徐凡的blog

有压力就有更多的动力，无压力将有更多的创新。微博：http://t.sina.com.cn/chief1985

个人资料

chief1985

公告：7月2日CSDN新版博客上线暂时停止服务

解剖Google搜索原理 The Anatomy of a Large-Scale Hypertextual Web Search Engine

分类： 搜索引擎 2007-04-30 16:08 阅读(1828) 评论(0) 收藏 举报

解剖Google搜索原理

By master 发表于 2006-12-13 11:06:00 [出自: 碧森尤信]

【摘要】 这篇文章中，我们介绍了google，它是一个大型的搜索引擎（of a large-scale search engine）的原型，搜索引擎在超文本中应用广泛。Google的设计能够高效地抓网页并建立索引，它的查询结果比其它现有系统都高明。这个原型的全文和超连接的数据库至少包含24000000个网页。我们可以从http://google.stanford.edu/ 下载。设计搜索引擎是一项富有挑战性的工作。搜索引擎为上亿个网页建立索引，其中包含大量迥然不同的词汇。而且每天要回答成千上万个查询。在网络中，尽管大型搜索引擎非常重要，但是学术界却很少研究它。此外由于技术的快速发展和网页的大量增加，现在建立一个搜索引擎和三年前完全不同。本文详细介绍了我们的大型搜索引擎，据我们所知，在公开发表的论文中，这是第一篇描述地如此详细。除了把传统数据搜索技术应用到如此大量级网页中所遇到的问题，还有许多新的技术挑战，包括应用超文本中的附加信息改进搜索结果。本文将解决这个问题，描述如何运用超文本中的附加信息，建立一个大型实用系统。任何人都可以在网上随意发布信息，如何有效地处理这些无组织的超文本集合，也是本文要关注的问题。

《The Anatomy of a Large-Scale Hypertextual Web Search Engine》

1 绪论

Web 给信息检索带来了新的挑战。Web上的信息量快速增长，同时不断有毫无经验的新用户来体验Web这门艺术。人们喜欢用超级链接来网上冲浪，通常都以象 Yahoo这样重要的网页或搜索引擎开始。大家认为List(目录)有效地包含了大家感兴趣的主题，但是它具有主观性，建立和维护的代价高，升级慢，不能包括所有深奥的主题。基于关键词的自动搜索引擎通常返回太多的低质量的匹配。使问题更遭的是，一些广告为了赢得人们的关注想方设法误导自动搜索引擎。

我们建立了一个大型搜索引擎解决了现有系统中的很多问题。应用超文本结构，大大提高了查询质量。**我们的系统命名为google，取名自googol的通俗拼法，即10的100次方**，这和我们的目标建立一个大型搜索引擎不谋而合。

1.1网络搜索引擎—升级换代（scaling up）：

1994-2000 搜索引擎技术不得不快速升级（scale dramatically）跟上成倍增长的web数量。1994年，第一个Web搜索引擎，World Wide Web Worm(WWWW)可以检索到110，000个网页和Web的文件。到1994年11月，顶级的搜索引擎声称可以检索到2‘000’000 （WebCrawler）至100‘000’000个网络文件（来自 Search Engine Watch）。可以预见到2000年，可检索到的网页将超过1‘000’000‘000。同时，搜索引擎的访问量也会以惊人的速度增长。在1997年的三四月份，World Wide Web Worm 平均每天收到1500个查询。

在1997年11月，Altavista 声称它每天要处理大约20’000’000个查询。随着网络用户的增长，到2000年，自动搜索引擎每天将处理上亿个查询。我们系统的设计目标要解决许多问题，包括质量和可升级性，引入升级搜索引擎技术（scaling search engine technology），把它升级到如此大量的数据上。

1.2 Google：

跟上 Web的步伐（Scaling with the Web）建立一个能够和当今web规模相适应的搜索引擎会面临许多挑战。抓网页技术必须足够快，才能跟上网页变化的速度（keep them up to date）。存储索引和文档的空间必须足够大。索引系统必须能够有效地处理上千亿的数据。处理查询必须快，达到每秒能处理成百上千个查询（hundreds to thousands per second.）。随着Web的不断增长，这些任务变得越来越艰巨。然而硬件的执行效率和成本也在快速增长，可以部分抵消这些困难。

还有几个值得注意的因素，如磁盘的寻道时间（disk seek time），操作系统的效率（operating system robustness）。在设计Google的过程中，我们既考虑了Web的增长速度，又考虑了技术的更新。Google的设计能够很好的升级处理海量数据集。它能够有效地利用存储空间来存储索引。优化的数据结构能够快速有效地存取（参考4.2节）。进一步，我们希望，相对于所抓取的文本文件和HTML网页的数量而言，存储和建立索引的代价尽可能的小（参考附录B）。对于象Google这样的集中式系统，采取这些措施得到了令人满意的系统可升级性（scaling properties）。

1. 3设计目标

1.3.1 提高搜索质量。我们的主要目标是提高Web搜索引擎的质量。

1994 年，有人认为建立全搜索索引（a complete search index）可以使查找任何数据都变得容易。根据Best of the Web 1994 -- Navigators，“最好的导航服务可以使在Web上搜索任何信息都很容易（当时所有的数据都可以被登录）”。然而1997年的Web就迥然不同。近来搜索引擎的用户已经证实索引的完整性不是评价搜索质量的唯一标准。用户感兴趣的搜索结果往往湮没在“垃圾结果Junk result”中。实际上，到1997年11月为止，四大商业搜索引擎中只有一个能够找到它自己（搜索自己名字时返回的前十个结果中有它自己）。导致这一问题的主要原因是文档的索引数目增加了好几个数量级，但是用户能够看的文档数却没有增加。用户仍然只希望看前面几十个搜

网页制作js(41)
虚拟机(28)
计算机图形学(16)
计算机图形学(0)
设计模式(0)
设计模式(0)
语言学习 (0)
语言学习 (2)
语音技术 (1)
语音技术 (2)
驱动(28)

文章存档

2011年06月(1)
2011年05月(2)
2011年03月(8)
2011年02月(2)
2011年01月(2)

展开

阅读排行

中国军队一个军多少人，一个师多少人，依次... (17869)
android makefile(and... (11903)
介绍几个流行的ASP编辑器 http:/... (9006)
DreamWeaver CS3 官方简体... (8864)
几款web版整合（QQ、msn、icq、... (8305)
适用于XP的AMD双核优化补丁集 (8301)
低级格式化硬盘工具(支持移动硬盘)htt... (7328)
★另人无比怀念的90年代动画片★ ‖最终... (6847)
让记录成为一种习惯 (6656)
Chrome编译 (6631)

