



这就是搜索引擎

核心技术详解

改变全世界人们生活方式的“信息之门”



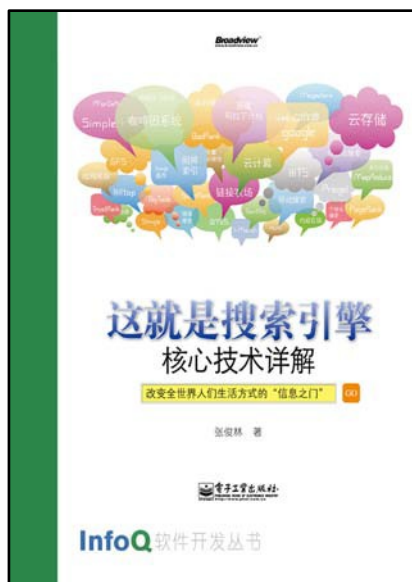
张俊林 著

国祭二对援丘;

InfoQ 软件开发丛书

免费在线版本

(非印刷免费在线版)



了解本书更多信息请登录[豆瓣图书相关页面](#)

InfoQ 中文站出品



本书由 InfoQ 中文站免费发放，如果您从其他渠道获取本书，请注册 InfoQ 中文站以支持作者和出版商，并免费下载更多 InfoQ 软件开发系列图书。

本迷你书主页为

<http://www.infoq.com/cn/minibooks/this-is-search-engine>

前言

互联网产品形形色色，有产品导向的，有营销导向的，也有技术导向的，但是以技术见长的互联网产品比例相对小些。搜索引擎是目前互联网产品中最具技术含量的产品，如果不是唯一，至少也是其中之一。

经过十几年的发展，搜索引擎已经成为互联网的重要入口之一，Twitter 联合创始人埃文·威廉姆斯提出了“域名已死论”：好记的域名不再重要，因为人们会通过搜索进入网站。搜索引擎排名对于中小网站流量来说至关重要。了解搜索引擎简单界面背后的技术原理其实对很多人都很重要。

为什么会有这本书

最初写本搜索引擎技术书籍的想法萌生于两年前，当时的场景是要给团队成员做搜索技术培训，但是我找遍了相关图书，却没有发现非常合适的搜索技术入门书籍。当时市面上的书籍，要么是信息检索理论方面的专著，理论性太强不易懂，而且真正讲搜索引擎技术的章节并不多；要么是 Lucene 代码分析这种过于实务的书籍，像搜索引擎这种充满算法的应用，直接分析开源系统代码并不是非常高效的学习方式。所以当时萌生了写一本既通俗易懂，适合没有相关技术背景的人员阅读，又比较全面，且融入最新技术的搜索引擎书籍，但是真正动手开始写是一年前的事情了。

写书前我给自己定了几个目标。首先内容要全面，即全面覆盖搜索引擎相关技术的主要方面，不仅要包含倒排索引、检索模型和爬虫等常见内容，也要详细讲解链接分析、网页反作弊、用户搜索意图分析、云存储及网页去重，甚至是搜索引擎缓存等内容，这些都是一个完整搜索引擎的有机组成部分，但是详述其原理的书籍并不多，我希望能够尽可能全面些。

第二个目标是通俗易懂。我希望没有任何相关技术背景的人也能够通过阅读这本书有所收获，最好是不懂技术的同学也能大致看懂。这个目标看似简单，其实很不容易达到，我也不敢说这本书已经达到了此目的，但是确实已经尽自己所能去做了。至于具体的措施，则包含以下三个方面。

- 一个是尽可能减少数学公式的出现次数，除非不得已不罗列公式。虽说数学公式具简洁之美，但是大多数人其实对于数学符号是有恐惧和逃避心理的，多年前我也有类似心理，所以但凡可能，尽量不用数学公式。

- 一个是尽可能多举例子，尤其是一些比较难理解的地方，需要例子来增进理解。
- 还有一个是多画图。就我个人的经验来说，尽管算法或者技术是很抽象的，但是如果深入理解其原理，去繁就简，那么一定可以把算法转换成形象的图片。如果不能在头脑中形成算法直观的图形表示，说明并未透彻了解其原理。这是我判断自己是否深入理解算法的一个私有标准。鉴于此，本书中在讲解算法的地方，大量采用了算法原理图，全书包含了超过 300 幅算法原理讲解图，相信这对于读者深入理解算法会有很大的帮助。

第三个目标是强调新现象新技术，比如 Google 的咖啡因系统及 Megastore 等云存储系统、Pregel 云图计算模型、暗网爬取技术、Web 2.0 网页作弊、机器学习排序、情境搜索、社会化搜索等在相关章节都有讲解。

第四个目标是强调原理，不纠缠技术细节。对于新手一个易犯的毛病是喜欢抠细节，只见树木不见森林，搞明白了一个公式却不了解其背后的基本思想和出发点。我接触的技术人员很多，十有七八会有这个特点。这里有个“道术孰优”的问题，何为“道”？何为“术”？举个例子的话，《孙子兵法》是道，而《三十六计》则为术。“道”所述，是宏观的、原理性的、长久不变的基本原理，而“术”则是在遵循基本原理基础上的具体手段和措施，具有易变性。技术也是如此，算法本身的细节是“术”，算法体现的基本思想则是“道”，知“道”而学“术”，两者虽不可偏废，但是若要选择优先级的话，无疑我会选择先“道”后“术”。

以上四点是写书前定下的目标，现在书写完了，也许很多地方不能达到最初的期望，但是尽了力就好。写书的过程很辛苦，起码比我原先想象的要辛苦，因为工作繁忙，所以只能每天早早起床，再加上周末及节假日的时间来完成。也许书中还存在这样那样的缺点，但是我可以无愧地说写这本书是有诚意的。

这本书是写给谁的

如果您是下列人员之一，那么本书就是写给您的。

1. 对搜索引擎核心算法有兴趣的技术人员

- 搜索引擎的整体框架是怎样的？包含哪些核心技术？
- 网络爬虫的基本架构是什么？常见的爬取策略是什么？什么是暗网爬取？如何构建分布式爬虫？百度的阿拉丁计划是什么？
- 什么是倒排索引？如何对倒排索引进行数据压缩？

- 搜索引擎如何对搜索结果排序？
- 什么是向量空间模型？什么是概率模型？什么是 BM25 模型？什么是机器学习排序？它们之间有何异同？
- PageRank 和 HITS 算法是什么关系？有何异同？SALSA 算法是什么？Hilltop 算法又是什么？各种链接分析算法之间是什么关系？
- 如何识别搜索用户的真实搜索意图？用户搜索目的可以分为几类？什么是点击图？什么是查询会话？相关搜索是如何做到的？
- 为什么要对网页进行去重处理？如何对网页进行去重？哪种算法效果较好？
- 搜索引擎缓存有几级结构？核心策略是什么？
- 什么是情境搜索？什么是社会化搜索？什么是实时搜索？
- 搜索引擎有哪些发展趋势？

如果您对其中三个以上的问题感兴趣，那么这本书就是为您而写的。

2. 对云计算与云存储有兴趣的技术人员

- 什么是 CAP 原理？什么是 ACID 原理？它们之间有什么异同？
- Google 的整套云计算框架包含哪些技术？Hadoop 系列和 Google 的云计算框架是什么关系？
- Google 的三驾马车 GFS、BigTable、MapReduce 各自代表什么含义？是什么关系？
- Google 的咖啡因系统的基本原理是什么？
- Google 的 Pregel 计算模型和 MapReduce 计算模型有什么区别？
- Google 的 Megastore 云存储系统和 BigTable 是什么关系？
- 亚马逊公司的 Dynamo 系统是什么？
- 雅虎公司的 PNUTS 系统是什么？
- Facebook 公司的 Haystack 存储系统适合应用在什么场合？

如果您对上述问题感兴趣，相信可以从书中找到答案。

3. 从事搜索引擎优化的网络营销人员及中小网站站长

- 搜索引擎的反作弊策略是怎样的？如何进行优化避免被认为是作弊？
- 搜索引擎如何对搜索结果排序？链接分析和内容排序是什么关系？
- 什么是内容农场？什么是链接农场？它们是什么关系？
- 什么是 Web 2.0 作弊？有哪些常见手法？
- 什么是 SpamRank？什么是 TrustRank？什么又是 BadRank？它们是什么关系？
- 咖啡因系统对网页排名有何影响？

最近有一批电子商务网站针对搜索引擎优化，结果被 Google 认为是黑帽 SEO 而导致搜索排名降权，如何避免这种情况？从事相关行业的营销人员和网站站长应该深入了解搜索引擎反作弊的基本策略和方法，甚至是网页排名算法等搜索引擎核心技术。SEO 技术说到底其实很简单，虽然不断发生变化，但是很多原理性的策略总是相似的，万变不离其宗，深入了解搜索引擎相关技术原理将形成您的行业竞争优势。

4. 作者自己

我的记性不太好，往往一段时间内了解的技术，时隔几年后就模糊了，所以这本书也是为我自己写的，以作为技术备查手册。沈利也参与了本书的部分编写工作。

致谢

感谢博文视点的付睿编辑，没有她也就没有本书的面世，付编辑在阅稿过程中提出的细致入微的改进点对我帮助甚大。

感谢翻开此书的读者，如果您在阅读本书的过程中发现一些纰漏或者错误，或者是意见建议，希望您能够不吝让我知晓，我会守在 mailjunlin@gmail.com 这个信箱旁敬候您的来信，如果给我微博发信也非常欢迎 <http://www.weibo.com/malefactor>。

特别感谢我的妻子，在近一年的写作过程中，我几乎把能用的所有业余时间都投入在本书的写作上，她为了不让我分心，承担了所有的家务，不介意没有时间陪她，这本书的诞生且算是送她的一个礼物吧。

于我而言，这本书的写作是一个辛苦而欣喜的过程，有如旅人远行，涉水跋山之际抬头远眺，总能看到曾经忽略的旖旎丽景，若您在阅读本书的过程中也能有此体会，那就是我的荣幸了。

张俊林 2011 年 6 月

QCon

全球软件开发大会

International Software Development Conference



www.qconferences.com

— 皇 回 固 @

中文 | 英文 | 日文 | 葡文 |


目录

前言.....	1
为什么会有这本书.....	1
这本书是写给谁的.....	2
致谢	4
目录.....	5
第 1 章 搜索引擎索引	7
1.1 索引基础	7
1.2 单词词典	13
1.3 倒排列表 (Posting List)	14
1.4 建立索引	15
1.5 动态索引	21
1.6 索引更新策略.....	22
第 2 章 链接分析.....	29
2.1 Web 图	29
2.2 两个概念模型及算法之间的关系.....	30
2.3 PageRank 算法.....	34
2.4 HITS 算法 (Hypertext Induced Topic Selection)	38
2.5 SALSA 算法	44
2.6 主题敏感 PageRank (Topic Sensitive PageRank)	50
2.7 Hilltop 算法	54
2.8 其他改进算法.....	60

本章提要.....	61
本章参考文献	62
第 3 章 网页反作弊.....	63
3.1 内容作弊	63
3.2 链接作弊	65
3.3 页面隐藏作弊.....	68
3.5 反作弊技术的整体思路.....	69
3.7 专用链接反作弊技术	73
3.8 识别内容作弊.....	74
3.9 反隐藏作弊	75
3.10 搜索引擎反作弊综合框架.....	76
本章提要.....	77
本章参考文献	77

第 1 章 搜索引擎索引

“吾有三剑，唯子所择；皆不能杀人，且先言其状。一曰含光，视之不可见，运之不知有。其所触也，泯然无际，经物而物不觉。二曰承影，将旦昧爽之交，日夕昏明之际，北面而察之，淡淡焉若有物存，莫识其状。其所触也，窃窃然有声，经物而物不疾也。三曰宵练，方昼则见影而不见光，方夜见光而不见形。其触物也，騄然而过，随过随合，觉疾而不血刃焉。此三宝者，传之十三世矣，而无施于事。匣而藏之，未尝启封。”

 《列子汤问》

索引其实在日常生活中是很常见的，比如书籍的目录就是一种索引结构，目的是为了让人们能够更快地找到相关章节内容。再比如像 hao123 这种类型的导航网站本质上也是互联网页面中的索引结构，目的类似，也是为了让用户能够尽快找到有价值的分类网站。

在计算机科学领域，索引也是非常常用的数据结构。其根本目的是为了在具体应用中加快查找速度。比如在数据库中，在很多高效数据结构中，都会大量采用索引来提升系统效率。

具体到搜索引擎，索引更是其中最重要的核心技术之一，面对海量的网页内容，如何快速找到包含用户查询词的所有网页？倒排索引在其中扮演了关键的角色。本章主要讲解与倒排索引相关的技术。

1.1 索引基础

本节通过引入简单实例，介绍与搜索引擎索引有关的一些基本概念，了解这些基本概念对于后续深入了解索引的工作机制非常重要。

1.1.1 单词—文档矩阵

单词—文档矩阵是表达两者之间所具有的一种包含关系的概念模型，图 1-1 展示了其含义。图 1-1 的每列代表一个文档，每行代表一个单词，打对钩的位置代表包含关系。

单词—文档矩阵					
	文档1	文档2	文档3	文档4	文档5
词汇1	✓			✓	
词汇2		✓	✓		
词汇3				✓	
词汇4	✓				✓
词汇5		✓			
词汇6			✓		

图 1-1 单词—文档矩阵

从纵向即文档这个维度来看，每列代表文档包含了哪些单词，比如文档 1 包含了词汇 1 和词汇 4，而不包含其他单词。从横向即单词这个维度来看，每行代表了哪些文档包含了某个单词。比如对于词汇 1 来说，文档 1 和文档 4 中出现过词汇 1，而其他文档不包含词汇 1。矩阵中其他的行列也可做此种解读。

搜索引擎的索引其实就是实现单词—文档矩阵的具体数据结构。可以有不同的方式来实现上述概念模型，比如倒排索引、签名文件、后缀树等方式。但是各项实验数据表明，倒排索引是单词到文档映射关系的最佳实现方式，所以本章主要介绍倒排索引的技术细节。

1.1.2 倒排索引基本概念

在本小节，我们会解释倒排索引常用到的一些专用术语，为了表达的便捷性，在本书后续章节会直接使用这些术语。

- 文档 (Document)：一般搜索引擎的处理对象是互联网网页，而文档这个概念要更宽泛些，代表以文本形式存在的存储对象。相比网页来说，涵盖更多形式，比如 Word、PDF、html、XML 等不同格式的文件都可以称为文档，再比如一封邮件、一条短信、一条微博也可以称为文档。在本书后续内容中，很多情况下会使用文档来表征文本信息。
- 文档集合 (Document Collection)：由若干文档构成的集合称为文档集合。比如海量的互联网网页或者说大量的电子邮件，都是文档集合的具体例子。
- 文档编号 (Document ID)：在搜索引擎内部，会为文档集合内每个文档赋予一个唯一的内部编号，以此编号来作为这个文档的唯一标识，这样方便内部处理。每个文档的内部编号即称为文档编号，后文有时会用 DocID 来便捷地代表文档编号。
- 单词编号 (Word ID)：与文档编号类似，搜索引擎内部以唯一的编号来表征某个单词，单词编号可以作为某个单词的唯一表征。

