个文档d_i中出现的次数

[2] 这里用“词项”而不用“单词项”是考虑到一般性。例如它可以是英文的单词，也可以是中文的字或词。

$T_N = \sum_{i=1}^{N}\sum_{j=1}^{M} f_{i,j}$：系统所有文档分解后包含词项的总量（包括重复，即一个多重集（multi-set））

$TF(t_j) = \frac{\sum_{i=1}^{N} f_{i,j}}{T_N}$：词项$t_j$在所有文档中出现的频度（词频）

$ITF(t_j) = -\log TF(t_j)$：倒置词频；越小表示出现频率越高。

作为数据结构，倒排文件分两部分：第一部分是由不同词项组成的索引，称为词表(vocabulary)，第二部分由每个词项出现过的文档集合构成，称为记录文件(posting file)，每个词项的对应部分称为倒排表，亦称记录表(posting lists)，可以通过词表访问。图 8-3 是一个示意。

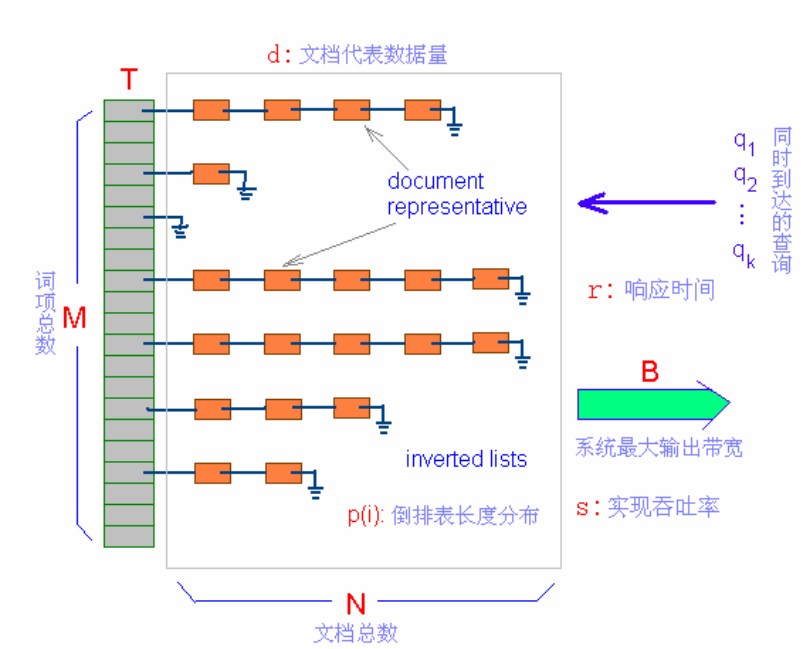


图 8-3 倒排文件结构示意图

图 8-3 的左边是词表，中间是记录文件。对应于词表的每一项，记录文件中

有若干个倒排表，一般长度记为s_j；统计分布为p(i)。至于$PL(t_j)$的每一项，取决于信息检索的方式（例如是否全文检索），内容会有不同，我们在此只用d表示其平均数据量，后面讨论中会适当展开。图 8-3 的右边我们还表示了查询的到达量k、响应时间要求r和系统的最大输出能力B，这是我们讨论性能模型时需要的。

三、 倒排文件的一种性能模型

所谓性能模型，在此就是要给出关于 N，M，p(i)，d，B，r 和 k 的一种关系，从而能够在给定系统内部参数的条件下对其外部行为（吞吐率）进行估计。

需要对 p(i)和 B，以及几个假设进行一下说明。

- p(i)是倒排表长度的统计分布函数，即$M \times p(i)$的长度表示 i 的记录表的个数，i ∈ [0, N]。于是倒排表的平均长度为 $\alpha = \sum_{i=0}^{N} i \times p(i) = \frac{1}{M}\sum_{j=1}^{M} s_j$。

- B 是支持倒排文件运行的下层系统的瓶颈带宽。取决于不同的情况，可能是磁盘的 I/O 带宽，也可能是网络带宽，我们不做区别。这里讨论的模型的思路是根据同时到达的查询量 k，得到一个数据量 D，然后看能否有$\frac{D}{B} \le r$。

- 为简单起见，我们假设查询$q_1, q_2, \ldots, q_k$都是单纯的，即它们都直接属于集合TERMS；同时假设它们是在TERMS上随机、独立的分布。

现在考察k个查询所导致的输出数据量，D。每个查询都可能落到M个词项中的任何一个中，k个查询可能涉及M的任何 1, 2, …, 或者k项，于是对应不同的数据量。如果能算出涉及i项的概率，记作$f_{M,k}(i)$，i = 1, 2, …, k, 则我们就能有

D = 一个倒排表的平均数据量 × k 个并发查询平均涉及的倒排表个数

$$= d \times \sum_{i=0}^{N} i \times p(i) \times \sum_{i=1}^{k} i \times f_{M,k}(i) = d \times \alpha \times \sum_{i=1}^{k} i \times f_{M,k}(i)$$

下面集中考虑$f_{M,k}(i)$。

首先看k个查询随机落在M个词项的所有可能总数，这相当于从集合{1,2,…,k}到{1,2,…,M}的映射的个数，即M^k。

然后对 i = 1, 2, …, k，考察 k 个查询恰好落在 i 个倒排表上的情况，这相当于是考虑集合{1，2，…，k}的 i-划分的个数，再加上这 i 个倒排表可能落在 M 中的任意 i 个上；前者即第二类斯特林数 S(k, i)。注意到查询在不同倒排表之间是可区分的，因此需要考虑的是排列，于是我们有

$$f_{M,k}(i)=\frac{S(k,i)\times P_M^i}{M^k}\text{，}\ \text{i = 1, 2, …, k}$$

$$D=d\times\alpha\times\sum_{i=1}^{k}i\times\frac{S(k,i)\times P_M^i}{M^k}$$

注意到，

$$S(k+1,i)=i\times S(k,i)+S(k,i-1),\qquad S(k,0)=0,\qquad S(k,1)=S(k,k)=1$$

还有，

$$M^k=\sum_{i=0}^{k}S(k,i)\times P_M^i$$

于是，

$$\sum_{i=1}^{k}i\times S(k,i)\times P_M^i=\sum_{i=1}^{k}(S(k+1,i)-S(k,i-1))\times P_M^i$$

$$=\sum_{i=1}^{k}S(k+1,i)\times P_M^i-\sum_{i=1}^{k}S(k,i-1)\times P_M^i$$

$$=\sum_{i=1}^{k+1}S(k+1,i)\times P_M^i-S(k+1,k+1)\times P_M^{k+1}-M\times\sum_{i=1}^{k}S(k,i-1)\times P_{M-1}^{i-1}$$

$$=M^{k+1}-P_M^{k+1}-M(\sum_{i=1}^{k+1}S(k,i-1)\times P_{M-1}^{i-1}-S(k,k)\times P_{M-1}^{k})$$

$$=M^{k+1}-P_M^{k+1}-M\times(M-1)^k+M\times P_{M-1}^k=M^{k+1}-M\times(M-1)^k$$

$$=M\times(M^k-(M-1)^k)$$

这样，

$$D = d \times \alpha \times \frac{M \times (M^k - (M-1)^k)}{M^k} = d \times \alpha \times M \times \left(1 - \left(1 - \frac{1}{M}\right)^k\right) \quad (8\text{-}1)$$

此即为我们得到的一种倒排文件性能基本模型[3]。它直接给出的是k个并发查询所导致的数据量。基于倒排文件的查询处理算法通常不复杂，不是计算密集型的任务。对于大规模倒排文件来说，数据从磁盘移动到内存，或者从内存通过网络送出是主要时间消耗所在，而 $\frac{D}{B}$ 即为响应输出完成所需时间。如果我们设 $\frac{D}{B} \le r$，即 $D \le B \times r$，我们就有可能对D中的各种情况进行讨论，诸如M的影响，p(i)的影响，等等。下面做些讨论，针对数据在磁盘和内存间的移动。

系统对文档信息检索的支持粒度，通常可分为“全文索引”和“非全文索引”两类。非全文索引只需告知哪些文档含有特定的词项，而全文索引则还需要给出该词项在相关文档中出现的位置等信息，多次出现就要多次记录。这样，对于公式（8-1）中的 d 来说，它的大小在全文索引情形下和词项在不同文档中出现的平均次数成比例，即 $\frac{\sum_{j=1}^{M}\sum_{i=1}^{N} f_{i,j}}{N \times M}$，而在非全文索引情形下则基本上是常数（核心信息是一个文档编号），我们记做 c，通常也就是几个字节。

也可以将 $d \times \alpha$ 一并考虑。对于每一个倒排表（对应于一个具体的词项t_j）来说，它的数据量在全文索引情形下正比于$N \times DF(t_j) + T_N \times TF(t_j)$，前一部分是倒排表中文档号和频率占用的长度，后一部分是位置信息占用的长度。因为T_N要远大于N，所以系统中每个词项倒排表的长度主要是由它的词频率TF和数据规模T_N决定的。非全文索引的情况下则只有$N \times DF(t_j)$。在平均情况下，

$$d \times \alpha = \begin{cases} c \times \alpha, & \text{非全文索引} \\ c \times \left(\alpha + \frac{T_N}{M}\right), & \text{全文索引} \end{cases} \quad (8\text{-}2)$$

这里，为简便起见（但无实质性影响），我们也用 c 表示了为记录一个词项在文档中一次出现位置信息所需的数据量。

为理解这个公式，对其中的符号所代表的量的数量级有些具体的概念是有益的。以天网搜索引擎的中文部分为例，$c \approx 10^1, \alpha \approx 10^5, M \approx 5 \times 10^4, N \approx 3 \times 10^7, T_N \approx 10^{10}$。非全文索引的倒排表数

[3] 这个结果也可以用概率论的观点简单推出，留作有兴趣的读者练习。

据量在 10^6数量级，而全文索引是它的若干倍。由此我们也能估计出，作为整个倒排文件的记录文件，即使非全文索引，内存也是放不下的。

另外，α 也可以单独考虑。按照我们的定义，它是倒排文件中倒排表长度的平均值，也就是和词表中词项的文档频率相关的一个量，即 $\alpha = \frac{\sum_{j=1}^{M} N \times DF(t_j)}{M}$。

由于 TF 和 DF 常常是和应用相关的统计量，可以在具体实现之前进行估计，从而使我们有可能在设计倒排表应用时就能根据式（8-1）对查询导致的数据量有个估计。

在标准的倒排文件查询处理算法中，系统要将用户查询中单词项对应的倒排表读到内存中执行集合操作[Frieder, et al.,1999]，因此倒排表的长度将首先影响操作执行的时间。当索引网页量增加时，高频词项的倒排表将急剧膨胀，占用大量 I/O 带宽、内存空间以及 CPU 时间，严重降低系统效率。理想情况下，所有词项的频率应该尽可能低，而且大小相近，使得所有倒排表保持同步增长，系统性能不会因为一部分单词记录表的长度快速增长而受损。实际情况中，词项的频率分布和文件的语言有关，下面我们以英语单词和汉语字符为例分析。

表 8-1 英汉词频统计排序对照

频率	英语单词(E)	汉语字符(H)
0.1%	103	252
0.07%	148	348
0.05%	220	475
0.02%	632	867
0.01%	1285	1215
0.007%	1780	1400
0.005%	2381	1609
0.002%	5474	2210
0.001%	6713	2676

为更好地表示它们的分布情况，取相应的ITF和IDF研究。将所有单词按照它

们的ITF（IDF）值从小到大排列，赋予[0，T]的序号x，作为坐标图的横轴，ITF（IDF）值作为坐标图的纵轴，就得到单词的ITF（IDF）分布图。图的起点越高，同时越平滑，单词的记录表长度趋于平均，越有利于索引更多的文件。我们以[CCF,2004],[EF,2004]统计出的英语单词和汉语单字的频率作为原始数据进行分析，两个统计结果都是按照单词的出现频率从高到低排序，表 8-1 是对它们进行数据抽样的结果，表示降低到某一个频率数值时的单词序号。可以看出，使用的汉字比英语单词要少的多（5,299 对 16006），汉语的高频字比英语的高频单词要多（指使用频率高于万分之一），低频词则反之。在万分之一频率处，两者的数量近似。在十万分之一处，英语有六千多个单词，汉语单字却不到三千个。在根据表 8-1 资料绘制出的ITF分布图 8-4 所示，两条曲线在接近ITF=4 处相交，汉语字符的曲线比英语单词的曲线要陡峭的多。这种特性决定了索引同样数量的单词（T_N相等），汉语字符的倒排表平均长度要大于英语单词的，而且增长更快（随着T_N的增长）。

用户在实际查询中词项的频率决定要读取的记录表长度，为了得到它们和系统吞吐量的关系，必须估计系统运行时查询词项的平均频率。汉语字符在 0.05%频率处的累计频率（Cumulative frequency，大于此频率所有单词出现频率累加）达到 76%，在 0.01%频率处的累计频率则达到 94%，这从侧面反映了在汉语查询中，用户查询词的平均频率要大于 0.01%，估计为 0.05%。根据英语单词的累计频率和实际情况，估计英语中用户查询的单词平均频率为 0.005%。

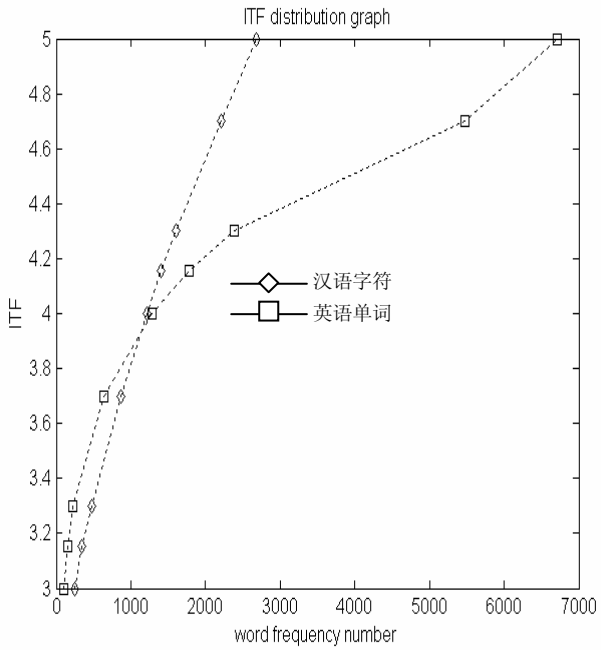


图 8-4 英语单词和汉语字符的 ITF 分布

在本节的最后，我们指出关于 D 的两个近似表达式，只要 M 相对于 k 较大，都应该是很有效的：

$$D \approx d \times \alpha \times M \times (1-(1-k \times \frac{1}{M})) = d \times \alpha \times M \times \frac{k}{M} = d \times \alpha \times k$$

这个结果和直觉如此相符：总数据量是文档单元数据量、倒排表长度和并发查询数的乘积。

$$D \approx d \times \alpha \times M \times (1-(1-k \times \frac{1}{M} + \frac{k(k-1)}{2M^2})) = d \times \alpha \times (k - \frac{k(k-1)}{2M})$$

在有些场合；例如面向领域的主题搜索引擎中，M 可能就没有必要很大（例如我们在有关中学数学网页搜索引擎的构建中，M 为 200 左右），此时 α 也不会很大，于是 k 有可能做得较大。

四、 结合计算机性能指标的考虑

在诸如搜索引擎和数字图书馆等信息检索应用中，主要特点是数据密集型，处理算法比较简单。因此与 CPU、内存等因素相比，I/O 乃是决定系统性能的关键。进而，在网络访问条件下，I/O 实际包含两个阶段，一是磁盘和内存之间，二是内存和网络之间。在不同条件下的，这两个阶段分别都可能成为系统性能瓶颈。下面主要讨论磁盘 I/O。

表 8-2 一些典型磁盘的性能数据

Disk Model	Size (GB)	Average access Time (msec)	RPM	Random IOPS	Internal Transfer Rate (MB/s)	Interface
IBM Ultrastar 36ZX	36	5.4	10,000	119	15-29	Ultra160
Quantum Atlas V	36.7	6.3	10,000	107.5	17-29	Ultra160
Seagate Cheetah X15	18.4	3.9	15,000	169.5	38-47	FC-AL
Seagate Cheetah 73	73.4	5.6	10,000	116	26-40	Ultra160

SCSI 是服务器常用的 I/O 总线，除了有较高的速度外，在 I/O 高负载下消耗 CPU 时间少也是一个重要优点（SCSI 消耗 CPU 时间 5%，IDE 可以达到 60-100%）。现在常用的 Ultra320 SCSI，最高带宽可达 320MBps。表 8-2 是一些高性能磁盘的相关数据，IOPS 项表示磁盘每秒钟可以执行的随机 I/O 操作，它与磁盘平均访问时间、读写的数据块大小、内部传输速率有关。

总的来说，因为磁盘结构中存在机械装置，决定了它的平均响应时间在毫秒级（平均 10ms），每秒钟可以完成的 I/O 操作也很有限（IOPS 平均是 100，即达到平均每秒种 100 次 I/O 访问，这个估计忽略了每次读写的数据块大小产生的影响）。当前单个磁盘的平均数据传输速率在 20-50MBps 之间，并没有完全利用 SCSI 总线的全部带宽，解决这个问题的方法是采用冗余磁盘阵列技术（RAID）。N 个磁盘组成的数组可以使数据传输速率获得接近 N 倍的提升，并且可以改善 I/O 请求的响应时间（同时也增加了 IOPS），不同的配置方案对 I/O 系统这两方面的影响可参考[Scheuermann, et al.,1998]。显然，这种提高 I/O 性能的方法是以增加系统硬件成本为代价的。

在检索系统的两个效率指标中，响应时间作为个体性能指标相对比较容易满足，它也受到系统吞吐量的影响，如果用户查询数量超过了系统可能的负载，会造成查询响应延迟增大。因此，吞吐量对系统设计和运行更为重要。根据前面论述，我们将查询词项的平均频率和 I/O 性能作为查询效率的决定因素，用它们估计系统的吞吐量和数据规模的关系。用户的大部分查询中的词项数量比较少，查询一个主题时用 2-3 个单词就可以描述，查询文章的题目时可能有 10 个词项以上。不妨设 Lq 表示用户查询中的词项个数，估计平均 Lq 等于 5。得出如下不等式：

$$(T_{latency} + T_N \times \frac{TF}{IO_{bandwidth}}) \times L_q \times m \le 1$$

$T_{latency}$: 磁盘访问平均延迟时间（s）

$IO_{Bandwith}$: I/O可用带宽（Bps）

m：吞吐量（个）

在这个不等式中，对词项的倒排表长度只考虑主要部分 $(T_N \times TF)$，将I/O时间估计为磁盘访问平均延迟时间和数据传输时间。假设将每个倒排表读入内存只需一次I/O，花费时间可以估计为 $(T_{latency} + T_N \times \frac{TF}{IO_{bandwidth}})$，每次读取倒排表的时间

乘 $L_q \times m$ 必定不大于 1 秒。当系统的I/O性能（$T_{latency}$、$IO_{Bandwith}$）和TF确定下来后，我们就得到T_N与m之间的反比关系。

更简化的模型是忽略数据块大小不同造成的传输时间变化，利用前面得出的一个磁盘的 IOPS=100，可以计算出在不考虑资料缓存情况下，系统平均每秒钟处理查询的上限 m=IOPS/Lq=20。根据磁盘的可用带宽大约是 20MBps，得出每个查询的 I/O 应不大于 1MB，也就是满足如下条件：

$$T_N \times TF \times L_q \le 1MB$$

代入以上得出的估计参数，有如下结论：

- 对汉语字符： $T_N \le 400MB$ （TF=0.05%，Lq=5 ）
- 对英语单词： $T_N \le 4GB$ （TF=0.005%，Lq=5 ）

由于在这个条件下，每个词项的倒排表长度达到 200KB，对 20MBps 的硬盘读取时间大约是 10ms，忽略它不太合适，因此实际中吞吐量要低于每秒钟 20 个。

依据上面结果，可以估计出能够索引的数据量。汉语的一个字符占两个字节，如果对汉语字符建索引，要维持每秒 20 个查询的系统吞吐量，最多只能索引大约 800MB 的文本数据库。英语的一个单词平均占用字节 6-8 个（包括空格符），同样情况下可以索引 24-32GB 的英语文本。由于汉语字符的 ITF 分布规律，对单个字符建索引难以提高检索系统的规模，解决的方法是对汉语文本做切词处理，按照词组建索引，有效地改善单词项的频率分布情况，能够取得和英语单词同样的结果。这种方法比用冗余磁盘阵列提高 I/O 性能的代价要小得多，效果也更显著。从成本上考虑，一台机器索引的数据量不应低于它内存的 10 倍，即在 10GB 以上，实际中要根据上面的结论在硬件设备选用、数据规模、响应时间、吞吐量之间做折中选择。

第三节 混合索引技术

一、 引言

大量的统计研究表明，搜索引擎用户输入查询长度平均较短，并且很少使用系统提供的查询操作符。这种情况下，检索结果排序考虑用户输入的查询词之间的短语关系或者位置邻近关系，对提高检索结果的效果十分重要。通过丰富倒排

文件的数据结构内容，这样的关系有可能在预处理阶段充分地表达出来，从而为检索服务算法的运行提供数据基础。文献[Anh and Moffat,2002]中介绍倒排索引的几种常见级别和索引的压缩技术，其中词级（Word-Level）的倒排索引记录索引词在文档中出现的每个位置信息，检索时通过这些位置信息来执行短语或邻近关系的检查。词级索引是倒排文件实现短语或邻近查询的一般组织方式。文献[Sadakane and Imai,2001]提出了邻近查询的通用算法，给出了最短邻近距离定义下的最优解算法。不过这个算法开销太大，在实际系统中并不实用。文献[Brin and Page,1998]介绍了邻近查询的一种近似实现方法，具有更好的检索效率。文献[Bahle, et al.,2002]，[Williams, et al.,1999] 提出了一种新的短语索引技术，对倒排索引词典里每个索引词，按其后续词组织倒排数据项，即为每个索引词与其后继的索引词建立辅助倒排索引。这一方法可以提高短语查询的效率，但它的缺点是基本索引词典和后继词词典分开存储，在查询过程中需要增加一次对后续词的词典数据读取，一定程度上抵消了对短语查询效率的提高。在中文信息检索领域，什么是最佳的索引单位一直是困扰中文全文检索的一个问题。研究表明使用中文分词，按词索引结合二元组(Bi-gram)索引是检索效率和效果较优的索引方式。这实际上是一种混合索引，不过对大规模文档集合，二元组索引的倒排索引词典膨胀迅速，索引文件中包含大量无用内容，对检索效率也带来有负面影响。在天网搜索引擎的实践中，针对上述各种技术的优劣，采用了一种基于未登录词自动识别技术的混合索引方法。实践表明，这一方法可以有效提高搜索引擎检索效率。

二、 混合索引原理

混合索引是在建立倒排索引过程中的一种索引词选择方法与技术。索引词的选择是检索系统实现的一个重要环节。现代搜索引擎普遍采用全文索引技术，把网页文档中提取出来的所有词语都选择参与索引。在理想情况下，索引词应该是表达文档内容的语义单位，对应着语言学里的**词汇词**的概念，它是专门表示含义，而其实际意义无法由组合成分相加得到的最小语言单位。但对于自动文档索引过程，识别文档中的词汇词，例如短语十分困难，因此通常选取语法意义上的最小语言单位为索引词。对英文文档，这一过程相对容易。对中文网页文档的索引过程，词间没有空格自然分隔，为索引词的选择带来了新问题。文献[Nie, et al.,2000]研究表明相对于按字索引，基于自动分词的索引方式具有良好的检索效果。对搜索引擎系统，更出于检索效率的因素，通常通过自动分词来选择索引词。在文档索引过程中，先通过中文自动分词程序的处理，把文档正文分割成为独立的分词单位，然后在这些分词单位基础上选择索引词。分词单位是指具有确定语义或语法功能的基本单位，通常被直接选作索引词。

目前，中文自动分词的成熟技术都是基于分词词典的机械型分词方法[刘开

瑛,2000]，这一方法的主要难点在于分词歧义处理和未登录词的识别，其分词词典规模是制约分词精度的重要因素。我们使用北京大学计算语言所的中文自动分词软件，该分词软件的基本词典规模为 6 万词。中文自动分词软件使用的词典选词十分严格，随意加入新词将影响分词软件的歧义处理过程，导致分词精度下降。文献[Stokoe, et al.,2003] 研究表明对英文文档数据，信息检索系统加入词歧义处理可以提高检索精度。对于中文文本，这一问题更加突出，任意扩大分词词典规模而忽略对分词精度的影响会对检索系统的检索效果带来负作用。同时，对处理 Web 数据，分词基本词典的规模是远远不够的。一方面，网上大量的常用词、新出现词、专业词汇等没有被收录，从而会被分词程序切分成分离的单字，每个单字被分别索引。这样的词在检索时会按短语查询执行，虽然可以检索出基本相同的结果集合，但执行过程需要从倒排文件中读取多个索引词的倒排项数据，然后执行位置检查，这大大降低了系统的检索效率。另一方面，分词词典中的分词单位一般很短，常用的短语也会被分词程序切分开，同样这一方式在对短语的查询上效率很低。如果分词程序使用的词典中分词单位过长，切分出短语，又可能使得组成短语的词无法被检索，导致检索系统召回率下降。如何扩大分词词典的规模，使得分词程序能够切分出更多的词，甚至短语，同时又不降低分词程序的分词精度，不降低检索效果是中文搜索引擎检索系统面临的一个基本问题。

天网检索系统采用混合索引技术解决上述问题[彭波,2004a]。这一技术首先用统计方法对索引文档中的未登录词进行识别，把识别出的新词（不被基本词典收录的字串）放入一个扩展词典。这可以有效扩大词典规模，但由于统计方法识别未登录词存在相当的错误率，扩展词典里面也存在不少被错误识别的词。系统目前控制扩展词典规模在 50 万词语左右。扩展词典在保存时，把识别的新词词条使用基本词典进行分词，保存切分开的基本词序列。

在索引创建过程中，对文档正文进行两趟分词。首先是基于基本分词词典的常规中文分词，采用北京大学计算语言所的分词软件。分词执行中包括复杂的歧义处理过程。第二趟再对基本分词结果使用基于扩展词典的分词，这一分词过程的最小单位是基本词典里的词条，采用正向最大匹配分词算法。两次分词的结果都被选择作为索引词，在倒排文件的创建中都被放入倒排索引词典，这一方法即混合索引。例如：基本词典有“国家”“图书馆”两个基本词条，无“国家图书馆”；系统通过识别，发现“国家图书馆”极为可能是一个词语，于是把它加入到扩展词典。对文档中出现的“…国家图书馆…”字串，第一趟基本分词步骤把它切分为“国家”和“图书馆”两个基本词条，第二趟扩展分词再把它切分为“国家图书馆”，最终索引词包括“国家”“图书馆”和“/2 国家图书馆”这样三个单位。扩展分词结果使用转义符“/”标识，转义符后紧接扩展词包含的基本分词词条个数，用于查询时位置关系的计算。

混合索引的检索过程对用户输入的查询串执行同样的两趟分词。首先是基本

分词，第二趟再对基本分词结果使用扩展分词。根据扩展分词结果词条包含的基本分词词条个数，标记被扩展分词结果覆盖的基本词条，它们在查询执行过程中无需处理。如上例，当用户输入查询“国家图书馆”，经过两趟分词，被切分为：“国家”“图书馆”，“/2 国家图书馆”。其中前两个基本词条被第三个扩展词条覆盖，查询执行中只需直接读取索引词“/2 国家图书馆”对应的倒排项数据，即可完成查询执行过程。相对于分别读取“国家”和“图书馆”的倒排项数据，然后按其中的位置数据验证短语关系的方法，使用混合索引大大提高了检索效率。在混合索引条件下，当用户查询“图书馆”时，检索将按正常的查询过程执行，混合索引也不会降低系统的查全率。

与文献[Bahle, et al.,2002] 的短语索引相比，混合索引使用统一的倒排索引词典，没有额外的二级索引词典访问开销；并且混合索引不限制扩展词条为两个基本词条长，可以索引更长的短语，更加灵活。与词索引+Bi-gram 索引相比，混合索引使用了未登录词的识别技术，可以有效控制倒排索引词典规模，避免了 Bi-gram 词典膨胀的问题。混合索引也是索引结构的规模与检索效率间的一种折衷。一方面，文档中的词被重复索引导致索引结构增大，占用更多的存储资源，另一方面，这些增加的索引，使得更多可能形成词语或短语的字串被索引，可以大大提高对它们的检索效率。实际环境中，系统存储开销相对于检索效率不是那么重要，所以混合索引成为一项可用的技术。

三、 混合索引实现

混合索引的实现主要包括未登录词识别、扩展词典组织和分词两个部分。除了两趟分词和扩展词对基本词条的覆盖处理外，索引系统的创建过程和检索过程同一般的索引实现没有区别。

1. 未登录词的识别

目前，从语料库中自动识别或者学习词典未登录新词，特别是面向领域的专业词汇以及人名、地名、机构名等专有名词等方面，已经有了大量的研究工作和实用的技术。对文本数据常规的未登录词识别算法[刘开瑛,2000]一般包括如下步骤：

1) 提取 n 元组：使用基本词典，将文本进行部分分词，从部分分词结果中提取 n 元组，即包含 n 个相邻基本词条的字串。一般 n 元组的规模很大，不利于后续处理，常通过设定 n 元组的提取规则加以限制。例如可以设置如下规则：对单字，只有连续出现的单字才能生成 n 元组；形成新词的 n 元组必须包含一个单字等。
2) 噪声剔除：删除那些包含低构词能力字的 n 元组，例如常见的助词“的”

“得”，介词“在”“把”等。对于这些字，由于数量少，可以人工收集包含它们的词。

3) 剔除 n 元重叠：把那些在 n 取不同值情况下重复被提取的 n 元组剔除。

4) 最后剩下的 n 元组按出现频次降序排列，为识别结果。

未登录词可以从网页文档集合与用户查询日志两种基本语料数据中得到。网页文档集合规模很大，用简单的 n 元组提取方式执行完整的未登录词识别算法会产生数量巨大的候选元组，难以处理。实际实现时，通过加入一些提取规则限制提取范围，使得在识别效率和识别效果间取得平衡。第一个规则的基本出发点是，常被用户查询的词或短语应该在文档集合中比较重要，而重要的词往往会在网页中的一些特殊位置和标签中间出现，例如标题，H 标签，加重的文字等。网页中这些文字首先被选择执行 n 元组提取。第二个规则，网页和普通文本不同，网页中通常会有大量已经被自然分隔的较短的字串，分隔包括标点，网页标签，例如导航链接上的文字，选择框中的文字，表格等等。这些较短的字串极可能就是一个词，被选取执行 n 元提取。第三个规则是从网页原文摘要文字中提取 n 元组。天网搜索引擎使用了文档模型技术[Zhang, et al.,2004] 对搜集系统得到的网页进行预处理，提取了网页正文的摘要。摘要文本是正文中最重要的内容，而且长度相对于正文全文小很多。通过上述三个规则，可以有效控制未登录词识别执行的数据规模，提高识别效率，而且提取的是整个文档集合中重要的文本内容，对识别效果没有太大损失。

从用户查询日志中识别未登录词，除了使用基本的识别方法外，还可以利用查询日志数据的特点。例如，直接从网页文档语料库中学习新词需要较复杂的处理算法，处理数据量大，而日志文件中用户的查询词通常比较简短，几乎没有完整的句子，只需做一些简单的处理，即可方便地学习新词。而且，如果用户查询词在词典内没有收录，很有可能就是潜在的新词，所以学习新词的准确率也比较高。搜索引擎日志中用户查询词在频度分布上是高度集中的。通过对天网用户查询日志的统计分析表明，前 5%的高频词占据了 64%以上的总查询次数；前 20%的查询词占据了 83%以上的总查询次数。我们可以利用查询词的频度分布特性来大量地减少 n 元组提取过程中处理数据量。而且从提高系统检索效率的角度，对长期保持高频度的用户查询可以直接加入到扩展词典里，越过识别过程。

2. 扩展词典组织与分词

识别出的新词保存在扩展词典中。基于扩展词典的分词实际上是一个对基本分词结果序列在扩展词典上的最长匹配查找过程，即输入基本分词结果序列，找到序列中在扩展词典里的所有最长的匹配词条。为了高效实现对分词的支持，先把每个新词字串用基本词典进行分词，转换为一个基本词条的序列。再使用一个

散列查找表把基本词条的字串转换为连续的整数编码。扩展词典保存每个词条中基本词条的整数编码，这相当于一个由数字组成的 n 元组的集合：$D=\{(t_1,t_2,...,t_n)\}$ $2\le n\le 6$。

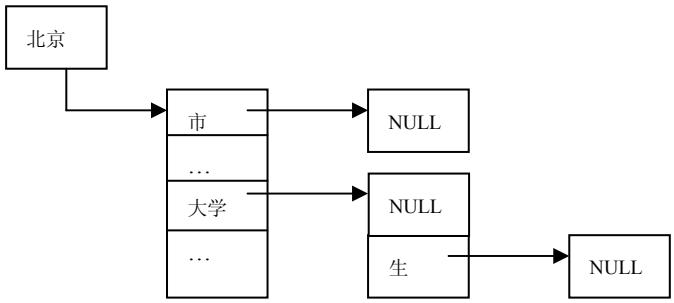


图 8-5 扩展词典树结构示例

系统中 n 取 2 到 6，式中 t 为基本词条编码，D 表示扩展词典。在扩展词典保存时，各元组按第 k（1≤k≤6）个元素的 t 值进行基数排序。再把排序结果每一层上相同 t 值的节点合并，转换为树结构保存。示例如图 8-5 所示。

输入：基本分词结果词条序列(t1,t2,t3….ti..)

输出：最长匹配扩展词条

算法：

```
1.    init_scoreboard();   初始化匹配任务表
2.    while(ti!=EOF){
3.        code = get_code(ti);   从编码散列表中取得 ti 的编码
4.        for each task in scoreboard {
5.            ret = search_token(code); 检测匹配任务追加一个词，是否结束?
6.            if (ret==NULL){
7.                clear_task
8.                add_hit                    ;得到一个匹配
9.            }
10.           else   update_task          ;根据检测结果更新匹配任务状态
11.       }
12.       check_hit            检查匹配结果，输出。
13.   }
```

图 8-6 扩展词典匹配查找算法

在图 8-5 中，树节点保存实际上是基本词条的编码，且每一层的兄弟节点按节点内编码数值有序。基于这样数据结构的扩展词典匹配查找算法如图 8-6 所示。

第四节 倒排文件缓存机制

一、 引言

缓存技术是提高系统性能和可扩展性的一种重要手段，在计算机各个应用领域都有广泛的应用。如何有效的在搜索引擎检索服务系统中使用缓存技术也是近年来学术界广泛关注的问题。

缓存技术的有效性建立在被缓存对象访问序列存在的局部性特征上。与操作系统内存管理、数据库系统和 Web 代理缓存这些领域大量的研究相比，搜索引擎检索系统上的缓存研究相对较少。它们之间有共性，但由于被缓存对象特征和对象访问模式的差异，又各自具有自己的特点。搜索引擎检索系统中通常被研究的缓存对象可分为三种，即查询结果、布尔操作的中间结果、以及倒排文件。文献[Xie and O'Hallaron,2002],[Wang, et al.,2001]详细分析了搜索引擎用户查询日志，发现用户查询具有很强的局部性，提出了缓存查询结果的可行性。在文献[Wang, et al.,2001],[Markatos,2001],[Saraiva, et al.,2001]中，进一步研究了缓存替换算法、缓存大小等因素对系统性能的影响。天网在[Wang, et al.,2001]的基础上实现了查询结果缓存，有效的提高了系统性能。文献[Chidlovskii, et al.,1999]提出语义缓存，把布尔查询的中间结果作为缓存对象，并利用查询结果间的语义关系加速后续查询的执行。这种方法可以充分利用不同查询之间的相关性提高缓存命中率，缺点是限制在布尔查询上，可能影响结果相关性排序。第三种是倒排文件的缓存，用户查询经过查询器执行，转换为对倒排文件数据的访问序列，这些数据也可以作为缓存对象。[Jonsson, et al.,1998]研究了 IR 背景下用户交互式查询的倒排文件缓存与查询执行结合的方法，[Saraiva, et al.,2001]研究了一个实际搜索引擎（TodoBR）中的倒排文件缓存对系统效率的影响。

下面我们基于天网的实际运行数据，重点讨论倒排文件缓存的优化设计。与[Saraiva, et al.,2001]相比一个差异在于它使用的是过滤向量空间模型查询处理技术[Persin, et al.,1996]，而天网的查询处理考虑查询词位置邻近关系，使用带位置数据的倒排索引，并使用索引压缩和块随机访问技术提高性能[Navarro, et al.,2000]。这种查询处理技术的不同，导致所产生的访问倒排文件数据序列性质的差异。文献中对倒排文件缓存的研究，基本以固定大小的页面为单位，忽略了

倒排文件访问数据是变长这一特点；并且缺乏替换策略、数据组织对缓存效率影响的分析。本节就如下问题展开讨论。

1) 缓存性能评估的指标如何选取？
2) 倒排文件缓存与操作系统的文件缓存相比是否有优势？
3) 倒排文件的数据组织方式对缓存效率及系统性能的影响如何？

二、 倒排文件缓存

1. 体系结构

天网检索系统采用分布式体系结构，按文档划分的方式组织数据到多个索引服务节点，它们独立的并行执行用户查询，把各自检索结果提交给查询服务器汇总返回给用户。各级缓存的位置如图 8-7。

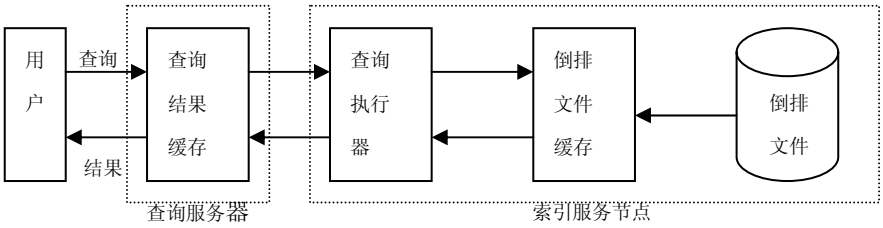


图 8-7 搜索引擎检索系统缓存结构

倒排文件缓存位于索引服务节点上，对查询执行器在执行用户查询过程中访问的倒排文件数据进行缓存。大量统计研究表明用户查询词序列具有良好的局部性，可以预期查询执行器发出的读取这些查询词倒排数据序列也具有同样的性质，这是人们研究倒排文件缓存的基本出发点。

在搜索引擎应用环境下，用户提交的查询中包含查询词个数通常很少，而词间的位置邻近关系对结果排序十分重要。与[Saraiva, et al.,2001]的过滤向量空间模型查询处理技术不同，天网使用带位置数据的全文倒排索引，对多个词的用户查询计算邻近权值。查询执行器访问倒排文件的数据分为两类，一是查询词对应的倒排表中的文档编号和文档内权值数据，称为文档数据；另一部分是查询词对应的出现在每篇文档中的位置数据，称为位置数据。执行过程中，各个查询词按倒置文档频率降序处理，先读取文档数据，执行文档集合的布尔运算（通常搜索引擎默认为 AND），得到一个小的结果集合，同时使用文档内权值数据计算每个结果文档对查询的相关性权值；再读取对应的位置数据，对结果集合进行邻近权值

排序。通过索引压缩技术，再结合对高频词使用位图记录文档数据，可以有效控制文档数据的长度。一般情况下，位置数据总量是文档数据量的 3～4 倍，查询执行中不必全部读取，通过随机访问的倒排索引组织技术[Navarro, et al.,2000]，可以减少数据读取量。