评论排行

让记录成为一种习惯 (48)
Google Wave的邀请名额(已经送... (19)

索结果。因此，**当集合增大时，我们就需要工具使结果精确**（在返回的前几十个结果中，有关文档的数量）。由于是从成千上万个有点相关的文档中选出几十个，实际上，相关的概念就是指最好的文档。高精确非常重要，甚至以响应（系统能够返回的有关文档的总数）为代价。令人高兴的是利用超文本链接提供的信息有助于改进搜索和其它应用 。尤其是链接结构和链接文本，为相关性的判断和高质量的过滤提供了大量的信息。Google既利用了链接结构又用到了anchor文本（见2.1和2.2节）。

1.3.2 搜索引擎的学术研究随着时间的流逝，除了发展迅速，Web越来越商业化。

1993 年，只有1.5%的Web服务是来自.com域名。到1997年，超过了60%。同时，搜索引擎从学术领域走进商业。到现在大多数搜索引擎被公司所有，很少技公开术细节。这就导致搜索引擎技术很大程度上仍然是暗箱操作，并倾向做广告（见附录A）。Google的主要目标是推动学术领域在此方面的发展，和对它的了解。另一个设计目标是给大家一个实用的系统。应用对我们来说非常重要，因为现代网络系统中存在大量的有用数据（us because we think some of the most interesting research will involve leveraging the vast amount of usage data that is available from modern web systems）。例如，每天有几千万个研究。然而，得到这些数据却非常困难，主要因为它们没有商业价值。我们最后的设计目标是建立一个体系结构能够支持新的关于海量Web数据的研究。为了支持新研究，**Google以压缩的形式保存了实际所抓到的文档**。设计Google 的目标之一就是要建立一个环境使其他研究者能够很快进入这个领域，处理海量Web数据，得到满意的结果，而通过其它方法却很难得到结果。系统在短时间内被建立起来，已经有几篇论文用到了Google建的数据库，更多的在起步中。我们的另一个目标是建立一个宇宙空间实验室似的环境，在这里研究者甚至学生都可以对我们的海量Web数据设计或做一些实验。

2. 系统特点

Google搜索引擎有两个重要特点，有助于得到高精度的搜索结果。
第一点，应用Web的链接结构计算每个网页的Rank值，称为PageRank，将在98页详细描述它。
第二点，Google利用超链接改进搜索结果。

2.1 PageRank:给网页排序：

Web 的引用（链接）图是重要的资源，却被当今的搜索引擎很大程度上忽视了。我们建立了一个包含518000000个超链接的图，它是一个具有重要意义的样本。这些图能够快速地计算网页的PageRank值，它是一个客观的标准，较好的符合人们心目中对一个网页重要程度的评价，建立的基础是通过引用判断重要性。因此在web中，PageRank能够优化关键词查询的结果。对于大多数的主题，在网页标题查询中用PageRank优化简单文本匹配，我们得到了令人惊叹的结果（从google.stanford.edu可以得到演示）。对于Google主系统中的全文搜索，PageRank也帮了不少忙。

2.1.1计算PageRank

文献检索中的引用理论用到Web中，引用网页的链接数，一定程度上反映了该网页的重要性和质量。PageRank发展了这种思想，网页间的链接是不平等的。

PageRank定义如下: 我们假设T1...Tn指向网页A（例如，被引用）。参数d是制动因子，使结果在0，1之间。通常d等于0.85。在下一节将详细介绍d。C（A）定义为网页A指向其它网页的链接数，

const的修饰规则（const便捷记忆... (13)
android java代码的启动：ap... (11)
webkit在vs2008中编译 (11)
《唐伯虎点秋香》随想----插件 (11)
C++函数指针 (9)
vs编译的一些技巧（持续更新中） (8)
gcc for Windows 开发环境... (7)
关于千千静听 (7)

最新评论

mark
回复 chief1985：一定...
用到的开源项目好多啊。。。
全部改为FE了,Unicode...
的确，我写了三四年的博客，发现...

好的网站

Voiz_经典语句
MSDN每日追踪 (RSS)
何宗键@BLOG (RSS)
雨林的专栏 (RSS)
windows核心编程 (RSS)
中国X黑客小组
孟岩
大学课程在线
web前端技术交流博客
研学论坛
I3S汉语词法分析系统ICTCLAS
80x86汇编小站
中国建站
北京大学搜索引擎与互联网信息挖掘组
java学习笔记
车东
像素精灵
编程王
亲亲我的土豆
Logo在线制作

网页A的PageRank值由下式给出： **PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))**

注意PageRank的形式，分布到各个网页中，因此所有网页的PageRank和是1。 PageRank或PR（A）可以用简单的迭代算法计算，相应规格化Web链接矩阵的主特征向量。中等规模的网站计算26‘000’000网页的 PageRank值要花费几小时。还有一些技术细节超出了本文论述的范围。

2.1.2 直觉判断 PageRank被看作用户行为的模型。

我们假设网上冲浪是随机的，不断点击链接，从不返回，最终烦了，另外随机选一个网页重新开始冲浪。随机访问一个网页的可能性就是它的PageRank值。制动因子d是随机访问一个网页烦了的可能性，随机另选一个网页。对单个网页或一组网页，一个重要的变量加入到制动因子d中。这允许个人可以故意地误导系统，以得到较高的PageRank值。我们还有其它的PageRank算法，见98页。

另外的直觉判断是一个网页有很多网页指向它，或者一些PageRank值高的网页指向它，则这个网页很重要。直觉地，在Web中，一个网页被很多网页引用，那么这个网页值得一看。一个网页被象Yahoo这样重要的主页引用即使一次，也值得一看。如果一个网页的质量不高，或者是死链接，象Yahoo这样的主页不会链向它。PageRank处理了这两方面因素，并通过网络链接递归地传递。