- 倒排索引 (Inverted Index)：倒排索引是实现单词—文档矩阵的一种具体存储形式。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。倒排索引主要由两个部分组成：单词词典和倒排文件。
- 单词词典 (Lexicon)：搜索引擎通常的索引单位是单词，单词词典是由文档集中出现过的所有单词构成的字符串集合，单词词典内每条索引项记载单词本身的一些信息及指向倒排列表的指针。
- 倒排列表 (PostingList)：倒排列表记载了出现过某个单词的所有文档的文档列表及单词在该文档中出现的位置信息，每条记录称为一个倒排项 (Posting)。根据倒排列表，即可获知哪些文档包含某个单词。
- 倒排文件 (Inverted File)：所有单词的倒排列表往往顺序地存储在磁盘的某个文件里，这个文件即被称为倒排文件，倒排文件是存储倒排索引的物理文件。

关于这些概念之间的关系，通过图 1-2 可以比较清晰地看出来。

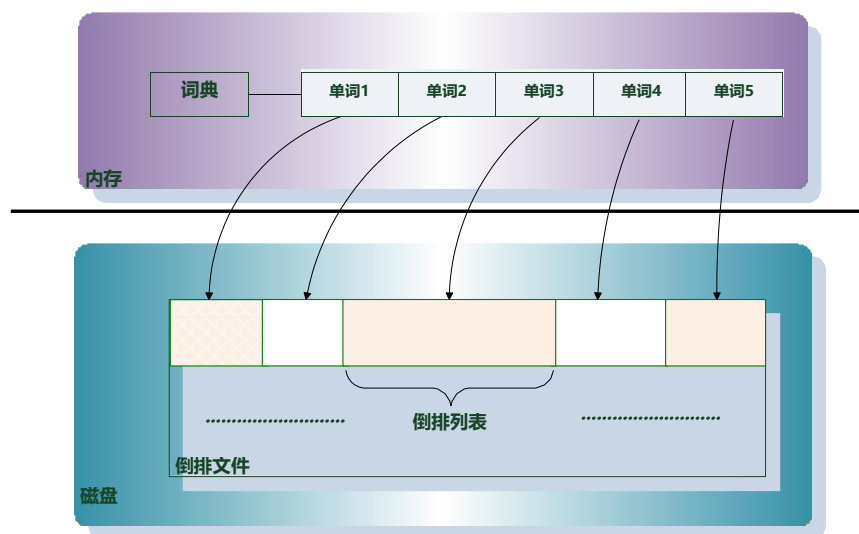


图 1-2 倒排索引基本概念示意图

1.1.3 倒排索引简单实例

倒排索引从逻辑结构和基本思路上讲非常简单。下面我们通过具体实例来进行说明，使读者能够对倒排索引有一个宏观而直接的感受。

假设文档集合包含 5 个文档，每个文档内容如图 1-3 所示，在图的最左端一栏是每个文档对应的文档编号，我们的任务就是对这个文档集合建立倒排索引。



文档编号	文档内容
1	谷歌地图之父跳槽Facebook
2	谷歌地图之父加盟Facebook
3	谷歌地图创始人拉斯离开谷歌加盟Facebook
4	谷歌地图之父跳槽Facebook 与 Wave项目取消有关
5	谷歌地图之父拉斯加盟社交网站 Facebook

图 1-3 文档集合

中文和英文等语言不同，单词之间没有明确的分隔符号，所以首先要用分词系统将文档自动切分成单词序列，这样每个文档就转换为由单词序列构成的数据流。为了系统后续处理方便，需要对每个不同的单词赋予唯一的单词编号，同时记录下哪些文档包含这个单词，在如此处理结束后，我们可以得到最简单的倒排索引（参考图 1-4）。在图 1-4 中，“单词 ID”一列记录了每个单词的单词编号，第 2 列是对应的单词，第 3 列即每个单词对应的倒排列表。比如单词“谷歌”，其单词编号为 1，倒排列表为{1,2,3,4,5}，说明文档集合中每个文档都包含了这个单词。

之所以说图 1-4 所示的倒排索引是最简单的，是因为这个索引系统只记载了哪些文档包含某个单词，而事实上，索引系统还可以记录除此之外的更多信息。图 1-5 是一个相对复杂些的倒排索引，与图 1-4 所示的基本索引系统相比，在单词对应的倒排列表中不仅记录了文档编号，还记载了单词频率信息（TF），即这个单词在某个文档中出现的次数，之所以要记录这个信息，是因为词频信息在搜索结果排序时，计算查询和文档相似度是一个很重要的计算因子，所以将其记录在倒排列表中，以方便后续排序时进行分值计算。在图 1-5 所示的例子中，单词“创始人”的单词编号为 7，对应的倒排列表内容有（3；1），其中 3 代表文档编号为 3 的文档包含这个单词，数字 1 代表词频信息，即这个单词在 3 号文档中只出现过 1 次，其他单词对应的倒排列表所代表的含义与此相同。



单词ID	单词	倒排列表 (DocID)
1	谷歌	1,2,3,4,5
2	地图	1,2,3,4,5
3	之父	1,2,4,5
4	跳槽	1,4
5	Facebook	1,2,3,4,5
6	加盟	2,3,5
7	创始人	3
8	拉斯	3,5
9	离开	3
10	与	4
11	Wave	4
12	项目	4
13	取消	4
14	有关	4
15	社交	5
16	网站	5

图 1-4 最简单的倒排索引



单词ID	单词	倒排列表 (DocID;TF)
1	谷歌	(1;1),(2;1),(3;2),(4;1),(5;1)
2	地图	(1;1),(2;1),(3;1),(4;1),(5;1)
3	之父	(1;1),(2;1),(4;1),(5;1)
4	跳槽	(1;1),(4;1)
5	Facebook	(1;1),(2;1),(3;1),(4;1),(5;1)
6	加盟	(2;1),(3;1),(5;1)
7	创始人	(3;1)
8	拉斯	(3;1),(5;1)
9	离开	(3;1)
10	与	(4;1)
11	Wave	(4;1)
12	项目	(4;1)
13	取消	(4;1)
14	有关	(4;1)
15	社交	(5;1)
16	网站	(5;1)

图 1-5 带有单词频率信息的倒排索引

实用的倒排索引还可以记载更多的信息，如图 1-6 所示的索引系统除了记录文档编号和单词频率信息外，额外记载了两类信息，即每个单词对应的文档频率信息（对应图 1-6 的第 3 列）及单词在某个文档出现位置的信息。



单词ID	单词	文档频率	倒排列表 (DocID;TF;<POS>)
1	谷歌	5	{1;1;<1>},{2;1;<1>},{3;2;<1;6>},{4;1;<1>},{5;1;<1>}
2	地图	5	{1;1;<2>},{2;1;<2>},{3;1;<2>},{4;1;<2>},{5;1;<2>}
3	之父	4	{1;1;<3>},{2;1;<3>},{4;1;<3>},{5;1;<3>}
4	吐槽	2	{1;1;<4>},{4;1;<4>}
5	Facebook	5	{1;1;<5>},{2;1;<5>},{3;1;<8>},{4;1;<5>},{5;1;<8>}
6	加盟	3	{2;1;<4>},{3;1;<7>},{5;1;<5>}
7	创始人	1	{3;1;<3>}
8	拉斯	2	{3;1;<4>},{5;1;<4>}
9	离开	1	{3;1;<5>}
10	与	1	{4;1;<6>}
11	Wave	1	{4;1;<7>}
12	项目	1	{4;1;<8>}
13	取消	1	{4;1;<9>}
14	有关	1	{4;1;<10>}
15	社交	1	{5;1;<6>}
16	网站	1	{5;1;<7>}

图 1-6 带有单词频率、文档频率和出现位置信息的倒排索引

文档频率信息代表了在文档集中有多少个文档包含某个单词，之所以要记录这个信息，其原因与单词频率信息一样，这个信息在搜索结果排序计算中是一个非常重要的因子。而单词在某个文档中出现位置的信息并非索引系统一定要记录的，在实际的索引系统里可以包含，也可以选择不包含这个信息，之所以如此是因为这个信息对于搜索系统来说并非必需，位置信息只有在支持短语查询的时候才能够派上用场。

以单词“拉斯”为例，其单词编号为 8，文档频率为 2，代表整个文档集中有两个文档包含这个单词，对应的倒排列表为{{3;1;<4>},{5;1;<4>}}, 其含义为在文档 3 和文档 5 出现过这个单词，单词频率都为 1，单词“拉斯”在两个文档中的出现位置都是 4，即文档中第 4 个单词是“拉斯”。

如图 1-6 所示的倒排索引已经是一个非常完备的索引系统，实际搜索系统的索引结构基本如此，区别无非是采取哪些具体的数据结构来实现上述逻辑结构。

有了这个索引系统，搜索引擎可以很方便地响应用户的查询，比如用户输入查询词“Facebook”，搜索系统查找倒排索引，从中可以读出包含这个单词的文档，这些文档就是提供给用户的搜索结果，而利用单词频率信息、文档频率信息即可对这些候选搜索结果进行排序，计算文档和查询的相似性，按照相似性得分由高到低排序输出，此即为搜索系统的部分内部流程，具体实现方案本书第 5 章会做详细描述。

1.2 单词词典

单词词典是倒排索引中非常重要的组成部分，它用来维护文档集中出现过的所有单词的相关信息，同时用来记载某个单词对应的倒排列表在倒排文件中的位置信息。在支持搜索时，根据用户的查询词，去单词词典里查询，就能够获得相应的倒排列表，并以此作为后续排序的基础。

对于一个规模很大的文档集合来说，可能包含几十万甚至上百万的不同单词，能否快速定位某个单词，这直接影响搜索时的响应速度，所以需要高效的数据结构来对单词词典进行构建和查找，常用的数据结构包括哈希加链表结构和树形词典结构。

1.2.1 哈希加链表

图 1-7 是这种词典结构的示意图。这种词典结构主要由两个部分构成，主体部分是哈希表，每个哈希表项保存一个指针，指针指向冲突链表，在冲突链表里，相同哈希值的单词形成链表结构。之所以会有冲突链表，是因为两个不同单词获得相同的哈希值，如果是这样，在哈希方法里被称做是一次冲突，可以将相同哈希值的单词存储在链表里，以供后续查找。

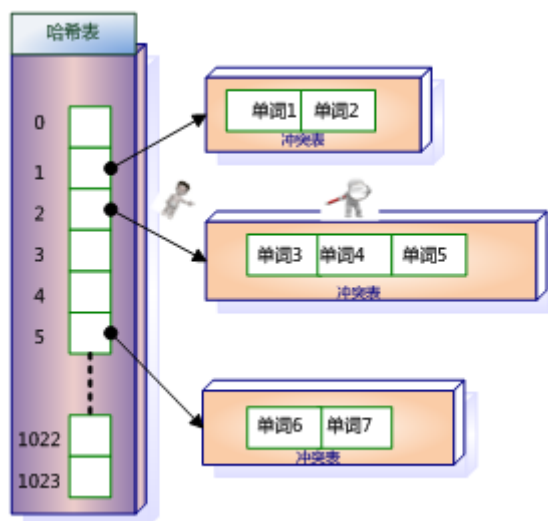


图 1-7 哈希加链表词典结构

在建立索引的过程中，词典结构也会相应地被构建出来。比如在解析一个新文档的时候，对于某个在文档中出现的单词 T ，首先利用哈希函数获得其哈希值，之后根据哈希值对应的哈希表项读取其中保存的指针，就找到了对应的冲突链表。如果冲突链表里已经存在这个单词，说明单词在之前解析的文档里已经出现过。如果在冲突链表里没有发现这个单词，说明该单词是首次碰到，则将其加入冲突链表里。通过这种方式，当文档集合内所有文档解析完毕时，

相应的词典结构也就建立起来了。

在响应用户查询请求时，其过程与建立词典类似，不同点在于即使词典里没出现过某个单词，也不会添加到词典内。以图 1-7 为例，假设用户输入的查询请求为单词 3，对这个单词进行哈希，定位到哈希表内的 2 号槽，从其保留的指针可以获得冲突链表，依次将单词 3 和冲突链表内的单词比较，发现单词 3 在冲突链表内，于是找到这个单词，之后可以读出这个单词对应的倒排列表来进行后续的工作，如果没有找到这个单词，说明文档集合内没有任何文档包含单词，则搜索结果为空。

1.2.2 树形结构

B 树（或者 B+树）是另外一种高效查找结构，图 1-8 是一个 B 树结构示意图。B 树与哈希方式查找不同，需要字典项能够按照大小排序（数字或者字符序），而哈希方式则无须数据满足此项要求。

B 树形成了层级查找结构，中间节点用于指出一定顺序范围的词典项目存储在哪个子树中，起到根据词典项比较大小的作用，最底层的叶子节点存储单词的地址信息，根据这个地址就可以提取出单词字符串。

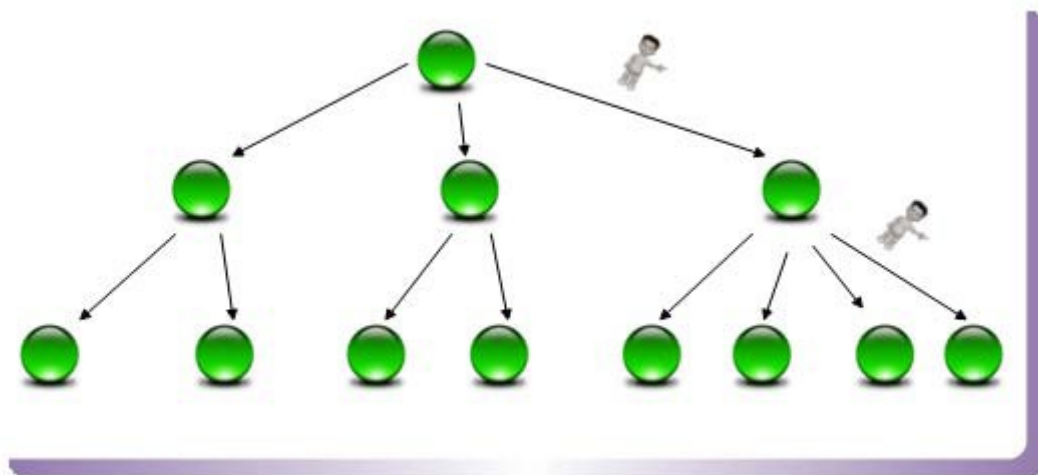


图 1-8 B 树查找结构

1.3 倒排列表 (Posting List)

在本章第一小节介绍的简单索引例子中，大致可以看到倒排列表起到的作用，倒排列表用来记录有哪些文档包含了某个单词。一般在文档集合里会有很多文档包含某个单词，每个文档会记录文档编号 (DocID)，单词在这个文档中出现的次数 (TF) 及单词在文档中哪些位置出

现过等信息，这样与一个文档相关的信息被称做倒排索引项（Posting），包含这个单词的一系列倒排索引项形成了列表结构，这就是某个单词对应的倒排列表。图 1-9 是倒排列表的示意图，在文档集中出现过所有单词及其对应的倒排列表组成了倒排索引。