天网查询执行器读取倒排文件的数据序列包括文档数据序列和位置数据序列。每次读取的数据长度不固定，但基本可以在一次磁盘系统 I/O 操作下完成（不考虑文件系统开销和数据组织碎片的条件下），这样的数据序列总称为 I/O 序列，它的项数代表磁盘系统执行的 I/O 次数。另外，以固定页面为单位，可以把 I/O 序列转换为页面访问序列，称为 PAGE 序列，它的项数对应磁盘系统实际读取数据总量，这可以衡量磁盘系统的带宽使用效率。搜索引擎的查询执行属于磁盘密集型应用，和数据库系统的事务处理应用类似，瓶颈在于磁盘系统每秒能够执行 I/O 次数的能力，即其 IOPS 参数；但平均每次访问的数据量比数据库事务处理要大，磁盘系统带宽参数也不能忽略。从实际系统的性能角度分析，与考察 PAGE 序列相比，考察 I/O 序列对缓存系统的性能评估更为有意义。

2. 负载数据

我们采用踪迹驱动（trace driven）的方法来研究倒排文件缓存的性质。采用天网 2002 年 11 月的用户查询日志，在北大燕穹提供的数据产品中的编号为 YQ-QUERYLOG.021203[InfoMall,2004]。查询日志记录了用户查询是否被天网的查询结果缓存命中。把被结果缓存命中的查询剔除，就得到实际到达索引服务节点图 8-7 所示的查询序列。

表 8-3 数据集基本统计信息

名称	数值	名称	数值
用户总查询数	7,341,383	I/O 序列长度	1,887,198
结果缓存未命中个数	3,522,968	I/O 序列唯一对象数	112,145
文档总数	2,603,035	PAGE 序列长度	20,808,025
文档数据原始大小	30.18 GB	PAGE 序列唯一对象数	965,929
倒排文件大小	5.77 GB		

同时，还需要形成一个该查询序列所针对的文档集合。为此，我们从天网搜集的网页集合中随机抽取一批网页，建立索引，修改查询程序，把访问倒排文件的每次操作记录到日志文件，内容包括访问数据的索引词编号、文件偏移、数据长度、访问类型。其中访问类型采用位置数据块的编号。取经过过滤的查询序列中连续的 10 万个查询，送入查询程序执行，记录得到此查询序列在天网查询执行

器下的倒排文件访问序列。根据访问日志中的索引词编号、访问类型与数据长度可以得到 I/O 序列，根据文件偏移和数据长度和指定的页面大小(例如 4KB)，可以得到 PAGE 序列。

最后，为了更有效的数据处理，把两个序列中的对象标识（I/O 序列中是索引词编号与访问类型，PAGE 序列是页面编号）转换为从 0 开始的连续整数。数据集的统计信息如表 8-3 所示。

三、 负载特性

这一部分分析负载数据的性质，讨论它们对倒排文件缓存和缓存替换算法的影响。

1. I/O 序列对象大小

I/O 序列中的对象大小不同，其中由位置数据访问产生的部分是固定长度(32KB)，而对文档数据访问产生的对象大小分布很不均匀，以 4KB 为单位对其分布统计如图 8-8 所示。其中值为 7.59KB，79%的请求对象长度在 64KB 以下，同时也有少数较大的数据访问。有效的缓存替换算法需要考虑对象的大小。对大量的小数据对象优先缓存，可以提高缓存的命中率，而对大对象优先缓存可以提高缓存的字节命中率。因为 I/O 序列反映的是系统 I/O 请求的次数，所以缓存命中率更为重要。在考虑缓存替换策略时，偏向小对象的方法预计可以获得更好的性能。

2. 序列中对象的频度分布

对象被访问的频率是缓存设计的一个重要因素。如果序列中对象访问频率分布非常不均匀，则需要考虑两个问题，一是缓存少数高频对象可以提高性能，另一个是不区分出大量低频对象将降低性能。实际上对象的访问频率和访问的时间局部性是相关的，可以推导出高频的访问对象也会具有较高的访问时间局部性[Jin and Bestavros,2000]。

I/O 序列和 PAGE 序列的访问频度对其访问频度的序号的分布如图 8-9。在计算数据点序号时，对同频度的数据使用最后一个数据点的序号，这样图中曲线平滑，没有尾部数据点堆积，便于分析和比较。总体看，两个序列都存在访问频度分布不均匀的现象，但和通常的 Zipf's 分布相比，这种差异还算是很平缓，可以预期频率是倒排文件缓存替换算法应该考虑的一个因素。但只考虑频率的替换算法，如 LFU，效果不会很好。两者间，I/O 序列的频率特性比 PAGE 序列更有利于缓存应用。

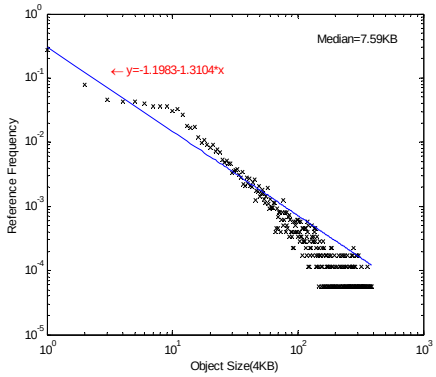


图 8-8 文档数据访问对象大小分布

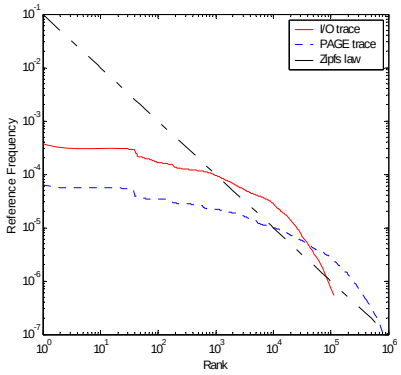


图 8-9 I/O 与 PAGE 序列序号-频度分布

3. 序列中对象的时间间隔分布

序列的时间局部性可以从序列中对同一个对象的两次连续访问的时间间隔分布来考察。使用访问在序列中的位置间隔，而不使用绝对时间，可以屏蔽用户查询密度在各个时间段内的周期性对分析的影响[Jin and Bestavros,2000]。

I/O 序列和 PAGE 序列的时间间隔分布如图 8-10。由于直接的时间间隔分布十分散乱，图中的处理是把距离数据按 2000 为单位分组，表现的是各组的频度。可以看到对数坐标下，序列的时间间隔分布接近直线，说明具有良好的时间局部性。I/O 序列的斜率为 1.039，PAGE 序列为 0.764，表明在同样的缓存大小比例下，I/O 序列可以预期得到比 PAGE 序列更高的缓存命中率。较强的时间局部性有利于缓存设计，对象访问的时新性(freshness)是替换算法需要考虑的一个重要因素。

4. 序列的重复模式

序列的空间局部性是指序列中固定模式的重复，这可以通过原始序列和随机排列处理后的序列中的唯一的定长串的个数来说明[Almeida, et al.,1996]。空间局部性也是缓存设计需要考虑的因素。

取 I/O 序列和 PAGE 序列前 10 万个数据，处理得到其中长度从 1 到 9 的连续串，统计唯一串的个数。再把序列进行随机重排，重复统计。得到序列中指定长度的唯一串的个数如图 8-11。随机排列破坏了序列中的重复模式，即破坏了序列的空间局部特性。图中随着串长度的增加，唯一串的个数也增加。随机排列的序列增加速度最快，其空间局部性最差；I/O 序列增加得最为平缓，其空间局部性

较强，PAGE 序列次之。

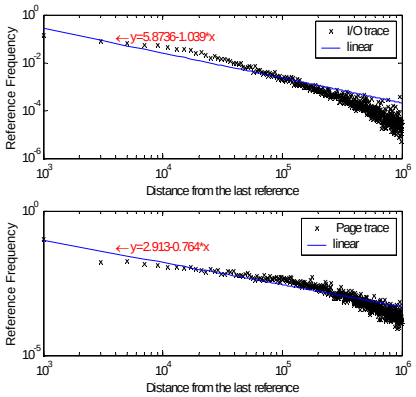


图 8-10 I/O 与 PAGE 序列时间间隔分布

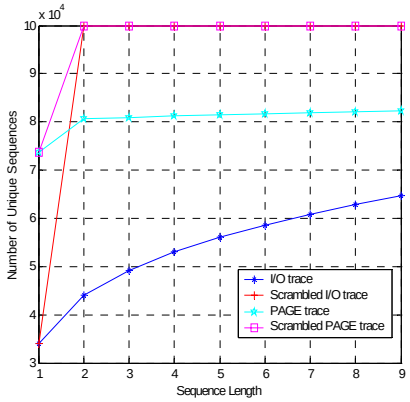


图 8-11 I/O 和 PAGE 序列中唯一模式串

四、 缓存策略的选择

我们注意到，现代操作系统的文件系统通常都提供 I/O 数据的缓存功能，通常以页为单位。也就是说，如果在应用层不安排缓存，应用中发生的 I/O 操作物理上也都可能在内存发生。文献[彭波,2004b]通过四组缓存仿真实验，验证了倒排文件缓存经过优化设计，可以比文件系统缓存性能更好。具体的方法可以通过缓存变长的 IO 序列对象，选择性能更好的 GD-SIZE1 替换算法，从优化磁盘系统 I/O 次数的角度来提高系统性能；也可以通过选取大的页面作为访问倒排文件的单位，从优化磁盘系统带宽利用率的角度提高系统性能。最后按页面对齐的方式组织倒排文件可以进一步优化缓存和系统性能。而这一组织方式下，可以直接把倒排文件页面存放在磁盘设备上，通过直接的设备访问接口，越过文件系统 read/write 调用来访问倒排文件数据。再加上优化设计的倒排文件缓存，使得索引服务的性能得到进一步的提高。

第五节 本章小结

本章第一节通过分析天网检索子系统的设计与实现，概述了检索系统所要关心的若干基本技术。检索系统的设计目标围绕检索效果和检索效率两个方面展开。系统通过一个集成框架把多种技术融合到一起，包括中文文本自动分类技术、中文信息提取技术等，以求能不断提高检索质量。天网检索分布式系统构架有效解

决了系统可扩展性问题，它是高效检索系统实现的物理基础。而检索系统在索引创建和检索上的相关实现技术，则是高效检索系统实现的保证。

第二节着重从 I/O 数据量的角度分析了影响倒排文件查询效率的各种因素，以及提高系统效率的一些技术，试图定量化地描述数据规模和查询效率之间关系。所得到的结论可以指导在计算机系统硬件、数据规模、系统性能（响应时间、吞吐量）之间做预测和评价，作为评估信息检索系统设计的一个依据。我们同时指出，所讨论的模型没有涉及其他一些优化技术，特别是压缩技术和缓存技术。这样一些优化技术实际上能够在相当程度上有效提高系统吞吐率。在这个意义上，依据该节模型所得的结论会比较偏于保守。

第三节在天网实践的基础上，提出了一种基于自动识别新词技术的混合索引技术。与其它几种常用索引词选择技术相比，这一技术能够有效提高搜索引擎检索效率，同时不会导致检索效果下降。

第四节研究了搜索引擎中倒排文件缓存技术。通过分析数据访问序列的局部性特征，以及基于真实数据的缓存仿真实验[彭波,2004b]，探讨了倒排文件缓存优化设计中的性能指标选择问题、替换算法、页面大小和倒排文件组织方式等对缓存性能的影响，得到如下结论：1) 通过缓存变长的 IO 序列对象，采用 GD-SIZE1 替换算法，可以明显减少磁盘系统 I/O 访问的次数；2) 通过按页面对齐方式组织倒排文件，选取大的页面作为访问倒排文件的单位，可以使磁盘系统带宽利用率得到优化。

第九章 用户行为的特征及缓存的应用

通过对大量用户行为的统计分析，我们发现搜索引擎用户输入的查询词语和查询过程中所点击到的网页 URL 均表现出明显的时间局部性；而且用户查询的分布符合幂函数特征并具有良好的自相似性。这些规律可能有多方面的价值，其中之一就是用来指导查询缓存的设计。而搜索引擎所访问数据的特殊性使得我们在缓存设计中有必要重新考察相关的细节。因此，本章除通过数据具体展示上述规律外，作为应用，还比较了查询缓存设计中 FIFO，LRU 及带衰减的 LFU 等 3 种缓存替换策略。最后，本章还讨论了基于用户行为考察海量网页信息的分布特征，并利用 URL 的入度、镜像度、目录深度等网页参数与用户行为反馈后的相关度的方差分析, 阐明其对优化搜索引擎系统排序算法的启示。

在对一些观察数据（或者测试数据）进行统计分析，以得到关于这些数据的某些总体性质的时候，问题常常可以如下抽象：给定一个集合，$S=\{e_1,e_2,...,e_n\}$，假设对其中的元素总共进行了 N 次观察（或者说这些元素在实验中总共出现了 N 次)；这 N 次出现分布到具体的元素上就形成了一个整数序列：$C=<c_1,c_2,...,c_n>$， 其中$0\le c_i\le N,\ \sum_{i=1}^{n}c_i=N$。将 C 的元素降序排列，可得到$\overline{C}=<\bar{c}_1,\bar{c}_2,...,\bar{c}_m,\bar{c}_{m+1},...,\bar{c}_n>$，其中 $\bar{c}_m>0,\bar{c}_{m+1}=...=\bar{c}_n=0$，亦即$\sum_{i=1}^{m}\bar{c}_i=\sum_{i=1}^{n}\bar{c}_i=N$。

根据不同的需求，我们可能关心如下几个统计量：

1) 集合 S 中元素出现频度（相当于概率密度）分布的递减情况，亦称为频度频级分布。

$$p_i=\frac{\bar{c}_i}{N},\qquad i=1,2,...n$$

2) 集合 S 中元素出现频度按降序排列后的前缀累积分布情况(相当于概率分布函数)

$$P_i = \frac{\sum_{j=1}^{i} \bar{c}_i}{N}, \qquad i = 1,2,...n$$

3) 有时为了需要，对（2）中的 i 做变换：$i = n \cdot x,$ 即有（相当于规格化）

$$y = \frac{\sum_{j=1}^{n \cdot x} c_i}{N}, \qquad x = \frac{1}{n}, \frac{2}{n}, ...1$$

在直角坐标系内画出有序对(x,y)的散点图，就可以很容易得到“A%的元素覆盖了 B%的观察”之类的结论。这样的统计分析方法在这一章会多次用到。

本章的主要内容安排如下，第一节简要介绍了一般搜索引擎系统中用户的查询日志与点击日志所记录的基本信息;第二节对近两个月的天网日志进行了分析，得到了用户行为的一些重要的统计特征；基于这些特征分析，第三节讨论了搜索引擎系统使用查询缓存的必要性，并比较了几种数据替换策略；第四节基于用户的行为对海量 Web 信息的分布特征进行了分析。

第一节 用户查询与点击日志

一般的搜索引擎系统主要维护了两类信息，一类是和搜集到的 Web 页面相关的信息，另一类是在服务过程中收集到的用户行为信息（记录在所谓的日志文件中）。前者指的是机器人从网上抓取的网页经过分析器分析处理后得到的信息，主要包括网页所包含的关键词、摘要信息、元信息（如网页作者、长度、修改时间等）以及 URL 超链信息，这类信息通常是作为输出信息给用户看的。而后一类信息主要包括用户输入的查询项，查询时间，用户的 IP 地址，用户在输出页面中所点击感兴趣页面的 URL。这两类信息的数据量都很大，在天网系统中它们都已超过千万量级。

天网日志文件分为**用户查询日志**[1]和**用户点击日志**。其中用户查询日志是在用户提交查询请求时记录的，它记录了用户查询时提交的关键词、提交时间、用户 IP、页号（查询结果分页显示，每页显示 10 个查询结果，用户首次查询页号为 1，

[1] 我们自 2000 年以来开始备份天网查询日志，每月整理一次，形成格式规整的纯文本文件，欢迎读者选用。

用户翻页时的页号即为用户选择的结果页面号)、是否在缓存中命中等信息。用户查询日志的一个简单的记录格式为：

```
Fri Mar 21 00:00:02 2003                    // 提交时间
218.24.100.77                               // 用户 IP
Database                                    // 是否在缓存中命中
老歌                                        // 查询词
3                                           // 页号
```

用户点击日志是用户浏览查询结果时点击页面时记录的，它记录了用户点击页面的时间、点击页面的 URL、用户 IP、点击页面的序号（该页面在查询结果中的位置)、该点击对应的查询词等信息。用户点击日志的一个简单的记录格式为：

```
Fri Mar 21 00:00:02 2003                                    // 点击时间
202.206.102.169                                             // 用户 IP
虫儿飞                                                      // 查询词
http://sports.163.com/tm/000828/000828_64264.html           // 点击的 URL
16                                                          // 点击页面的排序
```

根据天网系统维护的日志数据，如下统计分析了用户行为的分布特征，主要包括：

1) 用户查询词的分布情况
2) 雷同查询词的衰减统计
3) 相邻 N 项查询项的偏差分析
4) 用户点击 URL 的分布情况
5) 用户在输出结果中的翻页情况

在分析上述统计结果时，我们发现用户行为表现出极为强烈的局部性，这启发我们采用查询缓存和热点击缓存来提高系统性能。我们以日志中的用户行为作为输入，模拟实现 FIFO、LRU 以及带衰减的 LFU 等 3 种缓存替换策略，测试了其缓存命中率，比较了它们的优劣。另外，我们发现相邻 N 项查询项的偏差分布是稳定的，于是猜想用户查询项的分布过程符合自相似性，进而我们对此进行了验证，证明我们的猜测是对的。类似于互联网上网络流量的自相似性特征，该结论对于设计和评价一个搜索引擎系统具有很高的指导意义。

另外，我们还根据天网系统所搜集的网页信息统计分析了 Web 信息的一些重要参数的分布特征，这些参数包括网页入度、目录深度及镜像度等（这些参数将

在后面第四节中定义)。随后我们分别求出了这些参数的分布与用户点击 URL 的分布的差平方和，依此来度量这些参数对网页重要度的影响。同时，我们也得出了 URL 的入度、镜像度等参数与用户行为反馈后的相关度的方差分析对搜索引擎结果排序算法（ranking algorithm）的一些启示[单松巍,2003]。这些结论可以被用来提高搜索引擎的检索质量。

第二节　用户行为特征的统计分析

一、 用户查询词的分布情况

我们以天网 1999 年 4 月 15 日到 1999 年 6 月 10 日期间的日志记录作为分析对象，首先统计了用户查询词的分布情况。这里我们采用本章开头介绍的那种统计分析思路，假设用户的查询词序列为 S1={$q_1, q_2, \ldots, q_n$}，其中这 n 项查询词中共有 m 个不同的查询词，按其查询次数进行降序排列得到序列 S2={$Q_1, Q_2, \ldots, Q_m$}，而 S3={$C_1, C_2, \ldots, C_m$}是与 S2 对应的查询次数序列。我们希望考察序列 S2 中前某个百分比的查询词其对应查询次数占总查询次数的比率 Y，即计算公式（9-1）的值:

$$Y = \frac{\sum_{i=1}^{\lfloor m \times x \rfloor} C_i}{\sum_{j=1}^{m} C_j} \qquad (9\text{-}1)$$

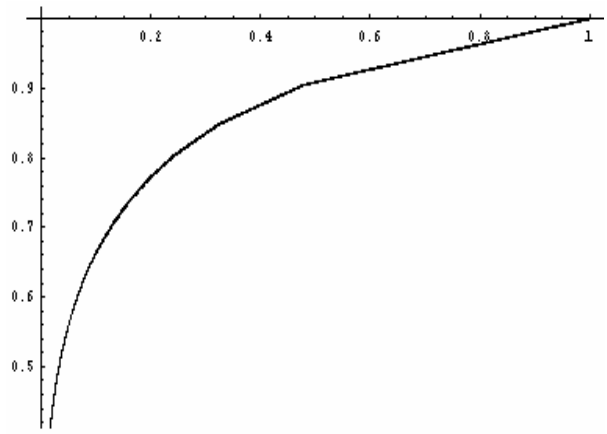


图 9-1 查询词的分布情况

通过对分析对象进行统计，我们得到$n=9.6\times10^5$，而$m=1.6\times10^5$。若以0.01作为x取值的跨度，可以计算得到一系列的Y值，如图9-1所示（其中X轴是用户查询词占查询词总数的百分比，Y轴是与X轴的百分比相对应的查询词的查询次数总和占总的查询次数的百分比）。该图显示，用户的查询词是非常集中的。例如，前20%的查询词的查询次数占了总查询次数的80%，即满足80/20规则[Baldi, et al.,2003]，也就是商业贸易中常提到的：80%的交易来自20%的客户。

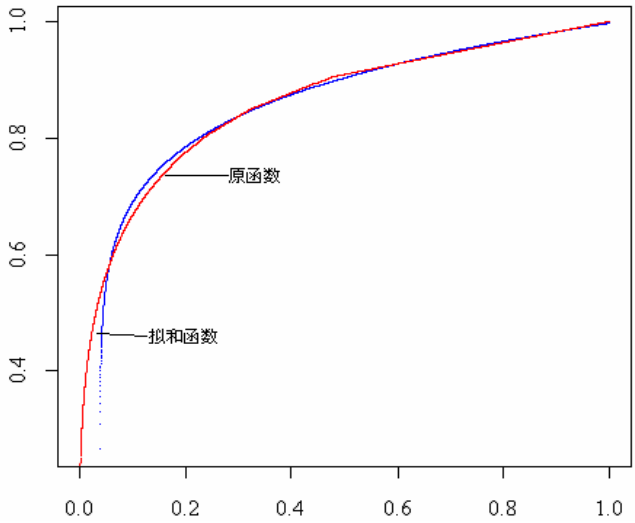


图9-2 查询词分布函数及其拟合函数

我们对图9-1中的查询词分布曲线进行函数拟合，得到其拟合函数，如图9-2所示。我们发现拟合函数具有幂函数的特征，其形式为：

$$y=(-0.04103+1.01689x)^{0.1346} \qquad (9\text{-}2)$$

这种幂函数$(x^a(a<1))$具有这样一个特征：在x越接近0的地方，y值增长越快，在x接近1的地方y的变化趋于平缓。这也表明了查询词的分布具有很强的局部性：绝大多数用户查询的关键词落在了相对很小的一个集合上。

二、 雷同查询词的衰减统计

基于天网1999年4月15日到1999年6月10日期间的日志记录，对用户雷同查询项的衰减情况进行统计分析，这里，我们把序列S1进行了分组，每相邻

1000 项分为一组，并假设第 i 组的查询序列为 $A_i=\{q_{i1},\ldots,q_{i1000}\}$，我们用 T_1 表示 A_1 中不同的查询项组成的集合，然后计算后面各组的查询项中有多少个查询项出现在 T_1 中，即对于 A_i，我们计算 Y_i 的值：

$$Y_i=\sum_{j=1}^{1000}c_{ij},\quad \text{其中}C_{ij}=\begin{cases}1 & if\ q_{ij}\in T_1\\ 0 & if\ q_{ij}\notin T_1\end{cases} \qquad (9\text{-}3)$$

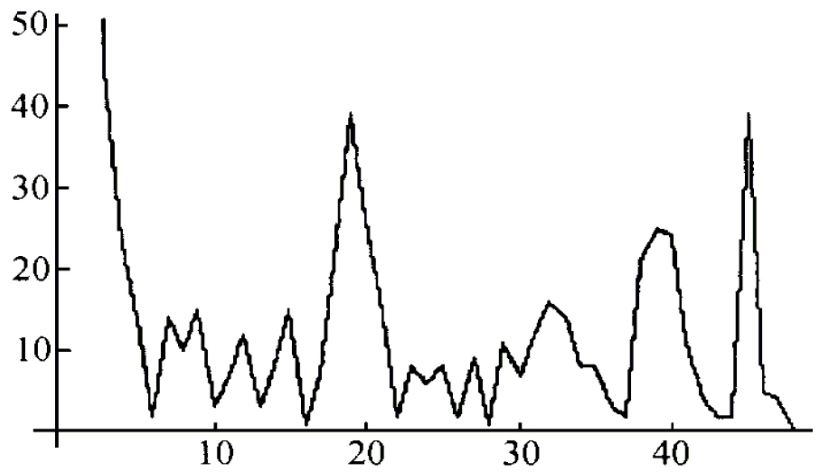


图 9-3 雷同查询词的衰减

当取不同的 *i* 值时就可以得到不同的 Y_i 值，其结果反映在图 9-3 中，其中横坐标表示组号，即第几组 1000 项，纵坐标表示该组查询项落在第 1 组查询项中的个数。从统计结果可以看出，第 1 组查询中的部分关键词或多或少地在其随后的多组查询中也出现了，直到第 48 组才完全消失，这表明用户的查询具有一定的稳定性。

三、 相邻 N 项查询词的偏差分析

根据天网 1999 年 4 月 15 日到 1999 年 6 月 10 日的查询日志，对其中相邻*N*项用户查询词的频率的差平方和进行了统计。具体做法如下：将用户查询每 1000 项分为一组，对于相邻的两组*A*和*B*，假设*A*组中出现的不同的用户查询是$(ab_1，ab_2，\ldots\ldots，ab_k，a_1，a_2，\ldots\ldots，a_n)$，其中$ab_i$是*A*组和*B*组中所共有的，$a_i$是*A*中出现但*B*中没出现的查询。而B组中出现的不同的用户查询是$(ab_1，ab_2，\ldots\ldots，ab_k，b_1，\ldots\ldots，b_m)$，其中$b_i$是*B*组中出现而*A*组中没出现的。

A 和 B 中的这些不同的查询项构成一个向量空间

$$(q_{ab_1}, q_{ab_2}, \Lambda, q_{ab_k}, q_{a_1}, \Lambda, q_{a_n}, q_{b_1}, \Lambda, q_{b_m})$$

我们假设某查询词q_i在A中出现的次数为F_{a_i}，对其规整化后作为其特征项A_i，这样就得到了A组的特征向量：

$(A_1, A_2, \ldots\ldots, A_k, A_{k+1}, \ldots\ldots, A_{k+n}, A_{k+n+1}, \ldots\ldots, A_{k+n+m})$，其中$A_i=F_{a_i}/1000$。

同样我们可以对这些不同的查询项在 B 组中出现的查询频率进行规整化后，得到 B 组的特征向量：

$(B_1, B_2, \ldots\ldots, B_k, B_{k+1}, \ldots\ldots, B_{k+n}, B_{k+n+1} \ldots\ldots, B_{k+n+m})$，其中$B_i=F_{b_i}/1000$。

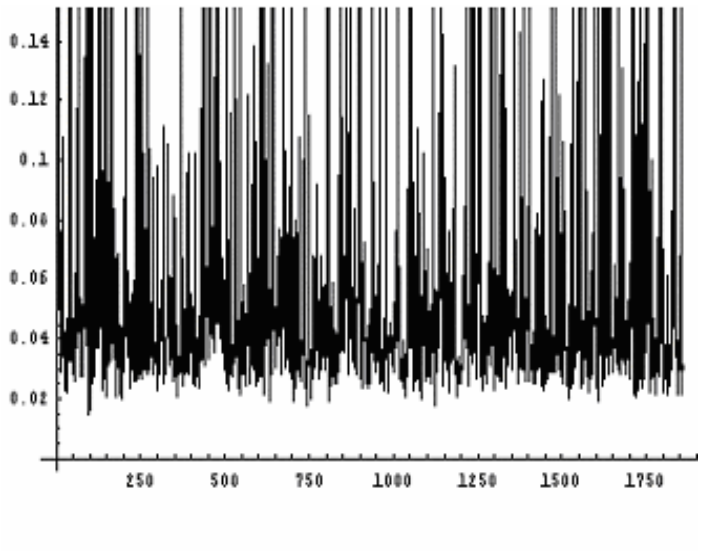


图 9-4 相邻 1000 项查询词的频率的差的平方和

我们计算以上两组特征向量的差平方和（如公式 9-4 所示），计算的结果如图 9-4 所示。该图显示：大部分的差平方和都是在 0.02 到 0.06 之间。它一方面说明了每相邻 1000 项之间的查询相差不是很大，二是说明了每相邻 1000 项之间的差别很稳定，即用户的查询不但在短时期内偏差不大，具有短期的相关性，而且这个偏差也比较稳定。

$$S = \sum_{i=1}^{k+n+m} (A_i - B_i)^2 \qquad (9\text{-}4)$$

四、 用户在输出结果中的翻页情况统计

我们用2000年4月份天网系统的查询日志来统计用户点击URL的翻页情况，其中该日志记录了近 50 万的用户点击情况，包括用户点击的 URL 及该 URL 所在输出结果中的页号。具体做法是：统计相同页号的页面点击次数占总点击次数的百分比。假设天网系统中能够提供 n 个显示页面（在当时的实际系统中 $n=2000$，每个页面包含 10 个网页信息），用$\{P_1,\ldots,P_n\}$来表示，它们对应的点击次数分别为 $C_1,\ldots,C_n$。对第 i 个页面，我们根据公式 9-5 计算其点击次数占总点击次数的百分比 Y_i。得到的结果如表 9-1 和图 9-5 所示（横坐标是页号，纵坐标是该页面的被点击次数占总点击次数的比率）。

$$Y=\frac{C_i}{\sum_{j=1}^{n}C_j} \qquad (9\text{-}5)$$

表 9-1 用户在前 5 页的翻页情况统计

页号	1	2	3	4	5
百分比	47.3%	12.2%	7.4%	5.0%	3.7%

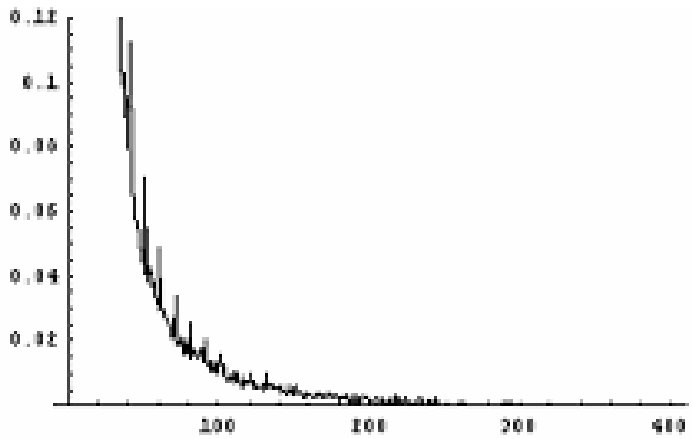


图 9-5 用户翻页情况统计

其中前面 5 页中 URL 点击次数占总点击次数的比例列在了表 9-1 中，可以看出大部分的用户点击都落在前面几页中，象第一页的用户点击占总点击的 47%，

而前面 5 页的点击占到了总点击的 75%以上。而图 9-5 表明用户很少浏览第 100 页以后的内容。这说明用户很少会在查询结果中翻很多页，用户一般就看看前面几页的内容而已。

五、 用户点击 URL 的分布情况

我们用 2000 年 4 月份天网的查询日志来统计用户点击 URL 的分布情况。这里我们假设用户点击的 URL 序列为 S1={$u_1, u_2, \ldots, u_n$}，其中这 n 个 URL 中共有 m 个是不同的，按其被点击次数进行降序排序得到序列 S2={$U_1, U_2, \ldots, U_m$}，而 S3={$C_1, C_2, \ldots, C_m$}是与 S2 对应的被点击次数序列。按公式（9-1）的计算方法，我们可以得到：统计序列 S2 中前某个百分比的 URL 其对应点击次数占总点击次数的比率 Y，其统计结果如图 9-6 所示，其中横坐标表示所选 URL 的数目占用户点击的 URL 总数的比率，纵坐标表示所选 URL 的被点击数目占用户点击总数的比率。

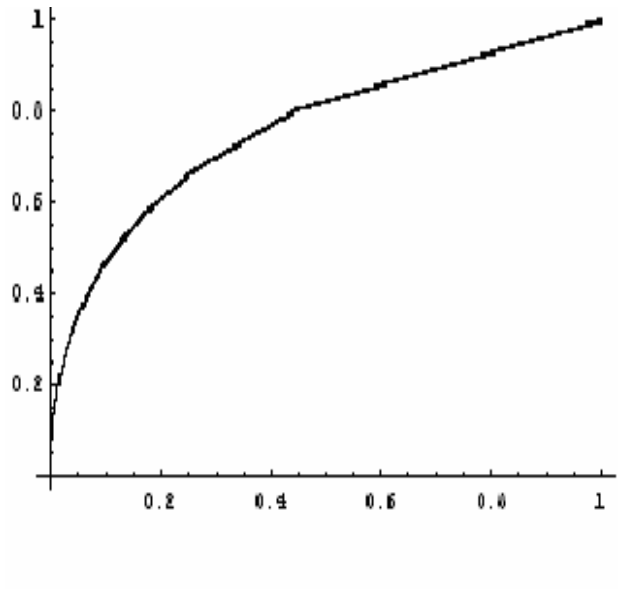


图 9-6 用户点击 URL 的分布情况

从图 9-6 的统计结果可以看出，用户点击的 URL 实际上也是非常集中的，2000 年 4 月天网 1.0 系统的数据库一共维护了 100 多万有效页面，但是在统计数据中被点击的 URL 只有 16 万多个，还不到总的有效页面的 1/6。而且在被点击的页面中常被用户点击的也是相当集中的，超过 50%的页面只被点击了一次，不到 1/3 的页面的点击次数占到了总点击次数的 2/3。这就表明了用户点击 URL 也具有很

强的局部性。

六、 考虑与不考虑查询项时点击 URL 分布的对比分析

对于点击 URL 的分布情况还有另外一种统计方法，即针对用户查询词的统计，因为用户的每一次点击都是在某个查询的结果中进行的，这种统计的方法就是把用户的点击和相应的查询联系起来。其具体方法是：将点击的 URL 按其对应的查询词分类，统计每个查询词下各个 URL 点击的次数。

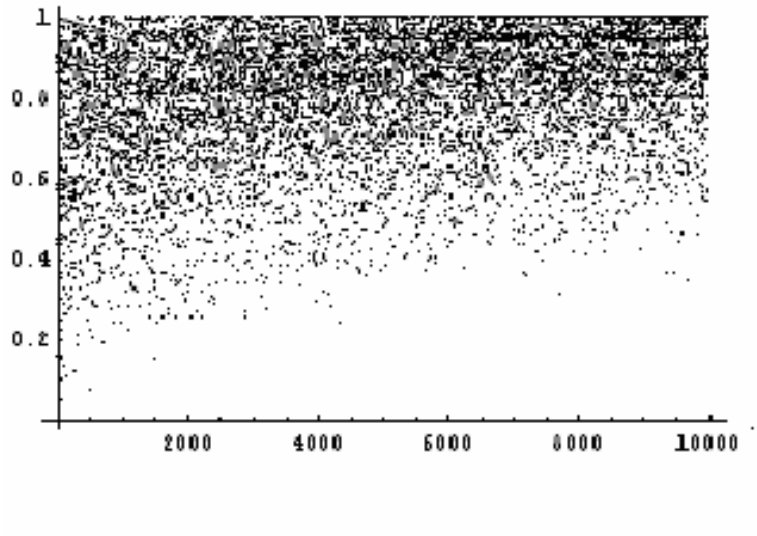


图 9-7 考虑查询项与否的 URL 分布情况

这样我们就得到了两种统计方法的结果，并且我们对这两种方法进行了比较。进行比较的方法是：在针对查询项的统计结果中，每个查询词Q_i下每个被点击的URL页面U_j都有一个点击次数W_{ij}，在不考虑查询的URL统计中，该URL也有一个点击次数W_j。在考虑查询项的点击次数的统计中，某个查询项Q_i下的URL点击次数形成一个向量：

$$\vec{P_i} = （W_{i1}，W_{i2}，……，W_{in}）$$

同时，这些 URL 在不考虑查询项时的 URL 点击次数也对应着一个向量：

$$\vec{P} = （W_1，W_2，……，W_n）$$

对于每个查询项，我们计算这两个向量的夹角余弦值：

$$COS\quad(\vec{P}_i,\vec{P}) = \frac{\vec{P}_i \bullet \vec{P}}{\left|\vec{P}_i\right| \times \left|\vec{P}\right|} \tag{9-6}$$

这个夹角的余弦值越接近 1，说明两个向量的夹角越小，两个向量的角度越接近，两个向量中各个分量占的比率越接近，即两种统计中，各个 URL 的点击次数占的百分比越接近。我们对用户点击次数最多的 10000 个词按上述方法做了比较，比较结果如图 9-7 所示（横坐标是查询词的编号，纵坐标是该查询词按上面方法计算的余弦值）。由图 9-7 可以看出，针对大部分的查询词计算出来的余弦值都是在 0.8 以上，这表明，在大部分的查询项下 URL 的点击频率和在所有查询项 URL 的总点击频率基本上是一致的。

七、 查询过程的自相似性

我们在统计每相邻 N 项查询项之间的频率的差平方和的时候，发现差平方和在长时间内一直比较稳定，似乎具有自相似性的特征，于是可以对查询日志做进一步分析，以验证用户的查询是否具有自相似性。

自相似性直观上说就是一组序列在很长的时间范围内表现出结构上的相似性。自相似模型的主要特点是长期依赖性，而不象泊松分布那样只能体现出来短期的依赖性。下面我们首先引入自相似性随机过程的定义[Crovella and Bestavros,1997]。

定义 9-1 设 X 是一个广义平稳随机过程，其均值为 μ，方差为 δ^2，自相关函数为 $\rho(\tau)$. 如果 $\rho(\tau)$ 具有以下形式：

$$\rho(\tau) \to \tau^{-\beta} L(\tau), \quad as \quad \tau \to \infty \tag{9-7}$$

其中 $L(\tau)$ 是一个在 τ 趋于无穷大时缓慢变化的函数，即 $\lim_{\tau\to\infty}\frac{L(x)}{L(\tau)}=1$，对所有的 x>0 成立。现将 X 分为大小为 m、非交叠的子块（聚合过程），用每个子块的均值所组成的序列表示一个随机过程，即：

$$X^{(m)}(t) = (X_{tm-m+1} + X_{tm-m+2} + \ldots + X_{tm})/m,\ t \ge 1 \tag{9-8}$$

对每一个 m， $X^{(m)}$ 都表示一个广义平稳随机过程，而 $\rho^{(m)}(\tau)$ 表示 $X^{(m)}$ 的自相关函数. 如果对所有的 m，聚合过程 $X^{(m)}$ 有着和 X 完全相同的自相关函数：

$$\rho^{(m)}(\tau) = \rho(\tau), \quad m \ge 1 \tag{9-9}$$

则称 X 为一个（严格二阶）自相似的随机过程，其自相似系数为 $H = 1-\beta/2$。

验证一个随机序列的自相似性，直观上的办法是取不同的 m 值，看不同聚合过程的分布图形是否相似，是否仍符合一般自相似性的序列的图形特征。进而我们可以采用如下的数学方法从理论上验证一个随机序列是否是自相似性的：

设$X = （X_1，X_2，……，X_n）$为待验证的随机序列，μ是这个序列的均值，$S^2(n)$为这个序列的方差。我们先按公式（9-10）计算R/S统计值（rescaled adjusted range statistic）然后计算$\log(R(n)/S(n))$，对于不同的n，取横坐标为$\log(n)$，纵坐标为$\log(R(n)/S(n))$作图，各个点和原点的连线的斜率应该比较接近，如果用一条直线来拟合，这条直线的斜率如果在 0.5 到 1 之间，这个序列就满足自相似性，如果在 0.7 以上，这个序列就具有很强的自相似性了。

$$R(n)/S(n) = [\max(0,W_1,...,W_n) - \min(0,W_1,...,W_n)]/S(n),$$
$$where \quad W_k = (X_1 + X_2 + ...,+X_k) - k\mu, (k \geq 1) \qquad (9\text{-}10)$$