2.2链接描述文字（Anchor Text）：

我们的搜索引擎对链接文本进行了特殊的处理。**大多数搜索引擎把链接文字和它所链向的网页（the page that the link is on）联系起来**。另外，把它和链接所指向的网页联系起来。这有几点好处。
第一，通常链接描述文字比网页本身更精确地描述该网页。
第二，链接描述文字可能链向的文档不能被文本搜索引擎检索到，例如图像，程序和数据库。有可能使返回的网页不能被抓到。注意哪些抓不到的网页将会带来一些问题。在返回给用户前检测不了它们的有效性。这种情况搜索引擎可能返回一个根本不存在的网页，但是有超级链接指向它。然而这种结果可以被挑出来的，所以此类的问题很少发生。链接描述文字是对被链向网页的宣传，这个思想被用在World Wide Web Worm 中，主要因为它有助于搜索非文本信息，能够用少量的已下载文档扩大搜索范围。我们大量应用链接描述文字，因为它有助于提高搜索结果的质量。有效地利用链接描述文字技术上存在一些困难，因为必须处理大量的数据。现在我们能抓到24000000个网页，已经检索到259000000多个链接描述文字。

2.3其它特点除了PageRank和应用链接描述文字外，Google还有一些其它特点。

第一,所有hit都有位置信息，所以它可以在搜索中广泛应用邻近性（proximity）。
第二，Google跟踪一些可视化外表细节，例如字号。黑体大号字比其它文字更重要。
第三，知识库存储了原始的全文html网页。

3 有关工作

Web 检索研究的历史简短。World Wide Web Worm（）是最早的搜索引擎之一。后来出现了一些用于学术研究的搜索引擎，现在它们中的大多数被上市公司拥有。与Web的增长和搜索引擎的重要性相比，有关当今搜索引擎

仿yahoo
VCASMO - 一个类似ppt的展示网站
网页链接快照
Linux那些事儿
我的另外一个博客

漂亮的网站

Begin to play with Ext Framework
古典flash

技术的优秀论文相当少。根据Michael Mauldin（Lycos Inc的首席科学家），“各种各样的服务（包括Lycos）非常关注这些数据库的细节。”虽然在搜索引擎的某些特点上做了大量工作。具有代表性的工作有，对现有商业搜索引擎的结果进行传递，或建立小型的个性化的搜索引擎。最后有关信息检索系统的研究很多，尤其在有组织机构集合（well controlled collections）方面。在下面两节，我们将讨论在信息检索系统中的哪些领域需要改进以便更好的工作在Web上。

3.1信息检索信息检索系统诞生在几年前，并发展迅速。

然而大多数信息检索系统研究的对象是小规模的单一的有组织结构的集合，例如科学论文集，或相关主题的新闻故事。实际上，信息检索的主要基准，（the Text Retrieval Conference），用小规模的、有组织结构的集合作为它们的基准。

大型文集基准只有20GB，相比之下，我们抓到的24000000个网页占147GB。在TREC上工作良好的系统，在Web上却不一定产生好的结果。例如，标准向量空间模型企图返回和查询请求最相近的文档，把查询请求和文档都看作由出现在它们中的词汇组成的向量。在Web环境下，这种策略常常返回非常短的文档，这些文档往往是查询词再加几个字。例如，查询“Bill Clinton”，返回的网页只包含“Bill Clinton Sucks”，这是我们从一个主要搜索引擎中看到的。网络上有些争议，用户应该更准确地表达他们想查询什么，在他们的查询请求中用更多的词。我们强烈反对这种观点。如果用户提出象“Bill Clinton”这样的查询请求，应该得到理想的查询结果，因为这个主题有许多高质量的信息。象所给的例子，我们认为信息检索标准需要发展，以便有效地处理Web数据。

3.2 有组织结构的集合（Well Controlled Collections）与Web的不同点

Web是完全无组织的异构的大量文档的集合。Web 中的文档无论内在信息还是隐含信息都存在大量的异构性。例如，文档内部就用了不同的语言（既有人类语言又有程序），词汇（email地址，链接，邮政编码，电话号码，产品号），类型（文本，HTML，PDF，图像，声音），有些甚至是机器创建的文件（log文件，或数据库的输出）。可以从文档中推断出来，但并不包含在文档中的信息称为隐含信息。**隐含信息包括来源的信誉，更新频率，质量，访问量和引用**。不但隐含信息的可能来源各种各样，而且被检测的信息也大不相同，相差可达好几个数量级。例如，一个重要主页的使用量，象Yahoo 每天浏览数达到上百万次，于此相比无名的历史文章可能十年才被访问一次。很明显，搜索引擎对这两类信息的处理是不同的。 Web与有组织结构集合之间的另外一个明显区别是，**事实上，向Web上传信息没有任何限制**。灵活利用这点可以发布任何对搜索引擎影响重大的信息，使路由阻塞，加上为牟利故意操纵搜索引擎，这些已经成为一个严重的问题。这些问题还没有被传统的封闭的信息检索系统所提出来。它关心的是元数据的努力，这在Web搜索引擎中却不适用，因为网页中的任何文本都不会向用户声称企图操纵搜索引擎。甚至有些公司为牟利专门操纵搜索引擎。

4 系统分析（System Anatomy）

首先，我们提供高水平的有关体系结构的讨论。然后，详细描述重要的数据结构。最后，主要应用：抓网页，索引，搜索将被严格地检查。 Figure 1. High Level Google Architecture

4.1Google体系结构概述

这一节，我们将看看整个系统是如何工作的（give a high level），见图1。本节不讨论应用和数据结构，在后几节中讨论。为了效率大部分**Google是用c或c++实现的，既可以在Solaris也可以在Linux上运行。**

Google 系统中，抓网页（下载网页）是由几个分布式crawlers完成的。一个URL服务器负责向crawlers提供URL列表。抓来的网页交给存储服务器 storeserver。然后，由存储服务器压缩网页并把它们存到知识库repository中。每个网页都有一个ID，称作docID，当新URL从网页中分析出时，就被分配一个docID。由索引器和排序器负责建立索引index function。索引器从知识库中读取文档，对其解压缩和分析。每个文档被转换成一组词的出现情况，称作**命中hits**。Hits 纪录了词，词在文档中的位置，最接近的字号，大小写。索引器把这些hits分配到一组桶barrel中，产生经过部分排序后的索引。索引器的另一个重要功能是分析网页中所有的链接，将有关的重要信息存在链接描述anchors文件中。该文件包含了足够的信息，可以用来判断每个链接链出链入节点的信息，和链接文本。 URL分解器resolver阅读链接描述anchors文件，并把相对URL转换成绝对URL，再转换成docID。为链接描述文本编制索引，并与它所指向的docID关联起来。同时建立由docID对组成的链接数据库。用于计算所有文档的PageRank值。用docID分类后的barrels，送给排序器sorter，再根据wordID进行分类，建立反向索引inverted index。这个操作要恰到好处，以便几乎不需要暂存空间。排序器还给出docID和偏移量列表，建立反向索引。一个叫DumpLexicon的程序把这个列表和由索引器产生的字典结合在一起，建立一个新的字典，供搜索器使用。这个搜索器就是利用一个Web服务器，使用由DumpLexicon所生成的字典，利用上述反向索引以及页面等级PageRank来回答用户的提问。