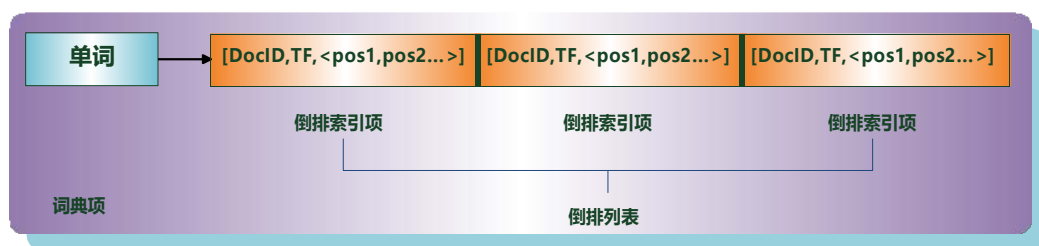


图 1-9 倒排列表示意图

在实际的搜索引擎系统中，并不存储倒排索引项中的实际文档编号，而是代之以文档编号差值（D-Gap）。文档编号差值是倒排列表中相邻的两个倒排索引项文档编号的差值，一般在索引构建过程中，可以保证倒排列表中后面出现的文档编号大于之前出现的文档编号，所以文档编号差值总是大于 0 的整数。如图 1-10 所示的例子中，原始的 3 个文档编号分别是 187、196 和 199，通过编号差值计算，在实际存储的时候就转化成了：187、9、3。

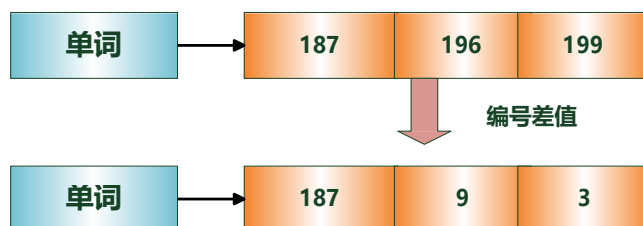


图 1-10 文档编号差值示例

之所以要对文档编号进行差值计算，主要原因是为了更好地对数据进行压缩，原始文档编号一般都是大数值，通过差值计算，就有效地将大数值转换为了小数值，而这有助于增加数据的压缩率。至于为何这样，本书第 4 章在专门讲述索引压缩时会叙述其原因。

1.4 建立索引

正如前面章节所述，索引结构如果建立好了，可以提高搜索的速度，那么给定一个文档集合，索引是如何建立起来的呢？建立索引的方式有很多种，本节叙述比较实用的 3 种建立索引的方法。

1.4.1 两遍文档遍历法（2-Pass In-Memory Inversion）

顾名思义，此方法需要对文档集合进行两遍扫描，图 1-11 是这种方法的示意图。值得注意

的一点是：此方法完全是在内存里完成索引的创建过程的，而另外两种方法则是通过内存和磁盘相互配合来完成索引建立任务的。

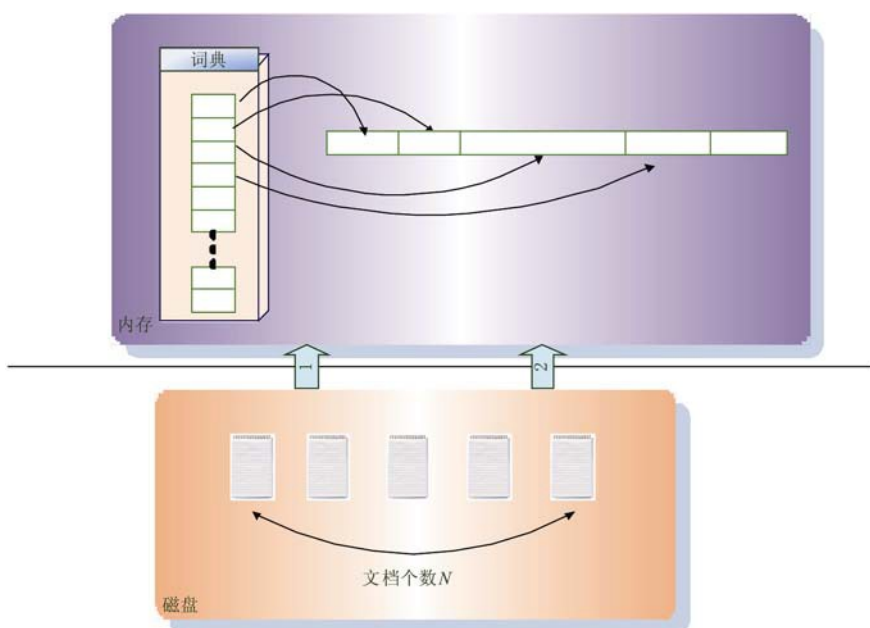


图 1-11 两遍文档遍历法

- 第一遍文档遍历

在第一遍扫描文档集合时，该方法并没有立即开始建立索引，而是收集一些全局的统计信息。比如文档集合包含的文档个数 N ，文档集合内所包含的不同单词个数 M ，每个单词在多少个文档中出现过的信息 DF 。将所有单词对应的 DF 值全部相加，就可以知道建立最终索引所需的内存大小是多少，因为一个单词对应的 DF 值如果是 10，说明有 10 个文档包含这个单词，那么这个单词对应的倒排列表应该包含 10 项内容，每一项记载某个文档的文档 ID 和单词在该文档对应的出现次数 TF 。

在获得了上述 3 类信息后，就可以知道最终索引的大小，于是在内存中分配足够大的空间，用来存储倒排索引内容。如图 1-11 所示，在内存中可以开辟连续存储区域，因为第一遍扫描已经获得了每个单词的 DF 信息，所以将连续存储区划分成不同大小的片段，词典内某个单词根据自己对应的 DF 信息，可以通过指针，指向属于自己的内存片段的起始位置和终止位置，将来在第二遍扫描时，这个单词对应的倒排列表信息会被填充进这个片段中。

综上所述，第一遍扫描的主要目的是获得一些统计信息，并根据统计信息分配内存等资源，同时建立好单词相对应倒排列表在内存中的位置信息，即主要做些资源准备工作。

- 第二遍文档遍历

在第二遍扫描的时候，开始真正建立每个单词的倒排列表信息，即对某个单词来说，获得包含这个单词的每个文档的文档 ID，以及这个单词在文档中的出现次数 TF，这样就可以不断填充第一遍扫描所分配的内存空间。当第二遍扫描结束的时候，分配的内存空间正好被填满，而每个单词用指针所指向的内存区域“片段”，其起始位置和终止位置之间的数据就是这个单词对应的倒排列表。

经过两遍扫描完成索引建立后，即可将内存的倒排列表和词典信息写入磁盘，这样就完成了建立索引的过程。从上述流程可以看出，索引的构建完全是在内存中完成的，这就要求内存一定要足够大，否则如果文档集合太大，内存未必能够满足需求。

从另外一个角度看，在建立索引的过程中，从磁盘读取文档并解析文档基本是最消耗时间的一个步骤，而两遍扫描法因为要对文档集合进行两遍遍历，所以从速度上不占优势，在实际中采用这种方法的系统并不常见。而下面介绍的两种方法都是对文档集合进行一遍扫描，所以在速度方面明显占优。

1.4.2 排序法 (Sort-based Inversion)

两遍遍历法在建立索引的过程中，对内存的消耗要求较高，不同的文档集合包含文档数量大小不同，其所需内存大小是不确定的。当文档集合非常大时，可能因内存不够，导致无法建立索引。排序法对此做出了改进，该方法在建立索引的过程中，始终在内存中分配固定大小的空间，用来存放词典信息和索引的中间结果，当分配的空间被消耗光的时候，把中间结果写入磁盘，清空内存里中间结果所占空间，以用做下一轮存放索引中间结果的存储区。这种方法由于只需要固定大小的内存，所以可以对任意大小的文档集合建立索引（参考图 1-12）。

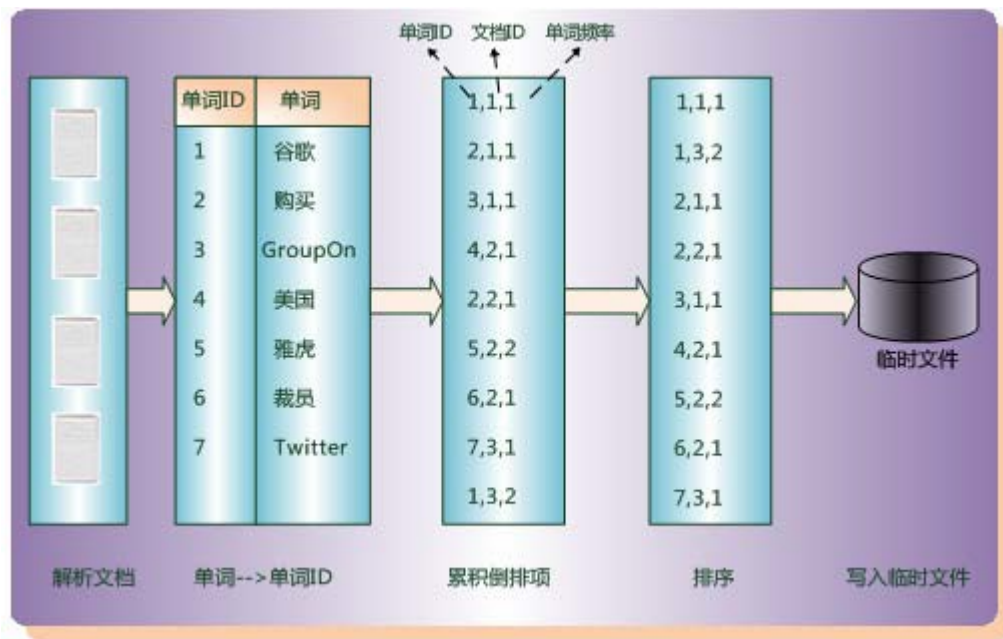


图 1-12 排序法

- 中间结果内存排序

图 1-12 是排序法在内存中建立索引中间结果的示意图。读入文档后，对文档进行编号，赋予唯一的文档 ID，并对文档内容解析。对于文档中出现的单词，通过查词典将单词转换为对应的单词 ID，如果词典中没有这个单词，说明是第一次碰到，则赋予单词以唯一的单词 ID 并插入词典中。在完成了由单词映射为单词 ID 的过程之后，可以对该文档内每个单词建立一个（单词 ID、文档 ID、单词频率）三元组，这个三元组就是单词对应文档的倒排列表项，将这个三元组追加进中间结果存储区末尾。如果文档内的所有单词都经过如此处理，形成三元组序列的形式，则该文档被处理完成，开始依次序处理下一文档，过程与此类似。

随着新的文档不断被处理完成，存储三元组集合的中间结果所占用的内存会越来越大，词典里包含的新单词也越来越多，当分配的内存定额被占满时，该方法对三元组中间结果进行排序。排序的原则是：主键是单词 ID，即首先要按照单词 ID 由小到大排序；次键是文档 ID，即在相同单词 ID 的情况下，按照文档 ID 由小到大排序。通过以上方式，三元组变为有序形式。为了腾出内存空间，将排好序的三元组写入磁盘临时文件中，这样就空出内存来进行后续文档的处理。这里需要注意的是：在建立索引的过程中，词典是一直存储在内存中的，每次清空内存只是将中间结果写入磁盘。随着处理文档的增加，词典占用的内存会逐渐增加，由于分配内存是固定大小，而词典占用内存越来越大，也就是说，越往后，可用来存储三元组的空间是越来越少的。

之所以要对中间结果进行排序，主要是为了方便后续的处理。因为每一轮处理都会在磁盘产生一个对应的中间结果文件，当所有文档处理完成后，在磁盘中会有多个中间结果文件，为了产生最终的索引，需要将这些中间结果文件合并。图 1-13 是如何对中间结果进行合并的示意图。

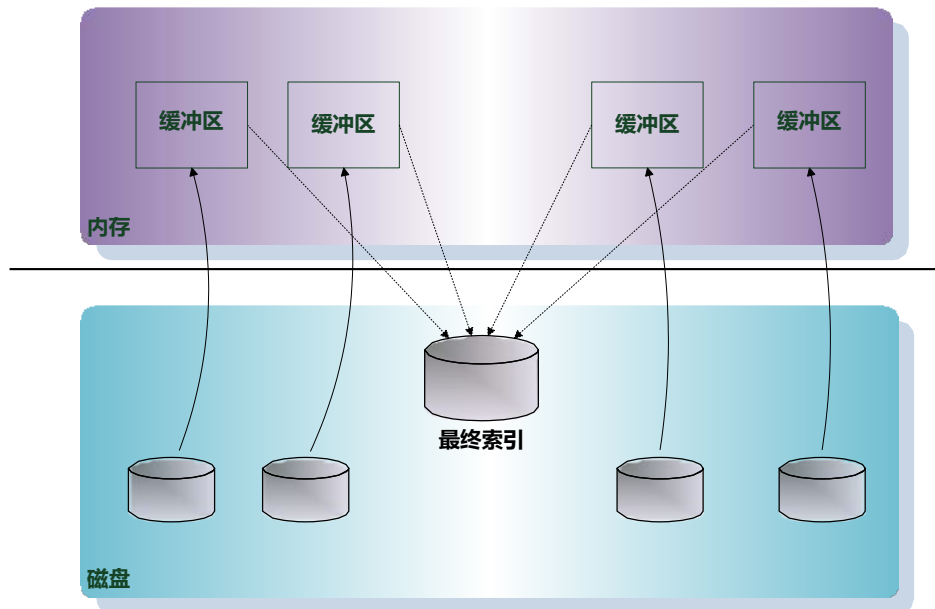


图 1-13 中间结果合并

- 合并中间结果

如图 1-13 所示，在合并中间结果的过程中，系统为每个中间结果文件在内存中开辟一个数据缓冲区，用来存放文件的部分数据。因为在形成中间结果文件前，已经按照单词 ID 和文档 ID 进行了排序，所以进入缓冲区的数据已经是有序的。合并过程中，将不同缓冲区中包含的同一个单词 ID 的三元组进行合并，如果某个单词 ID 的所有三元组全部合并完成，说明这个单词的倒排列表已经构建完成，则将其写入最终索引中，同时将各个缓冲区中对应这个单词 ID 的三元组内容清空，这样缓冲区就可以继续从中间结果文件中读入后续的三元组来进行下一个单词的三元组合并。当所有中间结果文件都依次被读入缓冲区，在合并完成后，就形成了最终的索引文件。

1.4.3 归并法 (Merge-based Inversion)

排序法分配固定大小内存来建立索引，所以无论要建立索引的文档集合有多大，都可以通过这种方法完成。但是如上所述，在分配的内存定额被消耗光时，排序法只是将中间结果写入磁盘，而词典信息一直在内存中进行维护，随着处理的文档越来越多，词典里包含的词典项

越来越多，所以占用内存越来越大，导致后期中间结果可用内存越来越少。归并法对此做出了改进，即每次将内存中数据写入磁盘时，包括词典在内的所有中间结果信息都被写入磁盘，这样内存所有内容都可以被清空，后续建立索引可以使用全部的定额内存。

图 1-14 是归并法的示意图。其整体流程和排序法大致相同，也是分为两个大的阶段，首先在内存里维护中间结果，当内存占满后，将内存数据写入磁盘临时文件，第二阶段对临时文件进行归并形成最终索引。