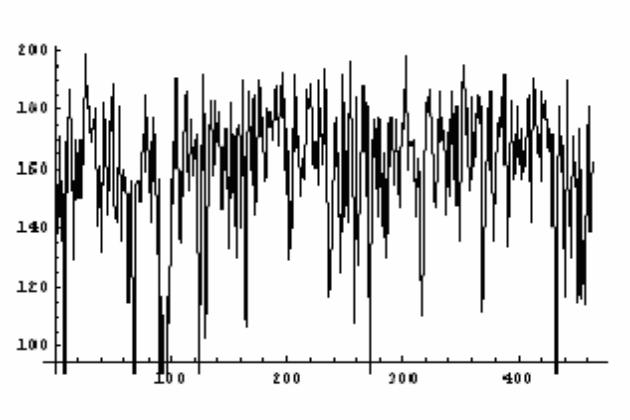


图 9-8 相邻 500 项中不同查询项的分布

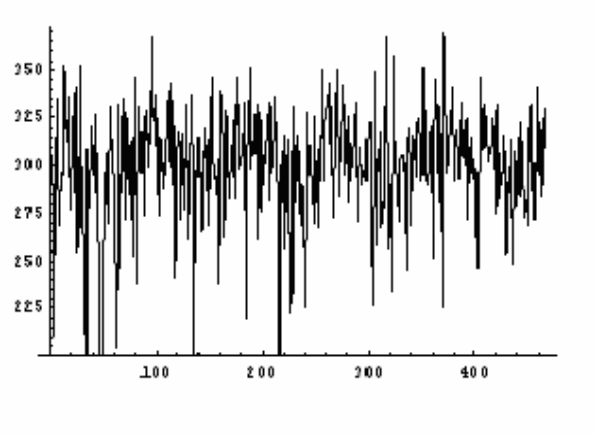


图 9-9 相邻 1000 项中不同查询项的分布

我们首先从直观上观察查询分布图是否符合自相似性特征。先将用户的查询每 500 项分为一组，统计每一组中不同的查询项的个数，以组号为横坐标，不同的查询个数为纵坐标，得到图 9-8。然后调整组的大小为 1000（即 m=2），2000(即 m=5)，做同样的统计，得到图 9-9 和 9-10。自相似性的序列其聚合后的分布图形仍能保持结构上的相似，其结果并不会因为聚合后就变的平缓了。而从图 9-9 和 9-10 中可以看出，用户查询分布满足上述特性。

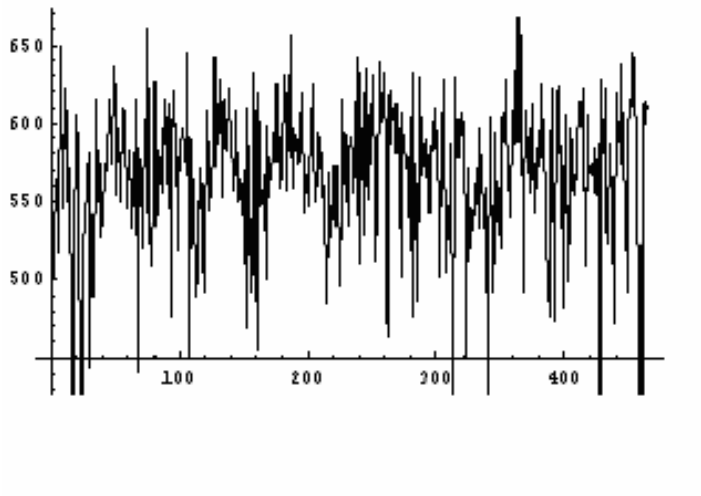


图 9-10 相邻 2000 项中不同查询项的分布

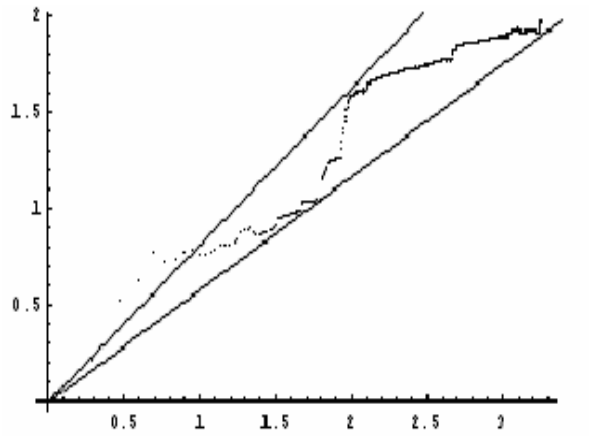


图 9-11 查询项分布的自相似性特征

接着，我们采用前面讲过的数学方法来严格地验证查询分布的自相似性。以天网日志中的数据为基础，利用公式(9-10)分别计算出 $\log(R(n)/S(n))$ 和 $log(n)$，并以 $\log(n)$ 为横坐标，$\log(R(n)/S(n))$ 为纵坐标得到函数图像如图 9-11 所示。可以看出几乎所有点都在斜率为 0.58 和 0.82 的直线之间，我们通过用最小二乘法来做直线拟和，求出拟合直线的斜率（即 Hurst 参数）是 0.67。当 hurst 因子介于 0.5 和 1 之间时，随机过程就是自相似的。这样，我们就验证了用户查询具备良好的自相似性。 本节的分析都是对天网某个时期的日志进行统计的，根据自相似性的特点，我们可以知道用户的查询是具有长期稳定性的，这样就可以将在前面的分析中得到的结果推广到搜索引擎长期以来的查询行为中，而且可以认为在今后相当长的时期，这些结果仍然有效。

第三节　查询缓存的使用

一、 基于用户行为的启示

用户查询分布的统计分析表明用户的查询词是非常集中的，这表明在查询中使用缓存的可行性：用户经常查询的词其实是很少的，把这些查询次数较高的词的查询结果放在缓存中，使用容量很小的缓存就能命中大部分的用户查询，这样就可以用较小的空间取得较大的缓存命中率。

假设在缓存中命中一个用户查询需要的延迟是T_m，在磁盘文件中查找一个用户查询需要的时间是T_d，缓存命中率是p，这样在引入缓存后，用户查询的平均响应时间变为原来（即未使用缓存）的η倍：

$$\eta = \frac{\overline{T_c}}{T_d} = \frac{p \times T_m + (1-p) \times T_d}{T_d} \qquad (9\text{-}11)$$

在天网系统中，一次访问硬盘的时间大约是一次访问内存的时间的几十倍，这样就有：

$$\eta \approx 1 - p \qquad (9\text{-}12)$$

由这个结果可以看出，如果在缓存中命中大部分的用户查询，即缓存命中率较高时，可以大大改善用户查询的平均响应时间。

用户雷同查询项的统计分析表明用户查询有一定的稳定性，这进一步说明了查询缓存的可行性，即在缓存中存放的查询信息及其结果不只是在很短的时间内才有效，可能经过一段时间后还被用户查询，这样缓存的实现就更加有价值了。而相邻 N 项查询项的查询频率偏差很小且非常稳定，从另一个角度说明了查询缓

存的可行性：缓存替换过程不会因为用户查询短期内的变化而产生颠簸现象。

我们对用户在输出结果中翻页情况的统计分析表明用户通常只浏览前几页的内容，这说明了对输出结果进行排序的重要性，即应当尽可能地把用户最想要的网页放在前几页。Direct Hit 技术能够跟踪用户对检索结果的后继行为，来获取大量的有用信息，以便提高查询结果排序的合理性。例如，那些经常为用户所浏览的网页应该被赋予较高的权值。这样做是非常合理的，因为目前几乎所有的搜索引擎系统在响应用户的检索请求并输出结果时，其返回页面中都包含了摘要信息，用户点击一个 URL，表明该 URL 符合他的要求。如果一个 URL 被很多用户所点击，表明该 URL 相对重要，我们应当提高其权值，使其排在输出结果的前面。为了根据一个 URL 被点击的次数来计算其权值，就需要在每次用户点击某个 URL 时，修改该 URL 的点击次数。如果计数器放在磁盘上，会引起大量的磁盘 I/O，严重影响系统性能（一个大型搜索引擎每秒的访问量通常超过几百次）。用户点击 URL 的局部性启发我们可以使用热点击缓存，即在内存中开辟一个空间，将用户点击过的 URL 放到里面，该 URL 若再次被点击，其点击次数和权值的修改都可以在内存中完成。因为用户的点击具有一定的局部性，所以只需要将很少的 URL 都放到内存中就可以在内存中命中绝大部分的用户点击，能够大大提高用户查询的响应速度。

然而如果在计算某个 URL 的被点击次数时没有具体到某个查询项，根据这样计算出来的 URL 权值来进行输出页面的 URL 排序显然是不合理的。然而如果对于用户提交过的每个查询项都维护一个 URL 列表，空间开销会很大。而我们对查询项考虑与否的 URL 点击分布的统计分析表明，大部分查询项下的 URL 点击频率和所有用户点击过的 URL 点击频率是大致相同的，这样我们在实现热点击缓存的时候，就没有必要再去记录查询项的信息了，只需要记录每个 URL 本身的信息即可，实现热点击缓存的空间代价和复杂程度就可以大大降低了。

另外，我们验证了搜索引擎系统中用户查询分布是一个自相似的随机过程，其自相关函数是以双曲函数衰减的，即其具备长期依赖性。查询分布的自相似性一方面表明查询分布的局部性特征是长期有效的，为引入查询缓存提供了理论基础，另一方面，类似于人们在发现了网络交通的自相似性后，利用自相似序列来测试 Web 服务器的性能，查询分布的自相似性对于设计和评价一个大型搜索引擎系统也具有重要的实用价值。

二、 缓存替换策略研究

前面的统计分析表明了查询缓存和热点击缓存的可行性，下面以天网 1999 年 4 月 15 日到 1999 年 6 月 10 日的查询日志作为输入来对几种缓存替换策略进行比较，选择一种最适合于搜索引擎系统的策略。我们评测的替换策略包括 FIFO

（First In First Out）、LRU（Least Recently Used）和 LFU（Least Frequently Used）三种，其中 LFU 是带衰减的 LFU，即每次发生替换时用某个衰减因子去衰减原来的查询次数并累加新的查询次数。对于 FIFO 和 LRU 这两种替换策略，主要是考察在不同的缓存大小下的命中率，而对于 LFU，还多了一个参数，那就是衰减因子，调整衰减因子的大小，LFU 策略下缓存的命中率可以有很大的差别。

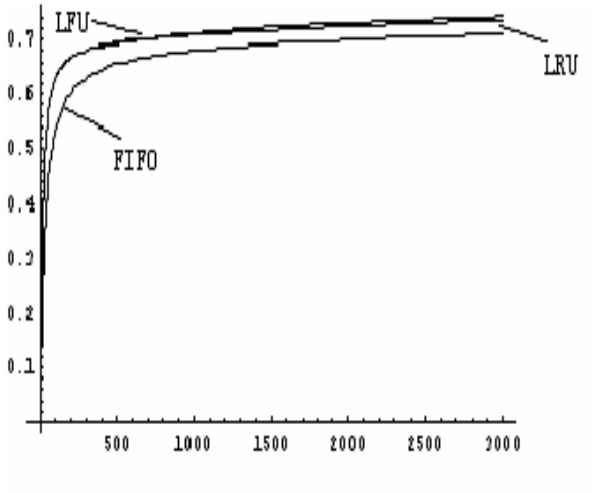


图 9-12 FIFO、LRU 和带衰减的 LFU 的缓存命中率比较

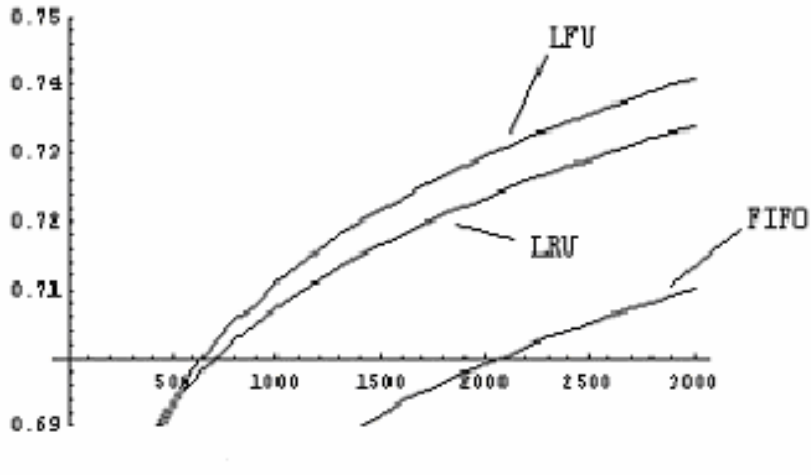


图 9-13 3 种替换策略的局部比较

表 9-2 调整后的 LFU 与 LRU 命中率的比较

缓存大小	100	300	500	1000	2000	3000
LRU 命中率	0.629381	0.680018	0.692691	0.707481	0.723485	0.733972
LFU 命中率	0.629934	0.680690	0.694037	0.711096	0.729509	0.741040

图 9-12 给出了 FIFO、LRU 和带衰减的 LFU 三种替换策略的缓存命中率，其中对于 LFU，我们将衰减因子固定为 0.998。从图中可以看出，当缓存的大小到达一定程度时，缓存的命中率可以很高，当缓存的大小为 500 时，这 3 种替换策略的缓存命中率都到了 60%以上，几乎 2/3 的查询都可以在缓存中命中，平均的用户查询响应速度就会大大提高。这说明查询缓存的效果很好。同时，我们可以看出 LRU 和 LFU 替换策略下的缓存命中率要比 FIFO 好得多，而在 0.998 的衰减因子下，LFU 比 LRU 的命中率相差不大，两条线几乎重叠在一起；我们把图 9-12 中的部分结果局部放大，得到图 9-13，可以看出， LFU 比 LRU 略好一些，但效果不是很明显。

实际上，LFU 在不同的缓存大小下，取得最佳命中率的衰减因子都不同，如果在不同的缓存大小下调整衰减因子，LFU 可以得到比 LRU 更好的命中率，表 9-2 是一些调整了衰减因子后的 LFU 的命中率，可以看出它要比 LRU 的效果好一些。

综合以上结论，LRU 和 LFU 的缓存命中率要明显好于 FIFO，LFU 如果固定了衰减因子，其效果和 LRU 相差不多，如果选取好的衰减因子，可以得到比 LRU 稍微好一些的效果。考虑到实现的复杂性，LRU 和 FIFO 都比较简单，而 LFU 在发生替换的时候要进行衰减，必须遍历整个缓存，其替换时间要远远大于 LRU 和 FIFO，而其效果和 LRU 相差的又不是很多。所以，综合考虑这几种替换策略，LRU 是最好的选择。

第四节 用户行为与 Web 信息的分布特征

一、 基本术语

在 2000 年 4 月上旬，天网搜集了 1,000,000 个国内网页，这些网页立刻作为新的数据对外提供服务。在随后的 14 天时间里，有 141,779 篇网页通过天网的引导被用户访问，总访问次数为 400,641。哪些网页被访问，访问了多少次都通过日志文件被记录了下来。我们将基于这些信息来考察 Web 信息（主要是入度、镜像度和目录深度）的分布特征及其与网页重要度之间的关系。

这里对网页重要度的度量规则定义为：用户访问越多的网页越重要。需要指

出的是，用户点击 URL 的行为是受天网系统的输出页面中结果排序的影响的，如 75%的用户点击落在前 5 个输出页面中，输出页面中，某个 URL 的权值是使用文档向量空间模型和 TF*IDF 算法计算出来的，它反映了该 URL 和用户查询项之间的相关程度，这种排序本身就有其合理性。而且用户在点击一个 URL 之前，通常先浏览该网页的摘要等信息，如果他对该网页内容感兴趣才去点击它。所以这种受相关度排序影响的用户行为能够很好地反映网页的重要程度。下面我们首先定义几个网页参数—网页入度、镜像度和目录深度，然后具体考察其与网页重要度之间的关系。

“网页 P 的入度 H(P)”是指整个网络中指向网页 P 的超链数目。正如 SCI 文章影响因子的计算，一篇文章若较多地被其他文章所引用，那它的影响因子就较大。同样地，我们也认为只有当其他网页的编辑者认为此网页重要时，才会在他们所编辑的网页中加入指向此网页的超链。

“网页 P 的镜像度 C(P)”是指整个网络中网页 P 的镜像个数。对于一则新闻，我们常常用“被多家报刊杂志转载”来形容它的被关注程度。对于网上的各种信息，如新闻、文学作品、技术文档等，如果它真的被大量网民所关注的话，就会有很多网站把这篇网页拷贝过来，当然也许会被略加改动。

“域名深度”是指域名中包含的子域的个数，“目录深度”是指域名中所包含目录的层数。下面我们将域名深度和目录深度统称为目录深度 D(P)。域名和目录都是人们用层次结构组织信息的一种方式。网页处在这种层次结构中的深浅是否与人们对它的关心程度有某种联系？下文也试图对这一问题进行回答。

二、 海量 Web 信息的特征分析

我们将天网 2000 年 4 月上旬搜集的 100 万网页按照被用户访问的次数按降序排序，设该 URL 序列为 $U_1,U_2,\ldots,U_{1000000}$，其对应的用户点击次数依次为 $V_1,V_2,V_3,V_4,\ldots\ldots V_{999999},V_{1000000}$，它们对应的网页入度为 $H_1,H_2,H_3,H_4,\ldots\ldots H_{999999},H_{1000000}$，网页镜像数为 $C_1,C_2,C_3,C_4,\ldots,C_{999999},C_{1000000}$，URL 目录深度是 $D_1,D_2,D_3,D_4,\ldots\ldots D_{999999},D_{1000000}$. 另外，我们还增加了一个参照序列，它对每一个 URL 赋予同等重要度，即 $S_1,S_2,S_3,S_4,\ldots\ldots S_{999999},S_{1000000}$，其中 $S_i=1$。图 9-14，图 9-15，图 9-16，图 9-17 分别示意了被用户访问的 14 万多网页按照用户行为（被点击次数）、网页入度、镜像度及目录深度进行排序后的分布情况。可以看出，被用户点击越多的 URL，其网页入度和镜像度也相对地越高，目录深度表现的则不是很明显。

对这 100 万网页消除镜像网页后得到 868,357 个有效网页，这其中有 131,906 个有效的被访问网页，然后我们分别计算出有效网页和有效的被访问网页的入度总和、镜像度总和以及目录深度总和，列于表 9-3 中。从表中的比例关系可以看

出，被访问网页的入度、镜像度都大于平均数（15.19%），而目录深度略小于平均数。如果我们根据入度 H(P)、镜像度 C(P)和目录深度 D(P)来计算网页 P 的搜索权值 W(P)，即：

$$W(P) = f(H(P), C(P), D(P)) \qquad (9\text{-}13)$$

则 W(P)应当与 H(P)和 C(P)呈现某种正比关系，与 D(P)成反比关系。这表明网页入度和镜像度越高，或目录深度越浅，网页重要度越大。我们优先搜集短目录的网页，还可以得到一个好处：避免了深度优先搜索，以搜集更多网站的重要网页。

表 9-3 各网页参数的分布

网页参数	总数	对被访问的部分求和	百分比
网页数	868357	131906	15.19%
入度	2876462	589868	20.51%
镜像数	999999	160896	16.09%
深度	6479796	947179	14.62%

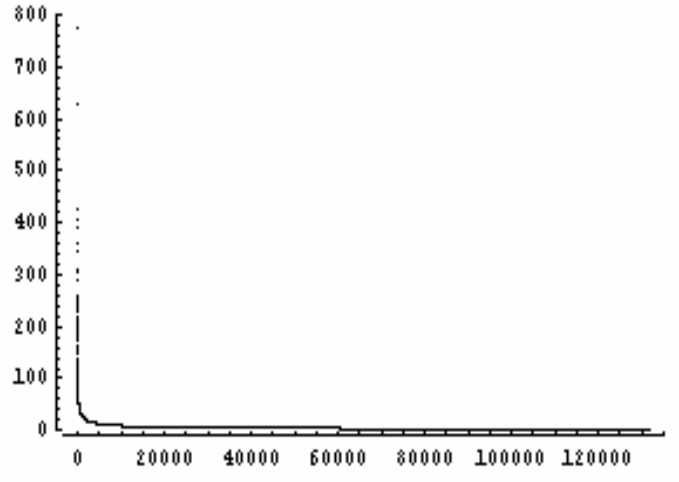


图 9-14 网页的被访问次数

天网系统的启发式搜集策略已考虑到目录深度这一因素，表明它的效果很好：当系统搜集到国内 1/10 的网页时已基本遍历完国内所有网站（CNNIC 2000 年 1 月的统计信息表明当时国内有 15153 个站点）。另外，在制定搜索导向策略时，还可以考虑一些其他措施。比如我们可以配置一些导向词（如用户经常查询的关键词），当导向词和一个网页的相关度很高时，可以赋予它较高的搜索权值。而如果一个 URL 的父 URL 有较高权值，则它也应被赋予较高权值。这些因素不是本节的主要考察对象。

我们把 URL 序列$\{U_i\}$对应的用户点击序列$\{Vi\}$、入度序列$\{H_i\}$、镜像度序列

{Ci}、目录深度序列{D_i}以及参考序列{S_i}分别进行规整化，得到新的序列{$V_i^{'}$}、{$H_i^{'}$}、{$C_i^{'}$}、{$D_i^{'}$}和{$S_i^{'}$}。即采用公式（9-14）来计算{V_i}，可得到子序列{$V_i^{'}$}（也可采用同样的方法得到其它规整化后的序列）：

$$V_i^{'} = \frac{V_i}{\sum V_j} \tag{9-14}$$

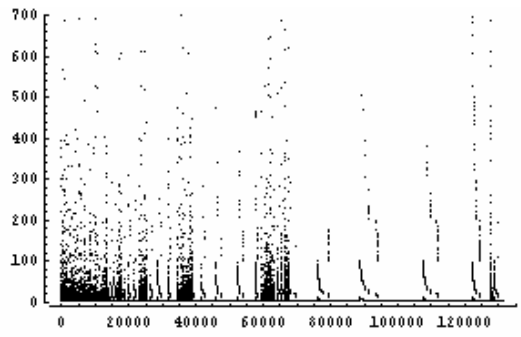


图 9-15 用户点击 url 对应网页的入度

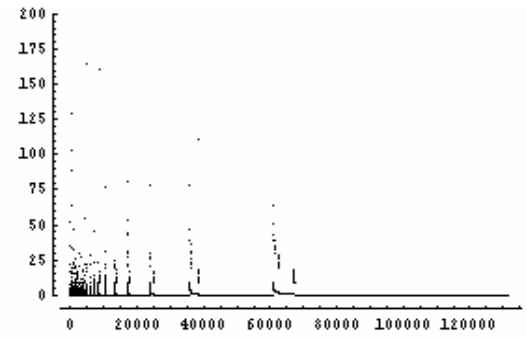


图 9-16 用户点击 url 对应网页的镜像度

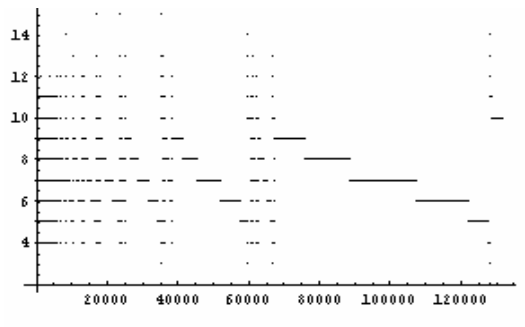


图 9-17 用户点击 url 对应网页的目录深度

对于$\{H_i^{'}\}$、$\{C_i^{'}\}$和$\{S_i^{'}\}$，我们分别求与$\{V_i^{'}\}$的差平方和，得到：

$$mis\_H = \sum (H_i^{'} - V_i^{'})^2 = 2.604480 \times 10^{-4} \qquad (9\text{-}15)$$

$$mis\_C = \sum (C_i^{'} - V_i^{'})^2 = 6.321719 \times 10^{-5} \qquad (9\text{-}16)$$

$$mis\_S = \sum (S_i^{'} - V_i^{'})^2 = 6.298261 \times 10^{-5} \qquad (9\text{-}17)$$

可以看出，网页入度与访问次数的偏差最大，而镜像度与访问次数的偏差和参考序列与访问次数的偏差比较接近。这一结果出乎我们的预料，尤其是我们认为网页入度最有可能与被访问次数相一致，以后可以象 Google 系统或 Clever 系统那样被用作影响结果排序（result ranking）的因素，但它与被访问次数的差平方和反而最大。这说明了一般情况下，URL 的入度与受用户查询相关度排序影响的用户点击行为呈现某种反比关系，在进行结果排序时不能简单地认为某个 URL 的入度越大，其检索权值就越高。

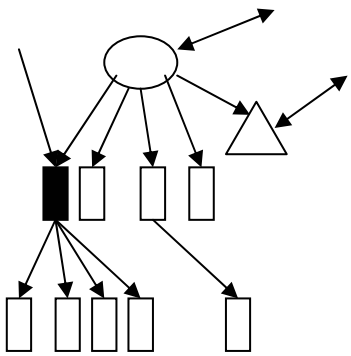


图 9-18 站内网页的树状结构

通过对实际数据的分析，我们发现，国内有影响的网站的主页、技术文档和书籍的目录主页多获得了比较高的入度，而一般的网页入度都较低。

进一步分析发现，网站一般组织成类似树形的结构，如图 9-18 所示。大多数载有文章的网页（图中的空心矩形），它们极少被外站的网页所链接，在本站一般也只被一个网页所链接。而主页（图中的椭圆），它们既有超链指向站外，也被站外网页所链接。图中的实心矩形是目录网页，如时代商城目录主页，它们被站外网页所指向，但自己的超链不向外指。图 9-18 中的三角为一些专门向外指的网站，常被命名为“网络导航”，“友情链接”之类。

进而分析天网的用户查询日志发现，绝大多数用户查询是针对普通网页的，只有少量的是在查找一些站点主页，如北大、清华、中科院、新浪网和方舟等。因而对所有的网页都用入度来和被访问次数计算差平方和是没有什么用处的。在这一启发下，我们对一百万网页中的一万多个主页来求它们被访问次数的差平方和，得到：

$$mis\_H' = \sum (H_i' - V_i')^2 = 7.16635 \times 10^{-4} \qquad (9\text{-}18)$$

$$mis\_C' = \sum (C_i' - V_i')^2 = 6.93268 \times 10^{-3} \qquad (9\text{-}19)$$

$$mis\_S' = \sum (S_i' - V_i')^2 = 7.17072 \times 10^{-3} \qquad (9\text{-}20)$$

从公式（9-18），（9-19），（9-20）可以看出，此时网页入度成为与用户访问次数相关度最好的指标，镜像度与用户行为的相关度也吻合的较好。综合公式（9-15）至（9-20）的结果，我们得到如下启示：在搜索引擎提供服务时，应当将网站查询和一般网页的查询区分处理，这样一方面可以缩小输出结果范围，提高检索质量外，还应当为这两类查询采用不同的相关度排序算法。如用户在进行网站查询时，除了根据查询项与网页的相关度计算该网页的基本权值外，还要根据其入度和镜像度计算附加权值，检索子系统综合这两个权值以进行结果排序。而对于普通网页，计算附加权值时就可以不考虑网页入度。

第十章 相关排序与系统质量评估

传统上，人们将信息检索系统返回结果的排序称为“相关排序”（relevance ranking），隐含其中各条目的顺序反映结果和查询的相关程度。在搜索引擎的情形，人们也这么讲，但内涵其实是有了差别。一方面，搜索引擎维护的内容十分繁杂且不规范，不像传统的图书、文献等有很好的分类体系管理。另一方面，搜索引擎面对的用户背景广阔，层次多样，不像传统的图书馆所面对的用户通常有相对比较整齐的用户群。因此，搜索引擎要给出的不是一个狭义的相关序，而是某种反映多种因素的综合统计优先序。检索质量评估的目标是对不同搜索引擎系统的检索质量评估其相对优劣次序。本章的第一节介绍传统的相关排序技术；第二节分析网络环境下影响排序的若干新的因素，并讨论如何利用 Web 间的链接关系进行相关度排序；第三节给出了考虑这些因素后的一个结果排序的具体实现方案，第四节介绍搜索引擎系统质量评估的一般技术与方法。

第一节 传统 IR 的相关排序技术

给定某个文档集合 D，大小为 M；设两篇文档 d1，d2 ∈ D，一个查询 q。用什么样的标准来讲“d1 与 d2 相比，前者和 q 更相关”？这方面最经典、最有影响的工作是 Gerald Salton 等在 30 多年前提出的“向量空间模型”（Vector Space Model，VSM）[Salton and Lesk,1968, Salton,1971]。该模型的基础是如下假设：文档 d 和查询 q 的相关性可以由它们包含的共有词汇情况来刻画。

这样，文档 d 和查询 q 就都被简化成词汇的集合（多重集）。不失一般性，令

$$\Sigma = \{t_1, t_2, ..., t_N\}$$

为一个词典，t_i为词项，N为它的规模，则：

$$d = < t_1^{m_1}, t_2^{m_2}, ..., t_N^{m_N} >$$

$$q = < t_1^{n_1}, t_2^{n_2}, ..., t_N^{n_N} >$$

其中，m_i，n_i，i=1, 2, …, N，表示相应词项出现的次数，即词频TF；如果次数为零，则表示该词项在文档或查询中没有出现。在实际应用中，人们通常去掉 t_i 而

直接用 m_i 和 n_i 来表示d和q。

d 和 q 相关程度的评价就以这样两个向量的某种“相近程度”为基础，这其中，有一些细节的变化。

1) 上述表示中，词项在文档和查询中出现的次数（词频）是一个基本量，我们称为“词频”模型。在实用中用规格化表示（以一篇文档为例）。

$$d =< w_1, w_2, ..., w_N > \text{其中，} \quad w_i = \frac{m_i}{\sum m_j}$$

查询q也有同样的表达形式。这里，w_i也称为词频，称这种方式为用词频来表示词项在文档或查询中的权重。

2) 在许多情况下，为了简便起见m_i, n_i只在集合{0,1}中取值，表示词项的出现与否，不关心其次数；此时的模型称为“二元模型”。

3) 若一个词项t_i在许多文档中都有出现（例如“我们”，“大家”等），尽管它可能在文档内部出现的频度也很高，它对于不同文档的区分能力就不会很强，因此它的权重应该相对较小。将这种观念刻画出来，引入词项的文档频率DF的概念。用k_i表示词项t_i在文档集合D中涉及的文档个数，M是集合D的大小，则文档频率为 $df(t_i) = \frac{k_i}{M}$ 。

我们需要的是和 df 成反比的一个量，称之为倒置文档频率 IDF，常用的一种定义为 $idf_i = \log\left(\frac{M}{k_i}\right)$。这样结合词频，就有了经典的 TF*IDF 词项权重的设计

$$w_i = tf_i \times idf_i = \frac{m_i}{\sum m_j} \times \log\left(\frac{M}{k}\right)$$

给定某种权重的定量设计，求文档和查询的相关性就变成了求 d 和 q 向量的某种距离，最常用的是余弦（cos）距离

$$\cos(d,q) = \frac{\sum w_i^{m_i} \cdot w_i^{n_i}}{|q| \times |d|}$$

上述这些理论，源于传统信息检索领域，针对的是普通文本。这样一种理论虽然从根本上看起来比较粗糙（将文本近似成一个词项集合，完全忽略语法和语义），但几十年来在大量真实语料评估的驱动下，其不断完善的实现在实践中得到

普遍认可[1]。

从具体实现的角度看，这样一种理论在倒排文件的数据结构上能够很容易得以实现。给定文档集合D和词典Σ，对D中的每一个d得到其权重表示（那些w_i）是预处理的工作，同时自然也得到了所需的M，N等。

现在的问题是，我们需要得到网页（用 p 来表示）和查询 q 的相关性。最初，一种简单的方法就是用普通文本作为网页的近似，即，$p \propto d$，只考虑网页中那些用户可见的文字部分，忽略标记和超链等内容，于是上面的理论和实践都可以马上照搬。但人们很快发现这有很大问题，其原因在于传统 IR 方法的成功有两个重要的内在假设：

1) 被索引的信息本身有很高的质量，至少在信息的组织和内容上有着比较高的质量。在 Web 出现以前，传统的 IR 之所以能够行之有效至少在部分上是依赖这一点的。很多的 IR 产品一般都是针对一个特定的领域，如法律信息、医疗信息、环保信息等等。这样它们可以针对这个领域进行算法的优化，同时，也避免了对一词多义的处理。
2) 检索信息的用户有一定的相关技能和知识。也就是说，当用户面对一个很大的信息源时，他知道通过什么样的手段去提高检索的准确率，但同时又不降低系统的查全率。与此相对应的，传统的 IR 系统总是提供一套相当复杂的检索语法来满足用户的不同要求。

然而，这些假设在 Web 上都已不再成立：

1) Web 上网页的质量参差不齐，大量的网页组织性、结构性比较差。同时，Web 又是一个无所不包的载体，它涉及到政治、经济、教育、生活等各个层面。这使得 IR 中的很多技术都没有了施展才能的余地。另外，网络上充斥着很多没有任何意义的网页，很多镜像的网页，如果不采取相应的技术处理，将会在很大程度上影响检索质量。
2) 大部分检索用户是没有任何经验的。他们经常只输入一个或者两个检索词来检索他们需要的网页，但会得到大量的返回结果，很难达到满意的程度。很少有用户愿意使用逻辑运算来提高检索的质量。即使这样，在不少用户的输入表达中，依然存在各种各样的问题[2]。

基于此，人们发现原有的 IR 技术已经不能适应 Web 的发展，必须改进原有的 IR 相应技术，研究新的适合 Web 的技术和算法，提高 Web 检索的质量。

[1] 这给我们一种启示：在应用性很强的领域，理论基础粗糙的先天不足可能由精细的实现技术来弥补。

[2] 例如，在天网日志统计中发现“搜索”这个词曾多次出现，反映了不少用户还不知道该如何用搜索引擎。

第二节 链接分析与相关排序

尽管 Web 页面的情况比传统 IR 面对的情况要复杂许多，但其中的复杂性也给我们带来了新的机会，主要体现在两个方面。首先可以利用网页间的链接关系进行链接分析，量化网页信息；其次，在 Web 查询模式下产生了许多新的信息可资利用，如 Web 用户行为信息等。

一、 链接分析

从开发利用的角度看，网页和普通文本的不同主要反映在两个方面：HTML 标签和网页之间的超链接。

1) 我们知道，HTML 设计有丰富的标签，是网页作者用于将网页的不同部分以不同的形式呈现给用户的手段，包括文字的布局，字体、字号的变化，等等，主要追求的是视觉效果。因此，标签能给我们提示其中文字的重要程度。例如，常识告诉我们，在同一篇文字中，比较大的字体往往是作者比较强调的内容；而在一版（以区别“一篇”，如同报纸）内容分块、且有一定布局的文字上，放在前面和中间的应该是作者比较强调的，等等。许多著名搜索引擎在网页的预处理阶段记录了这些信息，并用于结果排序。例如 Alta Vista, Inktomi, Excite, Infoseek，等等。
2) 链接反映的是网页之间形成的“参考”、“引用”和“推荐”关系。可以合理的假设，若一篇网页被较多的其他网页链接，则它相对较被人关注，其内容应该是较重要、或者较有用。因此，可以认为一个网页的“入度”（指向它的网页的个数）是衡量它重要程度的一种有意义的指标。这和科技论文的情况类似，被引用较多的就是较好的文章。同时，人们注意到，网页的“出度”（从它连出的超链个数）对分析网上信息的状况也很有意义的，因此可以考虑同时用两个指标来衡量网页。这些想法即是斯坦福大学 Google 研究小组和 IBM 公司的 Clever 系统开发小组几乎在同一时间分别提出著名的 PageRank 技术和 HITS 技术的基础。

可以用一种“随机冲浪”模型来作为 PageRank 的理论基础，该模型描述网络用户对网页的访问行为，假设如下：

1) 用户随机的选择一个网页作为上网的起始网页；
2) 看完这个网页后，从该网页内所含的超链内随机的选择一个页面继续进行浏览；
3) 沿着超链前进了一定数目的网页后，用户对这个主题感到厌倦，重新随

机选择一个网页进行浏览，如此反复。

按照以上的用户行为模型，每个网页可能被访问到的次数越多就越重要，这样的“可能被访问的次数”也就定义为网页的权值，PageRank 值。如何计算这个权值呢？PageRank 采用以下公式进行计算：

$$W_j = (1-d) + d\sum_{i=1,i\neq j}^{N} l_{i,j}\frac{W_i}{n_i}$$

其中W_j代表第j个网页的权值；l_{ij}只取 0、1 值，代表从网页i到网页j是否存在链接；n_i代表网页i有多少个连向其它网页的链接；d代表“随机冲浪”中沿着链接访问网页的平均次数。选择合适的初始数值，递归的使用上述公式，即可得到理想的网页权值。

IBM 研究院 Clever 系统中的相应技术称为 HITS（Hyperlink-Induced Topic Search）。Clever 描述两种类型的网页：

- “权威型（Authority）网页”：对于一个特定的检索，该网页提供最好的相关信息；
- “目录型（Hub）网页”：该网页提供很多指向其它高质量权威型网页的超链。

进而在每个网页上定义“目录型权值”和“权威型权值”两个参数。当遇到一个检索时，Clever 先利用检索的关键词得到一个网页的根集合，如从搜索引擎返回结果取前 200 个网页；然后根据这个集合在整个网页有向图中的位置来扩展这个根集合。具体办法是，将被链接（包括链出和链入）的网页加入到这个根集合中，形成一个新的集合；依据指定的网页规模进行扩展，如，使根集合扩展到一个包含 1,000 到 5,000 个网页的集合。

在得到这个集合后，就开始计算集合中每个网页的目录型权值和权威型权值。Clever的做法是采用目录型网页和权威型网页相互评价的办法进行递归计算。对于一个网页p，用x_p来表示网页p的权威型权值，用y_p来表示它的目录型权值，并且用如下公式进行计算：

$$x_p = \sum_{q\ such\ that\ q->p} y_q$$

$$y_p = \sum_{q\ such\ that\ q->p} x_q$$

这样的递归式也容易用矩阵方法表示。令所有选出来的网页都进行标号，我们得到所有网页的编号集{1,2,…,n}。令相邻矩阵A 为一个$n\times n$的矩阵，如果存在一个从网页i链接到网页j 的超链，就令矩阵中的第（i,j）个元素置为 1，其它各项置为 0。同时，我们将所有网页的权威型权值x和目录型权值y都表示成向量

形式x = (x_1, x_2, … x_n)，y = (y_1, y_2, … y_n)。由此我们可以得到计算x和y的简单矩阵公式：$y = A \cdot x$，$x = A^T \cdot y$，其中 A^T 是A的转置矩阵。进一步，我们有：

$$x = A^T \cdot y = A^T Ax = (A^T A)x$$
$$y = A \cdot x = AA^T y = (AA^T)y$$

经过一定次数的递归运算后，会得到集合中每个网页的权威型权值和目录型权值。按照这两个不同的权值，分别取出前 k 个返回给用户。根据 Clever 系统自己的测试数据，对于返回给用户的前 10 个检索结果，Clever 系统在 50%的情况下获得了高于 Yahoo!和 AltaVista 的用户评价。

通过上面的分析，我们发现这两种方法有很多相似之处。它们都利用了网页和超链组成的有向图，根据相互链接的关系进行递归的运算。但是，两者又有很大的区别，主要在于运算的时机。Google 是在网页搜集告一段落时，离线的使用一定的算法计算每个网页的权值，在检索时只需要从数据库中取出这些数据即可，而不用做额外的运算，这样做的好处是检索的速度快，但丧失了检索时的灵活型。Clever 使用即时分析运算策略，每得到一个检索，它都要从数据库中找到相应的网页，同时提取出这些网页和链接构成的有向子图，再运算获得各个网页的相应链接权值。这种方法虽然灵活性强，并且更加精确，但在用户检索时进行如此大量的运算，检索效率显然不高。