4.2 主要数据结构

经过优化的Google数据结构，能够用较小的代价抓取大量文档，建立索引和查询。虽然近几年CPU和输入输出速率迅速提高。磁盘寻道仍然需要10ms。任何时候Google系统的设计都尽可能地避免磁盘寻道。这对数据结构的设计影响很大。

4.2.1 大文件

大文件BigFiles是指虚拟文件生成的多文件系统，用长度是64位的整型数据寻址。多文件系统之间的空间分配是自动完成的。BigFiles包也处理已分配和未分配文件描述符。由于操纵系统不能满足我们的需要，BigFiles也支持基本的压缩选项。

4.2.2 知识库

Repository Data Structure **知识库**包含每个网页的全部HTML。**每个网页用zlib（见RFC1950）压缩**。压缩技术的选择既要考虑速度又要考虑压缩率。我们选择zlib的速度而不是压缩率很高的bzip。知识库用bzip的压缩率接近4：1。而用zlib的压缩率是3：1。文档一个挨着一个的存储在知识库中，前缀是docID，长度，URL，见图2。访问知识库不需要其它的数据结构。这有助于数据一致性和升级。用其它数据结构重构系统，我们只需要修改知识库和crawler错误列表文件。

4.2.3 文件索引

文件索引保存了有关文档的一些信息。索引以docID的顺序排列，定宽ISAM（Index sequential access mode）。每条记录包括当前文件状态，一个指向知识库的指针，文件校验和，各种统计表。如果一个文档已经被抓到，指针指向docinfo文件，该文件的宽度可变，包含了URL和标题。否则指针指向包含这个URL的URL列表。这种设计考虑到简洁的数据结构，以及在查询中只需要一个磁盘寻道时间就能够访问一条记录。还有一个文件用于把URL转换成docID。它是URL校验和与相应docID的列表，按校验和排序。要想知道某个URL的docID，需要计算URL的校验和，然后在校验和文件中执行二进制查找，找到它的docID。通过对这个文件进行合并，可以把一批URL转换成对应的docID。URL分析器用这项技术把URL转换成docID。这种成批更新的模式是至关重要的，否则每个链接都需要一次查询，假如用一块磁盘，322‘000’000个链接的数据集合将花费一个多月的时间。

4.2.4 词典

词典有几种不同的形式。和以前系统的重要不同是，词典对内存的要求可以在合理的价格内。现在实现的系统，一台256M内存的机器就可以把词典装入到内存中。现在的词典包含14000000词汇（虽然一些很少用的词汇没有加入到词典中）。它执行分两部分—**词汇表（用null分隔的连续串）和指针的哈希表。**不同的函数，词汇表有一些辅助信息，这超出了本文论述的范围。

4.2.5 hit list

hit list是一篇文档中所出现的词的列表，包括位置，字号，大小写。Hit list占很大空间，用在正向和反向索引中。因此，它的表示形式越有效越好。我们考虑了几种方案来编码位置，字号，大小写—简单编码（3个整型数），紧凑编码（支持优化分配比特位），哈夫曼编码。Hit的详细信息见图3。我们的紧凑编码每个hit用2字节。有两种类型hit，**特殊hit和普通hit**。特殊hit包含URL，标题，链接描述文字，meta tag。普通hit包含其它每件事。它包括大小写特征位，字号，12比特用于描述词在文档中的位置（所有超过4095的位置标记为4096）。字号采用相对于文档的其它部分的相对大小表示，占3比特(实际只用7个值，因为111标志是特殊hit)。特殊hit由大小写特征位，字号位为7表示它是特殊hit，用4比特表示特殊hit的类型，8比特表示位置。对于anchor hit八比特位置位分出4比特用来表示在anchor中的位置，4比特用于表明anchor出现的哈希表hash of the docID。短语查询是有限的，对某些词没有足够多的anchor。我们希望更新anchor hit的存储方式，以便解决地址位和docIDhash域位数不足的问题。

因为搜索时，你不会因为文档的字号比别的文档大而特殊对待它，所以采用相对字号。 hit表的长度存储在hit前。为节省空间hit表长度，在正向索引中和wordID结合在一起，在反向索引中和docID结合存储。这就限制它相应地只占8到5比特（用些技巧，可以从wordID中借8bit）如果大于这些比特所能表示的长度，用溢出码填充，其后两字节是真正的长度。 Figure 3. Forward and Reverse Indexes and the Lexicon

4.2.6 正向索引

实际上，正向索引已经部分排序。它被存在一定数量的barrel中（我们用64个barrels）。每个barrel装着一定范围的wordID。如果一篇文档中的词落到某个barrel，它的docID将被记录到这个barrel中，紧跟着那些词（文档中所有的词汇，还是落入该barrel中的词汇）对应的hitlist。这种模式需要稍多些的存储空间，因为一个docID被用多次，但是它节省了桶数和时间，最后排序器进行索引时降低编码的复杂度。更进一步的措施是，我们不是存储

docID本身，而是存储相对于该桶最小的docID的差。用这种方法，未排序的barrel的docID只需24 位，省下8位记录hitlist长。

4.2.7 反向索引

除了反向索引由sorter加工处理之外，它和正向索引包含相同的桶。对每个有效的docID，字典包含一个指向该词所在桶的指针。它指向由docID和它的相应hitlist组成的doclish，这个doclist代表了所有包含该词的文档。 doclist中docID的顺序是一个重要的问题。最简单的解决办法是用doclish排序。这种方法合并多个词时很快。另一个可选方案是用文档中该词出现的次数排序。这种方法回答单词查询，所用时间微不足道。当多词查询时几乎是从头开始。并且当用其它Rank算法改进索引时，非常困难。我们综合了这两种方法，建立两组反向索引barrel，一组barrels的hitlist只包含标题和anchor hit，另一组barrel包含全部的hitlist。我们首先查第一组索引桶，看有没有匹配的项，然后查较大的那组桶。