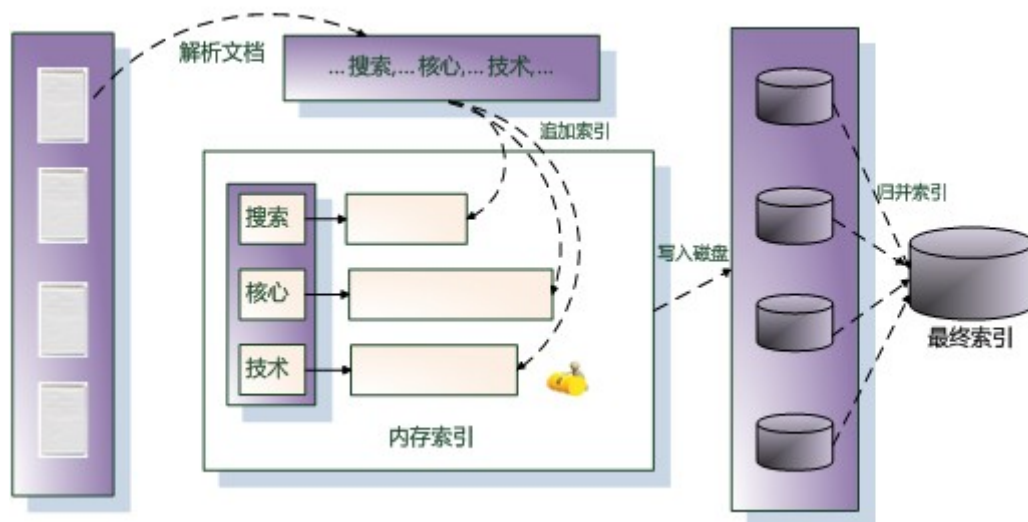


图 1-14 归并法

尽管从整体流程看，和排序法大致相同，但是在具体实现方式上有较大差异。

首先，排序法在内存中存放的是词典信息和三元组数据，在建立索引的过程中，词典和三元组数据并没有直接的联系，词典只是为了将单词映射为单词 ID。而归并法则是在内存中建立一个完整的内存索引结构，相当于对目前处理的文档子集单独在内存中建立起了一整套倒排索引，和最终索引相比，其结构和形式是相同的，区别只是这个索引只是部分文档的索引而非全部文档的索引。

其次，在将中间结果写入磁盘临时文件时，归并法会将整个内存的倒排索引写入临时文件，对于某个单词的倒排列表在写入磁盘文件时，将词典项放在列表最前端，之后跟随相应的倒排列表，这样依次将单词和对应的倒排列表写入磁盘文件，随后彻底清空所占内存。而排序法如上节所述，只是将三元组数据排序后写入磁盘临时文件，词典作为一个映射表一直存储在内存中。

在最后的临时文件合并为最终索引的过程中，两者也有差异。排序法因为保存的是有序三元组信息，所以在合并时，是对同一单词的三元组依次进行合并；而归并法的临时文件则是每

个单词对应的部分倒排列表，所以在合并时针对每个单词的倒排列表进行合并，形成这个单词的最终倒排列表。另外，归并法在最后的合并过程中形成最终的词典信息。

1.5 动态索引

如果搜索引擎需要处理的文档集合是静态集合，那么在索引建立好之后，就可以一直用建好的索引响应用户查询请求。但是，在真实环境中，搜索引擎需要处理的文档集合往往是动态集合，即在建好初始的索引后，后续不断有新文档进入系统，同时原先的文档集合内有些文档可能被删除或者内容被更改。典型的例子就是桌面搜索，当新下载一篇文档，那么搜索系统需要及时将其纳入，删除某篇文档也需要在非常短的时间内在搜索结果里体现出来。大多数搜索引擎面临的应用场景是类似的动态环境。

索引系统如何能够做到实时反映这种变化呢？动态索引可以实现这种实时性要求，图 1-15 是动态索引的示意图。在这种动态索引中，有 3 个关键的索引结构：倒排索引、临时索引和已删除文档列表。

倒排索引就是对初始文档集合建立好的索引结构，一般单词词典存储在内存，对应的倒排列表存储在磁盘文件中。临时索引是在内存中实时建立的倒排索引，其结构和前述的倒排索引是一样的，区别在于词典和倒排列表都在内存中存储。当有新文档进入系统时，实时解析文档并将其追加进这个临时索引结构中。已删除文档列表则用来存储已被删除的文档的相应文档 ID，形成一个文档 ID 列表。这里需要注意的是：当一篇文档内容被更改，可以认为是旧文档先被删除，之后向系统内增加一篇新的文档，通过这种间接方式实现对内容更改的支持。

当系统发现有新文档进入时，立即将其加入临时索引中。有文档被删除时，则将其加入删除文档队列。文档被更改时，则将原先文档放入删除队列，解析更改后的文档内容，并将其加入临时索引中。通过这种方式可以满足实时性的要求。

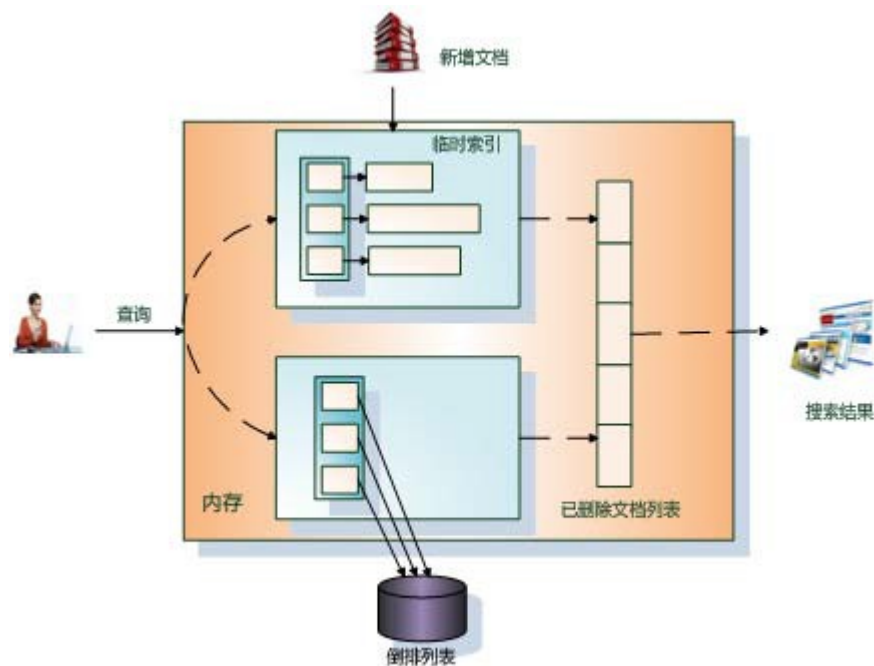


图 1-15 动态索引

如果用户输入查询请求，则搜索引擎同时从倒排索引和临时索引中读取用户查询单词的倒排列表，找到包含用户查询的文档集合，并对两个结果进行合并，之后利用删除文档列表进行过滤，将搜索结果中那些已经被删除的文档从结果中过滤，形成最终的搜索结果，并返回给用户。这样就能够实现动态环境下的准实时搜索功能。

1.6 索引更新策略

动态索引通过在内存中维护临时索引，可以实现对动态文档和实时搜索的支持。但是服务器内存总是有限的，随着新加入系统的文档越来越多，临时索引消耗的内存也会随之增加。当最初分配的内存将被使用完时，要考虑将临时索引的内容更新到磁盘索引中，以释放内存空间来容纳后续的新进文档，此时要考虑合理有效的索引更新策略。

常用的索引更新策略有 4 种：完全重建策略、再合并策略、原地更新策略及混合策略。

1.6.1 完全重建策略 (Complete Re-Build)

完全重建策略是一个相当直观的方法，当新增文档达到一定数量，将新增文档和原先的老文档进行合并，然后利用前述章节提到的建立索引的方式，对所有文档重新建立索引。新索引建立完成后，老的索引被遗弃释放，之后对用户查询的响应完全由新的索引负责。图 1-16

是这种策略的说明示意图。

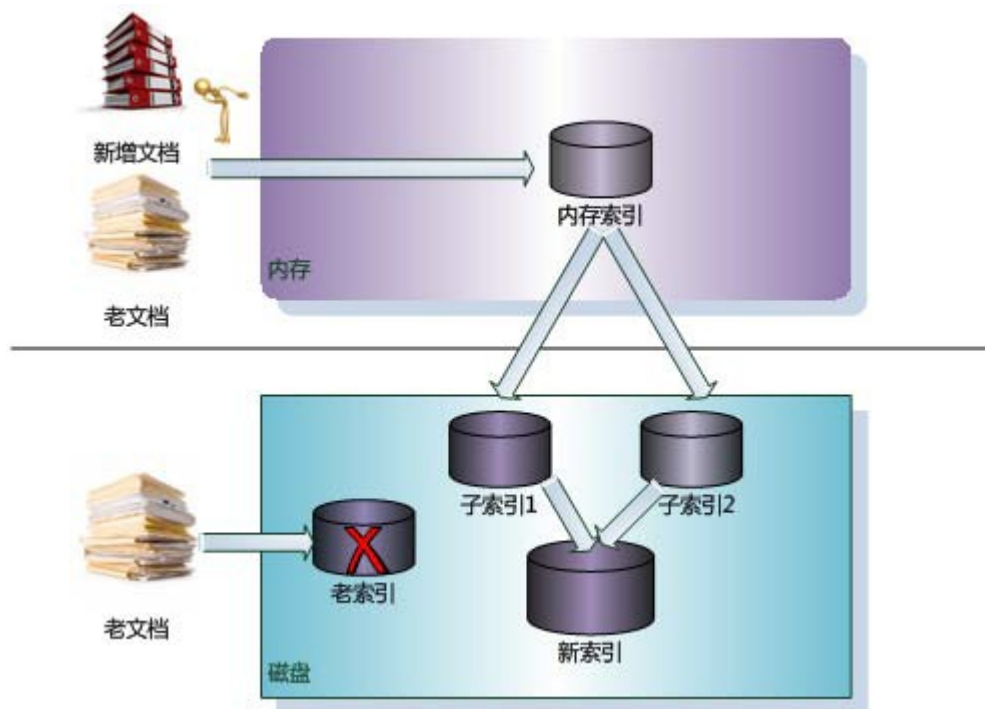


图 1-16 完全重建策略

因为重建索引需要较长时间，在进行索引重建的过程中，内存中仍然需要维护老的索引，来对用户的查询做出响应。只有当新索引完全建立完成后，才能释放旧的索引，将用户查询响应切换到新索引上。

这种重建策略比较适合小文档集合，因为完全重建索引的代价较高，但是目前主流商业搜索引擎一般是采用此方式来维护索引的更新的，这与互联网本身的特性有关。

1.6.2 再合并策略 (Re-Merge)

有新增文档进入搜索系统时，搜索系统在内存维护临时倒排索引来记录其信息，当新增文档达到一定数量，或者指定大小的内存被消耗完，则把临时索引和老文档的倒排索引进行合并，以生成新的索引。图 1-17 是这种策略的一种图示。

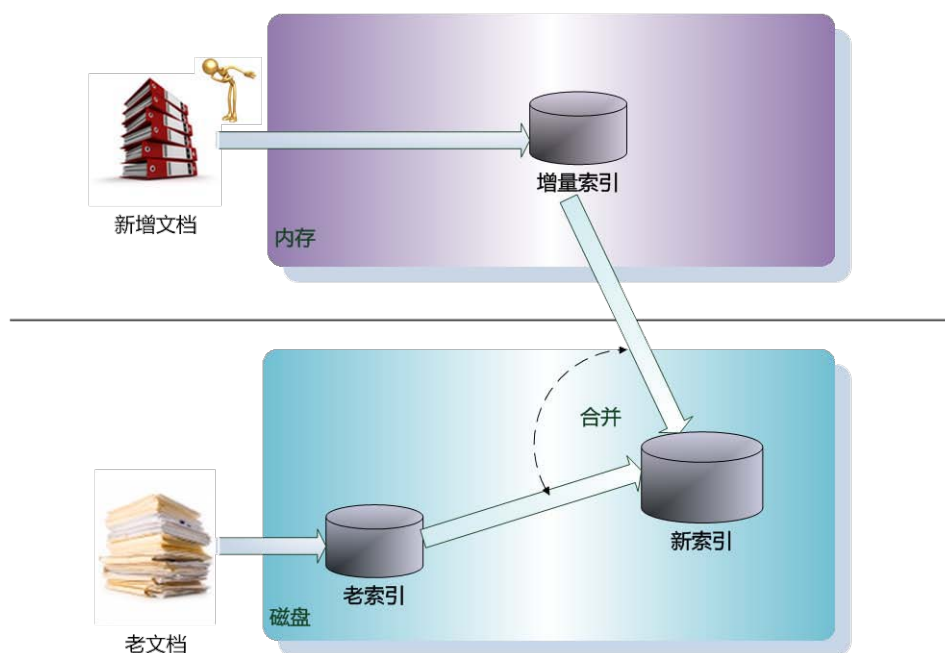


图 1-17 再合并策略

在实际的搜索系统中，再合并策略按照以下步骤进行索引内容的更新。

- 当新增文档进入系统，解析文档，之后更新内存中维护的临时索引，文档中出现的每个单词，在其倒排列表末尾追加倒排列表项，这个临时索引可称为增量索引。
- 一旦增量索引将指定的内存消耗光，此时需要进行一次索引合并，即将增量索引和老的倒排索引内容进行合并，图 1-18 是合并过程示意图。这里需要注意的是：倒排文件里的倒排列表存放顺序已经按照索引单词字典顺序由低到高进行了排序，增量索引在遍历词典的时候也按照字典序由低到高排列，这样对老的倒排文件只需进行一遍扫描，并可顺序读取，减少了文件操作中比较耗时的磁盘寻道时间，可以有效地提高合并效率。

在合并过程中，需要依次遍历增量索引和老索引单词词典中包含的单词及其对应的倒排列表，可以用两个指针分别指向两套索引中目前需要合并的单词（参考图 1-18 中箭头所指），按照如下方式进行倒排列表的合并。

考虑增量索引的单词指针指向的单词，如果这个单词在词典序中小于老索引的单词指针指向的单词，说明这个单词在老索引中未出现过，则直接将这个单词对应的倒排列表写入新索引的倒排文件中，同时增量索引单词指针后移指向下一个单词。