二、 Web 查询模式下的新信息

上述链接分析可以有效的计算网页的重要程度，但是带有明显的偏向，即不重视新出现的网页。新出现的网页，尽管可能很重要，但由于时间短，被链接的次数显然不可能很高，PageRank 的值就不会高。因此需要来补偿这个问题，人们注意到，除网页本身特性外，搜索引擎的应用环境和传统信息查询也有些不同，这可以从两个方面考虑。

1. 用户行为

和传统 IR 的用户群相比，虽然搜索引擎的用户群的经验少，但他们的数量却十分巨大。大型商业搜索引擎，如 Google，AltaVista，百度等，每天都有上 1000 万次的用户检索。通过对这些用户检索行为的统计分析，我们可以从中获取许多有用信息，这些信息可以大大提高搜索引擎检索结果的准确率，提高检索的质量。

Direct Hit 技术就是基于以上思想创立的。这项技术的主要特点是跟踪用户对检索结果的后继行为：哪些站点被用户选择浏览了？用户在这个站点上花费了多少时间？通过对这些数据的统计，搜索引擎就可以提高那些经常被用户选择，而

且花了大量时间去浏览的站点的权值，降低那些不太被用户关心的站点的权值。对于新加入系统的网页，系统则先给它们一个缺省的权值，然后由用户来决定它们的重要性。另外，系统还可以利用以前的用户检索行为来对以后的相似检索进行优化，帮助用户尽快发现自己需要的信息。

这个技术的另一个优点就是可以对一个固定的用户的行为进行跟踪和统计，进而发现这个用户的喜好和对检索结果的期待，从而产生专门针对该用户的检索结果。这就是我们后面会提到的个性化检索。例如，一个做建筑材料的工程师，当他检索“windows”，他最大的可能是关注有关窗户的问题；一个计算机工程师，同样的检索则更关心微软的 Windows 产品。通过几次用户检索行为的跟踪学习，我们就可以获得这些信息，进而在以后的检索中，我们就对检索的输出结果进行针对用户的适应性调整。

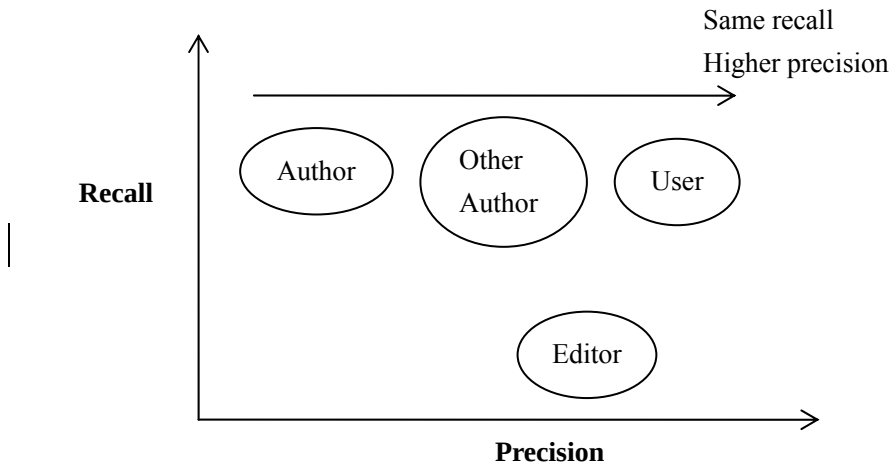


图 10-1 Inktomi 提供的几种搜索引擎技术的比较

Direct Hit 公司的 Gary Cullis 在搜索引擎 1999 年年会上将搜索引擎使用的四种技术，即：1）根据网页本身信息（Author）；2) 根据超链链接关系（Other Author）；3) 人工编辑产生的目录系统（Editor）；4) 根据用户行为（User）；进行了比较，如图 10-1 所示，得出根据用户行为的技术比其它几种技术无论在查准率和查全率上都有相当多的优势。

2. 新词的产生

在本章的第一节，我们提到过词典 Σ，但没有强调它的重要性。事实上，它是以关键词为查询表示的任何信息检索系统的基础，中文尤其如此。在传统信息检索场合，例如图书馆，信息资源相对稳定，信息内容相对成熟，词典也就相对

稳定，没有表现出突出的矛盾。网络环境下，用户的信息需求很宽泛，特别是“时代感”很强，关注的内容与社会新闻和事件经常紧密相关，因此在查询中常常会有那些时髦的新词，例如“大腕”，“美眉”之类。从我们前面介绍的倒排表工作原理可知，如果词典中没有相应的词，就不可能查到含有它们的网页（严格说是“不能有效地”查到它们，原因见后）。因此，获得新词，将它们及时地加入词典中，是维护运行搜索引擎的一个重要工作。下面是关于这一问题的论述。

简单的讲，词典就是一个译码器，它分配给词典中的每一个条目一个唯一的整型编码。前面我们已经论述过词典在预处理和查询服务两个阶段的必要性。在这里，我们从系统设计的高度再分析一下词典的用途。我们将系统分为核心和外围应用，图 10-2 显示了词典的地位，它相当于系统内核和外围之间的一座桥梁。外围应用通常是和系统输入和输出（广义的）打交道的，它们面临的数据千差万别。如果让核心直接处理这些形态各异的数据，就会导致系统核心代码的急剧膨胀，系统运行效率迅速降低。通过词典的处理，将各种数据以统一的整型编码的形式交给系统内核，使得系统内核的处理简单，保证了系统的运行效率。

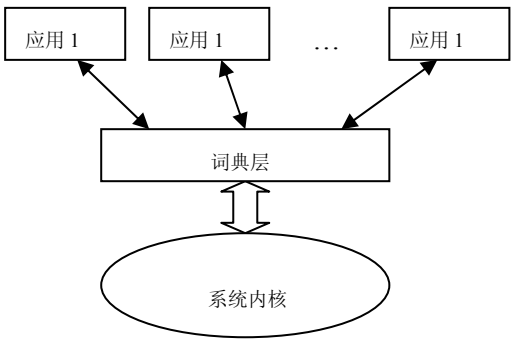


图 10-2 词典在系统中的地位

既然词典处于桥梁的地位，那么词典本身的设计就十分的关键。天网系统采取了Hash表的方法来实现系统的词典。对于每一个输入数据D_{input}，按照分布式搜集策略部分的分析，我们可以将其对应到一个大整数。我们记Hash表的大小为h_size，对任意一个输入数据，我们根据公式都可以获得它Hash表的入口地址：

$$Addr = F_{key}(D_{input}) = \text{int}(D_{input})\%h\_size$$

其中的F_{key}就是我们按照公式得到的Hash的散列函数。任何值域小于定义域的散列函数，都不能保证没有冲突，这就要求我们的散列函数造成冲突的概率尽量的小。通过选择适当的h_size和散列函数，我们可以控制Hash的平均拉链长度。

对于散列函数F_{key}，我们把 593,286 个项数据放入特征项大小为 100 万的Hash

表中散列，得到的平均拉链长度为 1.32。这意味着，我们对任何一个数据D_{input}操作时，需要的Hash相关的操作为 1.32 次。其它的相关操作，如F_{key}的运算、Hash表的增删改查，都是常数复杂度。因此，我们可以获得词典的运算复杂度为O(1)。

在进行中文分词时，就需要依赖中文词典。复旦大学在它们的文档分类系统[黄菁萱 and 吴立德,1998]中，使用基于 Church 提出的计算互信息和 χ^2 统计量的方法[Church and Hanks,1990]对文档进行专有词汇学习，所提取的专有词汇（5,000个）接近该系统使用的原始词典大小（11,000 个词条）的一半。结合关键词筛选的专有词汇学习技术使系统对开放语料的分类准确率提高了 15%。所以，按照需要不断扩大词典的容量是必须的。

如何扩大词典的容量？回顾我们对系统的分析，系统只有两个和外界数据的接口，即 Web 和用户检索。天网选择用户的检索进行学习，其理由如下：

首先，学习词汇是为了满足用户的检索需求，提高检索的质量。通过对用户检索数据的分析，从中学习新词，针对性强，更能提高检索的质量。

其次，从统计上来看，Web 上的数据和用户检索的字符串有着很大的差别。Web 网页中的中文大部分都是连写在一起的句子，以标点符号分开。而用户输入的检索字符串，根据我们的统计，大部分是词汇和词汇组成的短语。在网页中，面对一串连续的中文字符串，我们很难从中间提取出新词。而我们对用户的检索输入做一定的简单处理，即可以比较方便的学习到新的词汇。

我们可以通过如下步骤分析：

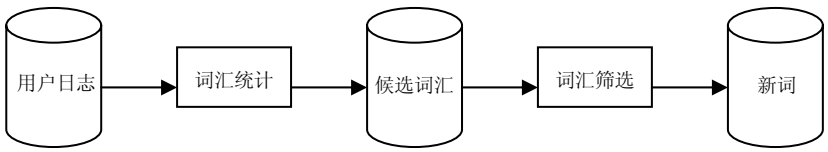


图 10-3 新词学习

词汇统计部分将用户的检索字符串进行一定的分析和筛选处理，并对通过筛选的词汇进行频率统计。1) 用户输入的检索有一部分是复杂的逻辑检索（大约20%），我们应该首先将这些带有逻辑运算符号的检索字符串转化为简单检索形式。2) 我们发现，检索中有大量的英文检索和中英混合检索，我们这里处理的是中文新词学习，因此我们要将所有的英文词汇过滤。3) 我们前面已经提到，对于过长的中文字符串，它是一个词汇的可能性极小，为了提高学习的准确度，我们定义一个学习词汇的最大长度 n，把所有检索字符串串长大于 n 的过滤掉。4) 我们对这些合法的“可能新词”进行学习，统计出每个词汇的检索频率。

词汇筛选部分主要有两个步骤。首先进行词频筛选，将低频的检索排除在新

词之外。众所周知，搜索引擎用户可以随便的检索任何内容，对于那些只有极少数人关心的生僻词汇，我们不用加入到词典中，因为词典的过度膨胀会在一定程度上降低系统运行的效率。另外，有很多的用户不能十分确定他们要检索的中文的正确拼写，或者在检索时不小心输入错误，如将“搜狐”输入成为“搜虎”。通过词频的筛选，可以在一定程度上解决这些问题。另外，统计得来的候选词汇，有一部分在我们的词典中已经存在，需要将它们过滤；还有一部分是两个或多个合法词汇组成的短语，一样需要将它们过滤掉，如“计算机网络”。

学得的新词将和搜集端从网页中提取的特征项共同组成新的词典，为以后的搜集和特征项提取服务。通过新词学习技术，使得系统对于新词检索的准确率大大提高，表 10-1 给出了新词学习前后检索准确率的对比数据。

表 10-1 新词学习对检索准确率的影响

词汇	学习前准确率	学习后准确率
克林顿	68%	82%
爱心社	49%	85%
宝洁	16%	73%
分布式	26%	60%
访美	58%	76%
非线性	66%	76%
编程	90%	94%
丰田杯	82%	100%
安阳	10%	50%
单片机	54%	88%

由于篇幅原因，我们仅列出了 10 个新词的检索结果. 通过我们对 100 个新词的检索结果的统计，我们获得更新前的平均检索准确率为 49.7%，更新后的平均准确率为 78.8%. 即搜索引擎系统对于新词的平均检索准确率提高了 58.6%.

第三节 相关排序的一种实现方案

将本章前述分析综合到一起，这一节我们给出相关排序的一种具体的实现方案。

一个网页是否重要，我们可以从其它网页上找出相应的线索。如果一个网页十分重要，那么会有大量的链接指向这个网页。因此，需要对一个还没有搜集的

URL 地址进行被链接次数的统计，以确定该 URL 获得的其它网页的评价，我们同时赋予其相应的权值W_l。另外，可以根据我们日常在网上的访问，来获得一些有价值的网站，加入到配置文件中。当一个网页属于这些重要网站时，我们就赋予它另外一个权值W_s。还有就是网页的编码类型。作为一个主要为华人服务的搜索引擎，我们主要的关注点在中文信息，所以我们应该优先搜集那些中文网页。即便是中文，仍然有不同的编码类型。例如，中国内地主要以 GB 为主，港台地区则以 Big5 为主，在北美及欧洲地区还存在 HZ 编码。按照搜索引擎服务的用户群，应该给相应的网页赋以不同的优先搜集次序，在我们的系统里，它体现为编码权值W_c。

基于以上三个主要方面的考虑，得到一个 URL 的权值评价：

$$W_t(p) = W_l(p) + W_s(p) + W_c(p)$$

这样，每个待搜集的网页都有自己的W_t，超链选择程序根据这些权值，从中选出一个或一批权值最大的来搜集，即达到了我们期望的目的[雷鸣,2000]。

一、 形成网页中词项的基本权重

前面提到了向量空间模型，但根据我们的讨论，并不能够将它完全照搬到搜索引擎系统中来。网页信息和正文文本最重要的差别就是在网页中含有大量的 HTML 标签（tag）。因此，我们在天网中提出了一个改进的 TF*IDF[Baeza-Yates and Ribeiro-Neto,1999]，[Church and Hanks,1990]算法用于检索和相关度评价算法。相对传统的 IR 而言，增加了对 HTML 标签和网页的可索引文本长度。可索引文本长度表示用户通过浏览器窗口看到的一个网页的文本内容长度。

考虑被 HTML 标签包围的一段文本内容，到底这些标签应该如何影响这段内容呢？天网将所有的标签分为两类：一类是影响文本权值的标签，如<FONT>、<H1>等；另一类是不影响文本权值的标签，如<IMG>、<FRAME>等。在此我们选择表 10-2 中的标签作为影响文本权值的标签。

表 10-2 影响权值的 HTML 标签

tag	Wt(tag)	tag	Wt(tag)
<TITLE>	40	<DT>	4
<CITE>	8	<LI>	4
<EM>	2	<UL>	4
<STRONG>	4	<A>	4
<B>	4	<FONT SIZE=7>	16
<I>	2	<FONT SIZE=6>	12
<BIG>	4	<FONT SIZE=5>	8
<H1>	12	<FONT SIZE=4>	4
<H2>	8	<FONT SIZE=3>	1
<H3>	4	<FONT SIZE=2>	1
<H4>	1	<FONT SIZE=1>	1
<H5>	1	<FONT SIZE=+1>	4

对于一个网页，首先给予该网页中的每个特征项一个缺省的权值W_0。如果一个特征项还被其它的有权标签包围，这些标签的权值将会影响到这个特征项的权值。例如，“hello”在<big>hello</big>环境中的权值应该为：

$$WBT = W_0 + W_t(big) + W_t(b)$$

通过此式，可以获得每个特征项在网页中每次出现的权值。假设特征项t在网页中出现n次，每次出现的权值分别为$WBT_1, WBT_2, \ldots WBT_n$，就可以得到特征项t在整篇网页中的权值：

$$WBT(t,p) = \sum_{i=1}^{n} WBT_i$$

对于一个网页，应用上式对每个特征项进行权值计算是公平的。但是，考虑到相同的特征项出现在不同的网页中，网页的长度越长，特征项可能获得的权值也就越高。所以，一个特征项的权值应该在某种程度上受到网页长度的影响。另外，为了区分高频词和低频词对网页的影响程度，我们沿用 IR 中的 IDF 项：

$$\begin{aligned} WB'(t,p) &= WBT(t,p) \times DS(p) \times IDF(t) \\ &= WBT(t,p) \times \log(S_{\max} / S(p)) \times \log(N / T(t)) \end{aligned}$$

S_{max}表示最大的网页可索引文本大小；S(p)代表网页p的可索引文本大小；N代表被索引网页的总量；T(t)是包含特征项t的网页的数量。

通过以上的分析，可以看到一个特征项的权值由三部分组成：第一部分是考虑了 HTML 标签影响的绝对权值；第二部分是考虑网页大小对权值的影响；第三部分是特征项出现频率对权值的影响。

最后，对WB(k,p)进行归一化处理。其中WB_{max}代表对于所有的k、p而言WB'(k,p)的最大值。

$$WB(k,p)=\frac{WB'(k,p)}{WB_{\max}}$$

WB(k,p)将作为基本权值来参与相关度评价的运算。

二、 利用链接的结构

网页之间的超链接是 Web 的基本特点，这也是从应用上区别现在的 Web 和以前的 Internet 最突出的特征。如果说 TCP/IP 协议组将上百万计算机无缝连接起来了，则 HTTP/HTML 协议组将上百亿信息（网页）无缝连接起来了。海量网页之间的相互链接形成了一个巨大的有向图，这个图的很多结构性特征既有趣，也有重要的意义。特别地，我们关心一个网页的入度。

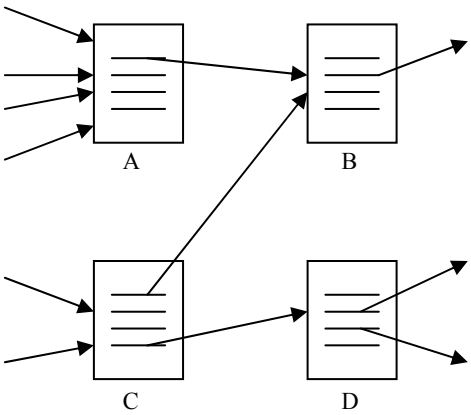


图 10-4 网页的互联结构示意

在这部分，我们主要考虑 WWW 中超链的互链关系对一个网页权值的影响。Web 有两个基本的构成因素：网页和超链。如果我们将网页认为是节点，超链是有向边的话，就可以将整个网络抽象为一个巨大的有向图。从图中可以看到，每个网页的入度是不同的，我们称每个网页的入度为网页的链接命中数（Link Hit Number, LHN）。

为什么 LHN 应该影响一个网页的权值呢？我们知道这些超链都是网页的编辑加入网页中的。他们之所以加入这些超链，是他们认为这些超链是有价值的，值得他们网页的浏览者去深入浏览。如果一个网页被大量的其它网页所链接（也就是说，被大量的网页编辑推荐），可以确定，这个被链接的网页是相对重要的，它们会对上网浏览的用户有更大的帮助。

所有的链接都应当按照如上所述的方法考虑吗？经过分析，天网将超链分为两类：链接向本网站内部网页的超链（自我推荐）和链接向其它网站上的网页的超链（他人推荐）。我们忽略第一类链接，理由如下：

1) 我们通过统计发现，很多网站的页面都是运用一定的页面模板实现的。在模版中会包含大量的该网站的索引超链，而这些超链会跟随模版被继承到该网站的每一个网页中。显然，这些超链不应该被考虑。
2) 有些大型网站（含有大量的网页）的主页会被本站点的其它网页大量链接，而获得很高的 LHN，尽管它有可能被极少的其它网站所链接。
3) 我们还要考虑网页编辑的欺骗行为。例如，它们在某些网页中包含大量的不可见链接指向自己的页面，进而获得较高的 LHN。

在下面的部分，我们将总的 LHN 值称作原始 LHN，将被其它站点链接的 LHN 值称为 LHN。

使用如上所述的策略，新网页将会面临不公正的待遇，因为一个新的网页，即使质量很高，由于知道它的网页编辑很少，也只能得到很小的 LHN 值。它们需要时间，才能被其他的网页编辑了解和评价。因此，对于新的网页，应用 LHN 去评价它们的优劣是不合适的。我们既然已经采用了 LHN 的方法，就应该按照一定的算法对新网页给以 LHN 的补偿。以下就是 LHN 补偿算法。

如果使用UNIX系统的时间格式(自 1970 年以来的秒数)，可以获得网页的发布时间T(p)，如果令当前时间为T_{now}，补偿的阈值时间为T_{st}，用如下公式获得补偿权值：

$$WLT(p)=\begin{cases}\log(T_{now}-T_{st})-\log(T_{now}-T(p)) & if \quad T(p)>T_{st}\\ 0 & otherwise\end{cases}$$

考虑了补偿权值后，得到新的 LHN 值：

$$WL'(p)=\log(LHN(p)+1)+WLT(p)$$

用WL_{max}来表示对于系统所有的p的WL'(p)的最大值，采用如下公式将LHN值进行归一化，得到期望的超链权值：

$$WL(p)=\frac{WL'(p)}{WL_{\max}}$$

三、 收集用户反馈信息

在搜索引擎中，当用户给出查询并得到一个返回结果列表之后，绝大多数的情况下他们都是扫描一下前面几个条目的摘要，感觉有他需要的内容，则点击对应的链接，去阅读网页全文。对来自于不同用户的同一个查询词来说，若某个链接虽然在返回结果表上出现的位置不太靠前，但被选取点击的次数比较多，于是系统应该感到该链接是比较受欢迎的，其位置应该往前调。举例来说，如果 80%输入查询词“北京大学”的用户都点击了输出结果的第 10 项，则系统应该认为第 10 项对于查询“北京大学”来说才是最相关的结果，应该将它排在前面。

具体实现起来，我们的办法是通过用户点击数（User Hit Number, UHN）来达到这一目的。对于一个检索q，会得到很多的检索结果网页，将这些网页表示为 $p_0, p_1, p_2, \dots, p_n$。假定检索q在一天内被提交了m次，定义$WUH_i$如下：

$$WUH_i(q,p)=\begin{cases}1 & if \ \text{hit by ith User}\\ 0 & \text{otherwise}\end{cases}$$

之后，我们定义检索 q 对应的一个网页 p 的 UHN：

$$WUD'(q,p)=\sum_{i=1}^{m}WUH_i(q,p)$$

但是，如果我们按照上述方法来统计用户评价权值的话，我们就忽略了返回结果中 URL 的位置信息。按照天网的统计，47.3%的用户只访问天网返回给他们的第一页（包含 10 个结果），12.2%的用户会继续访问第二页。这意味着一个结果在返回网页中的位置将会很大程度的影响用户点击的可能性。因此，如果一个结果网页在返回网页中排在尾部，即使这个网页和检索的相关度非常之高，它都只有很小的可能性被用户看到并点击。我们以一种补偿算法来弥补这个缺陷。这就是按照用户对每个返回页面访问的概率进行补偿。补偿因子 c(pos(q,p))按照表 10-3 来定义。考虑到补偿因子，定义计算公式（10-1）。

$$WUD(q,p)=\sum_{i=1}^{n}\left(c(pos(q,p))\times WUD_i(q,p)\right) \qquad (10\text{-}1)$$

公式（10-1）定义了检索q在一天的时间内，其结果页面p得到的用户评价，但是，这只考虑了一天的情况。应该如何来考虑长时期的用户评价呢？考虑n+1天的数据：$WUD_0, WUD_1, \dots, WUD_n$。最简单的方法是将这些数据求和得到这段时间的用户评价。

表 10-3 补偿因子定义表

页面号	pos(q,p)	被用户点击的概率	c (pos(q,p))
1	1-10	47.3%	1.00
2	11-20	12.2%	3.88
3	21-30	7.4%	6.39
4	31-40	5.0%	9.46
5	41-50	3.7%	12.78
6	51-60	2.9%	16.31
7	61-70	2.4%	19.71
8	71-80	2.0%	23.65
9	81-90	1.7%	27.82
≥10	≥91	≤1.4%	40.00

$$WUA'(q,p)=\sum_{i=0}^{n}WUD_i(q,p) \qquad (10\text{-}2)$$

但这是不合理的，因为用户在不同的时间感兴趣的网页是不同的。例如：当用户在奥林匹克运动会召开之前检索“奥林匹克运动会”，他会大量的点击那些讲述有关奥林匹克运动会准备情况和参赛运动员情况的网页。在奥林匹克运动会召开的过程中，大量的用户更关注关于世界纪录被打破的情况，各个国家获得的奖牌数和排名情况。当奥林匹克运动会结束后，他们的兴趣会转向一些有关奥林匹克运动会的评论的网页。如果仍使用公式（10-2），当用户在奥林匹克运动会进行时检索，他们将会看到大量的讲述有关奥林匹克运动会准备的网页排在前面，因为这些网页曾被给予了很高的用户评价。为了避免这个问题，我们使用一种衰减算法：

$$WUA(q,p)=\sum_{i=0}^{n}\left(k^i\times WUD_i(q,p)\right)$$

这里的 k 是衰减系数。系数 k 的值越大，先前的数据对结果的影响就越大。k=0 和 k=1 是两个极端情况。k=0 表示历史的数据不被考虑；k=1 表示所有的历史数据都和现在的数据有相同的重要性。

搜索引擎面对的是数以亿计的用户，而上式需要搜索引擎保留所有的历史数据，这样的代价十分巨大。所以我们通过转化成公式（10-3）来解决这个问题。

$$WUA(q,p)=\sum_{i=0}^{n}\left(k^{i}\times WUD_i(q,p)\right)$$

$$=k^{0}\times WUD_0(q,p)+k^{1}\times WUD_1(q,p)+\mathrm{K}+k^{n}\times WUD_n(q,p)$$

$$=WUD_0(q,p)+k\times WUA_1(q,p) \qquad (10\text{-}3)$$

利用一个递归程序，我们只需要记录两个数据：历史数据WUA_1 和当前数据WUD_0.

当一个新的网页刚刚被索引时，它没有被用户点击的机会，所以如果采用上述的方法时，它的用户评价值会是零，需要给予它们一些补偿。办法是给一个新的网页一个缺省的用户评价值：

$$WUC(q,p)=l\times WUA_{\max}(q)$$

补偿系数l反映了对于一个新的网页的重视程度。$WUA_{max}(q)$ 是对于所有的p取值最大的WUA(q,p) 值。考虑了补偿之后，新的用户评价值由如下公式（10-4）计算。

$$WU'(q,p)=WUA(q,p)+WUC(q,p) \qquad (10\text{-}4)$$

我们仍然需要对这个值进行归一化得到公式（10-5），其中WU_{max} 代表对于所有的p的WU'(q,p) 的最大值。

$$WU(q,p)=WU'(q,p)/WU_{\max} \qquad (10\text{-}5)$$

四、 计算最终的权重

以上已经给出了如何得到一个检索的相关网页集合，下面的工作是计算每个网页和查询 q 的相关度。相关度运算依赖三个方面，它们分别是：基本权值、链接权值和用户评价权值。首先计算每一个结果网页 p 的基本权值 WB(q,p)。

按照第一节的论述，每一个查询q可以被分解为m个特征项 $\{t_1, t_2, \ldots t_m\}$ 的逻辑运算。因此，对于每一个结果网页p，都可以获得每一个特征项在该网页中的权值$WB(t_i, p)$。我们按照如下方法定义权值的逻辑运算：

$$\mathop{\mathrm{I}}_{i=1}^{m} WB(t_i,p)=\sum_{i=1}^{m}WB(t_i,p)/m$$

$$\sum_{i=1}^{m} WB(t_i, p) = \max_{i=1}^{m}(WB(t_i, p))$$

任何一个用户的检索都可以表示为特征项的与（∩）和或（∪）的运算表达式，因此我们得到相应的权值运算公式（10-6）。

$$WB(q, p) = R\big(WB(t_0, p), WB(t_1, p), ...WB(t_m, p)\big) \qquad (10\text{-}6)$$

R 就是相应的逻辑运算表达。根据公式(10-2)，通过一定的运算过程，可以得到一个网页 p 的对应查询 q 的基本权值 WB(q,p)。

对于一个查询 q，还需要考虑链接权值 WL(p)和用户评价权值 WU(q,p)。有两种方法来处理这三者之间的关系：

1) 让 WB(q,p)作为基准权值，而链接权值和用户评价权值作为比例系数：

$$W(q, p) = WB(q, p) \times WL(p) \times WU(q, p)$$

2) 每一种权值按照一定的比例重新构成新的权值：

$$W(q, p) = k_{WB} \times WB(q, p) + k_{WL} \times WL(p) + k_{WU} \times WU(q, p)$$；

其中， $k_{WB} + k_{WL} + k_{WU} = 1$

天网选择了第二种方法。在这种方法里，每种权值都起到了影响结果权值的作用，但是，它们的影响又都被限制到一定的范围内。第二种方法的优点如下：

1) 几乎所有的网页拥有者，尤其是商业网站，期望它们的网页被排在搜索结果的前列。一些网站就利用一些手段来欺骗搜索引擎。例如，它们通过向网页中加入一些不可见的文本来提高它们基本权值部分的值：一个旅馆为了扩大影响，将它们的主页中加入大量的不可见的词汇“计算机”。如果这样，当用户检索计算机时，这个旅馆的主页将堂而皇之的排在前几个结果中。利用方法第二种方法可以在一定程度上避免这种欺骗。
2) 如果忽略一个站点内部的链接，这就使得网站的作者很难通过超链权值对搜索引擎进行欺骗。一般来讲，其它网页的编辑不会愿意以牺牲自己网页质量的代价来将一些不相关的超链加入到自己的网页中。
3) 用户评价权值也是一个容易被用来欺骗搜索引擎的特性。一些网站的所有者或许会雇佣一些职员，不停的在检索结果页面中点击它们自己的网页。如果按照方法一，经过一定的时间，这个总被点击的网页最终将升到第一的位置。但如果使用方法二，就可以有效的避免这个问题，因为用户的影响已经被限定到了一个合理的范围。

总的来讲，结果排序是搜索引擎技术最重要的一个方面，从概念上讲，主体

是本章第二节讨论的那些因素，但在实际系统中会衍生许多变化，其细节常常是商业机密。

第四节 搜索引擎系统质量评估

检索质量评估的目标是对不同搜索引擎系统的检索结果评估其相对优劣次序，本节介绍这方面的一般技术与方法。

一、 引言

公开有效的搜索引擎质量评估对指导用户选择搜索服务，对搜索引擎服务提供者与研究人员不断尝试新技术，提高服务质量十分重要。商业搜索引擎内部通常会有质量评估，一般不会公开。这一方向的工作与研究主要由信息检索领域的研究人员推动。

信息检索可以看作这样的过程和方法，通过它，一个需要信息的用户可以把他的信息需求转换成为对数据集中若干文档的引用，从而找到有用的信息。评估从这个研究方向创立开始就一直为人们关注。根据评估对象的不同，可以分为 6 个级别[Saracevic,1995]：工程级关注系统的效率，具体的评估指标和方法参见[Jian,1991]；输入级关注输入数据的覆盖率；处理级评估数据处理过程中的算法、技术和方法的效果；输出级评估结果输出后的交互、反馈等；应用级和社会级评估系统的应用和对生产率的影响。其中前三个级别的评估以系统为中心，后三个以用户为中心。进行评估有如下几个前提要求：被评估的系统，包括算法和数据；评估准则，如信息检索中的相关性；评估指标，如信息检索中的查准率和查全率；评估指标的获取，如信息检索中的相关性判别；评估方法，包括整个过程的设计与组织。

目前信息检索领域最重要的评估工作由 TREC 组织。TREC 主要在处理级进行检索效果评估，把相关性作为评估准则。相关性是一种复杂的感知和社会现象，它不是简单二元的是否判别，和环境、上下文有密切联系。相关性判别通过专门的评估人员人工判断，在大规模数据集上进行检索系统评估时，这种人工的工作成为最大的开销，并且使得对整个数据集进行完全的相关性判别成为不可能。为了让评估实验可重复，TREC 建立了大规模的评估数据集，包括数据集，查询集和相关结果集。其中相关结果集通过一种称为 pooling 的方法构造，对每个评估用的查询，从所有参加评估的结果序列中挑选出前一部分，人工判断其相关性，其它文档作不相关处理。在 TREC 的推动下，评估方法的研究得到了很大发展。文献[Zobel,1998]研究表明 pooling 方法下对系统相对性能评估具有稳定性，但同

时查全率被估计过高。文献[Voorhees and Buckley,2002]通过统计实验错误率研究了查询集大小对评估的影响，指出小的查询集会使评估结果具有高错误率；文献[Buckley and Voorhees,2000]研究了查询集大小、评估指标对评估结果稳定性的影响，指出对于 Web 环境，适合使用 P@10(前 10 个结果的精度)或 P@20 作评估指标，同时需要较多的查询以减少错误率，100 个查询对 P@20 较合适。文献[Voorhees,2001]研究了多级的相关性判别对评估的影响，指出 Web 环境下对高度相关的评估和一般相关的评估结果不一致，相比更不稳定。文献[Cormack, et al.,1998]研究了大规模评估数据集构建的技术，提出了改进的 pooling 方法。

Web 搜索引擎与传统信息检索有许多不同的特点，这些给传统的信息检索评估带来了新的挑战。Web 搜索处理的数据是整个 Web，即使采用 pooling 方法，TREC 建立静态评估测试数据集的方法也很难扩展到这样的数据规模。同时 Web 数据还在不断动态变化，对搜索引擎的评估很难建立在同样一个静态的数据集上，如何评估不同数据集上检索系统的质量也是一个新的研究内容。文献[Hawking, et al.,1999]研究了 TREC 的信息检索方法在 Web 环境下是否有效，不过实验建立在 TREC 的大规模数据集上，具有一定的偏向性。文献[Hawking, et al.,2001]对 11 个搜索引擎进行了评估，指出不同搜索引擎有显著性差异，不同评估指标间高度相关；特别提出了对未来 Web 搜索引擎质量评估需要针对不同用户的信息需求类型采用不同的评估技术。文献[Singhal and Kaszkiel,2001]详细分析了 TREC Web Track 的评估方法在 Web 环境下的不足，包括查询类型、相关性判别是按文档为基础，还是以用户为中心，按站点为基础，不同数据集上的评估结果是否可以比较等问题。文献[Craswell, et al.,2003]是 2003 年 TREC Web Track 的报告，分为主题提取类型和导航类型两种任务模式。主题提取类型以 P@N(前 N 个结果的精度)为评估指标，返回结果以站点为单位，以主题相关性和内容质量为评估准则。导航类型以 MRR(第一个正确答案的平均序号倒数)和 S@10(答案出现在前 10 个结果的查询比例)为评估指标。

二、 查询类别分析与查询集的构建

用户信息需求千差万别，以不同查询表达，这种类型差异对于检索系统十分重要，因为不同类型的需求可以有不同的检索方法更好的完成，对系统评估也同样如此，不同类型的用户信息需求和查询需要采用不同的评估方法。

对用户查询有不同的分类方法。文献[eTesting,2000]从查询语法特征上划分为 5 类：自然语言查询，单个查询词的简单查询，多个查询词的简单查询，包含操作符的复杂查询和主页查询。文献[Travis and Broder,2001]把用户信息需求分为三类：信息型，导航型和事务型。其中信息型是寻找主题相关的文档，也就是传统 TREC 评估的 ad-hoc 任务；导航型是寻找知道名称的站点或主页，对应 TREC 评

估的 homepage/named page 任务；事务型指用户期望找到一个服务入口，需要进一步进行服务访问，比如公交线路查询、歌曲下载、产品信息查询等。文献[Kang and Kim,2003]也使用同样的三个类别，研究了具体的分类识别算法。TREC 的评测不区分信息型和事务型。

可以通过对搜索引擎用户查询日志的统计分析，逆向推断用户的信息需求，得到查询类型的分布。对不同搜索引擎系统用户查询日志大量统计研究都表明，搜索引擎平均用户查询词长度很短，平均长度在 2~3 个词之间。短查询很难充分表达用户的信息需求，是搜索引擎为提高系统性能面临的一个重要挑战。这也为从查询日志中逆向推断用户信息需求带来了困难。短查询词在不同的上下文环境下存在多义，即使对相同含义，从用户角度也隐含着不同的信息需求。例如，查询“绿茶”，可能是指电影名，也可能是指茶叶；指电影名时，用户可能希望下载或观看，也可能希望了解影评等相关信息，这就分别对应上述事务型和信息型的查询类型。在统计过程中，使用普遍情况最大可能的用户需求作为查询类别的判断准则，如表 10-4 所示。

表 10-4 用户查询信息类别

类别	用户需求	判别准则
导航型	寻找指定名称的站点或主页	指定公司、机构、网站名称的查询
信息型	寻找主题相关的文档页面	页面内容可以满足信息需求的查询;人名、小说、电影查询归到这一类
事务型	寻找到一个服务入口，需要进一步进行服务访问	对软件、图片、音乐等资源的下载查询；信息服务的查询；

在一个查询样本中，可对不同类别的查询分别进行筛选，构建查询集。先去除那些可能对评测人员产生困扰的不良查询，比如性、暴力方面的查询；再各挑选查询意图明确的若干个查询（如 50 个），构成评估用的查询集。在 Web 搜索环境下，短查询现象普遍，因此很难对挑选出的每个查询补充上确切的查询意图描述，在实验过程中可以统一使用表 10-4 中的评估准则。

三、 评估实验的建立与分析

搜索引擎检索质量评估的目标是对不同搜索引擎系统的检索结果评估其相对优劣次序。对单独一个系统的评估，得到的评估指标的得分一般没有实际意义。搜索引擎的搜集和检索两大部分的性能对最终的检索质量都有影响。[Hawking, et al.,2001]指出以 P@N 为评估指标时，指定结果个数 N，检索精度随着文档集合大

小增长而增长。并且评估对象搜集的网页范围、数量都不相同，这种差异对评估有一定的影响。

可以考虑在实验中采用一种归一化的方法，把查询结果限定在一个固定的集合内，用来减小不同评估对象的搜集系统差异对检索效果带来的差异。例如采用InfoMall[InfoMall,2004]系统的网页集合为基准，对所建立的查询集，用一些工具抓取几个不同搜索引擎的前若干个（如 50 个）检索结果，同时向 InfoMall 系统请求这些结果 URL 的历史网页，当 InfoMall 系统的访问界面返回错误，通知一个 URL 不存在时（HTTP 的 404 错误码），则表明此检索结果不在 InfoMall 网页集合中。

评估实验的基本原则是盲测试性和可重复性。盲测试性是指评估实验应该尽量避免用户的主观偏向性，要求进行相关性评分的评测员不能区分评分对象属于那一个评估对象，甚至不知道评估对象差别的存在。如在实验中可以把抓取的不同搜索引擎的检索结果保持原始的顺序随机混合，即每个查询的结果序列（如只保留在 InfoMall 中命中的结果），对相同序号上的结果随机排列，合并成一个序列。同时使用统一的摘要提取程序，从网页原文中按查询词提取摘要，以相同的形式展现给评测人员，而且事先不透露评估对象的名称。这一过程可保证实验的盲测试性。

可重复性是一个科学实验的基本要求，TREC 的评估体系在这一方面做的很好，通过构建数据集，包括文档集、查询集和相关结果集，使得评估实验具有可重复性。Web 检索下，这一方法面临的最大问题是数据规模难于扩展，特别是对适合 Web 数据评测规模的文档集，难于有效构造出其中查询对应的相关结果集。TREC 的 pooling 方法提高了这一工作的效率，但仍然不能适应数据规模的进一步增长。实验中，可以由自愿参加评测工作的评测员，按相关性准则对查询结果的相关性评分。这一过程并不构建一个完全的相关结果集，对一个新的评估对象进行重复实验还必须让评测员重新对变化的查询结果进行相关性评分。

评估实验选择 DCV(Document cut-off value)类型的评估指标。检索性能评估通常基于 P(查准率)和 R(查全率)，有两类指标，一是 P-R 曲线和根据 P-R 曲线计算的平均精度 AVP，这一类指标对不同的查询在同一个查全率上计算平均精度；另一类是 DCV，它使用相同的评测过的文档数量进行归一化，用以表达不同查询结果中用户在同样的浏览代价下的性能。

对大规模数据量的 Web 检索评估，建立基于 Web 的评测环境和招募大量的自愿者参加评估是解决评估的数据规模扩展性问题的一个可能方法。实验中，通过使用相关度评测环境和工具，由志愿评测员对查询集合由各个搜索引擎返回的检索结果进行相关性判别。