4.3 抓网页运行

网络爬行机器人是一项具有挑战性的任务。执行的性能和可靠性甚至更重要，还有一些社会焦点。网络爬行是一项非常薄弱的应用，它需要成百上千的web服务器和各种域名服务器的参与，这些服务器不是我们系统所能控制的。为了覆盖几十亿的网页，Google拥有快速的分布式网络爬行系统。一个URL服务器给若干个网络爬行机器人（我们采用3个）提供URL列表。**URL服务器和网络爬行机器人都是用Python实现的**。每个网络爬行机器人可以同时打开300个链接。抓取网页必须足够快。最快时，用4个网络爬行机器人每秒可以爬行100个网页。速率达每秒600K。执行的重点是找 DNS。每个网络爬行机器人有它自己的DNS cache，所以它不必每个网页都查DNS。每一百个连接都有几种不同的状态：**查DNS，连接主机，发送请求，接收回答**。这些因素使网络爬行机器人成为系统比较复杂的部分。**它用异步IO处理事件，若干请求队列从一个网站到另一个网站不停的抓取网页**。运行一个链接到500多万台服务器的网页爬行机器人，产生1千多万登陆口，导致了大量的Email和电话。因为网民众多，总有些人不知道网络爬行机器人是何物，这是他们看到的第一个网络爬行机器人。几乎每天我们都会收到这样的Email“哦，你从我们的网站看了太多的网页，你想干什么？”还有一些人不知道网络搜索机器人避免协议（the robots exclusion protocol），以为他们的网页上写着“版权所有，勿被索引”的字样就会被保护不被索引，不必说，这样的话很难被web crawler理解。因为数据量如此之大，还会遇到一些意想不到的事情。例如，我们的系统曾经企图抓一个在线游戏，结果抓到了游戏中的大量垃圾信息。解决这个问题很简单。但是我们下载了几千万网页后才发现了这个问题。因为网页和服务器的种类繁多，实际上不在大部分Internet上运行它就测试一个网页爬行机器人是不可能。总是有几百个隐含的问题发生在整个web的一个网页上，导致网络爬行机器人崩溃，或者更糟，导致不可预测的不正确的行为。能够访问大部分Internet的系统必须精力充沛并精心测试过。由于象crawler这样大型复杂的系统总是产生这样那样的问题，因此花费一些资源读这些 Email，当问题发生时解决它，是有必要的。

4.4 Web索引分析

任何运行在整个Web上的分析器必须能够处理可能包含错误的大型集合。范围从HTML标记到标记之间几K字节的0，非ASCII字符，几百层HTML标记的嵌套，各种各样令人难以想象的错误。为了获得最大的速度，我们没有采用YACC产生上下文无关文法CFG分析器，而是采用灵活的方式产生词汇分析器，它自己配有堆栈。分析器的改进大大提高了运行速度，它的精力如此充沛完成了大量工作。把文档装入barrel建立索引—分析完一篇文档，之后把该文档装入barrel中，用内存中的hash表—字典，每个词汇被转换成一个 wordID。当hash表字典中加入新的项时，笨拙地存入文件。一旦词汇被转换成wordID，它们在当前文档的出现就转换成hitlist，被写进正向barrel。索引阶段并行的主要困难是字典需要共享。

我们采用的方法是，基本字典中有140万个固定词汇，不在基本字典中的词汇写入日志，而不是共享字典。这种方法多个索引器可以并行工作，最后一个索引器只需处理一个较小的额外词汇日志。排序—为了建立反向索引，排序器读取每个正向barrel，以 wordID排序，建立只有标题anchor hi t的反向索引barrel和全文反向索引barrel。这个过程一次只处理一个barrel，所以只需要少量暂存空间。排序阶段也是并行的，我们简单地同时运行尽可能多的排序器，不同的排序器处理不同的桶。由于barrel不适合装入主存，排序器进一步依据wordID和docID把它分成若干篮子，以便适合装入主存。然后排序器把每个篮子装入主存进行排序，并把它的内容写回到短反向barrel和全文反向barrel。

4.5搜索

搜索的目标是提供有效的高质量的搜索结果。多数大型商业搜索引擎好像在效率方面花费了很大力气。因此我们的研究以搜索质量为重点，相信我们的解决方案也可以用到那些商业系统中。

Google查询评价过程见图4。

1. 分析查询。
2. 把词汇转换成wordID。
3. 在短barrel中查找每个词汇doclist的开头。
4. 扫描doclist直到找到一篇匹配所有关键词的文档
5. 计算该文档的rank
6. 如果我们在短barrel，并且在所有doclist的末尾，开始从全文barrel的doclist的开头查找每个词，goto 第四步
7. 如果不在任何doclist的结尾，返回第四步。
8. 根据rank排序匹配文档，返回前k个。图4 Google查询评价在有限的响应时间内，一旦找到一定数量的匹配文档，搜索引擎自动执行步骤8。这意味着，返回的结果是子优化的。我们现在研究其它方法来解决这个问题。过去根据PageRank排序hit，看来能够改进这种状况。

4.5.1 Ranking系统

Google比典型搜索引擎保存了更多的web信息。**每个hitlist包括位置，字号，大小写。另外，我们还考虑了链接描述文字**。Rank 综合所有这些信息是困难的。ranking函数设计依据是没有某个因素对rank影响重大。首先，考虑最简单的情况—单个词查询。为了单个词查询中一个文档的rank，Goole在文档的hitlist中查找该词。Google认为每个hit是几种不同类型（标题，链接描述文字anchor，URL，普通大字号文本，普通小字号文本，……）之一，**每种有它自己的类型权重。类型权重建立了一个类型索引向量**。Google计算hitlist中每种hit的数量。然后每个hit数转换成count-weight。Count-weight开始随hit数线性增加，很快逐渐停止，以至于hit数与此不相关。我们计算count-weight向量和type-weight向量的标量积作为文档的IR值。最后IR值结合PageRank作为文档的最后rank。对于多词查询，更复杂些。现在，多词hitlist必须同时扫描，以便关键词出现在同一文档中的权重比分别出现时高。相邻词的hit一起匹配。对每个匹配hit 的集合计算相邻度。**相邻度基于hit在文档中的距离**，分成10个不同的bin值，范围从短语匹配到根本不相关。不仅计算每类hit数，而且要计算每种类型的相邻度，每个类型相似度对，**有一个类型相邻度权type-prox-weight**。Count转换成count-weight，计算count-weight、 type-prox-weight的标量积作为IR值。应用某种debug mode所有这些数和矩阵与查询结果一起显示出来。这些显示有助于改进rank系统。