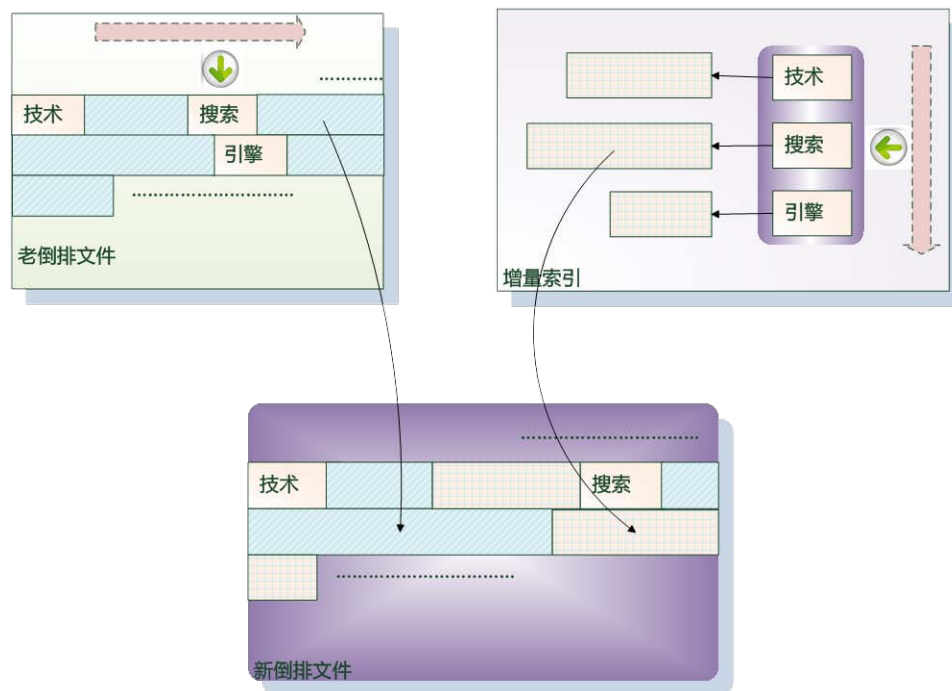


图 1-18 合并增量索引与老的倒排索引内容

如果两个单词指针指向的单词相同，说明这个单词在增量索引和老索引中同时出现，则将老索引中这个单词对应的倒排列表写入新索引的倒排列表，然后把增量索引中这个单词对应的倒排列表追加到其后，这样就完成了这个单词所有倒排列表的合并。图 1-18 中箭头所指单词是搜索，说明此时合并到了该单词，因为在老的索引系统和增量索引中都包含这个单词，所以首先将老索引对应的倒排列表追加到新索引倒排文件末尾，之后将增量索引中搜索这个单词对应的倒排列表追加在其后，这样就完成了这个单词索引项的合并。两个索引的单词指针都移动到下一个单词继续进行合并。

如果某个单词只在老索引中出现过，即发现老索引的单词指针指向的单词，其词典序小于增量索引单词指针指向的单词，则直接将老索引中对应的倒排列表写入新索引倒排文件中。老索引的单词指针后移指向下一个单词，继续进行合并。

当两个索引的所有单词都遍历完成后，新索引建成，此时可以遗弃释放老索引，使用新索引来响应用户查询请求。

同样地，在按照这个策略进行索引合并的过程中，为了能够响应用户查询，在合并索引期间，需要使用老索引响应用户查询请求。

再合并策略是效率非常高的一种索引更新策略，主要原因在于：在对老的倒排索引进行遍历时，因为已经按照索引单词的词典序由低到高排好顺序，所以可以顺序读取文件内容，减少磁盘寻道时间，这是其高效的根本原因。但是这种方法也有其缺点，因为要生成新的倒排索

引文件，所以对于老索引中的很多单词来说，尽管其倒排列表并未发生任何变化，但是也需要将其从老索引中读取出来并写入新索引中，这种对磁盘输入输出的消耗是没有太大必要且非常耗时的。

1.6.3 原地更新策略 (In-Place)

原地更新策略的基本出发点，可以认为是试图改进再合并策略的缺点。也就是说，在索引更新过程中，如果老索引的倒排列表没有变化，可以不需要读取这些信息，而只对那些倒排列表变化的单词进行处理。甚至希望能更进一步：即使老索引的倒排列表发生变化，是否可以只在其末尾进行追加操作，而不需要读取原先的倒排列表并重写到磁盘另一个位置？如果能够达到这个目标，明显可以大量减少磁盘读/写操作，提升系统执行效率。

为了达到上述目标，原地更新策略在索引合并时，并不生成新的索引文件，而是直接在原先老的索引文件里进行追加操作（参考图 1-19），将增量索引里单词的倒排列表项追加到老索引相应位置的末尾，这样就可达到上述目标，即只更新增量索引里出现的单词相关信息，其他单词相关信息不做变动。

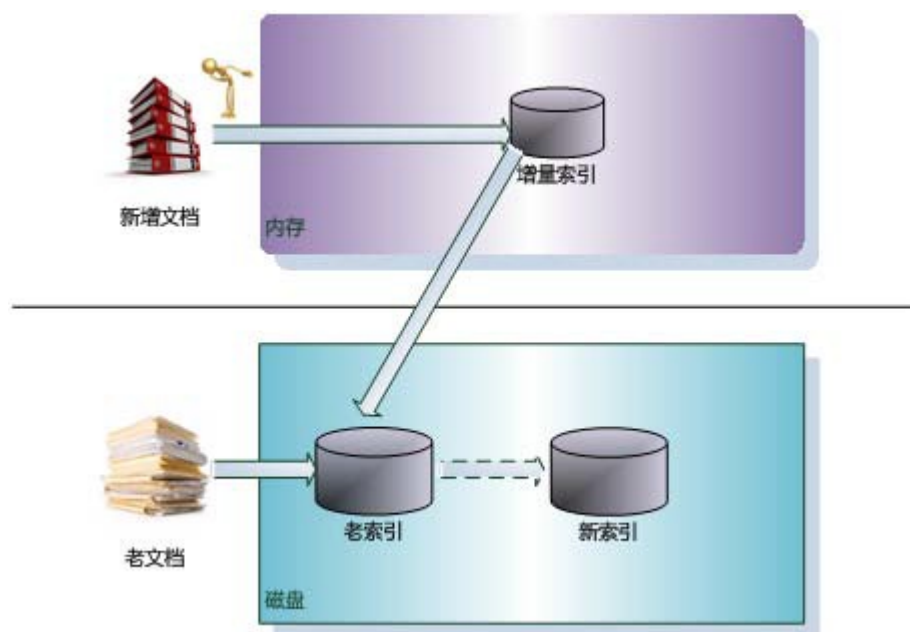


图 1-19 原地更新策略

但是这里存在一个问题：对于倒排文件中的两个相邻单词，为了在查询时加快读取速度，其倒排列表一般是顺序序列存储的，这导致没有空余位置用来追加新信息。为了能够支持追加操作，原地更新策略在初始建立的索引中，会在每个单词的倒排列表末尾预留出一定的磁盘空间，这样，在进行索引合并时，可以将增量索引追加到预留空间中。

图 1-20 是这种合并策略的一个说明。在图中，老索引中每个单词的倒排列表末尾都预留出空余磁盘空间，以作为信息追加时的存储区域。在对新增索引进行合并时，按照词典序，依次遍历新增索引中包含的单词，并对新增倒排列表的大小和老索引中相应预留空间大小进行比较，如果预留空间足够大，则将新增列表追加到老索引即可，如果预留空间不足以容纳新增倒排列表，那么此时需要在磁盘中找到一块完整的连续存储区，这个存储区足以容纳这个单词的倒排列表，之后将老索引中的倒排列表读出并写入新的磁盘位置，并将增量索引对应的倒排列表追加到其后，这样就完成了一次倒排列表的“迁移”工作。

在如图 1-20 所示的例子中，可以看出，单词“技术”和“引擎”在老索引中的预留空间足够大，所以对增量索引只需做追加写入即可，但是对于单词“搜索”来说，其预留空间不足以容纳新增倒排列表，所以这个单词的倒排列表需要迁移到磁盘另外一个连续存储区中。

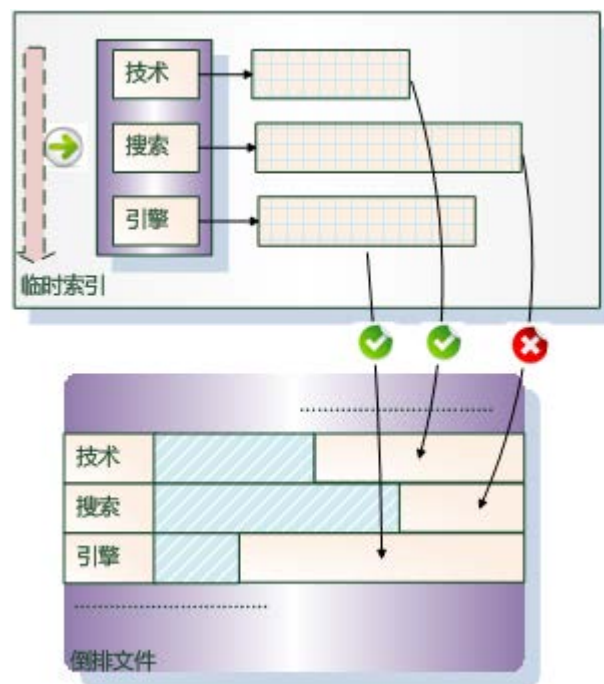


图 1-20 原地更新策略索引合并

原地更新策略的出发点很好，但是实验数据证明其索引更新效率比再合并策略低，主要是出于以下两个原因。

在这种方法中，对倒排列表进行“迁移”是比较常见的操作，为了能够进行快速迁移，需要找到足够大的磁盘连续存储区，所以这个策略需要对磁盘可用空间进行维护和管理，而这种维护和查找成本非常高，这成为该方法效率的一个瓶颈。

对于倒排文件中的相邻索引单词，其倒排列表顺序一般是按照相邻单词的词典序存储的，但是由于原地更新策略对单词的倒排列表做数据迁移，某些单词及其对应倒排列表会从老索引

中移出，这样就破坏了这种单词连续性，导致在进行索引合并时不能进行顺序读取，必须维护一个单词到其倒排文件相应位置的映射表，而这样做，一方面降低了磁盘读取速度，另外一方面需要大量的内存来存储这种映射信息。

1.6.4 混合策略 (Hybrid)

混合策略的出发点是能够结合不同索引更新策略的长处，将不同的索引更新策略混合，以形成更高效的方法。

混合策略一般会将单词根据其不同性质进行分类，不同类别的单词，对其索引采取不同的索引更新策略。常见的做法是：根据单词的倒排列表长度进行区分，因为有些单词经常在不同文档中出现，所以其对应的倒排列表较长，而有些单词很少见，则其倒排列表就较短。根据这一性质将单词划分为长倒排列表单词和短倒排列表单词。长倒排列表单词采取原地更新策略，而短倒排列表单词则采取再合并策略。

之所以这样做，是由于原地更新策略更适合长倒排列表单词，因为这种策略能够节省磁盘读/写次数，而长倒排列表单词的读/写开销明显要比短倒排列表单词大很多，所以如果采用原地更新策略，效果体现得比较显著。而大量短倒排列表单词读/写开销相对而言不算太大，所以利用再合并策略来处理，则其顺序读/写优势也能被充分利用。

InfoQ^{new}

促进软件开发领域知识与创新的传播



— 呈 — 固 — @ —
中文 | 英文 | 日文 | 葡文 | — — —

第 2 章 链接分析

“方以类聚，物以群分，吉凶生矣。”



《周易·系辞上》

搜索引擎在查找能够满足用户请求的网页时，主要考虑两方面的因素：一方面是由用户发出的查询与网页内容的内容相似性得分，即网页和查询的相关性，第 5 章已经就内容相似性计算做了介绍；另一方面就是通过链接分析方法计算获得的得分，即网页的重要性。搜索引擎融合两者，共同拟合出相似性评分函数，来对搜索结果进行排序。本章主要介绍一些著名的链接分析方法。

2.1 Web 图

互联网包含了海量网页，而网页和一般文本的一个重要区别是在页面内容中包含相互引用的链接（Link），如果将一个网页抽象成一个节点，而将网页之间的链接理解为一个有向边，则可以把整个互联网抽象为一个包含页面节点和节点之间联系边的有向图，称之为 Web 图。图 2-1 给出了 Web 图的形象化表示。

Web 图是对互联网的一种宏观抽象，其微观构成元素是一个个单独的网页。对于某个单独的网页 A 来说（参见图 2-2），在其内容部分往往会包含指向其他网页的链接，这些链接一般称为页面 A 的出链（Out Link）。因为

网页 A 是在一个网状结构中，所以不仅网页 A 会指向其他页面，也会有很多其他页面有链接指向网页 A，那么这些指向网页 A 的链接称为网页 A 的入链（In Link）。在图 2-2 中，网页 A 有两条出链和一条入链。

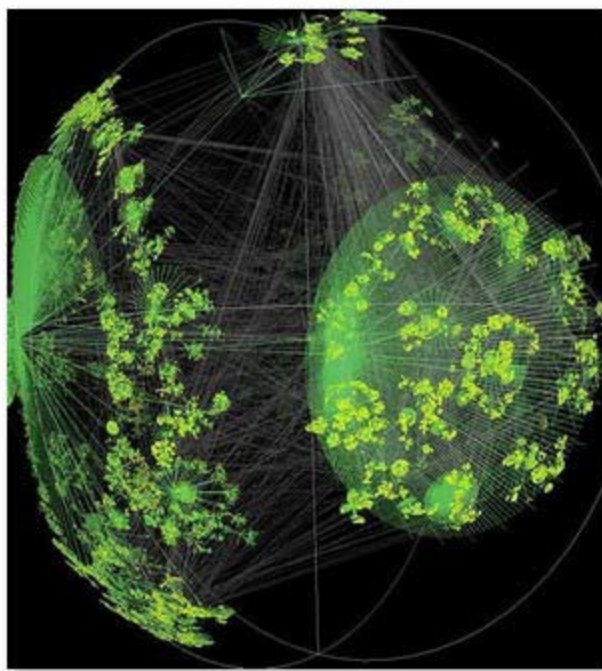


图 2-1 Web 图的形象化表示



图 2-2 入链与出链

锚文字也是网页中一种常见且非常有用的信息。所谓锚文字，就是页面内某个出链附近的一些描述文字。之所以锚文字会比较重要，是因为锚文字往往是对目标网页的一种概括性描述，所以在很多技术方法里都会利用这个信息来代表目标网页的含义。图 2-3 给出了一个锚文字与链接之间的关系示意图。

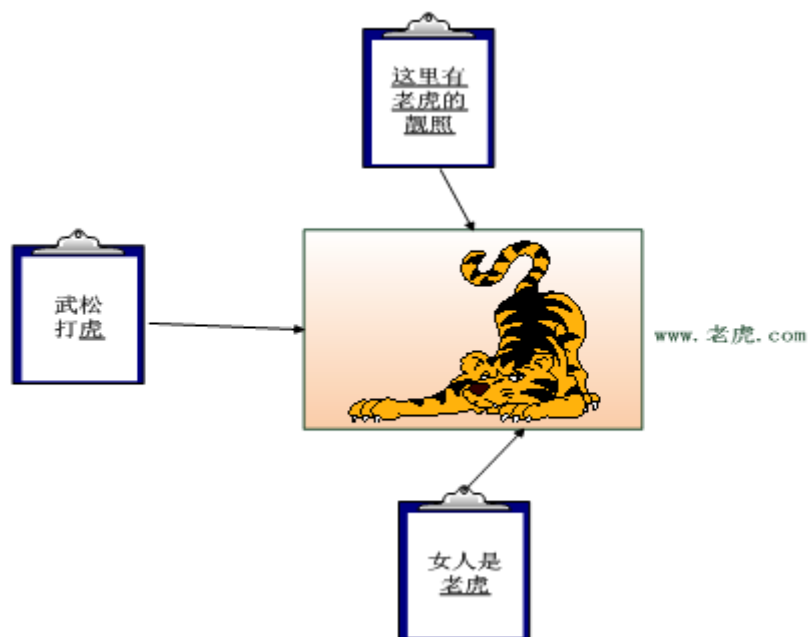


图 2-3 锚文字与链接之间的关系示意图

2.2 两个概念模型及算法之间的关系

在介绍具体链接分析算法之前，首先介绍两个概念模型，并对各个链接分析算法之间的关系进行说明，这样有助于读者从宏观角度理解各个算法的基本思路与传承关系。

2.2.1 随机游走模型 (Random Surfer Model)