对实验结果进行分析: 首先进行异常数据清理，然后根据评估指标计算出不同搜索引擎的分值，比较不同搜索引擎的检索质量优劣。由于实验过程存在的随

机因素，在结果比较分析中，还必须做统计的显著性检验以及实验的错误率分析。对两个不同评估对象，假设根据评估指标计算出的指标得分的差异符合正态分布，可以使用成对数据的 t 测试进行显著性检验，在实际应用中，即使正态分布假设并不成立，t 测试也基本有效[Hull,1993]。实验可以采用成对数据的 t 测试，通常置信度取 95%，对通过测试的结果认为存在显著性差异，否则没有显著性差异。实验的错误率是指在重复实验中，不同次实验得到相反的评估结果的次数与实验总次数的比率，它代表一次评估实验得到错误结果的可能性。[Buckley and Voorhees,2000]给出如下计算公式：

$$ErrorRate = \frac{\sum Min(|A > B|, |A < B|)}{\sum (|A > B| + |A = B| + |A < B|)}$$

其中 A,B 为评估对象，|A>B|表示重复实验得到 A 优于 B 的次数。如果实验在某一变化因素控制下，计算错误率，可以用来评估实验对这一变化因素的稳定性，如对评测员、查询集合大小以及评测指标等。

下篇　面向主题和个性化的 Web 信息服务

Web 信息服务就是根据用户的信息需求,为其提供相应的 Web 信息；而基于主题和个性化的信息服务就是指在特定的主题范围内, 能够根据用户个人独特的信息需求, 从互联网上搜索出有关的信息,并将它们整合在一起, 以便有针对性地满足各种不同用户的信息需求。目前，针对某一领域的小型主题搜索引擎、个性化智能搜索引擎的研究已成为下一代搜索引擎的两个研究热点。

本篇介绍我们在这方面所做的一些研究性工作，主要包括：

1. 中文网页分类技术：现已成为中文 Web 信息处理领域的基础性工作，例如将网页进行自动分类，可以为搜索引擎用户提供目录导航服务，进而提高系统的查准率。本篇第十一章将详细介绍中文网页分类的各种算法，并比较这些方法的优劣。利用网页分类实现搜索引擎的个性化查询服务是一项值得进一步研究的工作，其基本思想是：先对网页和不同用户分别进行分类，之后对两者进行类型匹配，从而实现个性化服务。
2. 个性化信息服务：Web 挖掘技术是实现 Web 个性化服务的核心技术之一，本篇第十二章首先介绍了这方面的研究进展；然后介绍了我们开发的天网知名度系统，该系统针对包含某一对象类的网页提供个性化检索服务，并在可能的时候以简报的形式推送给注册的用户。
3. 主题信息的搜集与应用： 第十三章首先对主题信息的搜集方法做了一个概括性介绍；然后提出了一种主题信息的搜集与处理模型，基于它并针对人们关心的一个热点主题，系统地对网上的信息进行搜集和分析，从不同的角度和层次得出互联网对该主题报道的强度。

第十一章 中文网页自动分类技术

网页自动分类技术已经成为 Web 领域的一个研究热点。本章主要讨论如何应用有指导的机器学习方法实现大规模中文网页的自动分类，以及如何应用中文网页自动分类方法实现搜索引擎目录导航服务。

第一节 引言

为了能够有效地组织和分析海量的 Web 信息，人们希望能够按照其内容实现对网页的自动分类。目前，网页自动分类技术在数字图书馆、主题搜索、个性化信息检索、搜索引擎的目录导航服务、信息过滤、主动信息推送服务等领域得到了广泛地应用。

在信息检索领域，评价一个系统的性能，通常有效果和效率两个方面的考虑。与此对应，评价一个分类器性能的优劣，通常也有两个基本的指标：分类质量（效果）和分类效率。对于分类质量，考察的指标通常为查准率和查全率；对于分类效率，考察的指标通常为分类器的训练效率和分类器的实际分类效率。分类质量和分类效率这两个指标，既相互独立，又相互影响。在理想的情况下，人们追求分类器不但要具有较高的分类质量，而且还要具有较高的分类效率。但是在实际的应用中，有时为了追求分类质量而不得不牺牲分类效率，有时为了保证一定的分类效率而不得不在一定范围内牺牲分类质量。因此，在设计分类器时，需要根据具体的应用环境来综合考虑这两个指标，重点解决主要矛盾。

本章首先系统地定量分析影响分类器性能的各种关键因素。根据实际的测试结果，并结合搜索引擎这一特定的应用环境，来寻找一种中文网页分类器的最佳设计方案：在具有较高分类质量的同时，还具有较高的分类效率。然后，根据这个方案实现了一个能够处理海量中文网页信息的分类器。最后，应用该分类器实现了天网搜索引擎中的目录导航服务[冯是聪,2003]。

第二节 文档自动分类算法的类型

在 Web 出现之前，人们已研究过许多普通文档分类的方法，形成了各种文档自动分类（Automatic Text Categorization，ATC）技术[Yang and Liu,1999]。随着

海量网页信息的涌现，ATC 技术的处理对象从普通文档扩展到网页信息，自然地，ATC 技术成了实现网页自动分类的基础。所谓文档自动分类就是用计算机程序来确定指定文档和预先定义类别之间的隶属关系[Sebastiani,1999]。

目前，主要的文档自动分类算法可以分为三类：

1) 词匹配法。词匹配法又可以分为简单词匹配法和基于同义词的词匹配法两种。简单词匹配法是最简单、最直观的文档分类算法，它根据文档和类名中共同出现的词决定文档属于哪些类。很显然，这种算法的分类规则过于简单，分类效果也很差。基于同义词的词匹配法是对简单词匹配法的改进，它先定义一张同义词表，然后根据文档和类名以及类的描述中共同出现的词（含同义词）决定文档属于哪些类。这种分类算法扩大了词的匹配范围，在性能上要优于简单词匹配法。不过，这种算法的分类规则仍然很机械，而且同义词表的构成是静态的，对文档的上下文不敏感，无法正确处理文档中其具体含义依赖于上下文的词，分类的准确度也很低。

2) 基于知识工程的方法。基于知识工程的文档分类方法，需要知识工程师手工地编制大量的推理规则，这些规则通常面向具体的领域，当处理不同领域的分类问题时，需要不同领域的专家制定不同的推理规则，而分类质量严重依赖于推理规则的质量。因此，在实际的分类系统中较少使用基于知识工程的学习法。

3) 统计学习法。统计学习法和词匹配法在分类机制上有着本质的不同。它的基本思路是先搜集一些与待分类文档同处一个领域的文档作为训练集，并由专家进行人工分类，保证分类的准确性，然后分析这些已经分好类的文档，从中挖掘关键词和类之间的联系，最后再利用这些学到的知识对文档分类，而不是机械地按词进行匹配。因此，这种方法通常忽略文档的语言学结构，而用关键词来表示文档，通过有指导的机器学习来训练分类器，最后利用训练过的分类器来对待分类的文档进行分类。这种基于统计的经验学习法由于具有较好的理论基础、简单的实现机制、以及较好的文档分类质量等优点，目前实用的分类系统基本上都是采用这种分类方法。

本章介绍的文档分类算法都属于统计学习法。根据分类结果的不同，基于统计学习法的分类系统在整体上可以被分为两类：独立二元（Independent Binary）分类系统和 m 元（m-ary）分类系统[黄菁萱 and 吴立德,1998]。所谓独立二元分类，就是给定一篇文档，分类系统对每一个类都独立地判断这篇文档是否属于该类：要么属于，要么不属于，而不存在其它的结果，并且在分类过程中，不同类别之间互不影响。所谓 m 元分类就是给定一篇文档，系统计算这篇文档与所有预

先定义的类的相似度，并按这篇文档和各个候选类的相似度排序，最后输出候选类列表。文档分类算法如图 11-1 所示，在第四节我们介绍其中几个典型的分类算法。

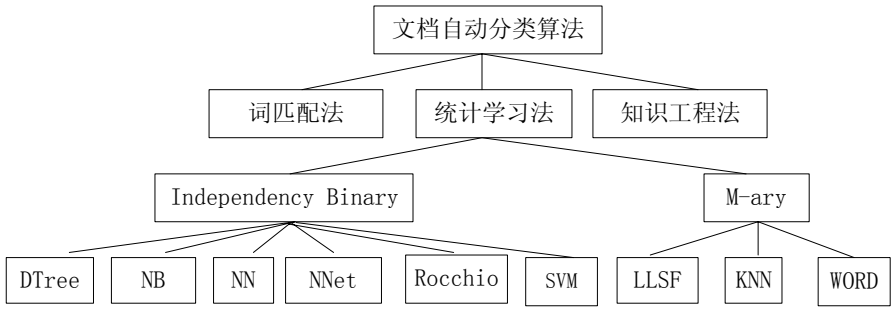


图 11-1 自动文档分类算法的分类

第三节 实现中文网页自动分类的一般过程

在应用基于案例的有指导的机器学习方法实现中文网页自动分类的过程中有一个基本的假设：文档的内容与其中所包含的词有着必然的联系，同一类的文档之间总存在多个共同的词，而不同类的文档所包含的词之间差异很大。因此，分类器的**训练过程**可以看作是在已知文档类别的情况下，统计不同类别内的词的分布，即在预先定义的类别集合C（$C=\{c_1, \ldots, c_k, \ldots, c_m\}$）与词项集合T（$T=\{t_1, \ldots, t_k, \ldots, t_n\}$）的幂集之间建立一种加权的映射关系，形成一种向量表示；相应的，分类器的**分类过程**，可以看作在已知一篇文档内所包含词项分布（用一个向量表示）的情况下，和在训练中形成的每个类别的向量表示进行对比，来确定该文档与类别的隶属关系。

根据对文档分类过程实质的分析，下面给出中文网页自动分类的一般过程。同普通英文文档相比，中文网页信息具有特性：⑴ 中文网页的内容使用中文书写，不像英文单词之间存在自然的形态间隔，中文需要分词处理。而且分词的效果能够显著地影响分类效果；⑵ 网页使用超文本设计。它包含大量的 HTML 标签和超链接。我们有可能利用这些信息来改进分类的质量。比如包含在标题<title>标签内的内容通常要比出现在网页正文<body>标签内的内容要重要的多。在 Web 上相邻的网页通常具有相关或相同的主题，因此网页之间的超链信息也可以给我们一些启发；⑶ 网页通常包含大量的“噪音”。同普通文本相比，网页的设计比较随

意，通常包含各类广告，设计人员的注释以及版权申明等无关信息。有时同一个网页甚至会包含多个不同的主题。在进行分类之前，需要自动清除这些“噪音”，否则这些“噪音”会降低分类质量。因此，需要对中文网页进行预处理后，才能应用相应的文档自动分类算法实现分类。

结合中文网页的特性，图 11-2 给出了中文网页自动分类的一般过程。其中：预处理过程主要包括中文分词以及网页内“噪音”的清除等处理；基于二元分类算法的分类器，可以把分类结果直接作为待分类网页的类别结果，而基于 M 元分类算法的分类器，还需要对该分类结果进行进一步的筛选后，才能作为待分类网页的类别结果。

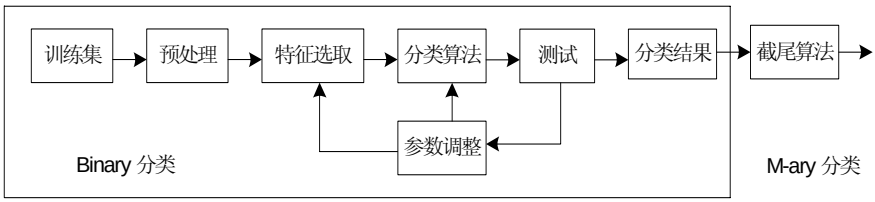


图 11-2 中文网页自动分类的一般过程

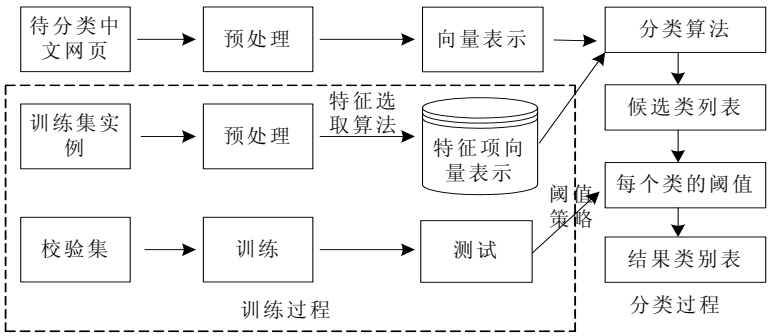


图 11-3 中文网页分类器的工作原理图

根据图 11-2 所示的中文网页分类的一般过程，我们设计了本章研究所使用的分类器，其工作原理如图 11-3 所示。从总体上，分类器的整个工作周期可以分成训练过程和分类过程。在训练过程中，训练集实例经过中文分词和特征选取处理后被表示成向量形式。该特征向量集用来描述类别模式，在分类过程中使用。校验集是训练集的一部分，通过应用相应的阈值策略来预先确定每个类别的截尾阈值。在分类过程中，一个待分类的中文网页经过中文分词并表示成向量后，应用

分类算法同训练过程得到的类别模式逐一比较，得到候选类别列表，然后同训练过程中得到的每个类别的阈值相比较，保留大于阈值的类别，并作为该网页的分类结果。

从图 11-3 可以看出，构建一个分类器的关键因素包括：预处理、训练集、特征选取算法、分类算法和截尾算法等。预处理部分的 HTML 网页净化方法已在第七章中介绍，如下将逐一定量地分析后 4 个因素对分类器性能的影响。

第四节　影响分类器性能的关键因素分析

一、 实验设置

为了定量地分析影响分类器性能的关键因素，我们首先实现了一个最基本的中文网页分类器。该分类器的具体设计方案如下：

1) 预处理。在预处理阶段，除了进行中文分词处理外，没有进行其它任何预处理；
2) 特征选取。在这里，直接把中文分词得到的所有关键词作为特征项，并由这些特征项构成特征向量，因此没有特征选取处理过程。
3) 分类算法。我们选用 kNN（k-Nearest Neighbor）分类算法来实现基本的分类器。在实验中我们取 k = 20，即仅保留相似度最大的 20 个实例网页。为确定待分类网页的类别，首先需要把具有相同类别的实例与待分类网页之间的相似度相加作为待分类网页的类别相似度，最后把相似度最高的类别作为该网页的结果类别，所以这里每个待分类网页仅取一个结果类别。
4) 截尾算法。因为上面的分类算法为每个待分类网页仅取一个结果类别，所以这里无需对分类结果应用截尾算法。
5) 分类质量的评价指标。在信息检索领域，通常采用查准率和查全率，人们通常借鉴这些标准来评价分类系统的优劣。

查准率表示在所有被检索出的文档结果集中，真正符合检索意图的文档所占的比率，它体现了系统检索结果的准确程度。查全率表示被检索出的文档集结果中真正符合检索意图的文档数在所有符合检索意图的文档集中所占的比率，它体现了系统检索所有相关文档的完备性。查准率和查全率这两个标准是互补的，单纯提高查准率就会导致查全率的降低，反之亦然。因此，尽管一个好的检索系统应该同时具有较高的查准率和较高的查全率，但是实际的检索系统往往需要在两

者之间做出一些折中，而避免其中一个指标过低。

为方便起见，人们还定义了一个F_1值[Yang and Liu,1999]，用以反映查准率和查全率的综合效果，其定义如公式(11-1)所示。根据计算方式的不同，F_1值可以分为宏观F_1值（Macro-F_1）和微观F_1值（Micro-F_1）。宏观F_1值的计算方式：首先需要根据公式（11-1）分别计算每个类别的F_1，然后再根据公式（11-2）来计算它们的平均值，即为宏观F_1值。此外，宏观F_1值还有一种计算方式：首先计算p和r的平均值，然后代入公式（11-1）来求宏观F_1值，这个过程可以用公式（11-3）来表示，本文将采用这种方式来计算分类器的宏观F_1值。微观F_1值的计算方式：首先需要在整个测试网页集合内分别统计p和r的值，然后根据公式（11-1）计算微观F_1值。

$$F_1 = \frac{2pr}{p+r} \quad (11\text{-}1) \qquad Macro - F_1 = \frac{1}{m}\sum_{i=1}^{m} F_{1i} \quad (11\text{-}2)$$

$$Macro - F_1 = \frac{2 \times \sum_{i=1}^{m} p_i \times \sum_{i=1}^{m} r_i}{\left(\sum_{i=1}^{m} p_i + \sum_{i=1}^{m} r_i\right) \times m} \qquad (11\text{-}3)$$

其中：p 为查准率；r 为查全率；m 为训练集类别数，这里为 12。虽然在我们使用的分类体系中共包含 733 个类别（样本集中类别及实例数量的分布情况详见表 11-2），但是为简单起见，我们把子类的分类结果分别统计到 12 个大类中，所以最后共有 12 个类的分类统计结果。

对于F_1值，从公式（11-3）可以看出，它反映了查准率p和查全率r之间的平衡关系：只有当p和r比较接近，并且取值都比较大时，F_1才比较大。反之，当p和r相差比较悬殊，或者取值都比较小时，F_1值就比较小。所以，F_1综合反映了分类器的整体性能。本章将使用宏观F_1值和微观F_1来评价分类器的质量。

二、 训练样本

为了推进信息检索领域的发展，由美国国家标准和技术研究院（NIST）、信息技术实验室（ITL）检索小组、美国国防部高级研究计划署（DARPA）信息技术处、高级研究开发机构（ARDA）等单位共同发起了有全球影响的信息检索会议 TREC，自 1992 年起每年一次；TREC 会议实际上是文本信息检索系统的擂台赛，可以说，在 TREC 上展示的文本分类系统代表了文本分类领域的最新研究成果。一些大学，如 CMU、BERKLEY、CORNELL 等和一些公司带着自己开发的文本分

类系统参加会议，由大会使用相同的训练集和测试集对这些系统进行评测。中国科学院计算所、清华大学、复旦大学等单位近几年也有派队参加，并取得了不错的成绩。同时我们注意到，由于 Web 技术的发展，TREC 也逐步开始提供标准的英文网页语料来评测 Web 信息检索系统。

表 11-1 样本集中类别及实例数量的分布情况表

类别编号	类别名称	类别数	训练样本数	测试样本数
1	人文与艺术	24	419	110
2	新闻与媒体	7	125	19
3	商业与经济	48	839	214
4	娱乐与休闲	88	1510	374
5	计算机与因特网	58	925	238
6	教育	18	286	85
7	各国风情	53	891	235
8	自然科学	113	1892	514
9	政府与政治	18	288	84
10	社会科学	104	1765	479
11	医疗与健康	136	2295	616
12	社会与文化	66	1101	301
共计		733	12336	3269

与面向英文的分类系统相比，中文分类系统的起步比较晚。从第五次TREC会议开始，增加了对中文分类系统的评测。实际上参加TREC-5 的中文分类系统处理的重点还停留在中文的分词问题上，而且处理的对象还是新华社的新闻稿这类普通的中文文本。基于案例的有指导的机器学习方法是实现中文网页自动分类的理论基础。因此，中文网页训练集是实现中文网页自动分类的前提条件。但是，到目前为止，还没有出现标准的中文网页语料库，因此也没有出现针对中文网页分类系统的评测。为了解决这一问题，我们通过动员不同专业的几十个学生，人工选取形成了一个基于层次模型的大规模中文网页样本集[1]。它包括 12,336 个训练网页实例和 3,269 个测试网页实例，分布在 733 个类别中，每个类别平均有 17 个训练实例和 4.5 个测试实例。样本集中类别及实例数量的分布情况如表 11-1 所示。

[1] 天网免费提供网页样本集给有兴趣的同行，燕穹产品号：YQ-WEBENCH-V0.8

此外，为了搜集网页的方便，我们开发了一个网页实例集的搜集和整理的工具 WebSmart，如图 11-4 所示。经过实际应用的检验，表明该工具操作方便，使用它可以非常方便地实现网页实例集的搜集和整理。

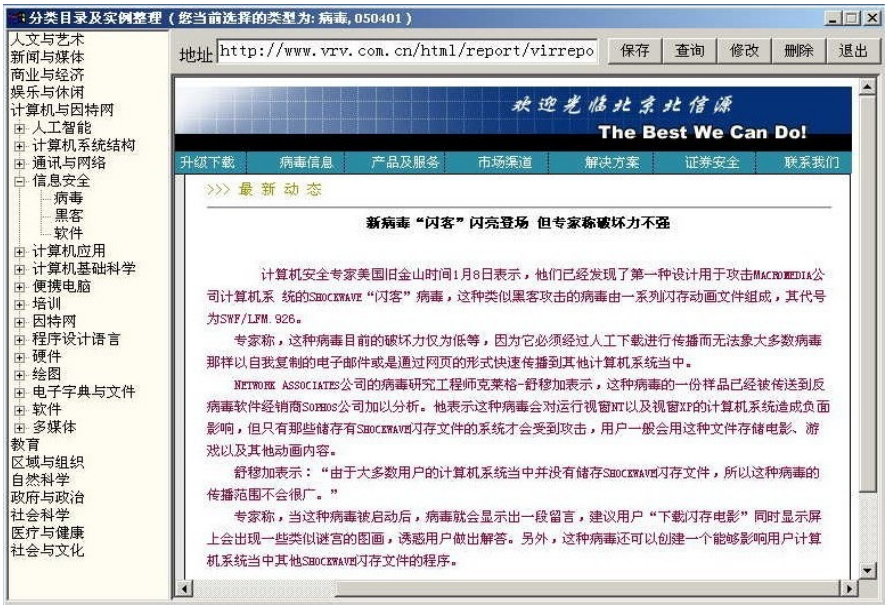


图 11-4 WebSmart ——一个网页实例集搜集和整理工具

下面简要地介绍上述中文网页样本集的分类体系。长期以来，国内外已存在一些可以借鉴的信息内容分类标准，通过比较分析它们的特点和对中文网页这一新型分类对象的适应性，提出了我们的分类体系。

国外具有代表性的分类标准有：《杜威十进分类法》、《美国科研系统常用分类法》、《联合国教科文组织大学学科分类法》等。由于文化背景、思维习惯等方面的不同，这些标准不能完全适应中文文献的分类。例如，分类名称和涵义存在着差异，国内多数分类法把人文科学包含在社会科学（哲学社会科学）领域内，而国外通常将社会科学和人文科学加以区别。

国内具有代表性的分类标准有：《中国图书馆分类法》（2000 年第四版）；国家标准 GB/T 13745-92《学科分类与代码》。这两种标准都不能满足我们的需要。《中国图书馆分类法》处理的对象是图书而非网页、分类体系比较复杂，因而不适合中文网页的分类。因为在 Web 上存在着大量的“新闻与媒体”类和“娱乐与休闲”类网页，用图书分类的方法来对网页分类是不精确的。而且对于非专业人员占绝大多数的普通 Web 用户而言，这些分类难以理解。《学科分类》标准于 1992 年制定，由于制定的时间过久，现在网上大量出现的新生概念，如：聊天

室、虚拟社区、个人主页、在线论坛等等，在这个标准中没有体现。同时，该标准还包括大量已经过时了的类别。这些类别在标准制定时体现了先进的技术，但是目前已经被其他技术所取代或很少被人提及。因此，上述的两个分类标准不能直接用作中文网页的分类。同时，我们还看到著名的分类目录网站，例如 Yahoo!, 新浪，搜狐等都有一个在网上实用的内容目录。经过分析整理，本文最终决定采用的分类体系如图 11-5 所示。它包含三个层次，12 个大类，共 733 个类别。从总体上可以分为学术性和非学术性两大类。其中学术性类别按国家标准 GB/T 13745-92《学科分类与代码》分类。选用该分类体系的主要原因是它分类层次关系简单明了，中国用户比较熟悉。

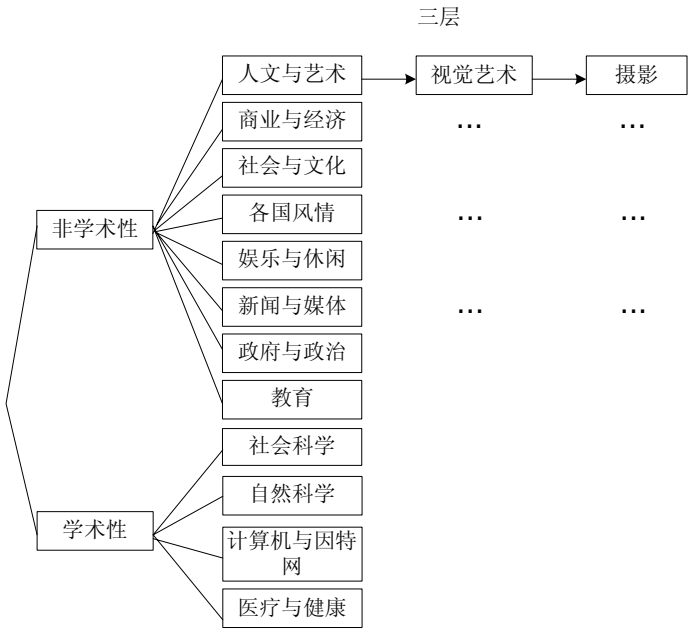


图 11-5 一种中文网页的分类体系

我们使用本节的设计方案实现了一个基本的中文网页分类器，并通过不断增加训练实例数，考察了每个类的训练样本数对分类器质量的影响，希望找到一个需训练的最小样本数。实际测试结果如图 11-6 和图 11-7 所示（这里需要说明的是，当训练样本数等于最右边的 19 时，表示训练集中的所有样本都被保留下来，而不是为每个类别取 19 个训练样本）。

从图 11-6 可以看出，当训练样本数大于等于 15 时，分类器的宏观F_1值就稳定下来了，尽管此时图 11-7 的微观F1 还在上升，但是增加的幅度已经很小了，

并且到 17 时平稳下来。因此，针对本文的训练集，最小训练样本数取 15 个。

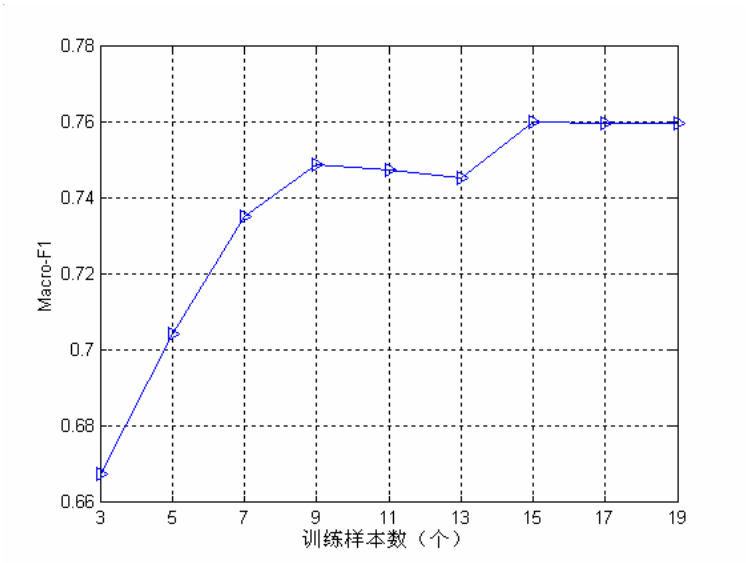


图 11-6 Macro-F_1值随样本数的变化

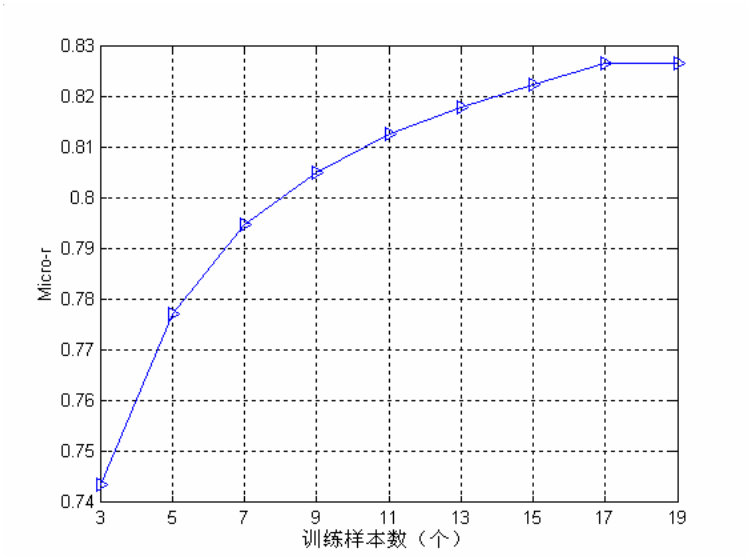


图 11-7 Micro-F_1值随样本数的变化

三、 特征选取

实现文本自动分类的基本困难之一是特征项空间的维数过高。所谓“特征项”在中文文本中主要指分词处理后得到的词汇，而特征项的维数则对应不同词汇的个数。数量过大的特征项一方面导致分类算法的代价过高，另一方面导致无法准确地提取文档的类别信息，造成分类效果不佳。因此，需要在不牺牲分类质量的前提下尽可能地降低特征项空间的维数。“特征选取”的任务就是要将信息量小，“不重要”的词汇从特征项空间中删除，从而减少特征项的个数，它是文本自动分类系统中的一个关键步骤。

为便于后面的描述，这里简要给出特征选取的一般过程。给定训练文档集合 $D = \{d_1,\ldots,d_n\}$，设 $T=\{t_1,t_2,\ldots,t_m\}$ 为对D中的文档做分词后得到的词汇全集，用[m]表示集合{1,2,…,m}。所谓“特征选取”可以看成是确定从TERMS到[m]的一个1-1 映射，即

$$\text{F-Selection：}\quad T \rightarrow [m]$$

然后根据计算开销的考虑，取一个 $i \in [m]$，认为T中那些函数值不小于i的词汇为“选取的特征项”，记做 T_s。

在完成了特征选取后，分类就是基于 T_s，即以其中的元素为基础，用一个向量来表达每一个文档。分类的过程就是按照某种算法来比较待分类文档的表示向量和训练集文档的表示向量，取最相近者所处的类为待分类文档的类。

[Yang and Pedersen,1997]研究了多种特征选取方法，如：文档频率（Document Frequency，DF）、信息增益（Information Gain，IG）、互信息（Mutual Information，MI）、开方检验（χ^2 test，CHI）、术语强度（Term Strength，TS）等。针对英文纯文本比较研究了上述五种经典特征选取方法的优劣。实验结果表明：CHI 和 IG 方法的效果最佳；DF 方法的性能同 IG 和 CHI 的性能大体相当，而且 DF 方法还具有实现简单、算法复杂度低等优点；TS 方法性能一般；MI 方法的性能最差。针对中文网页，其结论是否还正确，目前还没有很明确的结论。因此，本节使用同一个中文网页数据集评测了 DF、IG、MI 以及 CHI 等四种常见的特征选取方法。下面对这些典型的特征选取算法做一下简单地介绍：

1) 文档频率

DF 表示在训练集中包含某个特征项 t 的文档数。这种衡量特征项重要程度的方法基于这样一个假设：DF 较小的特征项对分类结果的影响较小。这种方法优先取 DF 较大的特征项，而 DF 较小的特征项将被剔除。即特征项按照 DF 值排序。这里，为物理意义清楚起见，我们并没有象本节开始那样严格的从 TERMS 到[m]的映射，但显然这是没有困难的，不赘述（后同）。不过我们注意到，这种策略不

符合被广泛接受的信息检索理论：高频词没有低频词对文档特征贡献大[Yang and Pedersen,1997]。DF 是最简单的特征项选取方法，而且该方法的计算复杂度低，能够胜任大规模的分类任务。

2) 信息增益

IG 通过统计某个特征项t在一篇文档中出现或不出现的次数来预测文档的类别。IG 的计算公式如公式(11-4)所示[Yang and Pedersen,1997]。

$$G(t) = -\sum_{i=1}^{m} P_r(c_i)\log P_r(c_i) + \mathrm{p_r(t)}\sum_{i=1}^{m} P_r(c_i \mid t)\log P_r(c_i \mid t)$$

$$+ P_r(\bar{t})\sum_{i=1}^{m} P_r(c_i \mid \bar{t})\log P_r(c_i \mid \bar{t}) \qquad (11\text{-}4)$$

其中：$P_r(c_i)$表示一篇文档属于类别c_i的概率；$P_r(t)$表示特征项t在一篇文档内出现的概率；$P_r(\bar{t})$表示特征项t不在一篇文档内出现的概率；$P_r(c_i \mid t)$表示特征项t在属于类别c_i的文档内出现的概率；$P_r(c_i \mid \bar{t})$表示特征项t不在属于类别c_i的文档内出现的概率。m 是文档类别数。G(t)值大则被选取的可能性大，即特征项按照 G 值排序。

3) 互信息

MI 使用公式（11-5）计算某个特征项t和类别c之间的相关性[Yang and Pedersen, 1997]。

$$I(t,c) \approx \log \frac{A \times N}{(A+C)\times(A+B)} \qquad (11\text{-}5)$$

其中：A为t和c同时出现的次数；B为t出现而c没有出现的次数；C为c出现而t没有出现的次数。N为所有文档数。如果t和c不相关，则$I(t,c)$值为 0。如果有 m 个类，于是对于每个 t 会有 m 个值，取它们的平均，就可得到特征选取所需的一个线性序。大的 I 平均值的特征被选取的可能性大。

4) CHI

使用 MI 衡量特征项的重要程度时，只考虑到了正相关对特征项重要程度的影响。如果特征项t和类别c反相关，就说明含有特征项t的文档不属于c的概率要大一些，这对于判断一篇文档是否不属于类别c也是很有指导意义的。为克服这个缺陷，CHI 使用公式（11-8）计算特征项t和类别c的相关性。

$$\chi^2(t,c)=\frac{N\times(AD-CB)^2}{(A+C)\times(B+D)\times(A+B)\times(C+D)} \qquad (11\text{-}8)$$

其中：A为t和c同时出现的次数；B为t出现而c没有出现的次数。C为c出现而t没有出现的次数；D为t和c同时没有出现的次数。N为训练集中的文档数。与 MI 类似，如果t和c不相关，则$\chi^2(t,c)$值为 0。同 MI 相同，如果有 m 个类，每个 t 就会有 m 个值，取它们的平均，就可得到特征选取所需的一个线性序。大的χ^2平均值的特征被选取的可能性大。

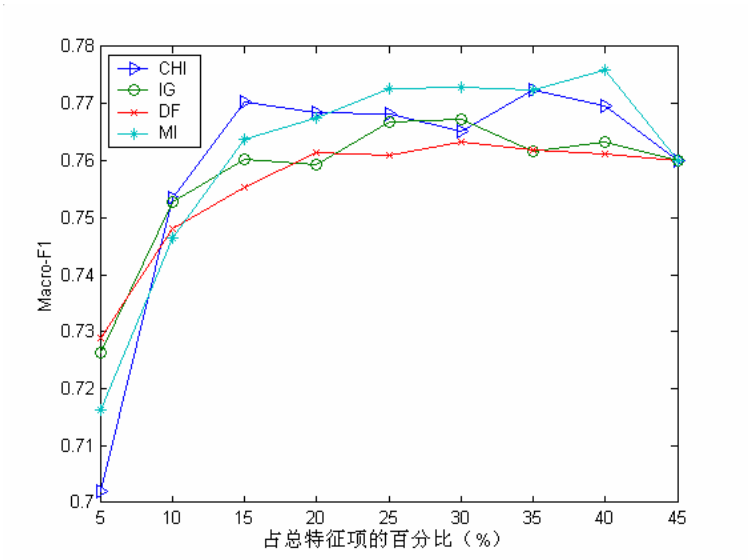


图 11-8 CHI、IG、DF、MI的比较（Macro-F_1）

为了观察特征项的个数对分类器性能的影响，我们分别使用CHI、IG、DF、MI特征选取算法挑选出不同个数的特征项。图 11-8 和图 11-9 分别表示当取不同百分比的特征项时，分类器宏观F_1和微观F_1的变化。这里需要说明的是，最右边的 45%表示的所有特征项都被保留下来，而不是仅保留前 45%个特征项。

从图 11-8 可以看出，CHI方法最优，在取前 15%的特征项时，分类器的质量就稳定下来了；使用DF方法时，分类器的宏观F_1波动最小，可以过滤掉 80%的特征项；MI方法和IG方法都在取前 25%的特征项时，分类器的质量才稳定下来了。

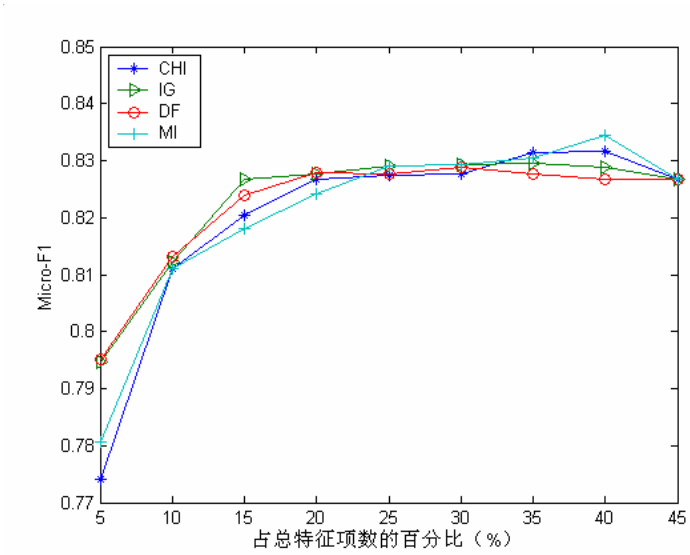


图 11-9 CHI、IG、DF、MI的比较（Micro-F_1）

从图 11-9 可以看出，IG方法最优，在取前 15%的特征项时，分类器的质量就稳定下来了；使用DF方法时，分类器的宏观F_1波动最小，可以过滤掉 80%的特征项；MI方法最差，在取前 25%的特征项时，分类器的质量才稳定下来了。

综合图 11-8 和图 11-9，可以看出，CHI、IG 和 DF 的性能明显优于 MI；CHI、IG 和 DF 的性能大体相当，都能够过滤掉 80%以上的特征项；DF 具有算法简单、质量高的优点，可以用来代替 CHI 和 IG，但是同被广泛接受的信息检索理论有些矛盾。我们这里得到的结论，同文献[Yang and Pedersen,1997]使用普通英文文本评测结果基本一致。

四、 分类算法

在本章第二节，我们有了一个关于各种文档自动分类算法的概貌。下面对几个比较典型的分类算法进行具体的介绍，并给出了 kNN 与 NB 算法的分类质量与效率的实验结果比较。

1. 典型分类算法

1) kNN 分类算法

kNN 分类算法是一种传统的基于统计的模式识别方法。算法思想很简单：对

于一篇待分类文档，系统在训练集中找到 k 个最相近的邻居，使用这 k 个邻居的类别为该文档的候选类别。该文档与 k 个邻居之间的相似度按类别分别求和，减去一个预先得到的截尾阈值，就得到该文档的类别测度。用 kNN 也表示所选 k 个最相近文档的集合，公式（11-9）刻画了上述思想[Yang and Liu,1999]。

$$y(x,c_j)=\sum_{\varpi d_i\in kNN} sim(x,d_i)y(d_i,c_j)-b_j \qquad (11\text{-}9)$$

其中，x为一篇待分类网页的向量表示；d_i为训练集中的一篇实例网页的向量表示；c_j为一类别；$y(d_i,c_j)\in\{0,1\}$（当d属于c_j时取 1；当d不属于c_j时取 0）；b_j为预先计算得到的c_j的最优截尾阈值；$sim(x,d_i)$为待分类网页与网页实例之间的相似度，由文档间的余弦相似度公式（11-10）计算得到：

$$\cos<x,d>=\frac{x\cdot d}{|x||d|} \qquad (11\text{-}10)$$

kNN 算法本身简单有效，它是一种 lazy-learning 算法，分类器不需要使用训练集进行训练，训练时间复杂度为 0。kNN 分类的计算复杂度和训练集中的文档数目成正比，也就是说，如果训练集中文档总数为 n，那么 kNN 的分类时间复杂度为 O(n)。