4.5.2 反馈

rank 函数有很多参数象type-weight和type-prox-weight。指明这些参数的正确值有点黑色艺术black art。为此，我们的搜索引擎有一个用户反馈机制。值得信任的用户可以随意地评价返回的结果。保存反馈。然后，当修改rank函数时，对比以前搜索的 rank，我们可以看到修改带来的的影响。虽然不是十全十美，但是它给出了一些思路，当rank函数改变时对搜索结果的影响。

5 执行和结果

搜索结果的质量是搜索引擎最重要的度量标准。完全用户评价体系超出了本文的论述范围，对于大多数搜索，我们的经验说明Google的搜索结果比那些主要的商业搜索引擎好。作为一个应用PageRank，链接描述文字，相邻度的例子，图4给出了Google搜索bill Clinton的结果。它说明了Google的一些特点。服务器对结果进行聚类。这对过滤结果集合相当有帮助。这个查询，相当一部分结果来自 whitehouse.gov域，这正是我们所需要的。现在大多数商业搜索引擎不会返回任何来自whitehouse.gov的结果，这是相当不对的。注意第一个搜索结果没有标题。因为它不是被抓到的。Google是根据链接描述文字决定它是一个好的查询结果。同样地，第五个结果是一个Email地址，当然是不可能抓到的。也是链接描述文字的结果。所有这些结果质量都很高，最后检查没有死链接。因为它们中的大部分PageRank值较高。PageRank 百分比用红色线条表示。没有结果只含Bill没有Clinton或只含Clinton没有Bill。因为词出现的相近性非常重要。当然搜索引擎质量的真实测试包含广泛的用户学习或结果分析，此处篇幅有限，请读者自己去体验Google，http://google.stanford.edu/。

5.1存储需求

除了搜索质量，Google的设计可以随着Web规模的增大而有效地增大成本。一方面有效地利用存储空间。表1列出了一些统计数字的明细表和Google存储的需求。由于压缩技术的应用知识库只需53GB的存储空间。是所有要存储数据的三分之一。按当今磁盘价格，知识库相对于有用的数据来说比较便宜。搜索引擎需要的所有数据的存储空间大约55GB。大多数查询请求只需要短反向索引。文件索引应用先进的编码和压缩技术，一个高质量的搜索引擎可以运行在7GB的新PC。

5.2 系统执行

搜索引擎抓网页和建立索引的效率非常重要。Google 的主要操作是抓网页，索引，排序。很难测试抓全部网页需要多少时间，因为磁盘满了，域名服务器崩溃，或者其它问题导致系统停止。总的来说，大约需要9天时间下载26000000网页（包括错误）。然而，一旦系统运行顺利，速度非常快，下载最后11000000网页只需要63小时，平均每天4000000网页，每秒48.5个网页。**索引器和网络爬行机器人同步运行**。索引器比网络爬行机器人快。因为我们花费了大量时间优化索引器，使它不是瓶颈。这些优化包括批量更新文档索引，本地磁盘数据结构的安排。索引器每秒处理54个网页。排序器完全并行，用4台机器，排序的整个过程大概需要24小时。

5.3搜索执行改进

搜索执行不是我们研究的重点。当前版本的Google可以在1到10秒间回答查询请求。时间大部分花费在NFS磁盘IO上（由于磁盘普遍比机器慢）。进一步说，Google没有做任何优化，例如查询缓冲区，常用词汇子索引，和其它常用的优化技术。我们倾向于通过分布式，硬件，软件，和算法的改进来提高Google的速度。我们的目标是每秒能处理几百个请求。表2有几个现在版本Google响应查询时间的例子。它们说明IO缓冲区对再次搜索速度的影响。

6 结论

Google设计成可伸缩的搜索引擎。主要目标是在快速发展的World Wide Web上提供高质量的搜索结果。Google应用了一些技术改进搜索质量包括PageRank，链接描述文字，相邻信息。进一步说，**Google是一个收集网页，建立索引，执行搜索请求的完整的体系结构。**

6.1 未来的工作

大型Web搜索引擎是个复杂的系统，还有很多事情要做。我们直接的目标是提高搜索效率，覆盖大约100000000个网页。一些简单的改进提高了效率包括请求缓冲区，巧妙地分配磁盘空间，子索引。另一个需要研究的领域是更新。我们必须有一个巧妙的算法来决定哪些旧网页需要重新抓取，哪些新网页需要被抓取。这个目标已经由实现了。受需求驱动，用代理cache创建搜索数据库是一个有前途的研究领域。我们计划加一些简单的已经被商业搜索引擎支持的特征，例如布尔算术符号，否定，填充。然而另外一些应用刚刚开始探索，例如相关反馈，聚类（Google现在支持简单的基于主机名的聚类）。我们还计划支持用户上下文（象用户地址），结果摘要。我们正在扩大链接结构和链接文本的应用。简单的实验证明，通过增加用户主页的权重或书签，PageRank可以个性化。对于链接文本，我们正在试验用链接周围的文本加入到链接文本。Web搜索引擎提供了丰富的研究课题。如此之多以至于我们不能在此一一列举，因此在不久的将来，我们希望所做的工作不止本节提到的。

6.2 高质量搜索

当今Web搜索引擎用户所面临的最大问题是搜索结果的质量。结果常常是好笑的，并且超出用户的眼界，他们常常灰心丧气浪费了宝贵的时间。例如，一个最流行的商业搜索引擎搜索“Bill Clillton”的结果是the Bill Clinton Joke of the Day: April 14, 1997。Google的设计目标是随着Web的快速发展提供高质量的搜索结果，容易找到信息。为此，Google大量应用超文本信息包括链接结构和链接文本。Google还用到了相邻性和字号信息。评价搜索引擎是困难的，我们主观地发现Google的搜索质量比当今商业搜索引擎高。通过PageRank分析链接结构使Google能够评价网页的质量。用链接文本描述链接所指向的网页有助于搜索引擎返回相关的结果（某种程度上提高了质量）。最后，利用相邻性信息大大提高了很多搜索的相关性。