互联网用户在网上时，往往有类似的网络行为：输入网址，浏览页面，然后顺着页面的链接不断打开新的网页。随机游走模型就是针对浏览网页的用户行为建立的抽象概念模型。之所以要建立这个抽象概念模型，是因为包括 PageRank 算法在内的很多链接分析算法都是建立在随机游走模型基础上的。

图 2-4 给出了随机游走模型的示意图。在最初阶段，用户打开浏览器浏览第 1 个网页，假设我们有一个虚拟时钟用来计时，此时可以设定时间为 1，用户在看完网页后，对网页内某个链接指向的页面感兴趣，于是点击该链接，进入第 2 个页面，此时虚拟时钟再次计时，时钟走向数字 2，如果网页包含了 k 个出链，则用户从当前页面跳转到任意一个链接所指向页面的概率是相等的。用户不断重复以上过程，在相互有链接指向的页面之间跳转。如果对于某个页面所包含的所有链接，用户都没有兴趣继续浏览，则可能会在浏览器中输入另外一个网址，直接到达该网页，这个行为称为远程跳转 (Teleporting)。假设互联网中共有 m 个页面，则用户远程跳转到任意一个页面的概率也是相等的，即为 $1/m$ 。随机游走模型就是一个对直接跳转和远程跳转两种用户浏览行为进行抽象的概念模型。

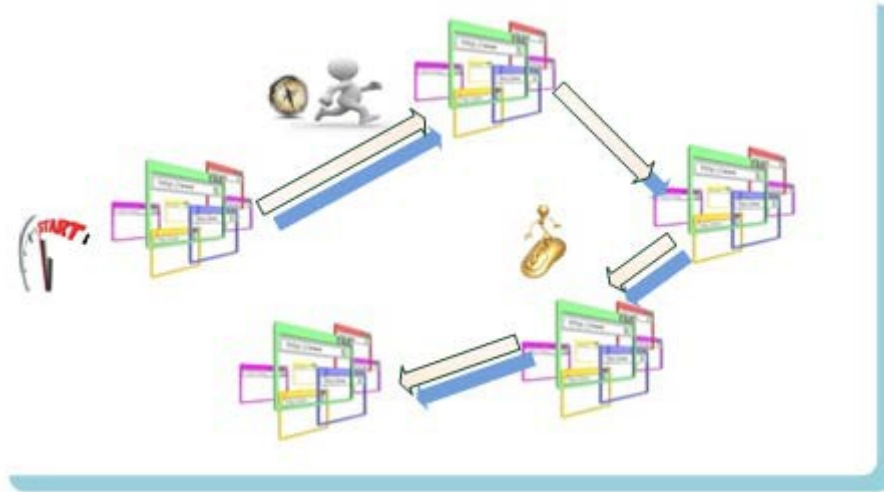


图 2-4 随机游走模型示意图

下面我们给出一个具体的随机游走模型的例子，为简单起见，该例子并未引入远程跳转行为。

在如图 2-5 所示的例子里，假设互联网由 A、B、C 3 个网页构成，其相互链接关系如图中页面节点之间的有向边所示。根据链接关系，即可计算页面节点之间的转移概率，比如对于节点 A 来说，只有唯一一个出链指向节点 B，所以从节点 A 跳转到节点 B 的概率为 1，对于节点 C 来说，其对节点 A 和 B 都有链接指向，所以转向任意一个其他节点的概率为 $1/2$ 。

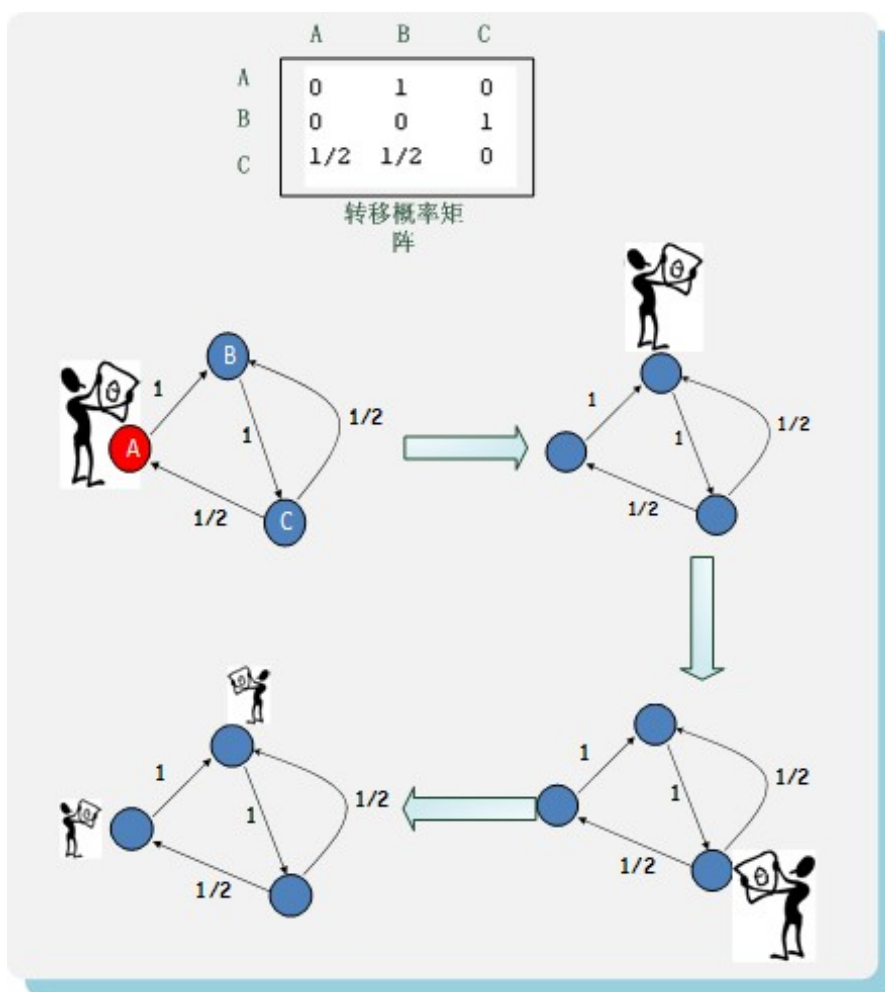


图 2-5 随机游走模型示例

假设在时刻 1，用户浏览页面 A，之后经由链接进入页面 B，然后进入页面 C，此时面临两种可能选择，跳转进入页面 A 或者页面 B 皆可，两者概率相同，都为 1/2。

假设例子中的互联网包含不止 3 个页面，而是由 10 个页面构成，此时用户既不想跳回页面 A，也不想跳回页面 B，则可以按照 1/10 的概率跳入其他任意一个页面，即进行远程跳转。

2.2.2 子集传播模型

子集传播模型是从诸多链接分析算法中抽象出来的概念模型（参见图 2-6）。其基本思想是在做算法设计时，把互联网网页按照一定规则划分，分为两个甚至是多个子集合。其中某个子集合具有特殊性质，很多算法往往从这个具有特殊性质的子集合出发，给予子集合内网页初始权值，之后根据这个特殊子集合内网页和其他网页的链接关系，按照一定方式将权值传递到其他网页。

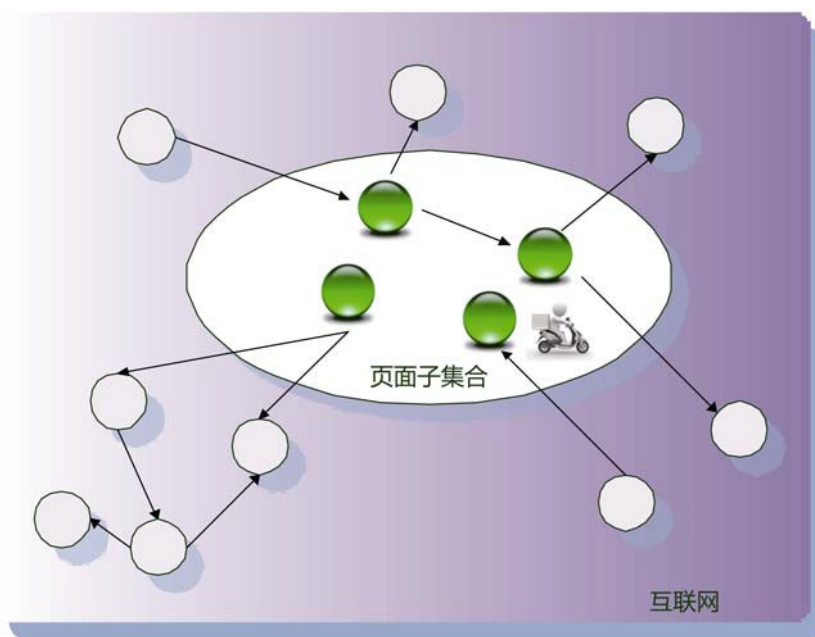


图 2-6 子集传播模型

本章介绍的一些链接分析算法符合子集传播模型，比如 HITS 算法和 Hilltop 算法及其衍生算法，在“网页反作弊”一章（第 8 章）会看到更多符合此模型的链接分析算法。

子集传播模型是个高度抽象的算法框架，很多算法可以认为是属于此框架的具体实例，即整体思路如上面所描述的流程，通常在以下几个方面各自存在不同。

- 如何定义特殊子集合，即在确定子集合内的网页应该有哪些特殊性质，具体算法规则不同。
- 在确定了特殊子集合所具有的性质后，如何对这个特殊子集合内网页给予一定的初始分值？不同算法打分方式各异。
- 从特殊子集合将其分值传播到其他网页时，采取何种传播方式？可传播的距离有多远？不同算法在此阶段也大都有差异。

注意：子集传播模型是本书作者从具体链接分析算法中归纳出的抽象模型，未见有文献明确提出，请读者阅读时谨慎参考。

2.2.3 链接分析算法之间的关系

到目前为止，学术界已经提出了很多链接分析算法，图 2-7 列出了其中影响力较大的一些算法及其相互关系，图中不同算法之间的箭头连接代表算法之间的改进关系，比如 SALSA 算法即融合了 PageRank 和 HITS 算法的基本思路。其他算法所代表的关系与此类似，可在图中明

显看出算法间的传承关系。

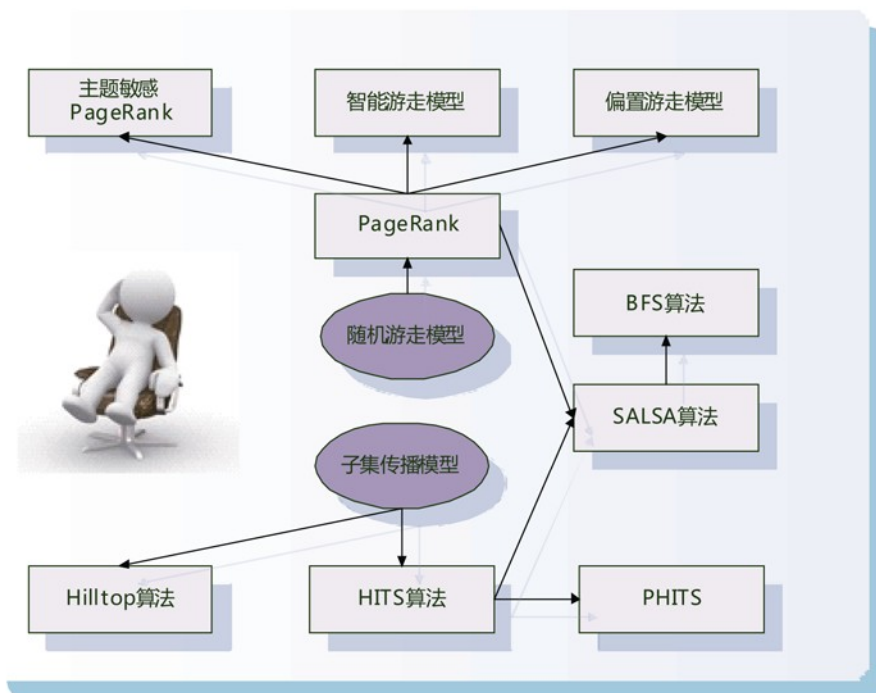


图 2-7 链接分析算法关系图

尽管链接算法很多，但是从其概念模型来说，基本遵循上述小节介绍的随机游走模型和子集传播模型。而从图中可看出，在众多算法中，PageRank 和 HITS 算法可以说是最重要的两个具有代表性的链接分析算法，后续的很多链接分析算法都是在这两个算法基础上衍生出来的改进算法。

在本章后续章节中，将详细介绍 PageRank 算法、HITS 算法、SALSA 算法、主题敏感 PageRank 算法和 Hilltop 算法。这些算法大都已被不同的商业搜索引擎所采用，在实际生活中发挥了很重要的作用。对于图 2-7 所列出的其他链接分析算法，将在本章末尾简述其原理。

2.3 PageRank 算法

PageRank 是 Google 创始人于 1997 年构建早期的搜索系统原型时提出的链接分析算法（参见图 28），自从 Google 在商业上获得空前的成功后，该算法也成为其他搜索引擎和学术界十分关注的计算模型。目前很多重要的链接分析算法都是在 PageRank 算法基础上衍生出来的。



图 2-8 Google 提出的 PageRank 算法

2.3.1 从入链数量到 PageRank

在 PageRank 提出之前，已经有研究者提出利用网页的入链数量来进行链接分析计算，这种入链方法假设一个网页的入链越多，则该网页越重要。早期的很多搜索引擎也采纳了入链数量作为链接分析方法，对于搜索引擎效果提升也有较明显的效果。

PageRank 除了考虑到入链数量的影响，还参考了网页质量因素，两者相结合获得了更好的网页重要性评价标准。

对于某个互联网网页 A 来说，该网页 PageRank 的计算基于以下两个基本假设：

- 数量假设：在 Web 图模型中，如果一个页面节点接收到的其他网页指向的入链数量越多，那么这个页面越重要。
- 质量假设：指向页面 A 的入链质量不同，质量高的页面会通过链接向其他页面传递更多的权重。所以越是质量高的页面指向页面 A，则页面 A 越重要。

通过利用以上两个假设，PageRank 算法刚开始赋予每个网页相同的重要性得分，通过迭代递归计算来更新每个页面节点的 PageRank 得分，直到得分稳定为止。

PageRank 计算得出的结果是网页的重要性评价，这 and 用户输入的查询是没有任何关系的，即算法是主题无关的。假设有一个搜索引擎，其相似度计算函数不考虑内容相似因素，完全采用 PageRank 来进行排序，那么这个搜索引擎的表现是什么样子的呢？这个搜索引擎对于任意不同的查询请求，返回的结果都是相同的，即返回 PageRank 值最高的页面。

2.3.2 PageRank 计算

本节探讨 PageRank 的具体计算过程。PageRank 的计算充分利用了上节提到的两个假设：数量假设和质量假设。网页通过链接关系构建起 Web 图，在初始阶段，每个页面设置相同的 PageRank 值，通过若干轮的计算，会得到每个页面所获得的最终 PageRank 值。随着每一轮的计算进行，网页当前的 PageRank 值会不断得到更新。