2) NB（Naïve Bayes）算法

NB 算法是基于贝叶斯全概率公式的一种分类算法。贝叶斯全概率公式的定义如公式(11-11)所示[Sebastiani,1999]。

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)} \qquad (11\text{-}11)$$

给定一个类c以及文档$d(a_1，a_2，\cdots，a_n)$，其中a_i表示文档d中出现的第i个特征项的权值，n为文档中出现的特征项的总数。根据全概率公式，可以得到公式(11-12)：

$$P(c|d)=\frac{P(d|c)P(c)}{P(d)}=\frac{P(a_1|c)P(a_2|c)...P(a_n|c)P(c)}{P(d)} \qquad (11\text{-}12)$$

其中，P(c|d)表示文档d属于类别c的概率；P(c)表示待分类的文档所处的领域中文档属于这个类的概率，在具体的计算时，可以分别用训练集中属于这个类的文档所占的比例代替。$P(a_i|c)$表示在类别c中特征项a_i出现的概率，可以近似地用训练集中包含有该特征项的类别c中的文档个数与训练集中类别为c的文档总数的

比值表示。

由此可以看出，NB 算法假设文档之间的特征项都是相互独立的。但是，这一假设对语义丰富的语言文字信息往往过于简单，这也在一定程度上限制了算法的性能。

NB算法需要使用训练集对分类器进行训练，也就是需要分别计算每个$P(a_i|c)$。假设训练集共有m个类别，n个特征项，待分类文档共有k个特征项，那么训练的时间复杂度为O(m*n)。分类的时间复杂度为O(k)。

3) 决策树（Decision Tree，Dtree）算法

决策树算法通过对训练数据的学习，总结出一般化的规则，然后再利用这些规则解决问题。用决策树进行文档分类的基本思路是这样的：先用训练集为预先定义的每一个类构造一棵决策树，构造方法如下：

① 以训练集作为树的根结点，它表示所有的训练文档，将它标记为“未被检测”；

② 找到一个标记为“未被检测”的叶结点，如果它表示的所有文档都属于这个类，或者都不属于这个类，将这个叶结点的标记改为“已被检测”，然后直接跳到第三步；否则，挑选当前最能区分这个结点表示的文档集中属于这个类的文档和不属于这个类的文档的特征项作为这个结点的属性值，然后以这个结点为父结点，增添两个新的叶结点，都标记为“未被检测”，父结点表示的训练文档集中含有这个特征项的所有文档用左子结点表示，所有不含有这个特征项的文档用右子结点表示；

③ 重复第二步操作，直到所有的叶结点都被检测过。

对每棵决策树，从它的根结点开始，判断结点的属性值（特征项）是否在待分类的文档中出现，如果出现，则沿着左子树向下走；否则沿着右子树向下，再继续判断当前结点的属性值是否在待分类的文档中出现，直到到达决策树的某个叶结点，如果这个叶结点表示的训练文档都属于这个类，则判定这篇待分类的文档也属于这个类；反之亦然。

4) Rocchio 算法

其基本思想是使用训练集为每个类构造一个原型向量，构造方法如下：给定一个类，训练集中所有属于这个类的文档对应向量的分量用正数表示，所有不属于这个类的文档对应向量的分量用负数表示，然后把所有的向量加起来，得到的和向量就是这个类的原型向量，定义两个向量的相似度为这两个向量夹角的余弦，逐一计算训练集中所有文档和原型向量的相似度，然后按一定的算法从中挑选某个相似度作为界。给定一篇文档，如果这篇文档与原型向量的相似度比界大，则这篇文档属于这个类，否则这篇文档就不属于这个类。Rocchio 算法的突出优点是容易实现，计算（训练和分类）特别简单，它通常用来实现衡量分类系统性能的基准系统，而实用的分类系统很少采用这种算法解决具体的分类问题。

2. kNN 与 NB 算法比较

文献[Yang and Liu,1999]比较研究了支持向量机（SVM）、kNN、NB、Linear Least Squares Fits（LLSF）、和 Neural Network（NNet）算法。研究结果表明，当训练集中每个类的正面实例比较少(少于 10 个)的情况下，SVM、kNN、LLSF 的性能明显优于 NNet 和 NB 算法。这里，我们将比较研究 kNN 和 NB 算法。

表 11-2 kNN 和 NB 算法的分类质量和分类效率比较

指标 / 算法	质量		效率（秒）	
	Micro-F1	Macro-F1	训练时间	测试时间
kNN	0.8266	0.7560	0	2426
NB	0.1934	0.1612	251	2129

从表 11-2 可以看出，kNN 的质量明显优于 NB 算法，但是，NB 算法的分类效率要比 kNN 方法略高。kNN 和 NB 分类算法对 12 个不同类别的分类情况见图 11-10 所示。

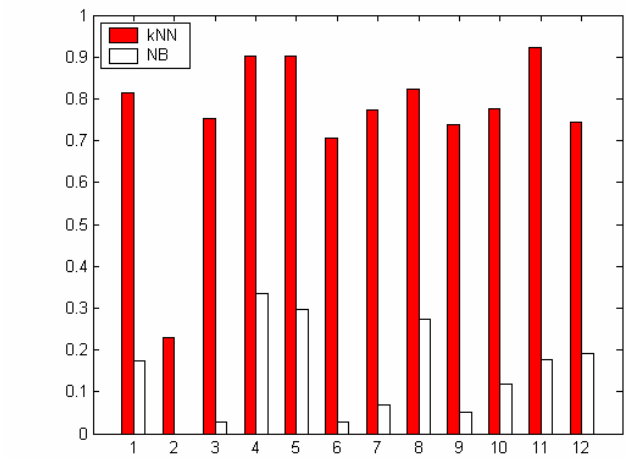


图 11-10 kNN 与 NB 分类结果的比较

从图 11-10 可以看出，kNN分类算法明显优于NB算法，对于所有类别，kNN的分类结果都优于NB。从图 11-10 还可以看出，即使是同一个算法对于不同类别的文档，其分类能力也是各有差异的。在“医疗与健康”领域，kNN的Macro-F_1达到最高值；在“新闻与媒体”领域，kNN的分类Macro-F_1达到最低值。对于“医疗与健康”领域，NB的Macro-F_1达到最高值；在“计算机与因特网”领域，NB的Macro-F_1达到最低值。从总体而言，NB算法对不同类别比较敏感, 是一种不稳

定的分类算法。kNN的分类质量受领域的影响不大。

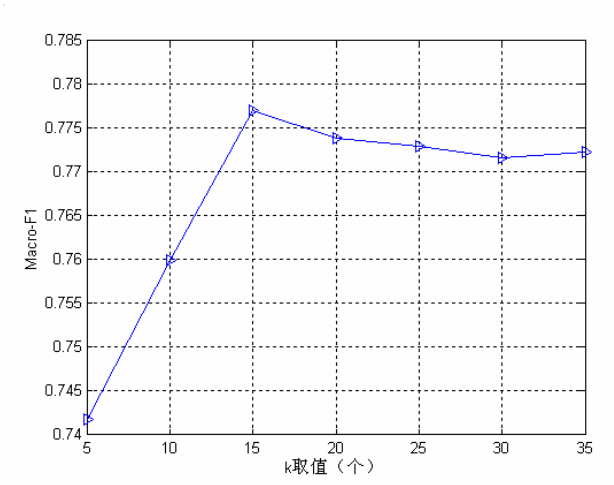


图 11-11 k的取值对分类器质量的影响（Marco-F_1）

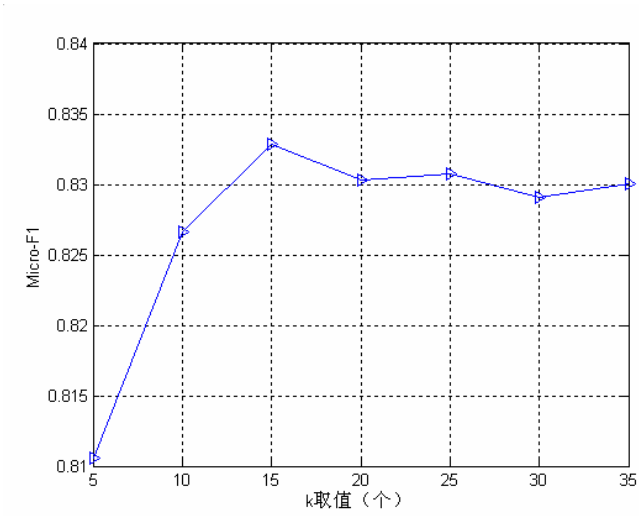


图 11-12 k的取值对分类器质量的影响（Micro-F_1）

针对 kNN 分类算法，我们分析了 3 种因素对分类质量的影响：k 的取值；衡量两篇文档之间相似度的方法：兰式距离法（Canberra metric）与欧式距离法（Euclidean distance）；分类目录中类别之间的层次关系。下面首先介绍 k 的取值对分类质量的影响。

1) k 的取值

在前文的实验中，我们一直取 k=20，但这不一定是 k 的最好取值。因此，这

里我们将通过具体实验来验证 k 的取值对分类质量的影响。

从图 11-11 和图 11-12 可以看出，当 k=15 时，分类器具有最佳的分类质量，随后，随着 k 值的增加，分类器分类质量有所下降，最后平稳下来。

2) 相似度的度量

为了计算两个文档之间的相似度可以有多种距离函数[卜东波,2000]，其中欧式距离和兰式距离在实际的分类系统中用的较多，通常使用的两个向量之间夹角的余弦就是兰式距离的一种。

表 11-3 欧式距离与兰式距离的比较

指标 / 算法	质量		效率（秒）
	Micro-F1	Macro-F1	测试时间
欧式距离	0.2419	0.1715	4790
兰式距离	0.8266	0.7600	2426

从表 11-3 可以看出，无论是分类质量，还是分类效率，应用兰式距离法来度量两个文档之间的相似度的分类效果明显优于欧式距离法。兰式距离法与欧式距离法对 12 个不同类别的分类情况如图 11-13 所示。

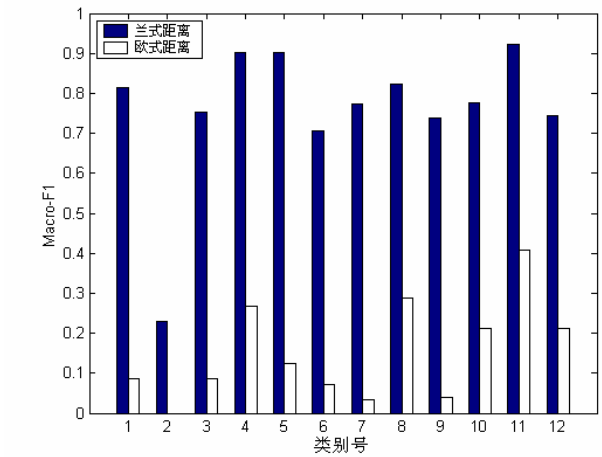


图 11-13 兰式距离法与欧式距离法对 12 个不同类别的分类情况

3) 分类目录中类别之间的层次关系

本节采用的分类体系包含三个层次，12 个大类，共 733 个类别（参见图 11-5），在分类过程中，可以利用这种层次关系来减少文档比较的次数：当某篇待分类的

文档与父类文档之间的相似度小于某个阈值时，就可以认为与该父类包含的子类是不相关的，因此可以直接减少文档比较的次数。表 11-4 给出了应用于层次模型的 kNN 与基本 kNN 的分类质量和分类效率的比较情况。从表 11-4 可以看出，这种基于层次模型的 kNN 分类方法的效率是基本 kNN 分类方法的 3 倍多，但是，分类质量也下降了许多。图 11-14 给出了两者在 12 个大类下的分类情况比较。

表 11-4 基于层次模型的 kNN 与基本 kNN 的比较

指标 / 算法	质量		效率（秒）
	Micro-F1	Macro-F1	测试时间
基于层次的 kNN	0.6723	0.5719	792
基本 kNN	0.8266	0.7600	2426

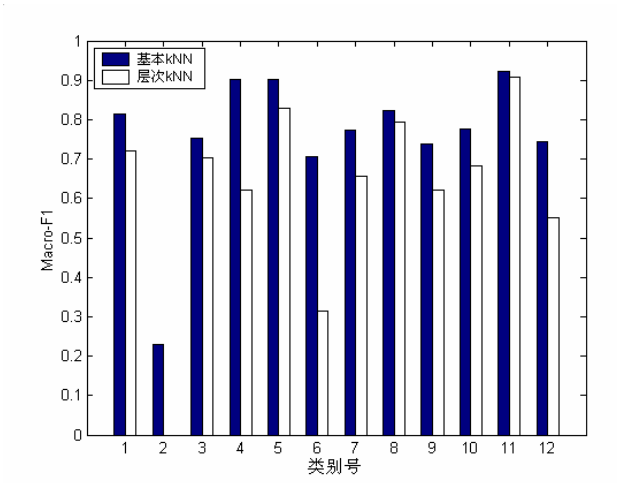


图 11-14 基于层次模型的 kNN 与基本 kNN 的比较

五、 截尾算法

对于一篇待分类文档，应用 m 元分类算法通常得到多个类别。一般情况下都要求从这些候选类别中选择部分类别为该文档的最终分类结果。这个过程使用的方法通常被称为阈值策略。下面简单介绍三个比较常见的阈值策略[Yang,2001]。

1. 位置截尾法（rank-based thresholding，记为 RCut）

假设分类系统预先定义的类别数为 m。整数 k 大于 1 并且小于 m。对于每一个待分类的文档 D，分类系统都返回一个长为 m 的候选类列表，取候选类列表的前 k 项（按类和文档的相似度排序），这篇文档就被认为属于这 k 个类。这种阈值

策略就被称为位置截尾法。RCut 方法的优点是实现非常简单，能够胜任在线分类工作。但它存在严重的缺陷：假设待分类的文档数目为 n，候选类列表的每个位置都对应 m 个候选类。即使 k 变化 1，每篇文档的类关系都要发生变化。因此，无法平滑地调整分类系统的性能。我们称 RCut 算法是以文档为中心的。

2. 比例截尾法（proportion-based thresholding，记为 PCut）

假设待分类的文档数目为n，预先定义的类别数为m。P_i表示训练集中属于类i的文档所占的比例。系统首先计算出每篇待分类文档的**候选类列表**，然后生成每个类的**候选文档列表**（按类和文档的相似度排序）。对于类i，取这个类的候选文档列表中的前$n*P_i*x$篇文档属于这个类，其他的文档则不属于这个类。其中x是经验比例因子（为一实数），通过改变它的大小，可以平滑地调整系统的性能。PCut算法的基本思想是控制分入各个类的文档数，使它们保持训练集中各个类文档数的比例关系。这种算法最大的问题是过分依赖于这种比例关系，而没有考虑类和文档的相似度以及类在候选类列表中的位置。可以看到，PCut算法是以类别为中心的。同RCut算法相比，PCut算法的系统性能比较平滑，但是不适用于在线分类。

3. 最优截尾法（score-based local optimization thresholding，记为 SCut）

同 PCut 算法一样，Scut 算法也是以类别为中心。假设待分类的文档数目为 n，预先定义的类别数为 m。系统首先计算出每篇待分类文档的候选类列表，然后生成每个类的候选文档列表（按类和文档的相似度排序）。对于候选类列表里的每一个类，如果这篇文档和这个类的相似度大于这个类的最优截尾相似度，那么这篇文档就属于这个类。否则，这篇文档就不属于这个类，其中，每个类的最优截尾相似度是这样预先取得的：将训练集分成两部分，其中一部分仍然作为训练集，另一部分作为测试集，对每一个类，评价分类系统在这个测试集下对于这个类的分类性能，调整截尾相似度，使得系统的性能达到最优，此时截尾相似度的值就是这个类的最优截尾相似度。SCut 算法性能比较优异，但是不能很好地处理那些稀有类别（就是比较少见的类别）。

表 11-5 RCut 和 SCut 截尾算法的比较

指标 / 算法	质量		效率（秒）
	Micro-F1	Macro-F1	测试时间
RCut	0.8266	0.7600	4324
SCut	0.8401	0.7849	5368
基本 kNN	0.8266	0.7600	2426

文献[Yang,2001]比较研究了上述三种阈值策略，结果发现 SCut 算法效果明显优于 PCut 和 RCut 算法。由于本文使用的训练样本分布比较均匀，每个类平均有 17 个训练网页，对于这种基本按比例分布的样本集，PCut 方法就没有什么作用了，因此，我们比较研究了 RCut 和 SCut 方法，总体分类结果如表 11-5 所示。

从表 11-5 可以看出，SCut 方法比 RCut 方法在分类质量上要好，而分类效率却要差些，但是两者的差别不是十分明显。这里，RCut 方法的分类质量同基本 kNN 方法的分类质量完全一样，因为通过实验测试发现，当 R=1 时，分类器的分类质量最好，RCut 的这种取大的一个文档类别的计算方法同普通 kNN 的计算方法一样。所以，两者的分类结果是一样的。具体达到 12 个大类，两者分类结果的比较见图 11-15 所示，从中可以看出，SCut 比 RCut 方法的效果要好一些。

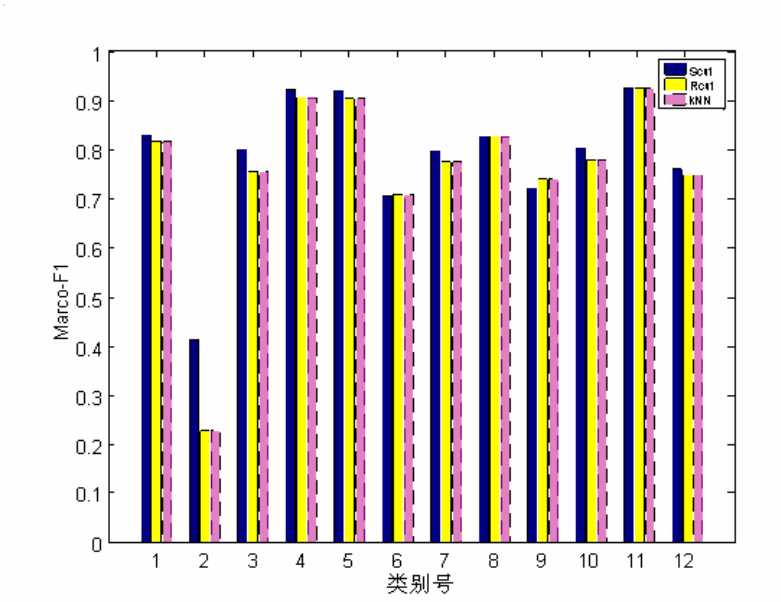


图 11-15 RCut 和 SCut 截尾算法的比较

六、 一个中文网页分类器的设计方案

在上一小节，我们定量地分析了影响分类器性能的各种关键因素。据此，我们设计了一个分类器。具体的要素和参数如表 11-6 所示。

表 11-6 一个分类器的设计方案

关键因素		方案
训练样本数		15
特征选取方法		CHI
分类算法	kNN & NB	kNN
	k	15
	相似度	兰式
	层次关系	层次关系
截尾算法：		SCut

第五节 天网目录导航服务

一、 问题的提出

目前提供 Web 浏览导航的系统主要分为两大类。第一类是目录导航系统。它主要是通过具有专业知识的网页编辑人员人工地对网站进行精选，建立一个内容分类索引目录，向用户提供目录导航服务。用户可以沿着分类目录的层次结构，逐步进入自己感兴趣的主题，进而找到所需的信息。这类系统的优点是提供的相关信息比例很高，而且信息内容的权威性较高，网页信息和用户的相关性较大，但是覆盖的范围小，系统维护的网页数量有限。其典型代表是 Yahoo!的目录系统。第二类是基于自动信息搜集和分析的搜索引擎系统（称为“自动式搜索引擎”）。它通过一个程序自动地在网上沿着超文档链递归地访问、搜集 Web 网页，分析页面的内容，生成索引和摘要，并向用户提供 Web 查询界面，根据用户的查询关键词在索引库中查找相关信息在网上的位置，最后将查询结果按照相似度排序后返回，帮助用户尽快地找到所需的信息。这类系统的优点是涵盖的网页数量巨大，但搜索的准确率相对比较低，其典型代表是 Google。

下面简要地比较这两类搜索引擎。

1) 自动式搜索引擎自动地搜集、分析和处理网页，因而它索引的网页数多，信息量大，并且能够定期或增量地搜集网页，更新索引库的内容，向用户提供最新的 Web 网页信息。但是它只提供基于关键词的检索，用户只有确切地知道自己感兴趣的网页含有哪些关键词时，查询的效果才比较理想。否则，返回的结果很可能和用户的实际需求“风马牛不相及”。
2) 目录式搜索引擎支持基于分类目录的查询。目录式搜索引擎对搜集的网页采用人工分类。由于这种人工方式对网页内容的理解比较准确，因此查询的准确性优于自动式搜索引擎。当用户对某个领域感兴趣但并不熟

悉这个领域的关键词时，这种查询方式能很好地为用户提供服务。由于人工分类效率低，网页更新困难，目录式搜索引擎在索引的网页规模上受到很大的限制。Google 等自动式搜索引擎索引的网页数量早以突破十亿级，而 Yahoo!要少得多。

天网搜索引擎是一种典型的自动式搜索引擎。下面介绍如何应用中文网页自动分类技术在天网中提供目录导航服务。

二、 天网目录导航服务的体系结构

天网目录导航服务的体系结构如图 11-16 所示。系统整体上可以分为两个部分：在线部分和离线部分。在线部分首先接受用户的查询条件，然后根据用户所处的网页目录自动地进行查询扩展，接着在预先分好类的网页库中检索，最后实时地返回查询结果。离线部分的核心部件为中文网页分类器，负责中文网页的自动分类。本文使用的中文网页分类器的工作原理如图 11-3 所示。分类器的整个工作周期可以分成训练过程和分类过程。在训练过程中，训练集实例经过中文分词和特征选取处理后被表示成向量形式。该特征向量用来描述类别模式，在分类过程中使用。校验集是训练集的一部分，通过应用相应的阈值策略来预先确定每个类别的截尾阈值。在分类过程中，一个待分类的中文网页经过中文分词并表示成向量后，应用分类算法同训练过程得到的类别模式逐一比较，得到候选类别列表，然后同训练过程中得到的每个类别的阈值相比较，保留大于阈值的类别，并作为该网页的分类结果。

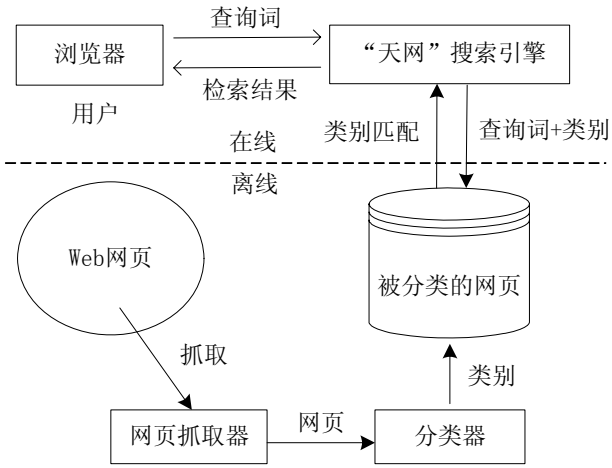


图 11-16 天网目录的体系结构

三、 天网目录的运行实例

图 11-17 展示了天网目录导航服务的用户界面。通过目录导航服务，用户可以非常容易地查找他感兴趣的主题：通过三层目录，用户可以逐步缩小他要检索内容的范围。实验测试表明，应用第四节介绍的分类器设计方案可以满足天网网页目录服务的要求。

图 11-17 天网目录导航服务

第六节 本章小结

本章我们系统地讨论了中文网页自动分类的相关技术及其它们在一个分类系统中的地位和相互关系，比较了在一个分类系统不同环节上多种算法的优劣，定量地得到了如下主要结论：

1) 对于规模比较大（例如几百个类）的层次型网页内容分类体系，训练集中每一类的样本数达到 15 个时得到的分类器的分类质量就基本稳定了，再增加样本数对分类器质量的改进不大。

2) 在旨在降低特征空间维度的各类特征选取算法中，IG、CHI 算法最有效；DF 方法在一定程度上可以用来替代 IG 和 CHI；MI 方法最差。
3) 比较 k 最近邻居（kNN）和朴素贝叶斯（NB）分类算法，我们发现 kNN 的分类质量明显优于 NB；而且 NB 算法对不同类别表现有明显差异，是一种不稳定的分类算法。但是，kNN 方法的分类效率要比 NB 差。
4) 三种因素对 kNN 算法的质量有影响。（1）k 的取值：当 k 取 15 时，分类器的分类质量最好；（2）衡量两篇文档之间相似度的方法：无论是分类质量，还是分类效率，兰式距离法都明显优于欧式距离法；（3）分类目录中类别之间的层次关系：基于层次的 kNN 分类算法的分类效率明显优于基本的 kNN 分类算法。但是，这需要牺牲一定的分类质量作为代价。
5) 比较 RCut 和 SCut 两种经典的截尾算法，实验表明 SCut 算法对分类器分类质量的影响明显优于 RCut 算法，但是算法费用要比 RCut 大。

这些研究成果系统地体现在天网提供的中文网页自动目录导航服务上。同时，作为本章各种实验研究基础的中文网页分类体系以及相应的训练测试集也是一项很好的工作成果。2003 年 3 月召开的首届“全国搜索引擎和网上信息挖掘学术研讨会”期间举行了由 10 个代表队参加的“中文网页自动分类竞赛”，该训练测试集为竞赛提供了有效的训练数据。

第十二章 搜索引擎个性化查询服务

一般的搜索引擎是基于关键词匹配的方式进行检索的，由于这种方法缺乏对关键词语义的理解，检索结果对用户而言不够理想。主要表现在两个方面：① 检索结果中无关的网页过多。在所有检索结果中经常是大多数结果与用户的需求无关。尽管某些网页含有检索关键词，实际上同用户的本意无关，但是也被返回给用户了。② 没有考虑不同用户的个性差异。目前，所有用户如果输入相同的查询条件，搜索引擎就会返回相同的结果，尽管这些用户的需求各不相同。事实上，不同的用户由于受教育水平、工作环境等因素的不同而具有鲜明的个性，希望搜索引擎能够提供个性化服务，使得查询结果符合用户的个性需求。因此，如何提高搜索引擎检索结果的精度并向用户提供个性化服务已成为搜索引擎技术的一个新的发展方向和研究热点。

本章第一节讨论基于 Web 挖掘的个性化服务技术，第二节介绍我们研究开发的一个网上针对某一对象类的信息进行个性化查询服务的系统——天网知名度系统。

第一节 基于 Web 挖掘的个性化技术

所谓 Web 个性化，实质上就是一种以用户需求为中心的 Web 服务。如图 12-1 所示。首先，不同 Web 用户通过各种途径访问 Web 资源，如图 12-1 中箭头 a 所示。其次，系统学习用户的特性，创建用户访问模型，如图 12-1 中箭头 b 所示。最后，系统根据得到的知识调整服务内容，以适应不同用户的个性化需求，如图 12-1 中箭头 c 所示。因此创建 Web 个性化服务系统的一般步骤为：（1）收集用户的各种信息，如注册信息，访问历史等；（2）分析用户数据，创建符合用户特性的访问模式；（3）结合用户特性，向用户提供符合其特殊需求的个性化服务。用户对系统提供的服务做出反馈信息，系统根据反馈信息调整服务。通过用户与系统之间循环往复的交互，系统最终能够为用户提供个性化服务。从上面的分析可以看出，通过分析用户的各种信息建立用户访问模式是建立个性化服务系统的关键。因为只有首先客观地描述了用户的需求，然后才能根据这些特性向用户提供个性化服务。

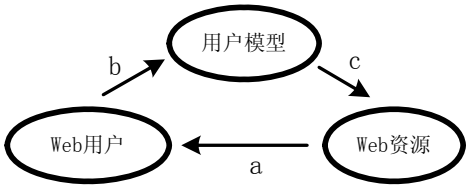


图 12-1 Web 个性化的实质

一、 Web 挖掘技术

Web 挖掘技术是实现 Web 个性化服务的核心技术之一。Web 挖掘的一般过程可以分成三个阶段:1) 预处理:需要对收集的数据进行必要的预处理,如清除“脏”数据。2) 模式发现:应用不同的 Web 挖掘算法发现用户访问模式。3) 模式分析:从发现的模式集合中选择有意义的模式。Web 挖掘通常可以分成三大类,如图 12-2 所示。

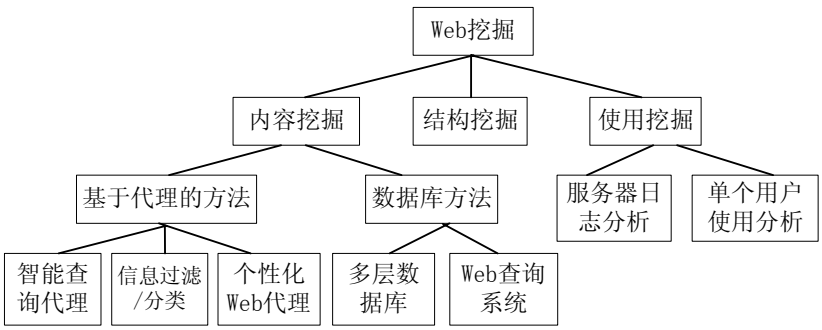


图 12-2 Web 挖掘的分类

Web 内容挖掘是从 Web 资源中发现信息或知识的过程。在创建个性化服务系统时，人们通常应用 Web 内容挖掘对网页内容进行分析，其中网页的自动分类技术在搜索引擎、数字化图书馆等领域得到了广泛的应用。根据实现方法的不同可以分成基于代理的方法和数据库方法。Web 内容挖掘由于直接处理数据对象的内容，因此得到的结果一般比较精确，在个性化系统中得到较广泛的应用。

Web 使用挖掘技术通常可以应用到两个领域: 当用来分析 Web 服务器的访问日志时，可以利用挖掘得到的服务模型来设计适应性 Web 站点；当应用到单个用

户时，通过分析用户的访问历史来发现有用的用户访问模式。Web 使用挖掘由于处理数据对象通常为用户的访问历史或服务器的访问日志，无法得知数据对象代表的内容，因此得到的结果一般比较粗糙，但是由于该方法比较成熟而且实现起来也较内容挖掘简单，在个性化系统中也得到了较广泛的应用。Web 使用挖掘的基本方法包括：聚类、关联规则、序列模式、分类、依赖性建模、统计分析等。

Web 结构包括页面内部的结构以及页面之间的结构。挖掘 Web 结构信息对于导航用户浏览行为、改进站点设计、评价页面的重要性等都非常重要。PageRank 算法和 HITS 算法利用 Web 页面间的超链接信息计算“权威型”（Authorities）网页和“目录型”（Hubs）网页的权值。Web 结构挖掘通常需要整个 Web 的全局数据，因此在个性化搜索引擎或主题搜索引擎研究领域得到了广泛的应用。

二、 典型个性化 Web 服务系统的比较

目前已经出现了多个应用 Web 挖掘技术创建的个性化 Web 服务系统。这些系统应用的 Web 挖掘类型包括使用挖掘、内容挖掘和结构挖掘；收集数据的方式有三种：从客户端、代理或服务器方得到原始数据；最后提供的服务有两类：过滤服务和导航服务。表 12-1 根据这三个方面的不同，比较了基于 Web 挖掘的典型 Web 个性化系统。

表 12-1 典型 Web 个性化系统的比较

系统名称	数据收集方式			挖掘类型			服务方式	
	客户端	代理	服务器	使用	内容	结构	过滤	导航
WBI		√		√	√		√	
Web Watcher		√		√	√			√
Pitkow			√	√				√
WebMinier			√	√				√
Site Helper			√	√	√			√
ParaSite			√			√		√
Analog			√	√				√
Letizia	√			√	√		√	
Net Perceptions	√			√	√		√	√
WebTagger	√			√			√	
PowerBookmarks	√			√	√		√	√
WebVCR	√			√				√

尽管 Web 挖掘技术已经在 Web 个性化系统中得到了广泛的应用，但是还存在着以下几个方面的问题。

1） 隐私问题

这是一个不可回避的问题。因为要想建立个性化 Web 系统就必须有用户的参与同时还要分析用户反馈的信息。这就可能涉及到用户的隐私。目前的 Web 个性化技术还不能很好的解决这个问题：即在实现个性化服务的同时又不侵犯用户的隐私。

2） 性能问题

Web 个性化系统都不同程度地扩展了传统的浏览器/服务器体系结构，Web 信息经过相应处理后才返回客户端，就必然会延长响应时间。实时个性化系统对响应时间要求比较高，特别是采用中间代理方式的系统，如果中间处理过程费时过多或用户数量过大，系统性能将是一个不可忽视的问题。而且针对 Web 系统,无论是其用户量,还是系统维护的网页通常都是海量的,目前的 Web 挖掘算法在处理这些数据时通常都采用离线方式，因此对于要求在线实时处理的情况还不能很好地解决。

3） 质量评价问题

应用 Web 挖掘技术实现 Web 个性化服务,不同系统采用不同的 Web 挖掘技术，如何评价它们的建模效果以及系统最终的服务质量也是一个非常重要的问题。目前对个性化系统服务质量的评价，不同系统采用不同的方式和测试数据，因此，还没有一个通用的标准来客观评价多个不同个性化系统服务质量的优劣。需要研究一种通用的性能指标并开发相应的Benchmark来评价各种不同的Web挖掘技术。

三、 基于 Web 挖掘的个性化技术的发展

基于 Web 挖掘的个性化技术发展有如下趋势：

1） 与人工智能技术的结合

个性化系统领域的许多问题最终都可归结到机器学习、知识发现等问题上。用户建模过程通常都应用到代理和多代理技术。因此人工智能技术与 Web 挖掘技术的结合将会促进 Web 个性化系统的飞速发展。

2） 与交互式多媒体 Web 技术的结合

随着下一代互联网技术的飞速发展与应用，未来的 Web 将是多媒体的世界。Web 个性化技术和 Web 多媒体系统结合出现了交互式个性化多媒体 Web 系统。支持海量多媒体数据流的内容挖掘将成为 Web 挖掘技术的基本功能之一。由于这种基于内容的交互式个性化多媒体 Web 系统更能满足用户需要，因此也将成为 Web 个性化系统的发展方向之一。

3） 与数据库等技术的结合

Web 挖掘技术的基础是数据挖掘。尽管 Web 数据由于自身的特性（如海量、半结构、超链信息等）使得 Web 挖掘面临着新的挑战，但是随着数据库技术，特别是数据挖掘技术的发展，Web 挖掘技术也将得到快速的发展。当然，为解决诸如质量评价、性能以及隐私问题，基于 Web 挖掘的个性化技术在这些方面也将得到长足的发展。

第二节 天网知名度系统

天网知名度系统[Fame,2004]是在“北大-IBM 创新研究院”项目支持下研究开发的一个个性化信息检索系统。该系统是天网搜索引擎技术和先进的中文信息处理技术的结合。它针对特定的（命名）实体及其特性，建立起相关的信息资源模型，通过基于该模型的网页过滤和相关度评价，提供个性化检索和定制信息的主动推送服务。

天网知名度系统可以根据用户注册的实体信息，对搜集到的原始网页进行分析和整理，依用户指定的实体属性对每个网页内容进行相关度及正负面评价，把相关的网页进行汇集、排序，并把满足要求的网页以指定的方式加工、存储，向用户提供 Web 信息检索服务，或主动地以简报或邮件的形式定期向用户推送有关网页信息。

一、 系统结构

在天网知名度系统中，我们将用户关心的对象称为命名实体，如：个人、公司、机构等。对我们已经完成的一个实验系统（FAME）来说（主要针对个人），它要求用户将其要查询的实体信息分类注册，即使是其他用户已经注册的名命实体也要求再加以注册，系统将为每个用户登记专用的实体信息，形成个人信息库和实体信息库，以保证尽量满足每个用户的个性化检索需求。

实体信息的分类由系统根据实体的特性予以划分，如个人信息可划分为以下八类：

1) 个人所在的领域（政府、科教、业界、影视等）
2) 个人的名字，包括别名、笔名、艺名等，保证检索的完整性
3) 个人所在的工作单位
4) 个人的职业描述（主席、书记、教授、记者、演员等）
5) 个人的兼职单位（可以有多个）
6) 个人的社会形象
7) 特征词（用户关心的特征描述）

8) 个人的代表作（著作、作品名、产品名等）

我们认为以上信息基本涵盖了在社会上有一定知名度的个人的相关特征，系统将根据用户注册的这些信息去分析过滤每个网页，计算网页的相关度，如图 12-3 和图 12-4 所示。

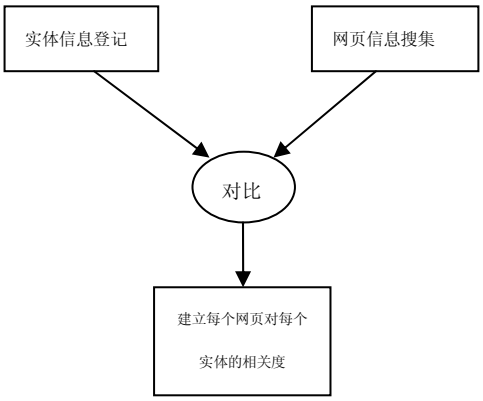


图 12-3 网页与实体相关度的建立

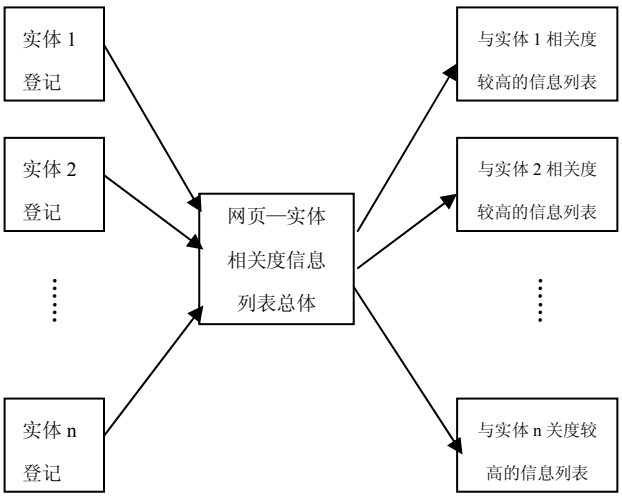


图 12-4 个性化知名度示意图

天网知名度系统的系统结构如图 12-5 所示，该图的上半部分除了网页信息提

取外，基本上是天网原有的系统功能模块。其中，网页搜集利用了天网的抓取功能模块，为天网知名度系统搜集中文的原始网页库 Url.dat。原始网页库保存为一定结构（每个网页有固定格式的附加头信息，说明网页的地址、时间、格式等信息）的文本文件，每个网页顺序存放。原始网页经过净化预处理后输入中文分词和信息提取模块，该模块完成中文分词、词性标注、实体识别以及实体关系的提取功能，形成网页表示库，其中每个原始网页被表示成网址、网页长度（以词计）、网页文本词串、词串对应的词性标记串和 HTML 标记串、网页中提取的实体及其实体属性之间关系的各种信息列表（如：人名列表、单位机构名列表、人名与单位关系列表、人名与职务关系列表等），为进一步的网页评价做好准备。网页索引根据不含信息提取的分词模块所处理的结果并依据词频和位置等权重信息而进行，形成索引库。