6.3可升级的体系结构

除了搜索质量，Google设计成可升级的。空间和时间必须高效，处理整个Web时固定的几个因素非常重要。实现Google系统，CPU、访存、内存容量、磁盘寻道时间、磁盘吞吐量、磁盘容量、网络IO都是瓶颈。在一些操作中，已经改进的Google克服了一些瓶颈。Google的主要数据结构能够有效利用存储空间。进一步，网页爬行，索引，排序已经足够建立大部分web索引，共24000000个网页，用时不到一星期。我们希望能在一个月内建立100000000网页的索引。

6.4 研究工具

Google不仅是高质量的搜索引擎，它还是研究工具。Google搜集的数据已经用在许多其它论文中，提交给学术会议和许多其它方式。最近的研究，例如，提出了Web查询的局限性，不需要网络就可以回答。这说明Google不仅是重要的研究工具，而且必不可少，应用广泛。我们希望Google是全世界研究者的资源，带动搜索引擎技术的更新换代。

7、致谢

Scott Hassan and Alan Steremberg评价了Google的改进。他们的才智无可替代，作者由衷地感谢他们。感谢Hector Garcia-Molina, Rajeev Motwani, Jeff Ullman, and Terry Winograd和全部WebBase开发组的支持和富有深刻见解的讨论。最后感谢IBM，Intel，Sun和投资者的慷慨支持，为我们提供设备。这里所描述的研究是Stanford综合数字图书馆计划的一部分，由国家科学自然基金支持，合作协议号IRI-9411306。DARPA，NASA，Interva研究，Stanford数字图书馆计划的工业合作伙伴也为这项合作协议提供了资金。

参考文献。

- Best of the Web 1994 -- Navigators http://botw.org/1994/awards/navigators.html
- Bill Clinton Joke of the Day: April 14, 1997. http://www.io.com/~cjburke/clinton/970414.html.
- Bzip2 Homepage http://www.muraroa.demon.co.uk/
- Google Search Engine http://google.stanford.edu/
- Harvest http://harvest.transarc.com/

- Mauldin, Michael L. Lycos Design Choices in an Internet Search Service, IEEE Expert Interview http://www.computer.org/pubs/expert/1997/trends/x1008/mauldin.htm
- The Effect of Cellular Phone Use Upon Driver Attention http://www.webfirst.com/aaa/text/cell/cell0toc.htm
- Search Engine Watch http://www.searchenginewatch.com/
- RFC 1950 (zlib) ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html
- Robots Exclusion Protocol: http://info.webcrawler.com/mak/projects/robots/exclusion.htm
- Web Growth Summary: http://www.mit.edu/people/mkgray/net/web-growth-summary.html
- Yahoo! http://www.yahoo.com/

- [Abiteboul 97] Serge Abiteboul and Victor Vianu, *Queries and Computation on the Web*. Proceedings of the International Conference on Database Theory. Delphi, Greece 1997.
- [Bagdikian 97] Ben H. Bagdikian. *The Media Monopoly*. 5th Edition. Publisher: Beacon, ISBN: 0807061557
- [Chakrabarti 98] S.Chakrabarti, B.Dom, D.Gibson, J.Kleinberg, P. Raghavan and S. Rajagopalan. *Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text.* Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.
- [Cho 98] Junghoo Cho, Hector Garcia-Molina, Lawrence Page. *Efficient Crawling Through URL Ordering.* Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.
- [Gravano 94] Luis Gravano, Hector Garcia-Molina, and A. Tomasic. *The Effectiveness of GlOSS for the Text-Database Discovery Problem.* Proc. of the 1994 ACM SIGMOD International Conference On Management Of Data, 1994.
- [Kleinberg 98] Jon Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, Proc. ACM-SIAM Symposium on Discrete Algorithms, 1998.
- [Marchiori 97] Massimo Marchiori. *The Quest for Correct Information on the Web: Hyper Search Engines.* The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.
- [McBryan 94] Oliver A. McBryan. GENVL and *WWWW: Tools for Taming the Web. First International Conference on the World Wide Web.* CERN, Geneva (Switzerland), May 25-26-27 1994. http://www.cs.colorado.edu/home/mcbryan/mypapers/www94.ps
- [Page 98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Manuscript in progress. http://google.stanford.edu/~backrub/pageranksub.ps
- [Pinkerton 94] Brian Pinkerton, *Finding What People Want: Experiences with the WebCrawler.* The Second International WWW Conference Chicago, USA, October 17-20, 1994. http://info.webcrawler.com/bp/WWW94.html
- [Spertus 97] Ellen Spertus. *ParaSite: Mining Structural Information on the Web.* The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.

- [TREC 96] *Proceedings of the fifth Text REtrieval Conference (TREC-5).* Gaithersburg, Maryland, November 20-22, 1996. Publisher: Department of Commerce, National Institute of Standards and Technology. Editors: D. K. Harman and E. M. Voorhees. Full text at: http://trec.nist.gov/
- [Witten 94] Ian H Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* New York: Van Nostrand Reinhold, 1994.
- [Weiss 96] Ron Weiss, Bienvenido Velez, Mark A. Sheldon, Chanathip Manprempre, Peter Szilagyi, Andrzej Duda, and David K. Gifford. *HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering.* Proceedings of the 7th ACM Conference on Hypertext. New York, 1996.