在一轮更新页面 PageRank 得分的计算中，每个页面将其当前的 PageRank 值平均分配到本页面包含的出链上，这样每个链接即获得了相应的权值。而每个页面将所有指向本页面的入链所传入的权值求和，即可得到新的 PageRank 得分。当每个页面都获得了更新后的 PageRank 值，就完成了一轮 PageRank 计算。

下面以如图 2-9 所示的例子来说明 PageRank 的具体计算过程。图中包含 7 个页面，其页面编号分别从 1 到 7，页面之间的链接关系如图所示。这里要注意的一点是：如图 2-9 所示的情况已是经过若干轮计算之后的情形，每个页面已经获得了当前的 PageRank 分值，比如页面 1 当前的 PageRank 分值为 0.304，页面 2 当前的 PageRank 分值为 0.166，其他页面的对应 PageRank 数值也在图中标出。我们接下来看看从当前的状态出发，如何进行下一轮的 PageRank 计算。

在 PageRank 计算中，从页面 A 指向页面 B 的某个链接的权值，代表了从页面 A 传导到页面 B 的 PageRank 分值。而对于页面 A 来说，如果有多个出链，那么页面 A 本身的 PageRank 分值会均等地分配给每个链接。比如图 2-9 中的页面 4，其当前 PageRank 分值为 0.105，包含 3 个出链，分别指向页面 2、页面 3 和页面 5，所以每个出链获得的分值为 0.035。图 2-9 中其他链接的权值也与页面 4 的出链一样，通过类似计算可以得到。

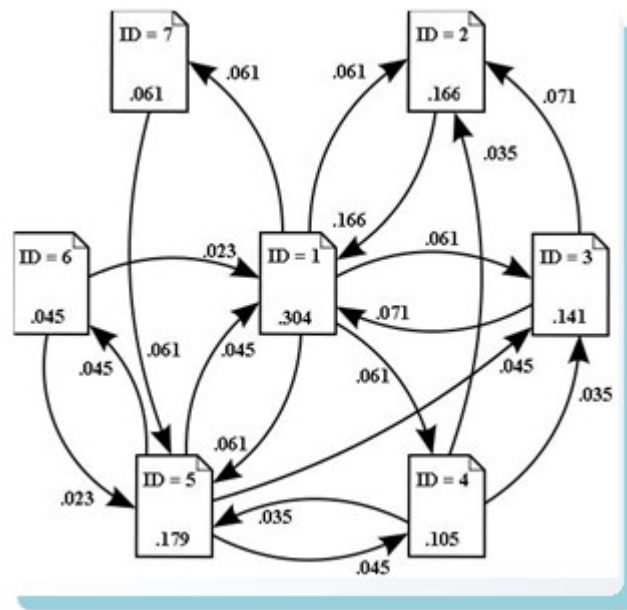


图 2-9 PageRank 计算实例

获得了每个链接传导的权值后，即可进行新一轮的 PageRank 计算。要想得到各个页面节点的得分，只需将网页入链所传入的分值汇总即可。比如图 2-9 中的页面 1，有 4 个入链，分别来自于页面 2、3、5、6。每个链接传导的权值也如图上所标，那么页面 1 的新的 PageRank 值为 4 个入链传入的权值之和，即：

$$PR(1)=0.166+0.071+0.045+0.023=0.305$$

即得分从上一轮的 0.304 更新到 0.305。

其他页面节点也依次如此计算，以获得新的 PageRank 分值，当所有页面节点的分值得到更新，就完成了一轮 PageRank 计算。如果在新的一轮 PageRank 计算之后，发现总体而言，页面节点的 PageRank 值基本稳定，不再发生较大变化，即可结束 PageRank 的计算，以此时得到的得分，作为最后排序时可以利用的 PageRank 得分。

2.3.3 链接陷阱 (Link Sink) 与远程跳转 (Teleporting)

互联网页面之间的链接结构实际上很复杂，上一小节介绍了 PageRank 的计算过程，但是对于某些特殊的链接结构，按照上述方法计算 PageRank 会导致问题，一个典型的例子就是“链接陷阱”（参见图 2-10）。

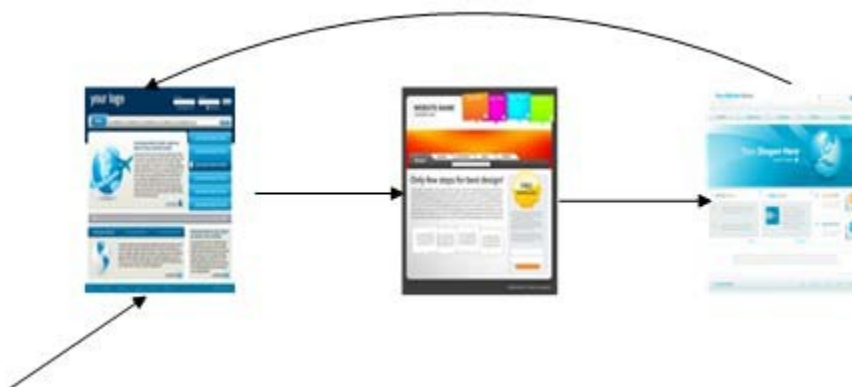


图 2-10 链接陷阱

图 2-10 包含了 3 个网页，相互有链接指向，形成了一个环形结构。这种结构类似于天体中的黑洞，在计算 PageRank 的时候，该结构将导致系统只会吸收传入的分值，而不能将获得的分值传播出去，随着 PageRank 一轮轮地连续运算，链接陷阱内的页面 PageRank 得分越来越高，这与 PageRank 的设计初衷相违背。

远程跳转是解决链接陷阱的通用方式，所谓的远程跳转，即在网页向外传递分值的时候，不限于向出链所指网页传递，也可以以一定的概率向任意其他网页跳转。对于链接陷阱内的网页来说，增加了远程跳转措施后，就像为每个页面增加了指向互联网任意其他页面的虚拟边，权值可以通过这种虚拟边向外传递，以此来避免链接陷阱导致的问题。

2.4 HITS 算法 (Hypertext Induced Topic Selection)

HITS 算法也是链接分析中非常基础且重要的算法，目前已被 Teoma 搜索引擎 (www.teoma.com) 作为链接分析算法在实际中使用。

2.4.1 Hub 页面与 Authority 页面

Hub 页面和 Authority 页面是 HITS 算法最基本的两个定义。所谓 Authority 页面，是指与某个领域或者某个话题相关的高质量网页。比如搜索引擎领域，Google 和百度首页即该领域的高质量网页；比如视频领域，优酷和土豆首页即该领域的高质量网页。所谓的 Hub 页面，指的是包含了很多指向高质量 Authority 页面链接的网页，比如 hao123 首页可以认为是一个典型的高质量 Hub 网页。

图 2-11 给出了一个 Hub 页面实例，这个网页是斯坦福大学计算语言学组维护的页面，这个网页收集了与统计自然语言处理相关的高质量资源，包括一些著名的开源软件包及语料

库等，并通过链接的方式指向这些资源页面。这个页面可以认为是“自然语言处理”这个领域的 Hub 页面，相应地，被这个页面指向的资源页面，大部分是高质量的 Authority 页面。



图 2-11 自然语言处理领域的 Hub 页面

HITS 算法的目的是通过一定的技术手段，在海量网页中找到与用户查询主题相关的高质量 Authority 页面和 Hub 页面，尤其是 Authority 页面，因为这些页面代表了能够满足用户查询的高质量内容，搜索引擎以此作为搜索结果返回给用户。

2.4.2 相互增强关系

很多算法都是建立在一些假设之上的，HITS 算法也不例外。HITS 算法隐含并利用了两个基本假设：

- 基本假设 1：一个好的 Authority 页面会被很多好的 Hub 页面指向。
- 基本假设 2：一个好的 Hub 页面会指向很多好的 Authority 页面。

到目前为止，无论是从 Hub 页面或者 Authority 页面的定义也好，还是从两个基本假设也好，都能看到一个模糊的描述，即“高质量”或者“好的”，那么什么是“好的”Hub 页面？什么是“好的”Authority 页面？两个基本假设给出了所谓“好”的定义。

基本假设 1 说明了什么是“好的”Authority 页面，即被很多好的 Hub 页面指向的页面是好的 Authority 页面，这里两个修饰语非常重要：“很多”和“好的”，所谓“很多”，即被越多的 Hub 页面指向越好，所谓“好的”，意味着指向该页面的 Hub 页面质量越高，则页面越好。这综合了指向本页面的所有 Hub 节点的数量和质量因素。

基本假设 2 则给出了什么是“好的”Hub 页面的说明，即指向很多好的 Authority 页面的网页是好的 Hub 页面。同样地，“很多”和“好的”两个修饰语很重要，所谓“很多”，即指向的 Authority 页面数量越多越好；所谓“好的”，即指向的 Authority 页面质量越高，则该页面越是好的 Hub

页面。这也综合考虑了该页面有链接指向的所有页面的数量和质量因素。

从以上两个基本假设可以推导出 Hub 页面和 Authority 页面之间的相互增强关系（参考图 2-12），即某个网页的 Hub 质量越高，则其链接指向的页面的 Authority 质量越好；反过来也是如此，一个网页的 Authority 质量越高，则那些有链接指向本网页的页面 Hub 质量越高。通过这种相互增强关系不断迭代计算，即可找出哪些页面是高质量的 Hub 页面，哪些页面是高质量的 Authority 页面。

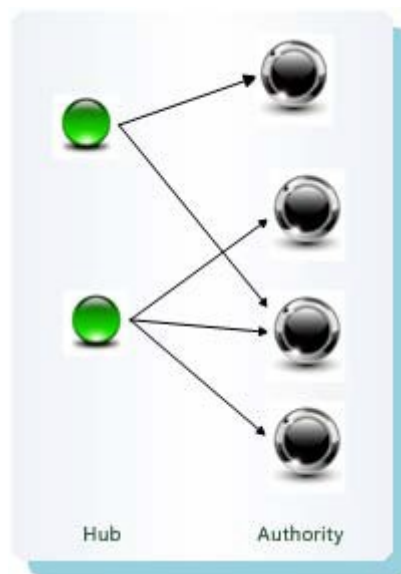


图 2-12 相互增强关系

2.4.3 HITS 算法

HITS 算法与 PageRank 算法一个显著的差异是：HITS 算法与用户输入的查询请求密切相关，而 PageRank 算法是与查询无关的全局算法。HITS 后续计算步骤都是在接收到用户查询后展开的，即是与查询相关的链接分析算法。

HITS 算法接收到了用户查询之后，将查询提交给某个现有的搜索引擎（或者是自己构造的检索系统），并在返回的搜索结果中，提取排名靠前的网页，得到一组与用户查询高度相关的初始网页集合，这个集合被称做根集（Root Set）。

在根集的基础上，HITS 算法对网页集合进行扩充（参考图 2-13），扩充原则是：凡是与根集内网页有直接链接指向关系的网页都被扩充进来，无论是有链接指向根集内页面也好，或者是根集页面有链接指向的页面也好，都被扩充进入扩展网页集合。HITS 算法在这个扩展网页集合内寻找好的 Hub 页面与好的 Authority 页面。

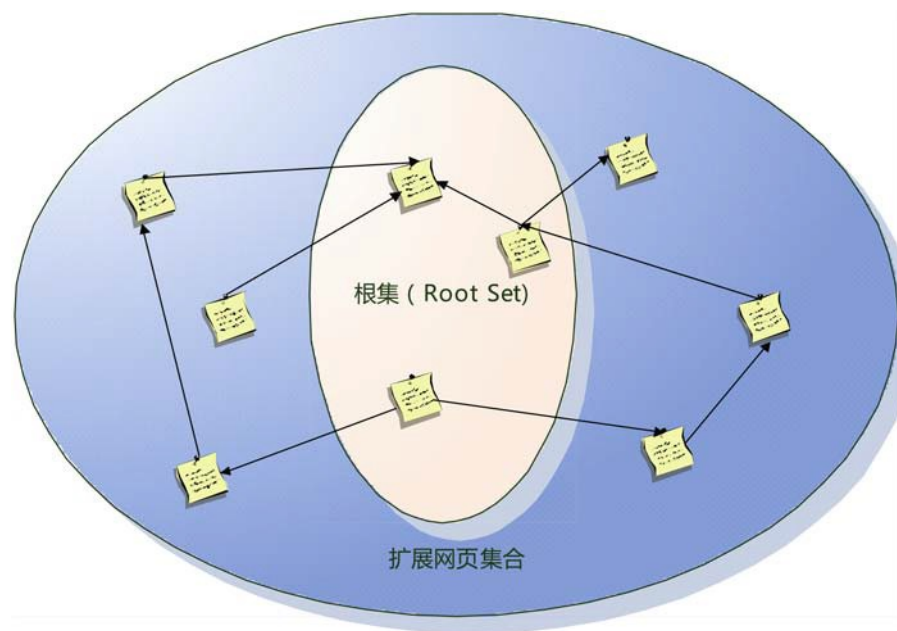


图 2-13 根集与扩展网页集合

对于扩展网页集合来说，我们并不知道哪些页面是好的 Hub 页面或者好的 Authority 页面，每个网页都有潜在的可能，所以对于每个页面都设立两个权值，分别来记载这个页面是好的 Hub 页面或者 Authority 页面的可能性。在初始情况下，在没有更多可利用信息前，每个页面的这两个权值都是相同的，可以都设置为 1。

之后，即可利用上面提到的两个基本假设，以及相互增强关系等原则进行多轮迭代计算，每轮迭代计算更新每个页面的两个权值，直到权值稳定不再发生明显的变化为止。

图 2-14 给出了迭代计算过程中，某个页面的 Hub 权值和 Authority 权值的更新方式。假设以 $A(i)$ 代表网页 i 的 Authority 权值，以 $H(i)$ 代表网页 i 的 Hub 权值。在如图 2-14 所示的例子中，扩展网页集合有 3 个网页有链接指向页面 1，同时页面 1 有 3 个链接指向其他页面。那么，网页 1 在此轮迭代中的 Authority 权值即为所有指向网页 1 页面的 Hub 权值之和；类似地，网页 1 的 Hub 分值即为所指页面的 Authority 权值之和。

扩展网页集合内其他页面也以类似的方式对两个权值进行更新，当每个页面的权值都获得了更新，则完成了一轮迭代计算，此时 HITS 算法会评估上一轮迭代计算中的权值和本轮迭代之后权值的差异，如果发现总体来说权值没有明显变化，说明系统已进入稳定状态，则可以结束计算。将页面根据 Authority 权值得分由高到低排序，取权值最高的若干页面作为响应用户查询的搜索结果输出。如果比较发现两轮计算总体权值差异较大，则继续进入下一轮迭代计算，直到整个系统权值稳定为止。