图 12-5 的下半部分中，左边是用户注册信息的搜集模块。在此基础上，进行实体信息的分级概念扩展，提取用户实体信息补充词典，并形成实体信息库，为每个注册实体产生实体的描述模板（profile）。图的中间部分是网页评价模块，该模块依据网页表示库、实体信息库和实体描述模板，过滤出包含注册实体的网页，根据网页信息和实体属性为每个实体的相关网页进行相关度评价，得到网页评分文件，进一步得到网页评分库。图的右边部分是用户服务界面，用户通过该界面登录系统，提交所要查询的注册实体，系统通过检索网页评分库和网页索引库，反馈给用户按照相关度排序的网页地址和摘要的列表，用户对得到的检索结果可以给系统提交相关度评判意见，该信息将反馈给网页评价模块，用于改进今后对该实体的网页评价模型的参数；另外，用户还可以注册新的实体，从而增加实体信息库的内容；也可以提交指定的网页，交付系统给予即时评价。

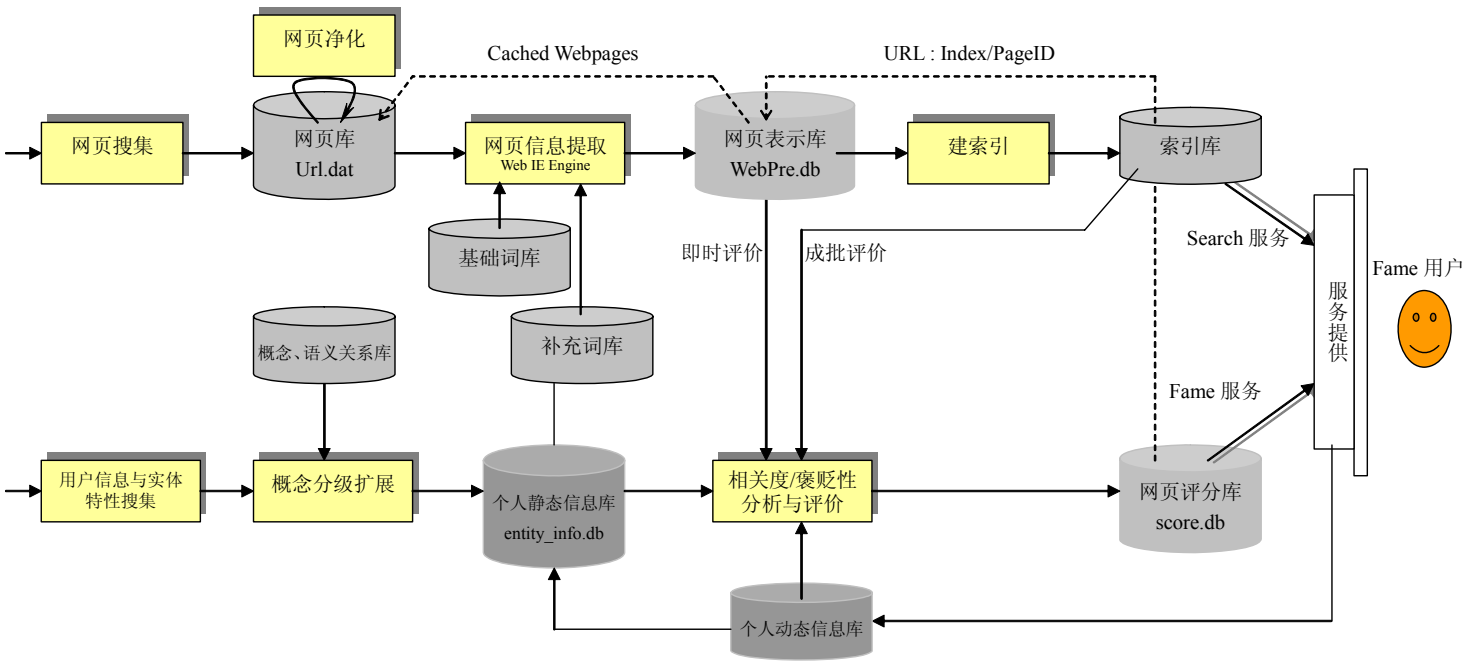


图 12-5 “天网知名度”系统结构

二、 网页与命名实体的相关度评价

对网页与命名实体的相关度进行评价，可以决定网页的检索相关排序。在网页相关度评价之前，天网知名度系统前期的处理工作有：对天网搜来的原始网页进行标记过滤，中文分词，同时进行实体名识别、实体名与实体特性关系的识别等，如在 FAME 实验系统中含有：人名识别、人名与单位（Employee_of）以及人名与职务（Post_of）等二元关系的识别，进而形成网页表示库。

网页对实体的相关度可以用一个 32 位整数表示，所有网页对注册实体的相关度评价结果存放在网页相关度评分库中。网页相关度评分库的结构是：网页编号（网页在索引库中的 ID）、实体编号（实体在实体信息库中的 ID）、该网页对该实体的相关度评分值。有了以上的准备工作，网页对实体的相关度评价流程如下：

针对网页表示库中的每一个网页：

1) 检查其实体名列表，检索用户信息库，对其中已注册的实体名建立一个该网页对该实体名的相关度评分初值；
2) 对检索出的注册实体列表，检查该网页中的二元关系和实体信息库，对符合匹配的关系为该网页的相关度评分增加一定分值，同时利用排除词表过滤掉重名的无关网页；
3) 对网页分词中的有效词（对语义理解有效的大部分实词）分别检索实体信息库的各类信息，分不同情况为该网页对实体的相关度评分增加不同分值；
4) 对网页分词中的有效词检查其 HTML 标记，分不同情况为该网页对实体的相关度评分增加不同分值；
5) 根据网页长度（按词计算）、网页中的实体名个数等因素调整其相关度评分值；
6) 形成网页相关度评分库。

基于上述思想开发的 FAME 实验系统，小规模实现了在 75 万中文简体网页的范围内提供近 300 个名人的网页搜索服务。网页相关度评价模块采用标准 C++ 编码实现，在 PIII700,内存 512MB，SCSI 硬盘的 Red Hat Linux 7.2 系统下运行正常，对 75 万网页全部处理一遍需要约 80 分钟；另外，系统还实现了对个别网页单独的相关度评价功能，从而保证了系统的时新性。

该系统采用了基于内容的浅层分析技术，提取网页中人名、人的职业描述以及人所在的工作单位描述，对查询信息建立了合理的结构，大大增加了网页中有关人物分析的准确性。同时，不同的用户可以根据个人的关心焦点对同一个人注

册不同的实体。

我们首先来看天网知名度系统与其它搜索引擎的横向比较结果。表 12-2 为部分实体在各主要检索系统中的搜索结果，其中只比较检索了前 20 条结果，数据 x/y 意为在前 y 条结果中有 x 条与实体相关；在 Fame 一列显示的是对现有 75 万网页的检索结果，在 Google、百度、天网下均有两列结果，分别显示将 Fame 的 [人名+单位+职业] 和 [人名+单位] 等信息作为关键词的查询结果（因为若把 Fame 的全部信息作为关键词，很多实体将无匹配网页），其中 y<20 的说明检索出的结果不足 20 条。

表 12-2 天网知名度系统与其他检索系统的横向比较结果

实体	领域	Fame	Google		百度		天网	
朱镕基	政府	20/20	11/11	20/20	19/20	20/20	20/20	20/20
王刚	政府	16/20	6/6	19/20	20/20	19/20	16/20	18/20
迟惠生	科教	20/20	7/7	20/20	8/8	20/20	3/3	19/20
叶天正	业界	20/20	3/3	20/20	0/0	0/0	0/0	0/0
杨澜	媒体	20/20	20/20	20/20	0/0	0/0	10/10	4/4
周大新	文学	18/20	20/20	16/20	20/20	17/20	20/20	17/20
崔健	歌星	20/20	0/0	20/20	0/0	20/20	0/0	20/20
王刚	影视	16/20	0/0	15/20	5/6	18/20	2/3	12/20
巩俐	影视	20/20	20/20	20/20	20/20	20/20	20/20	20/20
郝海东	体育	20/20	1/1	12/12	2/2	20/20	2/2	20/20

表 12-2 中数据仅为网页与实体是否相关的结果比较，可以看出，Fame 仅对 75 万网页检索的检索结果是不错的，与 Google 的海量网页库检索结果基本相当，在有几项上还要好一点，并优于百度和天网的海量网页库检索结果；而 Fame 的优势在于网页与实体相关程度的排序。由于各检索系统检索出的网页集合有很多差别，目前尚未找到一个合理的定量比较方法。但从实际检索的结果看，Fame 的排序结果优于其他检索系统，政府、科教等类的排序效果尤其突出。

其次我们来看一下不同的相关度评价策略对天网知名度检索结果的纵向影响。我们可以利用的信息有网页的词频、HTML 标记、用户注册的信息以及从网页中提取的二元关系，实验是将这些信息逐步加入我们的评价策略，观察它们检索结果的影响。表 12-3 给出了对若干个实体的六种（A-F）评价结果，具体为

A 纯文本(词频)

B A+结构化用户信息

C A+HTML 标记信息

D A+二元关系

E A+B+C

F A+B+C+D

表 12-3 实际上是一个封闭的评测结果。我们对每个实体搜集了 20 至 30 个相关网页，并且人工给出了每个网页的相关性评价，分为高[high]、中[mid]、低[low]三个等级，其中遵循的标准（可以根据用户的要求再行调整）为：相关度高的可以是有关目标实体名人的个人信息（生平介绍、作品介绍等），名人领域内的专题报道等；相关度中的可以是名人发表的评论或文章，名人的花边新闻，在其他人的报道中目标实体名人所占份额较多者，或其他不容易归到相关度高或低的情况；相关度低的可以是在其他报道（如新闻）中偶尔提到目标实体人名，引文中偶尔出现目标实体名字，或很多其他人名（编委、会议名单等情况）中偶尔出现目标实体人名等情况。

如下我们对天网知名度的检索结果与人工评判结果进行比较。基本思想是：在高、中、低三种不同的相关度的网页中，相关度得分应该满足下列关系：

Relation: [high] ≥ [mid]

[high] ≥ [low]

[mid] ≥ [low]

如果定义已知网页集合中相关度为高、中、低的个数分别

|high|=m; |mid|=n; |low|=k

那么总的关系个数为：

$$Total = m \times n + n \times k + k \times m$$

评价结果仍以 x/y 形式给出，其中 y 为有关实体网页中总的关系个数（各个实体总关系不同的原因是我们搜集的网页个数不同），x 为天网知名度系统正确评价的关系个数：

从表 12-3 的结果数据可以看出，加入结构化用户信息和二元关系之后，大部分结果有所改进，这说明利用命名实体识别和实体关系提取的浅层分析技术，在网页内容分析中有一定的效果。但是，表 12-3 中也有些不好的结果，特别是将所有的信息指标逐步加入相关度评价策略时评价的结果并没有呈现明显的单调上升趋势，而是有一定的波动，这需要再实验，调整各种信息指标的参数[昝红英,2004]。

为增加天网知名度系统检索的智能性，现准备对实体信息库中出现的词语进行人工的同义词集概念扩展，比如对“北京大学”扩展到“北大”等，以提高系统的查全率和完整性。从长远计议，可以考虑利用北京大学计算语言学研究所开发与的 WordNet 同构的中文概念语义词典（Chinese Concept Dictionary, CCD）实现对实体信息库中词语的自动概念分级扩展。

表 12-3 天网知名度系统的纵向比较结果

Entity_Name	Result_A	Result_B	Result_C	Result_D	Result_E	Result_F
朱镕基	32/38	29/38	15/38	32/38	32/38	19/38
王刚（中共中央办公厅）	41/44	41/44	29/44	41/44	41/44	38/44
李晓明	55/76	56/76	58/76	56/76	56/76	59/76
迟惠生	66/74	64/74	47/74	61/74	63/74	59/74
王玮	16/18	16/18	15/18	16/18	16/18	15/18
叶天正	57/69	56/69	50/69	55/69	55/69	55/69
杨澜	49/90	48/90	49/90	50/90	52/90	52/90
崔永元	42/98	43/98	39/98	48/98	44/98	43/98
周大新	31/40	32/40	31/40	32/40	33/40	33/40
刘墉	15/15	15/15	15/15	15/15	15/15	15/15
田震	45/66	43/66	45/66	50/66	48/66	48/66
崔健	33/45	30/45	33/45	39/45	35/45	35/45
王刚（主持人、演员）	41/48	41/48	34/48	40/48	40/48	41/48
巩俐	62/96	62/96	62/96	66/96	66/96	66/96
王刚（辽宁足球队）	51/99	51/99	51/99	56/99	54/99	54/99
郝海东	49/69	48/69	49/69	47/69	49/69	49/69

第十三章 面向主题的信息搜集与应用

Web信息分布的局部专题化是互联网信息所呈现的特征之一，伴随着面向主题信息获取的需求越来越多，用户希望主题信息获取能够做到领域信息搜集更完备、更新速度更快、并能够自动发现领域内的主要资源,进而研究主题信息的变化及其分布特征。由于主题信息一般只占整个Web很小的一部分，并且具有分散性，因此传统的基于宽度优先或深度优先的搜索策略在Web信息搜集的效率上难以达到期望要求。面向主题的信息搜集系统的主要任务是利用有限的网络带宽、存储容量和较少的时间，抓取尽可能多的主题网页。

本章第一节介绍面向主题的信息搜集方法，第二节介绍一种主题信息的搜集与处理模型。

第一节 主题信息的搜集

综合性搜索引擎如同一个公共图书馆，它试图满足各类用户的查询需求，所搜集的网页内容广而泛；而由面向主题的搜集系统所建立的主题搜索引擎，则相当于一个专业图书馆，它只搜集与主题内容相关的页面。

目前，Web 主题信息搜集的主要方法来源于 S. Chakrabarti 于 1999 年构建的 Foused Crawling 系统[Chakrabarti, et al.,1999]，该系统采用基于样例网页驱动的主题信息的搜集方法，所搜集的主题信息由用户通过选定样例网页来确定，并基于如下的假设：

如果页面 u 是一个与主题相关的页面（正例），u 到页面 v 有一个超链，则页面 v 是正例的概率远远大于在 Web 上随机抽取的一个页面。

一、 主题信息分布的局部性

为考察 Web 上主题信息分布的特征，Davison 从一个称为 DiscoWeb 的研究型搜索引擎的网页库中抽样获取了 10 万个页面[Davison,2000]，通过大量的反复试验，得到了不同情形下，两个页面之间的平均相关度。该试验利用向量空间模型中向量夹角的余弦值来度量两个页面之间的相似度,其中词条的权重用 TF*IDF 来计算。

设随机变量 u 和 v 表示 Web 上的两个页面，分别用如下统计量考察页面对之

间相似性的平均值。

当 u 和 v 为 Web 上随机抽取的两个页面时，用 Random 表示他们之间相似度的均值；当 u 和 v 被同一个页面所链接，即：u 和 v 具有相同的父节点页面时，用 Sibling 表示他们之间相似度的均值；当 u 为 Web 上随机抽取的一个页面，u 是 v 父节点且 u 和 v 具有同一个主机（由主机名确定）时，用 SameDomain 表示他们之间相似度的均值；当 u 为 Web 上随机抽取的一个页面，u 是 v 父节点,但 u 和 v 位于不同主机（由主机名确定）时，用 DiffDomain 表示他们之间相似度的均值。统计试验结果如图 13-1 所示。

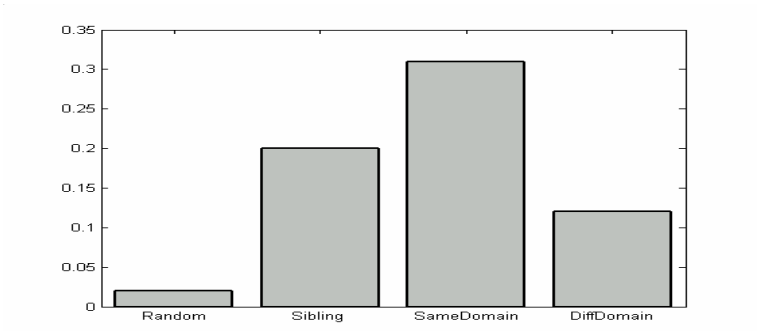


图 13-1 页面对的平均相关性

图 13-1 显示：SameDomain 对具有较高的相似度，Sibling 对次之，DiffDomain 略差，随机抽取的两个页面（Random）的相关度极低。由此亦可看到 Web 信息组织的局部化特征。从一个网页开始的随机冲浪，其主题迁移的可能性极大。

二、 一种主题信息搜集系统

Foused Crawling 系统[Chakrabarti, et al.,1999]抓取网页的过程如下：首先由用户从某一开放的分类目录体系如 Yahoo!中选取若干个子类节点作为主题信息，这些节点所包含的一些页面作为训练集，构造一个分类器（采用诸如第十一章的方法构造分类器）。当抓取到一个新的页面 u 时，首先提交到分类器进行相关度预测，如果页面 u 是一个正例（页面与主题信息的相关度超过某一个阈值），则由 u 指向的超链放入工作池（work pool）作为待抓取的超链。否则 u 被剪枝，其所指向的超链亦不进一步抓取。

Foused Crawler 主要有三部分组成：1）分类器（classifier），用于判定所抓取网页的相关性，进一步可确定是否对该网页所包含的超链进行扩展；2）提取器（distiller），用于找到已抓取网页集的 Hub，并确定待抓取 URL 的优先级；3）

抓取器（crawler），在分类器和提取器指导下、基于具有动态可配置的优先控制策略下抓取网页。系统结构如图 13-2 所示。

基本抓取思想可表述为：整个 Web 可以看作一个有向图 G，确定一个层次分类目录体系 C，如 Yahoo!等，则每一个主题 c∈C 可以看作由 G 中一些样例页面构成，记这些样例页面的集合为 D(c)。这些页面可以被系统进行预处理，用户的兴趣是 C 的一个子集 C*，用于表示用户定义的主题信息。对任意一个网页 q，q 关于 C*的相似度即为 q 与主题信息的相关程度，规定：父节点与主题信息的相关度定义为各子节点相关度之和。系统开始运行时，优先抓取 D(C*)中的页面，之后按与 D(C*)的距离及各页面相关度的大小次序进行抓取。系统的目标是抓取尽量多的相关网页，亦即追求 R(V)/|V|的最大化，其中，V 表示系统搜集到的网页集（显然，D(C*)是 V 的子集），R(V)表示 V 中与主题相关（大于某一阈值）的网页数量，或 V 中各网页的相关度之和。

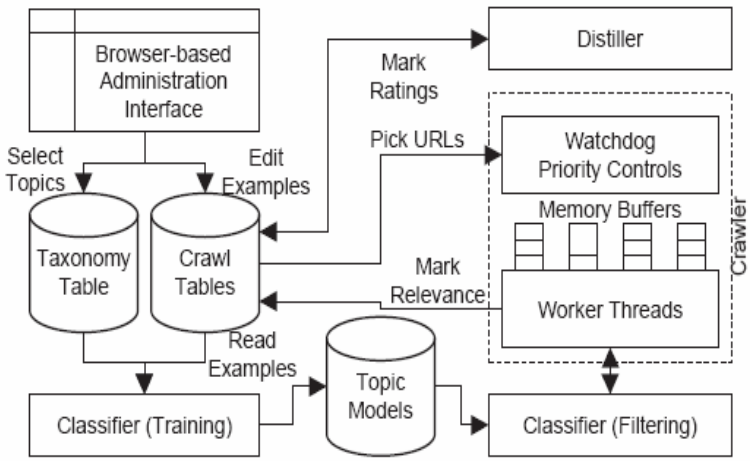


图 13-2 Foused Crawler 的系统结构

上面的系统中，分类器的选择不是必须的；如 Aggarwal 等的 Intelligent Crawling 系统就是通过若干关键词来定义主题信息的[Aggarwal, et al.,2001]，但其效果在搜集页面达到相当规模后才能体现出来，且对主题关键字较敏感。用户也可以自己挑选若干样例网页作为主题信息的定义，但其效果往往低于从某一开放的分类目录体系中选择样例并构造分类器的结果，这主要在于几个词或页面不能较好的特征化一个主题信息，特别是负例的选择面太广。

马亮等设计了一个处理中文信息的主题信息搜集系统 Irobot[马亮 and 陈群秀,2002]，该系统在对已搜集页面的主题相关度评价时综合考虑了页面的标题、

段落标题(通常由特殊字体确定)、Anchor 文本(所引用 URL 的说明文本)等对于页面评价具有较高价值的特征区域，并赋予了相对较高的权重系数，以此期望提高评价结果的准确性。对待搜集 URL 的相关性预测，考虑了一些启发因素如：一个 URL 父节点的主题相关度，URL 所对应 Anchor 文本的主题相关度，URL 的 Sibling 因素以及该 URL 所在 Web 位置的主题相关性密度等 4 个方面的因素，经加权计算后进行待搜集 URL 相关性预测，取得了较好的效果。

第二节　主题信息的一种搜集与处理模型及其应用

人们现在普遍认为，自然科学各学科的研究都有三个支柱：理论、实验和模拟。例如我们有理论物理学，实验物理学，计算物理学。随着计算机应用的不断普及，这样的方法论也在向社会科学拓展，例如在经济学研究领域用计算机来模拟市场行为已经不是新鲜事了。我们这里要指出的是，计算机在社会科学领域的应用不仅仅是模拟，在更多的场合是拓展其实验的深度和广度。我们知道，社会科学的实验主要通过采样调查，统计分析来实现。有了计算机，这样的工作在范围和规模上就可以大大扩展。不仅如此，我们还看到，由于有了计算机，社会科学工作者还有可能设计全新的实验。这是和自然科学不同的。在那里，规模更大、精度更高的实验往往意味着更加昂贵的专门实验设备。

本节以新闻传媒领域为例，提出一个实验模型。该模型的基本精神是利用一个明确定义的信息搜集与分析的过程，来确定某个主题在网络媒体上不同层面表现的强度，相当于考察传播学中“议程设置”的情况[陈燕, et al.,2002]。这里的要点是，没有非平凡的计算机技术的应用，这样的模型就不可能实现；而这个模型的程序实现，相当于为网络传播学的研究构建了一个强大的**实验设备**。

一、 模型设计

宏观上看，本节描述的模型是一个过程，它针对人们关心的热点主题，系统地对网上的信息进行搜集和分析，从不同的角度和层次得出互联网对该主题报道的强度。它包含如下几个步骤：样本空间的选取，主题特征的提取，设置目标参量，网页的搜集以及数据的后处理。下面分别说明这些步骤。

1. 样本空间的选取

当我们要系统地研究一个主题在互联网上表现强度的时候，最理想、最彻底的办法是将网上的所有信息考察一遍。但这显然是不现实的。通常，我们只能取一个样板空间来研究。具体来说，样板空间就是网页总体集合的一个子集，对应

于若干特定的网站中的若干特定的网页。这有两个因素需要考虑：

1) 样板空间的选取要和主题宣传的设计受众相关。这里强调“设计受众”是因为我们关心的主题的表现强度只是针对它的设计受众才有意义。例如，若主题是“F4”，则要选取 20 岁左右的青年人喜欢上的网站；若主题是“创建世界一流大学”，则要选取大学师生和管理人员经常上的网站。
2) 样本空间本身有“时间”和“空间”两方面的含义。所谓“时间”含义是指对一个主题的舆论强度研究常常要有一个时间区间，在时间上如何采样是需要斟酌的，每隔一小时、一天，还是一个星期；所谓“空间”含义是指不仅要考虑对哪些网站的选取，还可能要考虑对所选取的网站中哪些内容采样。

2. 主题特征的提取

在本节模型中，主题的特征分为两个层次，各为一个词组的集合。第一个层次叫做**主题词组**，第二个层次叫做**主题相关词组**；按定义，后者总是包含前者。为提取主题特征，我们借鉴[Glover, et al.,2002]实验方法中的基于全文分类方法, 并根据中文信息的特点进行了简化和改进。

首先人工判断整理少量和主题相关的典型网页（训练集），然后对它们的内容进行串频统计。串频统计以任意相邻的 2－15 个字为一个串，统计它们在训练集中出现的频率。提取出现频率较高的串，删除其中的常用普通词和无意义的串后，把它们作为该主题的特征（这些串被称为特征串，下同），进而人工将它们分成上述两个层次。这样形成的两个集合，主题词组和主题相关词组，就构成了后面判断的基础：如果一个网页含有主题词组中任意一个词，则我们称它为主题网页，若一个网页含有主题相关词组中任意一个词，则我们称它为主题相关网页。

3. 目标参量

作为模型的输出目标，我们对样本空间的网页从 3 个正交的维度，每个维度分两个层次进行考察。

第一维度：宏观统计，变化过程

不仅考虑整个时间区间内搜集到的主题网页的总量，还考虑每个时间片上的主题网页数量的变化情况。整个时间区间上的主题网页的总量宏观地表现了该主题在新闻媒体报道中的重视程度。而每个时间片上的主题网页数量的变化情况，又可以详细的刻画出该主题报道的孕育、产生、增长、高潮、渐退直至消亡整个过程。

第二维度：绝对数量，相对数量

不仅考虑主题网页的绝对数量，还考虑主题网页在同期内全部网页数量中所

占的比例。绝对数量和相对数量为我们比较不同时期的不同主题的报道力度提供了两种视野。比较主题网页的绝对数量，我们可以得出谁是更强大的主题报道；而比较主题网页的相对数量，我们可以得出所关心主题在媒体报道中的突出程度，两种既有区别又有联系。

第三维度：总体信息，独立信息

不仅考虑总的聚合信息量，还考虑从时间、空间上消重后的独立信息量。所谓“消重”是针对网上内容存在极大的复制现象而言的。通过对我国网上信息的一次全面搜集，我们曾经发现网页的平均复制律达到 4。相同的一篇报道可能被转载在不同的网页中，这种信息复制无疑给互联网用户的信息获取提供了极大的方便，但相对单个用户来说，还存在能最多获取多少条不同信息的问题。我们对搜集来的网页进行内容消重，把所有内容相同的网页算作一篇独立信息的网页，得到独立信息的网页数量，从而可以看出对该主题报道的丰富性、综合性。

注意到这三个维度的正交性，我们得到如图 13-3 所示的立方体，即我们有 8 个方面的数据要产生。

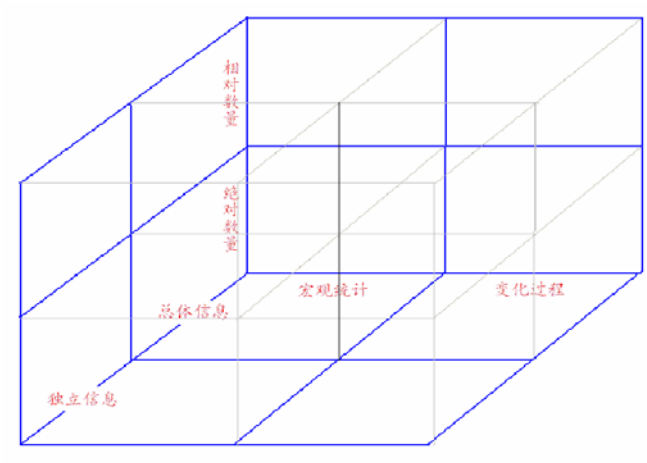


图 13-3 用于表达网上主题新闻强度指标的立方体

例如“宏观统计，绝对数量，总体信息”指的是在确定的时空样板空间中所有和主题相关网页的总量、平均值、标准差等；而“变化过程，相对数量，独立信息”则表示的是在时间轴上展开样板空间的内容，考察不重复计算的主题相关信息量和总体信息量之间的相对关系。

4. 网页的搜集

网页搜集策略直接影响主题的研究结果，不同的搜集策略可能得出完全不同的研究结果。好的搜集策略应该与样本空间有很好的对应，满足样本空间的“时

间”和“空间”两方面的需求。我们先讨论在搜集中可能会遇到的问题，并给出一个好的面向主题的搜集策略应该满足哪些条件。

在网页搜集过程中，我们经常会遇到如下问题：

1) 网页的流逝性

根据 Junghoo Cho 的实验结果[Cho and Garcia-Molina,2000]，对于热门站点10%以上的网页的生命周期不超过一个星期。而这些短暂的网页可能是对主题的及时报道，它们的消失对主题研究是一个损失。

2) 普通的搜索引擎的周期太长

目前的搜索引擎大多采用广度搜集策略，定期对网页进行全面的搜集。而在两次搜集期间，可能会有一大批网页因为更新或删除而无法搜到。

3) 搜集的网页重复度大

Web 页面的平均生命周期为 138 天，最常见的生命周期为 62 天[Cho and Garcia-Molina,2000]。按普通的搜索策略，如果把搜索周期缩短的话，则搜集的大多数网页都是与前一次搜集相同的网页。这种搜集的效率太低，开销太大。

鉴于此，我们认为，面向主题的高效率的搜集策略应该有以下特点：

1) 搜集频率足够高，保证周期短暂的网页仍然能有很大的概率被搜集到。（满足样本空间“时间”性的需求）
2) 搜集策略必须保证有较高的数量覆盖率和质量覆盖率[Meng, et al.,2003]。（满足样本空间“空间”性的需求）
3) 搜集策略必须具有较好的技术特性，以保证搜集过程的有效。例如，通常搜集是由多个进程长时间协同工作完成，如果在同一时间这些进程都集中到一个网站上，则可能造成阻塞和拒绝服务，从而使得搜集过程出现异常。

5. 数据的后处理

1) 消重算法

前面提到过，要得到独立信息量，必须对具有相同内容的网页进行消重。对于搜集到的网页，存在两种形式的内容重复。第一种是在同一次搜集的网页中，可能存在着相同内容的网页，它们的产生是由于同一篇报道在不同网站上的复制，对应不同的 URL。这种复制是互联网常见的现象，我们把对这种情况的消重称作**空间上消重**。另一种是在多次搜集中产生的，有些网页可能在多次搜集中都没有变化（即 last-modify-time 没变），这样它就被搜集了多次，我们把对这种情况的消重称作**时间上消重**。

本模型采用第七章第三节中的消重算法：对每一篇网页进行切词，然后统计

词频，并按词频由高到低的顺序对词进行排序，把这个序列看作一个字符串，截取前 2048 个字节生成 MD5（16 个字节）作为网页的一个属性，认为具有相同的 MD5 属性的网页它们的网页内容是相同的。这样就可以根据 MD5 属性对网页进行消重。该算法具有 98%的准确率 。

消重工作分为两步：第一步是对每个时间片上搜集的网页进行消重（空间上消重），统计出每个时间片上的独立信息量；第二步再将所有时间片上消重后的网页合并在一起再进行一次消重（时间上消重），统计出整个时间区间内的独立信息量。

2) 残差数据的处理

经上述过程得到的数据，绝大多数都是令人满意的，但仍会有个别数据出现了反常，这主要有两方面的因素。第一个因素是非可预测的破坏因素，比如网络故障、机器故障等，它直接影响了该时间片上的搜集效果；第二个因素与搜集策略有关，如果搜集策略的稳定性不够好，就会在个别时间片上出现明显的异常。

对于上述所产生的“噪音”数据，可以通过插值去估计真实的数据。我们发现，把邻近的四个时间片作为插值点，利用三次拉格朗日插值基本上可以理想的估计出反常时间片上的数据。

经过这样的处理，根据上述 3 维分析的方法，一般就可以比较详细的得出一个主题从信息量的角度上在网上被反映的力度。

二、 应用实验：以“十六大”为主题

利用上述模型，我们对最近召开的党的十六大网上信息量进行了一次实验性研究。

样本空间：其基础选自CNNIC发布的“中国互联网络网站影响力调查报告”中的有影响力站点列表，包含了全国最有影响力的 143 个网站[1]。我们认为十六大这样的宣传主题，受众是全国人民，选择这样的样本空间是最合适的。同时，我们确定时间跨度为以 11 月 8 日为中心，前后各 16 天，即从 10 月 22 日至 11 月 24 日。搜集的时间片单位是天。

主题特征：首先我们从新浪网获得若干对“十六大”的专题报道网页（共262个），利用串频统计得到“十六大”主题相关特征串为：十六大、16大、第十六次全国代表大会、16届一中全会。把包含其中任意一个特征串的网页称为“十六大网页”，它们直接报道了十六大这个主题。“十六大”主题相关特征串除包含上述4个外，还有：三个代表、小康社会、与时俱进、党的建设等21个串。把包含其中任意一个特征串的网页称为“十六大相关网页”，其中有些网页虽然没包含“十六

[1] http://tech.sina.com.cn/internet/china/2000-07-27/31930.shtml

大”等字样，但我们认为它们也是和十六大主题紧密相关的。

目标参量：和前述模型一致，我们考察8个方面的数据。本节的后边给出其中几个有代表性的，完整的结果见参考文献[Li, et al.,2003]。

网页搜集：我们设计了一种按层搜索策略，它的方法如下：

1) 对于一个网站的网址，我们把主页作为第一层；把主页链向的网页作为第二层；把第二层链向的网页作为第三层；……
2) 在每次搜集中，把上述 143 个网站的主页（第一层）作为第一次搜集目标，完成后再搜集第二层网页，然后搜集第三层和第四层。
3) 每天搜集一次，晚上 10 开始搜集，一般到第二天中午四层搜集完成。在这期间得到的网页都算搜集启动那一天的。

我们从 10 月 22 日到 11 月 24 日利用一台计算机进行了按层搜索。平均每天搜集得到约 32 万网页。

后处理：在这34天的网页搜集中，有4天因为网络故障或机器故障而导致数据偏差过大；有3天因为搜集策略不够稳定而有偏差（比如说，我们在搜集第一层的某个主页时，这时对方的服务器忙或其它暂时的故障而无法返回该主页，这样我们就会无法搜集这个网站的所有网页，搜集来的网页数会明显减少。虽然我们把每天的搜集开始时间放在晚上10点以避免网络高峰，但这种因素仍是存在的。）对此，我们利用三次拉格朗日插值对这几天的网页数进行了估计（对每个插值点，取周围的4个有效数据为参考），形成了一套完整的数据。我们发现：

在搜集的约一千万个网页中，包含有“十六大”等和党的十六大直接相关字样的网页 31 万多篇，平均每天 9 千多篇，占总数的 2.9%；包含有“三个代表”等和十六大精神相关字样的网页 79 万多篇，平均每天 2.3 万多篇，占总数的 7.3%。经过消重，共有信息独立性网页 230 多万篇，其中包含有“十六大”等和党的十六大直接相关字样的信息独立性网页 13 万多篇，平均每天近 4000 篇，占总数的 5.6%；发现包含有“三个代表”等和十六大精神相关字样的信息独立性网页 27 万多篇，平均每天 8,000 多篇，占总数的 11.6%。我们注意到，总体网页的复制率要比十六大主题网页复制率高得多，前者约为 4.67，而后者约为 2.39，这也是主题新闻性很强的一个证据。并发现十六大网页并不是在 11 月 8 日达到高峰，而是在 11 月 2 日之前基本平缓，从 11 月 2 日开始逐日递增，在 11 月 20 日达到最高，如图 13--4 所示。

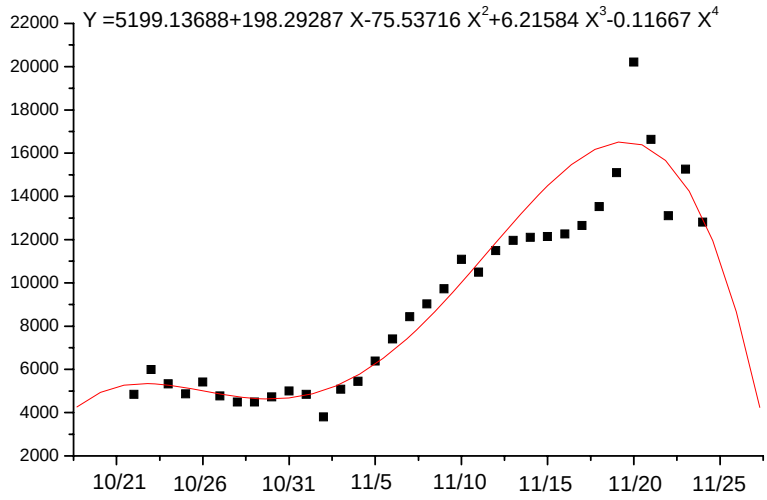


图 13-4 十六大网页数量在 10 月 22 至 11 月 24 期间的变化情况

三、 总结与讨论

从对海量网络信息内容的实验性研究出发，作为计算机技术在社会科学中的一种应用，本节提出了一种针对特定主题的研究模型。该模型的基本目标是主题信息的强度。针对体现主题信息在时空意义上强度的三个角度，这个模型表现为一个对信息进行系统的搜集和处理的过程。将这个过程用计算机程序来实现，我们就得到了一个可用于海量网络信息研究的“实验设备”。

基于该“设备”，我们对网上十六大信息进行了 34 天的观察实验，得出了一系列有意义的数据。这个应用结果说明了上述模型是可行的、有用的。同时我们碰巧注意到，2002 年 11 月 6 日江泽民主席在会见“电视与广播博物馆国际理事会 2002 年北京年会”代表时讲“因特网已成为中国新闻传媒的重要组成部分”。本项实验的结果数据是该论断的一个具体体现。

从进一步的工作看，可以有两个方面。一是在模型中引入优化指标，例如考虑为得到特定的统计数据最少需要花费的计算资源和时间。二是更深层次地考察这次得到的数据，以及选择其它主题进行类似实验，看能否得出有一定规律的结论，例如“一个预定事件网上舆论的高峰在该事件发生后 X 天左右出现”之类；这需要在样板空间的选择和搜集策略上做进一步的研究。

参考文献

[Aggarwal, et al.,2001] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "Intelligent crawling on the World Wide Web with arbitrary predicates," in *Proceedings of the tenth international conference on World Wide Web*. Hong Kong, Hong Kong: ACM Press, 2001, pp. 96--105.

[Almeida, et al.,1996] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," presented at Proceedings of the 1996 4th International Conference on Parallel and Distributed Information Systems, Dec 18-20 1996, Miami Beach, FL, USA, 1996.

[Anh and Moffat,2002] V. N. Anh and A. Moffat, "Impact transformation: effective and efficient web retrieval," presented at Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, Tampere, Finland, 2002.

[Ashish and Knoblock,1997] N. Ashish and C. Knoblock, "Wrapper generation for semistructured Internet sources," presented at Proc. PODS/SIGMOD'97, 1997.

[Baeza-Yates and Ribeiro-Neto,1999] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley-Longman, 1999.

[Bahle, et al.,2002] D. Bahle, H. E. Williams, and J. Zobel, "Efficient phrase querying with an auxiliary index," presented at Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.

[Baldi, et al.,2003] P. Baldi, P. Frasconi, and P. Smyth, *Modeling the Internet and the Web, probabilistic methods and algorithms*: John Wiley, 2003.

[Bergman,2000] M. K. Bergman, "The Deep Web: Surfacing Hidden Value," 2000. (http://www.dad.be/library/pdf/BrightPlanet.pdf).

[Brin and Page,1998] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," presented at Proceedings of the 7th Internation World Wide Web Conference/Computer Networks, Amsterdam, 1998.

[Broder, et al.,2000] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and a. J. Wiener, "Graph structure in the web: experiments and models," presented at Proceedings of the 9th World-Wide Web

Conference, Amsterdam, 2000.

[Buckley and Voorhees,2000] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," presented at Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000), Athens, Greece, 2000.

[CCF,2004] Chinese Character Frequency statistics. http://lingua.mtsu.edu/chinese-computing/statistics/.

[Chakrabarti, et al.,1999] S. Chakrabarti, M. v. d. Berg, and B. Dom, "Focused Crawling : a new approach to topic-specific Web resource discovery," presented at Proceedings of the 8th World-Wide Web Conference, Toronto, Canada, 1999.

[Chakrabarti, et al.,2001] S. Chakrabarti, M. Joshi, and V. Tawde, "Enhanced topic distillation using text, markup tags, and hyperlinks," presented at 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sep 9-13 2001, New Orleans, LA, 2001.