图片附录：

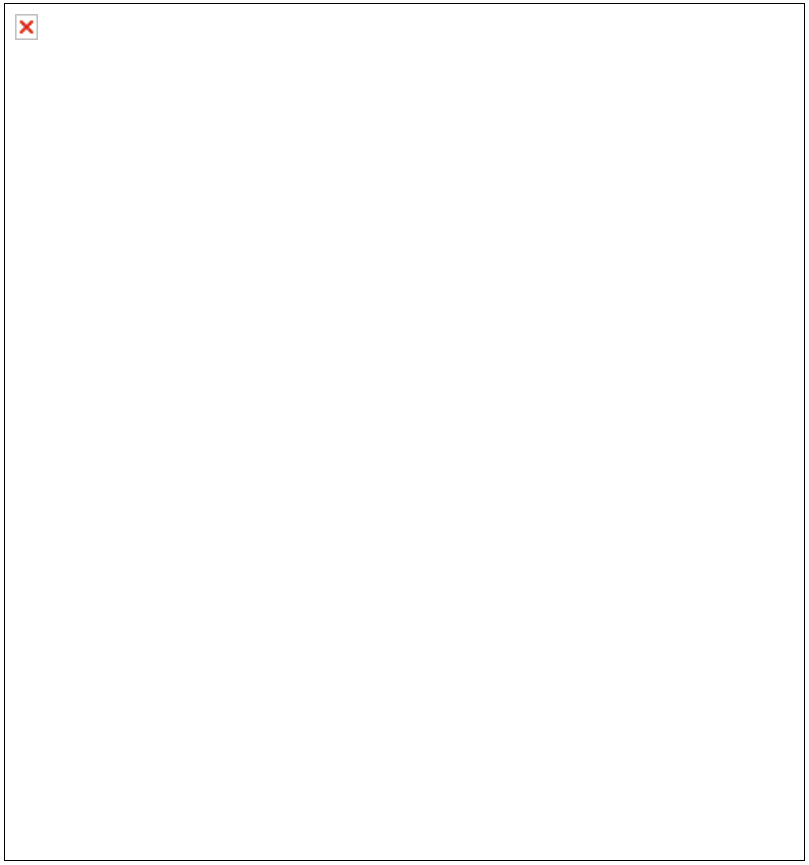

图1 Google系统的工作流程图

(注：原图来自Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual. Web Search Engine, 1998.http://www-db.stanford.edu/%7Ebackrub/Google.html)

①Google使用高速的分布式爬行器(Crawler)系统中的漫游遍历器(Googlebot)定时地遍历网页，将遍历到的网页送到存储服务器(Store Server)中。

② 存储服务器使用zlib格式压缩软件将这些网页进行无损压缩处理后存入数据库Repository中。Repository获得了每个网页的完全Html 代码后，对其压缩后的网页及URL进行分析，记录下网页长度、URL、URL长度和网页内容，并赋予每个网页一个文档号(docID)，以便当系统出现故障的时候，可以及时完整地进行网页的数据恢复。
③索引器(Indexer)从Repository中读取数据，以后做以下四步工作：
④(a) 将读取的数据解压缩后进行分析，它将网页中每个有意义的词进行统计后，转化为关键词(wordID)的若干索引项(Hits)，生成索引项列表，该列表包括关键词、关键词的位置、关键词的大小和大小写状态等。索引项列表被存入到数据桶(Barrels)中，并生成以文档号(docID)部分排序的顺排档索引。

索引项根据其重要程度分为两种：当索引项中的关键词出现在URL、标题、锚文本(Anchor Text)和标签中时，表示该索引项比较重要，称为特殊索引项(Fancy Hits)；其余情况则称为普通索引项(Plain Hits)。在系统中每个Hit用两个字节(byte)存储结构表示：特殊索引项用1位(bit)表示大小写，用二进制代码111(占3位)表示是特殊索引项，其余12位有4位表示特殊索引项的类型(即hit是出现在URL、标题、链接结点还是标签中)，剩下8位表示hit在网页中的具体位置；普通索引项是用1位表示大小写，3位表示字体大小，其余12位表示在网页中的具体位置。
顺排档索引和Hit的存储结构如图3所示。

图3 顺排档索引和Hit的存储结构

值得注意的是，当特殊索引项来自Anchor Text时，特殊索引项用来表示位置的信息（8位）将分为两部分：4位表示Anchor Text出现的具体位置，另4位则用来与表示Anchor Text所链接网页的docID相连接，这个docID是由URL Resolver经过转化存入顺排档索引的。
(b)索引器除了对网页中有意义的词进行分析外，还分析网页的所有超文本链接，将其Anchor Text、URL指向等关键信息存入到Anchor文档库中。
(c)索引器生成一个索引词表(Lexicon)，它包括两个部分：关键词的列表和指针列表，用于倒排档文档相连接(如图3所示)。
(d) 索引器还将分析过的网页编排成一个与Repository相连接的文档索引(Document Index)，并记录下网页的URL和标题，以便可以准确查找出在Repository中存储的原网页内容。而且把没有分析的网页传给URL Server，以便在下一次工作流程中进行索引分析。
⑤URL分析器（URL Resolver）读取Anchor文档中的信息，然后做⑥中的工作。
⑥(a) 将其锚文本(Anchor Text)所指向的URL转换成网页的docID；(b)将该docID与原网页的docID形成“链接对”，存入Link数据库中；(c)将 Anchor Text指向的网页的docID与顺排档特殊索引项Anchor Hits相连接。
⑦数据库Link记录了网页的链接关系，用来计算网页的PageRank值。
⑧文档索引(Document Index)把没有进行索引分析的网页传递给URL Server，URL Server则向Crawler提供待遍历的URL，这样，这些未被索引的网页在下一次工作流程中将被索引分析。
⑨排序器（Sorter）对数据桶(Barrels)的顺排档索引重新进行排序，生成以关键词(wordID)为索引的倒排档索引。
倒排档索引结构如图4所示：

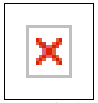

图4 倒排档索引结构

⑩ 将生成的倒排档索引与先前由索引器产生的索引词表(Lexicon)相连接产生一个新的索引词表供搜索器(Searcher)使用。搜索器的功能是由网页服务器实现的，根据新产生的索引词表结合上述的文档索引(Document Index)和Link数据库计算的网页PageRank值来匹配检索。

在执行检索时，Google通常遵循以下步骤（以下所指的是单个检索词的情况）：

(1)将检索词转化成相应的wordID；

(2)利用Lexicon，检索出包含该wordID的网页的docID；

(3)根据与Lexicon相连的倒排档索引，分析各网页中的相关索引项的情况，计算各网页和检索词的匹配程度，必要时调用顺排档索引；

(4)根据各网页的匹配程度，结合根据Link产生的相应网页的PageRank情况，对检索结果进行排序；

(5)调用Document Index中的docID及其相应的URL，将排序结果生成检索结果的最终列表，提供给检索用户。

用户检索包含多个检索词的情况与以上单个检索词的情况类似：先做单个检索词的检索，然后根据检索式中检索符号的要求进行必要的布尔操作或其他操作。

- 上一篇：基于既定词表的自适应汉语分词技术研究
- 下一篇：asp中的字符串函数示例

上一篇：ASCII码全接触http://www.elook.net.cn/handbook/this_content/ascii.htm

下一篇：基于既定词表的自适应汉语分词技术研究

http://www.pipcn.com/blog/user1/master/archives/2006/1099.shtml

查看评论

暂无评论

发表评论

用 户 名： greenstorm2010

评论内容：

提交

以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场