[Chidlovskii, et al.,1999] B. Chidlovskii, C. Roncancio, and M.-L. Schneider, "Semantic cache mechanism for heterogeneous Web querying," *Computer Networks*, vol. 31, pp. 1347-1360, 1999.

[Cho and Garcia-Molina,2000] J. Cho and H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler," presented at Proceedings of 26th International Conference on Very Large Databases (VLDB), 2000.

[Cho,2002] J. Cho, "Crawling the Web: Discovery and maintenance of large- scale Web data," Stanford University,PhD, 2002, pp. 188.

[ChSeg,2003] Chinese Segmentation. http://net.cs.pku.edu.cn/~webg/src/ChSeg/.

[Church and Hanks,1990] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Comput. Linguist.*, vol. 16, pp. 22--29, 1990.

[CNNIC,2004] China Internet Network Information Center. (中国互联网络信息中心) http://www.cnnic.net.cn.

[Cormack, et al.,1998] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke, "Efficient construction of large test collections," presented at Proceedings of the 1998 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98),, Melbourne, Vic., Aust, 1998.

[Cormen, et al.,2001] T. H. Cormen, C. E.Leiserson, R. L.Rivest, and C. Stein, Introduction *to Algorithms, Second Edition: The MIT* Press, 2001.

[Craswell, et al.,2003] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu, "Overview of the TREC-2003 Web Track," presented at Proceedings of TREC 2003,

Gaithersburg, Maryland USA, 2003.

[Crovella and Bestavros,1997] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," IEEE/ACM T*ransactions on Networking, vol. 5,* pp. 835-846, 1997.

[Davison,2000] B. D. Davison, "Topical locality in the web," presented at SIGIR Forum (ACM Special Interest Group on Information Retrieval),
Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Infornation Retrieval (SIGIR 2000), Athens, Greece, 2000.

[Dublincore,2003] Dublin Core Metadata Element Set, Version 1.1: Reference Description. http://dublincore.org/documents/dces/.

[EAD,2002] Encoded Archival Description (EAD). Official EAD Version 2002 Web Site, http://www.loc.gov/ead/.

[EF,2004] English word Frequency from Editorials. http://www.supervoca.net/editorial/.

[eTesting,2000] L. eTesting, "Google Web Search Engine Evaluation," 2000. (http://www.veritest.com/clients/reports/google/google.pdf).

[Fame,2004] 天网知名度. http://net.pku.edu.cn/~fame/.

[Frieder, et al.,1999] O. Frieder, D. A. Grossman, A. Chowdhury, and G. Frieder, "Efficiency considerations of scalable information retrieval servers," Journal of Digi*tal Information, 1999.*

[Glover, et al.,2002] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake, "Using web structure for classifying and describing web pages," presented at Proceedings of the eleventh international conference on World Wide Web, Honolulu, Hawaii, 2002.

[Google,2004] Google Search Engine. http://www.google.com.

[Hammer, et al.,1997] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha, "Extracting Semistructured Information from the Web," presented at Workshop on Management of Semistructured Data, Tucson, Arizona, 1997.

[Hawking, et al.,1999] D. Hawking, N. Craswell, P. Thistlewaite, and D. Harman, "Results and challenges in Web search evaluation," Computer Networ*ks, vol. 31, pp.* 1321-1330, 1999.

[Hawking, et al.,2001] D. Hawking, N. Craswell, P. Bailey, and K. Griffiths, "Measuring search engine quality," Information Retr*ieval, vol. 4, pp.* 33-59, 2001.

[Heinz and Zobel,2003] S. Heinz and J. Zobel, "Efficient single-pass index construction for text databases," Journal of the *American Society for Information Science and Technology, vol. 54, pp.* 713-729, 2003.

[HTMLTidy,2004] HTML Tidy Library Project. http://tidy.sourceforge.net/.

[Hull,1993] D. Hull, "Using statistical testing in the evaluation of retrieval experiments," presented at Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Jun 27-Jul 1 1993, Pittsburgh, PA, USA, 1993.

[InfoMall,2004] China Web InfoMall. http://www.infomall.cn.

[Jian,1991] R. Jian, The Art of Comp*uter Systems Performance Analysis:Techniques for Experimental Design, Measurement, Simulation, and Modeling. NewYork: John* Wiley & Sons, Inc., 1991.

[Jin and Bestavros,2000] S. Jin and A. Bestavros, "Sources and characteristics of Web temporal locality," presented at Proceedings of the 8th Interantional Symposium on Modeling, Analysis and Simulation of Computer and Telecomunication System, Aug 29-Sep 1 2000, San Francisco, CA, USA, 2000.

[Jonsson, et al.,1998] B. T. Jonsson, M. J. Franklin, and D. Srivastava, "Interaction of query evaluation and buffer management for information retrieval," presented at Proceedings of the ACM SIGMOD International Conference on Management of Data, Jun 1-4 1998, Seattle, WA, USA, 1998.

[Josuttis,1999] N. M. Josuttis, The C++ Standar*d Library A Tutorial Reference: Addison Wesle*y Longman, Inc., 1999.

[Kang and Kim,2003] I.-H. Kang and G. Kim, "Query Type Classification for Web Document Retrieval," presented at Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2003, Jul 28-Aug 1 2003, Toronto, Ont., Canada, 2003.

[Kleinberg,1998] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," presented at Proceedings of the 1998 9th Annual ACM SIAM Symposium on Discrete Algorithms, Jan 25-27 1998, San Francisco, CA, USA, 1998.

[Knuth,1973] D. Knuth, The Art of Comp*uter Programming, Volume 3: Sorting and Searching (2nd Edition): Addison-Wesle*y Pub Co; 2nd edition (April 24, 1998), 1973.

[Kowalski,1997] G. Kowalski, Information Ret*rieval Systems: Theory and Implementation: Kluwer Academi*c Publishers, Boston, MA, 1997.

[Lawrence and Giles,1998] S. Lawrence and C. L. Giles, "Searching the world wide web," Science, vol. 2*80, pp.* 98-100, 1998.

[Lei, et al.,2001] M. Lei, J. Y. Wang, B. J. Chen, and X. M. Li, "Improved relevance ranking in WebGather," Journal of Comp*uter Science and Technology, vol. 16, pp.* 410-417, 2001.

[Li and Shi,2002] X. Li and Z. Shi, "Innovating web page classification through reducing noise," Journal of Comp*uter Science and Technology, vol. 17, pp.* 9-17, 2002.

[Li, et al.,2003] X. M. Li, J. J. Zhu, and H. F. Yan, "A Model for Collecting and Processing Topical Information in the Web and Its Application," Journal of Comp*uter Research and Development (in Chinese), vol. 40, pp.* 1667-1671, 2003. (李晓明，朱家稷，闫宏飞，互联网上主题信息的一种收集与处理模型及其应用，《计算机研究与发展》2003 年 12 月 Vol.40 No.12).

[Lin and Ho,2002] S.-H. Lin and J.-M. Ho, "Discovering informative content blocks from Web documents," presented at KDD - 2002 Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Jul 23-26 2002, Edmonton, Alta, Canada, 2002.

[Liu, et al.,2000] J. Liu, M. Lei, J. Wang, and B. Chen, "Digging for gold on the web: Experience with the WebGather," presented at Proceedings of the 4th International Conference on High Performance Computing, The Asia-Pacific Region, IEEE Computer Society Press, 2000.

[Lu,1999] Z. Lu, "Scalable distributed architectures for information retrieval," University of Massachusetts Amherst,PhD, 1999, pp. 178.

[Manber,1994] U. Manber, "Finding similar files in a large file system," presented at Usenix Winter 1994 Technical Conference, San Francisco, 1994.

[Markatos,2001] E. P. Markatos, "On caching search engine query results," Computer Communications, vol. 24, pp. 137-143, 2001.

[Meng, et al.,2003] T. Meng, H. F. Yan, and X. Li, "An Evaluation Model on Information Coverage of Search Engines," ACTA Electronica Sinaca, vol. 31, pp. 1168-1172, 2003. (孟涛，闫宏飞，李晓明，搜索引擎信息覆盖率模型研究，《电子学报》2003(8) Vol.31 No.8).

[Moffat and Zobel,1992] A. Moffat and J. Zobel, "Parameterised compression for sparse bitmaps," in Proceedings of the 15th annual international ACM SIGIR conference on Research and development *in information retrieval. Copenhagen, Denmark: ACM Press, 1992, pp. 274--285.*

[Moffat and Zobel,1994] A. Moffat and J. Zobel, "Compression and fast indexing for multi-gigabyte text databases," Australian Computer Journal, vol. 26(1), 1994.

[Moffat and Zobel,1996] A. Moffat and J. Zobel, *"Self-indexing inverted f*iles for fast text

retrieval," Acm Transactions on Information Systems, vol. 14, pp. 349-379, 1996.

[Najork and Heydon,2001] *M. Najork and A. Heydon, "High-Perform*ance Web Crawling," Compaq Systems Research Center Sep 2001.

[Najork and Wiener,2001]M. Najork and J. L. Wiener, "Breadth-first crawling yields high-quality pages," in Proceedings of the tenth international conference on World Wide Web: ACM Press, 2001, pp. 114--*118.*

[Navarro, et al.,2000] *G. Navarro, E. S. De Moura, M. Neube*rt, N. Ziviani, and R. Baeza-Yates, "Adding compression to block addressing inverted indexes," Information Retrieval, vol. 3, pp. 49-77, 2000.

[Newman,2001] H. Newman, "Application Perform*ance and I/O," Storag*e,SW Expert, 2001. (http://swexpert.com/CB/SE.C11.APR.01.pdf).

[Nie, et al.,2000] J.-Y. *Nie, J. Gao, J.* Zhang, and M. Zhou, "On the use of words and n-grams for Chinese information retrieval," presented at Proceedings of the fifth international workshop on on Information retrieval with Asian languages, Hong Kong, China, 2000.

[Page, et al.,1998] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project 1998.

[Persin, et al.,1996] M. Persin, J. Zobel, and R. Sacks-Davis, "Filtered document retrieval with frequency-sorted indexes," Journal of the American Society for Information Science, vol. 47, pp. 749-764, 1996.

[RFC2045*, et al.,2004]* *The primary definition of MIME.* *http://*www.faqs.org/rfcs/index.html.

[RFCs,2004] RFCs Collection. http://www.faqs.org/rfcs/index.html.

[Sadakane and Imai,2001]K. Sadakane and H. Imai, "Fast algorithms for k-word proximity search," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. *E84-A, pp. 2311-2318, 2001.*

[Salton and Lesk,1968] *G. Salton and M. E. Lesk, "Computer* Evaluation of Indexing and Text Processing," J. Acm, vol. 15, pp. 8--36, 1968.

[Salton,1971] G. Salton, The SMART Retrieval System - Exper*iments* in Automatic Document Processing: Prentice Hall Inc., *Englewood Cliffs, NJ, 1971.*

[Saracevic,1995] *T. Saracevic, "Evaluation* of evaluation in information retrieval," presented at Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Jul 9-13 1995, Seattle, WA, USA, 1995.

[Saraiva, et al.,2001] P. C. Saraiva, E. S. De Moura, N. Ziviani, W. Meira, R. Fonseca, and B. Ribeiro-Neto, "Rank-preserving two-level caching for scalable search engines," presented at 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sep 9-13 2001, New Orleans, LA, 2001.

[Scheuermann, et al.,1998] P. Scheuermann, G. Weikum, and P. Zabback, "Data partitioning and load balancing in parallel disk systems," the VLDB Journal, 1998.

[Searchenginewatch,2004] Searchenginewatch. http://www.searchenginewatch.com/.

[Sebastiani,1999] F. Sebastiani, "A tutorial on Automated Text Categorization," presented at Analia Amandi and Alejandro Zunino (eds.), Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence, Buenos Aires, AR, pp. 7-35, 1999.

[Shivakumar and Garca-Molina,1998] N. Shivakumar and H. Garca-Molina, "Finding near-replicas of documents on the web," presented at Proceedings of Workshop on Web Databases (WebDB'98), Mar, 1998.

[Silverstein, et al.,1998] C. Silverstein, M. Henzinger, J. Marais, and M. Moricz, "Analysis of a Very Large AltaVista Query Log," Hewlett Packard Laboratories SRC-TN-1998-014, oct 1998.

[Singhal and Kaszkiel,2001] A. Singhal and M. Kaszkiel, "A case study in web search using TREC algorithms," presented at Proceedings of the tenth international conference on World Wide Web, Hong Kong, Hong Kong, 2001.

[Spink, et al.,2001] A. Spink, D. Wolfram, B. J. Jansen, and a. T. Saracevic, "Searching the web: The public and their queries," Journal of the American Society for Information Science, vol. 53, pp. 226--234, 2001.

[Stevens,1996] W. R. Stevens, TCP for Transactions, HTTP, NNTP, and the UNIX(R) Domain Protocols (TCP/IP Illustrated, Volume 3): Addison-Wesley Pub Co, 1996.

[Stokoe, et al.,2003] C. Stokoe, M. P. Oakes, and J. Tait, "Word Sense Disambiguation in Information Retrieval Revisited," presented at Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2003, Jul 28-Aug 1 2003, Toronto, Ont., Canada, 2003.

[Sullivan,2004] Major Search Engines and Directories. http://searchenginewatch.com/links/article.php/2156221.

[Tomasic and Garcia-Molina,1993] A. Tomasic and H. Garcia-Molina, "Performance of

inverted indices in shared-nothing distributed text document information retrieval systems," presented at Proceedings of the Second International Conference on Parallel and Distributed Information Systems, 1993.

[Travis and Broder,2001] B. Travis and A. Broder, "The Need Behind the Query: Web Search vs Classic Information Retrieval," presented at The Sixth Search Engine Meeting, Boston, Massachusetts, 2001.

[TSE,2004] Homepage of Tiny Search Engine. http://net.pku.edu.cn/~webg/src/TSE/.

[Voorhees,2001] E. M. Voorhees, "Evaluation by highly relevant documents," presented at 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, LA, 2001.

[Voorhees and Buckley,2002] E. M. Voorhees and C. Buckley, "The effect of topic set size on retrieval experiment error," presented at Proceedings of the Twenty-Fifth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Aug 11-15 2002, Tampere, Finland, 2002.

[W3C,1997] W3C, "A Lexical Analyzer for HTML and Basic SGML," WWW Consortium (W3C), 1997. (http://www.w3.org/MarkUp/SGML/sgml-lex/sgml-lex).

[W3C,1999] W3C, "HTML 4.01 Specification," WWW Consortium (W3C), 1999. (http://www.w3.org/TR/1999/REC-html401-19991224/).

[Wang, et al.,2001] J. Wang, S. Shan, M. Lei, Z. Xie, and X. Li, "Web search engine: characteristics of user behaviors and their implication," Science in China, Series F, vol. 44, pp. 351--365, 2001. (王建勇、单松巍、雷鸣、谢正茂、李晓明，海量 web 搜索引擎系统中用户行为*的分布特征及其启示，《中国科学》E 辑，2001 年 8 月*，第 31 卷，第四期，372-384 页。).

[Wemble, et al.,1999] D. Wemble, Y. Jiang, and Y. K. Ng, "Record-Boundary Discovery in Web Documents," presented at Proc.SIGMOD'99, 1999.

[Williams and Zobel,1999] H. E. Williams and J. Zobel, "Compressing integers for fast file access," Computer Journal, vol. 42, pp. 193-201, 1999.

[Williams, et al.,1999] H. E. Williams, J. Zobel, and P. Anderson, "What's Next? - Index Structures for Effi*cient Phrase Quer*ying," presented at Proc. Australasian Database Conference, Auckland, New Zealand, 1999.

[Witten, et al.,1994] I. H. Witten, A. Moffat, and T. C. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images. New York, NY: Van Nostrand Reinhold, 1994.

[Wong,1997]C. Wong, "The Robot Exclusion Sta*ndard," in Web Client Programming with Perl: O'Reilly, 1997.*

*(htt*p://www.oreilly.com/openbook/webclient/appc.html).

[Xie and O'Hallaron,2002] Y. Xie and D. O'Hallaron, *"Locality in search engine que*ries and its implications for caching," presented at IEEE Infocom 2002, Jun 23-27 2002, New York, NY, United States, 2002.

[Yan, et al.,2001a] H. F. Yan, J. Y. Wang, X. M. Li, and L. Guo, "Architectural design and evaluation of an efficient Web-crawling system," presented at Proceedings of 15th International Parallel and Distributed Processing Symposium, San Francisco, California, USA, 2001a. (Also published in Journal of Systems and Software, vol. 60, pp. 185-193, Feb 15, 2002.).

[Yan, et al.,2001b] H. F. Yan, J. Y. Wang, and X. M. Li, "A dynamically reconfigurable model for a distributed Web crawling system," presented at International Conference on Computer Networks and Mobile Computing, Beijing, China, 2001b.

[Yan and Li,2002] H. F. Yan and X. Li, "On the Structure of Chinese Web 2002," Journal of Computer Research and Development, vol. 39, pp. 958-967, 2002. (闫宏飞，李晓明，关于中国 Web 的大小、形状和结构，《计算机研究与发展》（Grid 特刊）2002 年 8 月，http://net.cs.pku.edu.cn/~yhf/chinaweb_yhf_lxm.pdf).

[Yang,1995] Y. Yang, "Noise reduction in a statistical approach to text categorization," presented at Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Jul 9-13 1995, Seattle, WA, USA, 1995.

[Yang and Pedersen,1997] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," presented at Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997.

[Yang and Liu,1999]Y. Yang and X. Liu, "A re-examination of text categorization methods," presented at Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), 1999.

[Yang,2001] Y. Yang, "A study on thresholding strategies for text categorization," presented at 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sep 9-13 2001, New Orleans, LA, 2001.

[Yang and Zhang,2001] Y. Yang and H. Zhang, "Page Analysis Based on Visual Cues," presented at Poster Proceedings of the 10th International WWW Conference, Hongkong, 2001.

[Zhang, et al.,2004] Z. Zhang, J. Chen, and X. Li, "A preprocessing framework and

approach for web applications," Journal of Web Engineering, vol. 2, pp. 175--191, 2004.

[Zobel, et al.,1992] J. Zobel, A. Moffat, and R. S. Davis, "An efficient indexing techniques for full-text database systems," presented at *Proc. 18th International* Conference on Very Large Databases, 1992.

[Zobel,1998] J. Zobel, "How reliable are the results of large-scale information retrieval experiments?," presented at Proceedings of the 1998 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98),, Melbourne, Vic., Aust, 1998.

[卜东波,2000] 卜东波, "聚类/分类理论研究及其在文本挖掘中的应用," 中科院,博士论文, 2000.

[陈燕, et al.,2002] 陈燕, 陶丹, and 李广增, 传播学研究方法: 科学出版社, 2002.

[单松巍,2003] 单松巍, "搜索引擎的日志分析：方法、技术和应用," 北京大学,硕士论文, 2003.

[冯是聪,2003] 冯是聪, "中文网页自动分类技术研究及其在搜索引擎中的应用," 北京大学,博士论文, 2003, pp. 88.

[冯是聪, et al.,2003] 冯是聪, 张志刚, and 李晓明, "一种中文网页自动分类方法的实现及其应用," 计算机工程, 2003. (已收录).

[黄菁萱 and *吴立德,1998]* *黄菁萱* and 吴立德, "基于向量空间模型的文档分类系统," 模式识别与人工智能, vol. 1, 1998.

[雷鸣,2000] 雷鸣, "一个大规模、高性能的搜索引擎系统及索引和检索子系统的实现," 北京大学,硕士论文, 2000.

[刘开瑛,2000] 刘开瑛, 中文文本自动分词和标注. 北京: 商务印书馆, 2000.

[马亮 and 陈群秀,2002] 马亮 and 陈群秀, "智能 Web 中文主题信息收集系统 IRobot 的设计," 中文信息学报, vol. 16, pp. 23-29, 2002.

[彭波,2004a] 彭波, "搜索引擎检索系统的的效率优化和效果评估研究," 北京大学,博士论文, *2004a, pp*. 106.

[彭波,2004b] 彭波, "搜索引擎中倒排文件索引的缓存机制," Pku_Cs_Net_Tr2004005, 2004b. (http://162.105.80.88/cra*zysite/home/report*/upload/1820180625.doc).

[天网,2004] 北京大学天网中英文搜索引擎. http://e.pku.edu.cn.

[谢正茂,2003] 谢正茂, "Web 数据模型以及获取、存储方法研究," 北京大学,硕士论文, 2003.

[闫宏飞,2002] 闫宏飞, "可扩展 Web 信息搜集系统的设计、实现与应用初探," 北京大学,博士论文, 2002, pp. 116.

[咎红英,2004] 咎红英, "基于实体属性的中文网页检索研究," 北京大学,博士论文, 2004, pp. 143.

[张志刚,2004] 张志刚, *"基于网页的信息系统的*一种预处理过程," 北京大学,硕士论文, 2004.

[赵江华,2002] 赵江华, "天网高性能分布式检索系统的设计与实现," 北京大学,硕士论文, 2002, pp. 48.

附录. 术语

B:

半结构化数据（semi-structured data），和普通纯文本相比，Web 上的网页数据具有一定的结构性，表现在其中的 HTML 标注上；但和具有严格理*论模型的关系数据库的数据相比，这种 HTM*L 标注带来的结构性又要弱很多，因此人们称 Web 上的数据为半结构化数据，这是 Web 上数据的基本特点。

布尔模型（boolean model），在信息检索领域，不同的场合有不同的含义。当我们讨论用户提交查询的时候，指的是为形成最终查询结果集合，由一个查询的各个成分对查询结果子集之间所要求的一种运算关系；而在讨论文档比较的向量空间模型中，布尔模型指的是构成一个文档向量的各个分量只取 1 和 0 两个值，分别代表对应特征项的出现与否。

C:

查全率（recall），判断检索系统质量的一种度量，表示系统所检索到的与查询相关的文档数占与查询相关的总文档数的百分比。

查询（query），用户使用信息系统提供的输入语言和规则对自己信息需求的一种表达。常用的输入语言包含关键词规范和一些布尔连接符。

查准率（precision），判断检索系统质量的一种度量。系统所检索到的与查询相关的文档数占检索出的所有文档数的百分比，即反映检索结果“正确性”的度量。

词典（vocabulary），文档（或文档集合）中所有不同词项的集合。

词频（term frequency，tf或TF），TF(i,j)是指一个词项t_i在一篇文档d_j中出现的次数。

D:

代理（agent），或称代理程序，在应用中，接收到用户的请求后，能代表用户完

成任务并返回结果，但不受用户监督的程序、进程或部分系统。在中，代理程序用于从存档或信息库中搜索与用户所给主题词相关的内容，所以有时又称为智能代理（Intelligent Agent）。

倒排文件（inverted file），组织和索引文件，以便于检索的一种方法。在该方法中，一个关键字的集合是基础，该集合中每一个关键字对应一串记录项，其中每一项包含一个文档编号、该关键字在该文档中出现的情况等信息。

倒置文档频率（inversed document frequency，idf或IDF），通常IDF(t_i) 取值为 $\log(N/n_i)$，其中N是所有文档的总数，n_i是在N个文档中包含词项t_i的文档数。

动态网页（dynamic Web page），需要通过提交查询信息才能获取的网页。

动态摘要（dynamic abstract），做文档摘要的一种方法。对于搜索引擎来说，就是在响应用户查询的时候，根据查询词在文档中出现的位置，提取出查询词周围相关的文字并返回给用户。由于一篇文档会含有不同的查询词，因此动态摘要技术可能把同一个文档形成不同的摘要文字。

G:

共有词汇假设（shared bag of words），信息检索技术的一个最基本假设，即认为文档的含义可以由它所包含的关键词的集合来表达。

H:

HTML（hypertext markup language），超文本标记语言，是 Web 的关键技术之一，它为 ASCII 格式的超文本文档提供了一种标准表述方式。

缓存（cache），在计算机科学领域经常出现的一个概念，其基本含义是利用局部性原理实现的一种匹配两种不同速度的中间机制。它可以出现在 CPU 和 RAM 中间，也可以出现在应用系统的 I/O 操作与磁盘之间。在搜索引擎中，为缓解查询要求的高速度和磁盘访问低速的矛盾，常会在内存中设计各种缓存，包括查询缓存、点击缓存，以及倒排表缓存等。

J:

静态网页（static Web page），不需要通过提交查询信息即可获得的页面。

镜像网页（mirror Web page），网页的内容完全相同，未加任何修改。

局部性原则（locality principle），是程序行为的一种性质。它包括：时间局部性和空间局部性。前者指的是，如果某数据刚才被访问，则它很可能在近期内还要被访问；后者指的是，如果某数据刚才被访问，则和它在位置上相邻的数据很可能将被访问。

拒绝服务攻击（denial of service，DoS），是一种攻击行动，使网站服务器充斥大量要求回复的信息，消耗网络带宽或系统资源，导致网络或系统不胜负荷以至于瘫痪而停止提供正常的网络服务。

L:

链接分析（link analysis）： Web 上的网页及其相互之间的链接可以看成一个巨大的有向图，链接分析指的是利用网页之间的链接信息来评判其重要性（或者相关性）的技术。常用的链接信息包含网页的出度、入度，锚文本内容等；常用的链接分析算法有：PageRank，HITS，SALSA，PHITS，Bayesian 等。

M:

MD5（message digest 5），报文摘要，用于报文编码的一种算法。MD5 算法在 RFC1321 中定义，其基本功能是将一个任意长的报文变换为一个 128 位的摘要，两个不同的报文对应的摘要相同的概率极小，两个摘要之间的相近程度和对应两个报文的相近程度没有任何关系。

锚文本（anchor text），HTML 文本中的链接描述信息，向读者提示该链接所指向网页的性质或特征。例如，在一篇网页中书写有新闻频道，则“新闻频道”就是链接 href= “http://www.cctv.com”在本网页中的锚文本。

目录型网页（hub page），该网页提供很多指向其它权威型网页的超链接。是与权威型网页相对应的。

Q:

齐普夫定律（Zipf's law），由美国学者 G.K.齐普夫于本世纪 40 年代提出的词频分布定律。它可以表述为：如果把一篇较长文档中每个词出现的频次统计起来，按照高频词在前、低频词在后的递减顺序排列，并用自然数给这些词编上等级序号，即频次最高的词等级为 1，频次次之的等级为 2，……。若用 f 表示频次，r 表示等级序号，则有 f＝C/r（C 为常数）。

切词（word segmentation），或称分词，主要在中文信息处理中使用，即把一句话分成一个词的序列。如，“网络与分布式系统实验室”，分词为“网络 与 分布式 系统 实验室”。

全文检索（full text retrieval），文本信息检索的一种方法（或者说是一种精细程度），其特点是不仅文档中出现的每一个词都可以被检索出来，而且每一个词的每一次出现也可以被检索出来。

权威型网页（authority page），网页内容通常有一个特定的主题，并且被许多其它网页链接，是与目录型网页相对应的一个概念。

S:

散列表（Hash Table），或称哈希表，是一种数据结构，它便于快速的信息查找。散列表生成时为表中的每项数据分配一个随机索引代码。这种索引代码的随机性使得数据的分布比较均匀，从而可能大大节省后续查找的时间。

数字图书馆（digital libarary），一个数字信息对象收藏、组织和表现这些对象的方法以及将这些对象提供给用户的相关的信息技术。它包括支持用户进行定位、检索和获取这些信息对象的服务。

搜索引擎（search engine，SE），Web 上的一种应用软件系统，它以一定的策略在 Web 上搜集和发现信息，对信息进行处理和组织后，为用户提供 Web 信息查询服务。

索引词载体信息（index term carrier），HTML 的标签信息标识了文档中索引词的字体和大小写等信息。

T:

停用词（stop word），指文档中出现的连词，介词，冠词等并无太大意义词。例如在英文中常用的停用词有 the，a, it 等；在中文中常见的有“是”,“的”,“地”等。

吞吐量（throughput），或称吞吐率，是指在单位时间里系统完成的总任务量。对于搜索引擎来说，就是指系统在单位时间（秒）里可以服务的最大用户查询数量。

U:

URL（uniform resource locator），用来定位互联网上信息资源的一种协议（或者说描述规范），网页的定位通常就是以形如“http://host/path/file.html”的 URL 来描述的，而 FTP 资源则用形如“ftp://host/path/file”的 URL 来描述。

URL 域名深度，网页对应的 url 中域名部分包含的子域个数。

URL 目录深度，网页对应的 url 中除去域名部分的目录层次，即 url = schema://host/localpath 中的 localpath 部分。如 url 为 http://www.pku.edu.cn，则目录深度为 0；如果是 http://www.pku.edu.cn/cs，则目录深度为 1。

W:

网页出度（page outdegree），针对一个网页，该网页指向其他网页的超链接数目。

网页净化（noise reduction），识别并去除网页噪音的过程；即去除网页内与该网页主题内容无关的信息，如广告、版权信息等。

网页爬取器（gatherer），指网页搜集子系统中根据 url 完成一篇网页抓取的进程或者线程，通常一个搜集子系统上会同时启动多个 gatherer 并行工作。

网页入度（page indegree），针对一个网页，整个网络中指向该网页的超链接数目。

网页搜集子系统（crawler system），尤指在搜索引擎系统中，利用 HTML 文档之间的链接关系，在 Web 上依照网页之间的超链关系一个个抓取网页的程序。鉴于其在 Web 上沿超链“爬行”的工作方式，这种程序有时也称为“蜘蛛”（spider）。Crawler，spider，robot，bot 一般都指的是相同的事物。

文档对象模型（document object model，DOM），DOM 将一个 XML 文档转换成一个对象集合，然后可以任意处理该对象模型。这一机制也称为“随机访问”协议，因为可以在任何时间访问数据的任何一部分，然后修改、删除或插入新数据。

文档自动分类（automatic text categorization，ATC），用计算机程序来确定指定文档和预先定义文档类别之间的隶属关系。

X:

先进先出（first in first out，FIFO），是一种页面替换算法，选择最先装入主存储器的那一页调出，或者说是把驻留在主存时间最长的那一页调出。

相关排序（relevance ranking），指信息检索系统返回结果的排序，其中条目的顺序反映了系统确定的结果和查询的相关程度。

向量空间模型（vector space model，VSM），按照共有词汇假设，一组文档有一个总词语集合 Σ，一篇文档可以用一个向量表示，其元素是对应词语在该文档中出现情况的一种定量描述，一组文档就可以看成是一个向量空间中的若干元素，于是可以应用向量空间中距离的概念来考察两篇文档之间的相似程度等。

响应时间（response time），在计算机系统中，从提交请求（或询问）到开始看到回答之间所经历的时间。对于搜索引擎来说，就是用户提交查询到他看到返回结果之间所经历的时间。在搜索引擎的具体实践中，由于这个时间和动态变化的网络状态有关，常常用检索系统为完成一个查询所消耗的响应时间来近似。

消重（replicas or near-replicas detection），清除所搜集网页集合中的镜像或转载网页的过程。

协议（protocol），为实现通信而制定的能够协调各功能单元操作的一组规则。

信息检索（information retrieval，IR），将信息按一定的方式组织和存储起来，并根据用户的需要找出有关信息的过程。

信息检索模型（IR model），依照用户查询，对文档集合进行相关排序的一组前提假设和算法。IR模型可形式地表示为一个四元组< D, Q, F, $R(q_i,d_j)$ >，其中D是一

个文档集合，Q是一个查询集合，F是一个对文档和查询建模的框架，$R(q_i,d_j)$ 是一个排序函数，它给查询q_i和文档 d_j 之间的相关度赋予一个排序值。常用的信息检索模型有：集合论模型、代数模型、概率模型等。

Y:

用户查询日志（user query log），是在用户提交查询请求时由系统自动记录的相关信息，它包括用户查询时提交的关键词、提交时间、用户 IP 地址、页号（通常查询结果分页显示，每页显示 10 个查询结果，用户首次查询页号为 1，用户翻页时页号即为用户选择的结果页面号）和是否在缓存中命中等信息。

用户点击日志（user hit log），是用户浏览查询结果并点击页面时由系统自动记录的相关信息，它通常包括用户点击页面的时间、点击页面的 URL、用户 IP 地址、点击页面的序号（该页面在查询结果中的位置）、该点击对应的查询词等信息。

元数据(meta data)，描述某种类型资源（或对象）的属性、并对这种资源进行定位和管理、同时有助于数据检索的数据。

元搜索引擎（meta search engine），又称集成型搜索引擎，它将用户的查询发送给多个独立的搜索引擎，收集它们产生的结果，然后按照一定的算法进行选择和重新排序以形成一个最终结果返回给用户。

Z:

中文信息处理（Chinese information processing），用计算机对汉语的音、形、义等语言文字进行信息的加工和操作，包括对字、词、短语、句、篇章的输入、输出、识别、转换、压缩、存储、检索、分析、理解和生成等各方面的处理技术。

主题搜集（topic-specific/focused crawling），即面向主题的信息搜集系统，其主要任务是利用有限的网络带宽、存储容量和较少的时间，抓取尽可能多的与主题内容密切相关的网页。

转载网页（near-replicas Web page），内容基本相同但可能有一些额外的编辑信息等。虽然网页做了部分改动，但其主题内容未变；即去除网页的噪声（如广告、版权等信息）外，其它正文内容相同。转载网页也称为近似镜像网页。

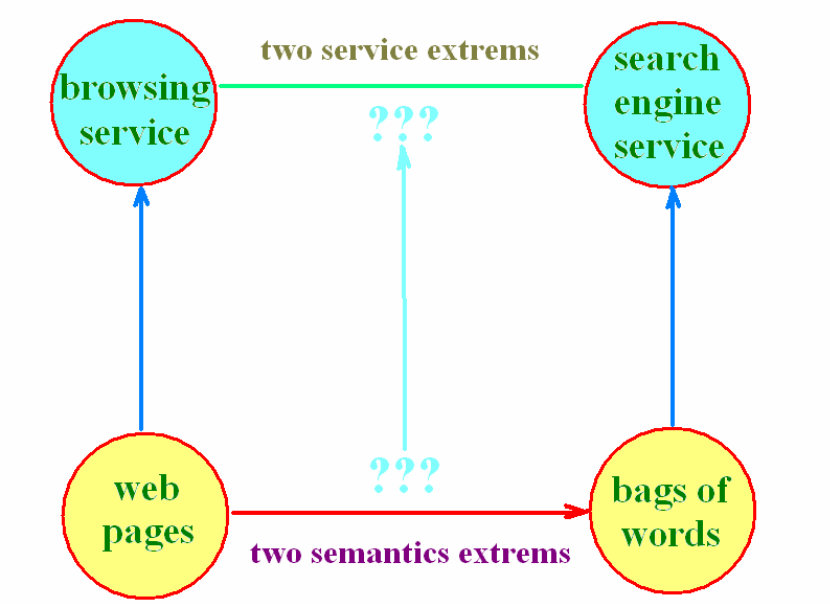
最低频使用（least frequently used，LFU），缓存内容维护的一种数据替换策略，当缓存满，且有新的数据要进来时，它总是淘汰现有数据中在过去使用频率最低的数据。数据替换的粒度可以根据应用场合确定。

最近最少使用（least recently used，LRU)，缓存内容维护的一种数据替换策略，当缓存满，且有新的数据要进来时，它总是淘汰现有数据中在过去最长时间未被使用过的数据。

后记

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，这从近几年来 WWW 学术年会上总有大量相关论文发表可见一斑。Google 在短短的几年里在全世界范围的成功，在展示搜索引擎技术巨大潜力的同时，也似乎给人们带来了一种“赢家通吃”的认识；不少人认为互联网上只需要为数不多的几个搜索引擎。在现在常见的应用模式下，在大规模通用搜索引擎的意义上，我们也同意这样的认识。但如果有了创新的应用模式，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。

为此，我们不妨暂时离开身在其中的互联网。当我们越走越远的时候，万千气象逐渐淡开去，但留下了 E-mail，Web 浏览，搜索引擎。这里不谈 E-mail。我们能否有一种将 Web 浏览和搜索引擎放在一起考察的视角，从而有可能解释为什么搜索引擎能如此成功，引导我们思考还可能有什么样的东西也可能取得很大成功？



上图就是这样一种视角。由一篇篇相互链接的网页构成了一张 Web，它使得人们能够在上面浏览。而每一篇网页被变成了一个词的集合以后，将它们集中起

来，就构成了搜索引擎工作的数据基础。如果说，原始的网页自然具有最丰富的语义，那么当它变成了一个词的集合以后，语义就“丧失殆尽”了。因此我们可以讲搜索引擎作用的对象和 Web 浏览作用的对象是两个语义极端。而引导 Web 浏览的是地址（URL），引导搜索引擎工作的是内容（query），又使得我们可以认为它们是两个服务的极端。进一步地，如果认为地址是没什么语义的，而查询词是有一定语义的，那么又能看到没什么语义的对象和丰富语义的对象相结合形成了有意义的信息服务（Web 浏览），而相对有语义的对象在相对语义很弱的对象上也可以形成有意义的信息服务（搜索引擎）。从这个角度看问题，就使我们很容易去想是否能够发展出某种折衷的服务来：其基础数据的语义在网页和词语集合之间，服务引导的语义在地址和自由词语之间。

这样，我们就有了一个很大的发挥想象力的空间。

同时我们还看到，搜索引擎的发展和应用也为人们针对海量网络信息的相关研究提供了一个生动的背景。如同在第一章引论中提到的，我们在不断发展“天网”搜索引擎的过程中，注意到互联网上的信息除了无比丰富外，还具有如下特征：

- 信息发布者和享用者的数量差不多是同一个数量级的；这和诸如报纸、书籍、电视等传统媒体大大不同；
- 相当一部分内容的产生是比较随意的，没有经过专业的编辑整理，发布人员也不需要负什么责任；
- 互联网上的信息在时间上具有流逝性，网页的出现和消失是一个动态的过程；例如我们现在对下面的问题难以有一个清楚地回答：1995 年中国互联网上都有过些什么内容？
- 网页和其他互联网信息与生俱来就是数字化、网络化的，因此十分便于获取、处理和二次增值开发。

对这些特征的认识促使我们产生了一个构想，即全面搜集中国互联网上的信息，对它们进行高效的组织，并且提供超越搜索引擎范畴的信息服务。我们简称这个构想为“北大燕穹”，它激发了一系列相关的科研和系统建设的实践[1]。

我们认为，随着互联网深入到社会的各个角落，网络信息内容的全面性、流逝性和随意性使其成为一类珍贵的社会资源，对互联网信息的研究有可能成为研究我们社会的一个独特的途径。因此，从 2001 年底开始，利用天网技术，我们开始全面地搜集和保存中国互联网上的网页信息资源，建设形成了“中国互联网信息博物馆”，http://www.infomall.cn。 这是一个动态成长的大规模互联网网页信息历史档案，现收藏有从 2001 年以来的约 7 亿中国互联网上的网页，容量 8TB，

[1] 国外一些学者也有类似的工作。最早的，有Internet Archive；另一个很有名的是IBM的WebFountain。

而且以每天几十万到几百万左右网页的速度增加；还收藏有从 2001 年以来的约 4000 万条搜索引擎访问日志，而且以每天 20 万左右的速度增加。“北大燕穹”就是围绕“中国互联网信息博物馆”的建设、发展和应用的活动和成果的总称。

在这样一个海量信息资源库之上，人们不但能够方便地浏览中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，管理学等领域的学者都有可能从中发现自己感兴趣的宝藏，发展出新的研究方法，取得创新的成果。本书第十三章第二节的内容就是一个实例。另有一些相关的工作见参考文献 [Li and Zhu,2003]。

“中国互联网信息博物馆”向各相关领域的研究人员开放资源，欢迎大家一起来开采这样一个巨大的矿藏。

时间将证明，“中国互联网信息博物馆”将在研究中国互联网信息建设的历程及其对我们社会的影响方面发挥重要的作用。我们也设想未来会有一本著述，展示在“中国互联网信息博物馆”上的多学科研究成果。