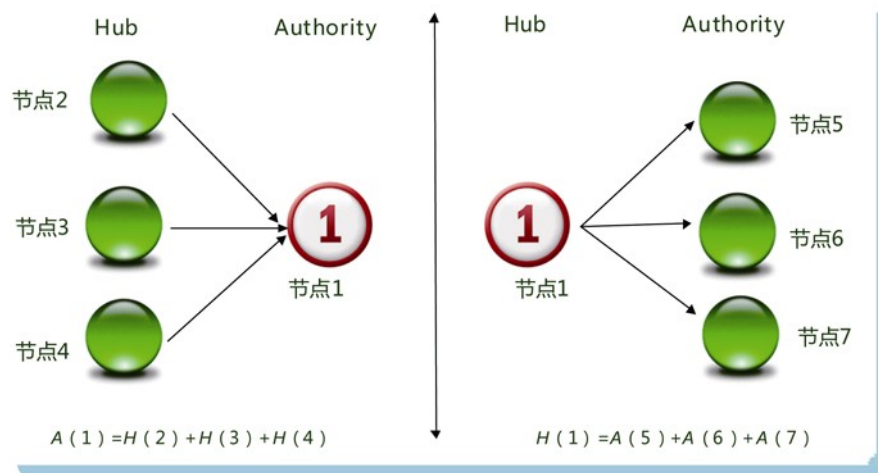


图 2-14 Hub 与 Authority 权值计算

2.4.4 HITS 算法存在的问题

HITS 算法整体而言是个效果很好的算法，目前不仅在搜索引擎领域应用，而且被自然语言处理及社交分析等很多其他计算机领域借鉴使用，并取得了很好的应用效果。尽管如此，最初版本的 HITS 算法仍然存在一些问题，而后续很多基于 HITS 算法的链接分析方法，也是立足于改进 HITS 算法存在的这些问题而提出的。

归纳起来，HITS 算法主要在以下几个方面存在不足。

- 计算效率较低

因为 HITS 算法是与查询相关的算法，所以必须在接收到用户查询后实时进行计算，而 HITS 算法本身需要进行很多轮迭代计算才能获得最终结果，这导致其计算效率较低，这是实际应用时必须慎重考虑的问题。

- 主题漂移问题

如果在扩展网页集合里包含部分与查询主题无关的页面，而且这些页面之间有较多的相互链接指向，那么使用 HITS 算法很可能会给予这些无关网页很高的排名，导致搜索结果发生主题漂移，这种现象被称为紧密链接社区现象（Tightly-Knit Community Effect）。

- 易被作弊者操纵结果

HITS 算法从机制上很容易被作弊者操纵，比如作弊者可以建立一个网页，页面内容增加很多指向高质量网页或者著名网站的网址，这就是一个很好的 Hub 页面，之后作弊者再将这个网页链接指向作弊网页，于是可以提升作弊网页的 Authority 得分。

- 结构不稳定

所谓结构不稳定，就是说在原有的扩展网页集合内，如果添加删除个别网页或者改变少数链接关系，则 HITS 算法的排名结果就会有非常大的改变。

2.4.5 HITS 算法与 PageRank 算法比较

HITS 算法和 PageRank 算法可以说是搜索引擎链接分析的两个最基础且最重要的算法。从以上对两个算法的介绍可以看出，两者无论是在基本概念模型，还是计算思路及技术实现细节都有很大的不同，下面对两者之间的差异进行逐一说明。

- HITS 算法是与用户输入的查询请求密切相关的，而 PageRank 与查询请求无关。所以，HITS 算法可以单独作为相似性计算评价标准，而 PageRank 必须结合内容相似性计算才可以用来对网页相关性进行评价。
- HITS 算法因为与用户查询密切相关，所以必须在接收到用户查询后进行实时计算，计算效率较低；而 PageRank 则可以在爬虫抓取完成后离线计算，在线直接使用计算结果，计算效率较高。
- HITS 算法的计算对象数量较少，只需计算扩展集合内网页之间的链接关系；而 PageRank 是全局性算法，对所有互联网页面节点进行处理。
- 从两者的计算效率和处理对象集合大小来比较，PageRank 更适合部署在服务器端，而 HITS 算法更适合部署在客户端。
- HITS 算法存在主题泛化问题，所以更适合处理具体的用户查询；而 PageRank 算法在处理宽泛的用户查询时更有优势。
- HITS 算法在计算时，对于每个页面需要计算两个分值，而 PageRank 算法只需计算一个分值即可；在搜索引擎领域，更重视 HITS 算法计算出的 Authority 权值，但是在很多应用 HITS 算法的其他领域，Hub 分值也有很重要的作用。
- 从链接反作弊的角度来说，PageRank 从机制上优于 HITS 算法，而 HITS 算法更易遭受链接作弊的影响。
- HITS 算法结构不稳定，当对扩展网页集合内链接关系做出很小改变，则对最终排名有很大影响；而 PageRank 算法相对 HITS 而言表现稳定，其根本原因在于 PageRank 计算时的远程跳转。

2.5 SALSA 算法

上一小节介绍了 PageRank 算法和 HITS 算法之间的异同之处，SALSA 算法的初衷希望能够结合两者的主要特点，既可以利用 HITS 算法与查询相关的特点，也可以采纳 PageRank 的随机游走模型，这是 SALSA 算法提出的背景。由此可见，SALSA 算法融合了 PageRank 和 HITS 算法的基本思想，从实际效果来说，很多实验数据表明，SALSA 的搜索效果也都优于前两个算法，是目前效果最好的链接分析算法之一。

从整体计算流程来说，可以将 SALSA 划分为两个大的阶段：首先是确定计算对象集合的阶段，这一阶段与 HITS 算法基本相同；第 2 个阶段是链接关系传播过程，在这一阶段则采纳了随机游走模型。

2.5.1 确定计算对象集合

PageRank 的计算对象是互联网所有网页，SALSA 算法与此不同，在本阶段，其与 HITS 算法思路大致相同，也是先得到扩展网页集合，之后将网页关系转换为方向二分图形式。

- 扩展网页集合

SALSA 算法在接收到用户查询请求后，利用现有搜索引擎或者检索系统，获得一批与用户查询在内容上高度相关的网页，以此作为根集。并在此基础上，将与根集内网页有直接链接关系的网页纳入，形成扩展网页集合（参考图 2-15）。之后会在扩展网页集合内根据一定的链接分析方法获得最终搜索结果排名。

- 转换为无向二分图

在获得了扩展网页集合之后，SALSA 根据集合内的网页链接关系，将网页集合转换为一个二分图。即将网页划分到两个子集合中，一个子集合是 Hub 集合，另一个子集合是 Authority 集合。划分网页节点属于哪个集合，则根据如下规则。

- 如果一个网页包含出链，这些出链指向扩展网页集合内其他节点，则这个网页可被归入 Hub 集合。
- 如果一个网页包含扩展网页集合内其他节点指向的入链，则可被归入 Authority 集合。

由以上规则可以看出，如果某个网页同时包含入链和出链，则可以同时归入两个集合。同时，Hub 集合内网页的出链组成了二分图内的边，根据以上法则，将扩展网页集合转

换为二分图。

图 2-15 和图 2-16 给出了一个示例，说明了这个转换过程。假设扩展网页集合如图 2-15 所示，由 6 个网页构成，其链接关系如图所示，同时为了便于说明，每个网页给予一个唯一编号。图 2-16 则是将图 2-15 中的网页集合转换为二分图的结果。以网页 6 为例，因为其有出链指向网页节点 3 和网页节点 5，所以可以放入 Hub 集合，也因为编号为 1、3、10 的网页节点有链接指向网页节点 6，所以也可以放入 Authority 集合中。网页节点 6 的两个出链保留，作为二分图的边，但是这里需要注意的是，在转换为二分图后，原先的有向边不再保留方向，转换为无向边，而 HITS 算法仍然保留为有向边，这点与 SALSA 略有不同。

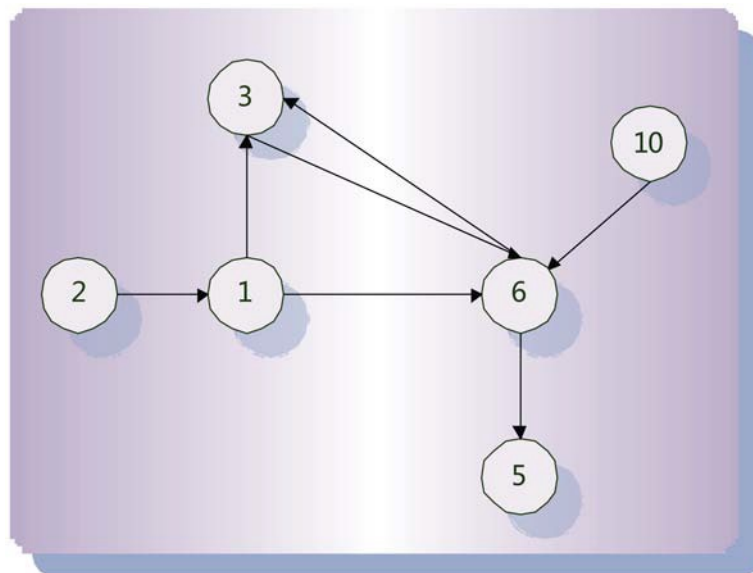


图 2-15 扩展网页集合示例

到这一步骤为止，除了 SALSA 算法将扩展网页集合转换为无向二分图，而 HITS 仍然是有向二分图外，其他步骤和流程，SALSA 算法与 HITS 算法完全相同，因此，SALSA 算法保证是与用户查询相关的链接分析算法。

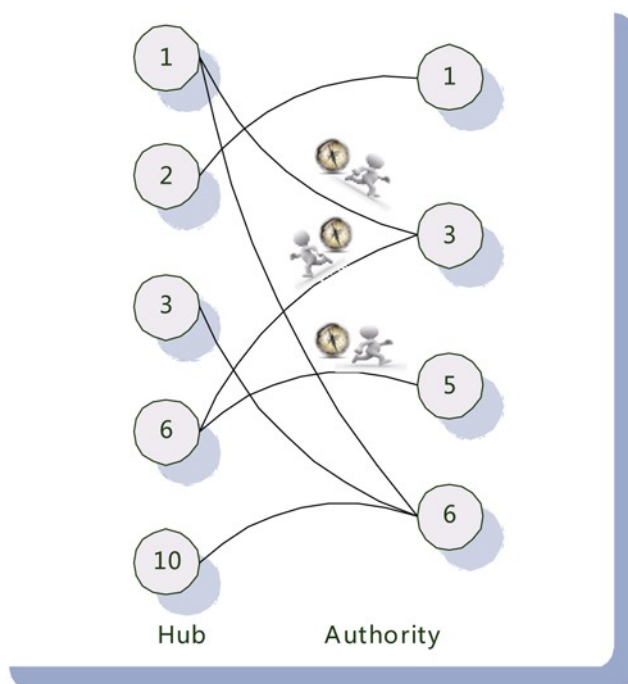


图 2-16 二分图

2.5.2 链接关系传播

在链接关系传播阶段，SALSA 算法放弃了 HITS 算法的 Hub 节点和 Authority 节点相互增强的假设，转而采纳 PageRank 的随机游走模型。

- 链接关系传播概念模型

如图 2-16 所示，假设存在某个浏览者，从某个子集中随机选择一个节点出发（为方便说明，图中所示为从 Hub 子集合的节点 1 出发，实际计算往往是从 Authority 子集合出发的），如果节点包含多条边，则以相等概率随机选择一条边，从 Hub 子集合跳到 Authority 集合内节点，图中所示为由节点 1 转移到节点 3 之后从 Authority 子集合再次跳回 Hub 子集合，即由节点 3 跳到节点 6。如此不断在两个子集之间转移，形成了 SALSA 自身的链接关系传播模式。

尽管看上去与 PageRank 的链接传播模式不同，但其实两者是一样的，关键点在于：其从某个节点跳到另外一个节点的时候，如果包含多个可供选择的链接，则以等概率随机选择一条路径，即在权值传播过程中，权值是被所有链接平均分配的。而 HITS 算法不同，HITS 算法属于权值广播模式，即将节点本身的权值完全传播给有链接指向的节点，并不根据链接多少进行分配。

SALSA 的上述权值传播模型与 HITS 模型关注重点不同，HITS 模型关注的是 Hub 和 Authority 之间的节点相互增强关系，而 SALSA 实际上关注的是 Hub-Hub 及 Authority-Authority 之间的节点关系，而另一个子集合节点只是充当中转桥梁的作用。所以，上述权值传播模型可以转化为两个相似的子模型，即 Hub 节点关系图和 Authority 节点关系图。

- Authority 节点关系图

图 2-17 是由 6-16 的二分图转化成的 Authority 节点关系图，Hub 节点关系图与此类似，两者转化过程是相似的，我们以 Authority 节点关系图为例来看如何从二分图转化为节点关系图。

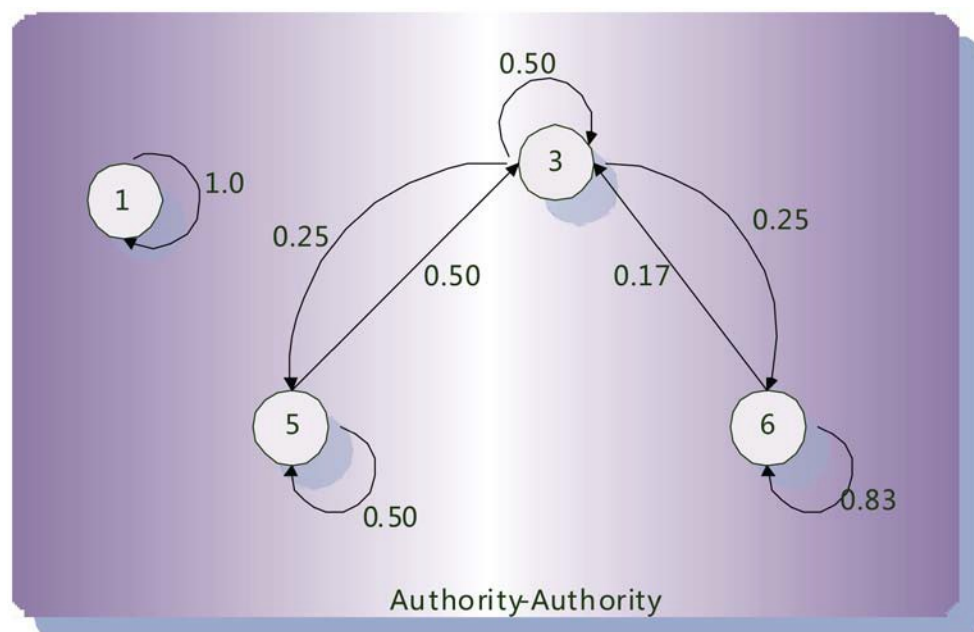


图 2-17 Authority 节点关系图

这里需要注意的是：Authority 集合内从某个节点 i 转移到另一个节点 j 的概率，与从节点 j 转移到节点 i 的概率是不同的，即非对称的，所以转换后的 Authority 节点关系图是个有向图，以此来表示其转移概率之间的差异。

对于图 2-17 这个 Authority 节点关系图来说，图中包含的节点就是二分图中属于 Authority 子集合的节点，关键在于节点之间的边如何建立及节点之间转移概率如何计算。

- 节点关系图中边的建立

之所以在 Authority 节点图中，节点 3 有边指向节点 5，是因为在二分图中，由节点 3 通过 Hub 子集合的节点 6 中转，可以通达节点 5，所以两者之间有边建立。

这里需要注意的是：在二分图中，对于 Authority 集合内的某个节点来说，一定可以通过 Hub 子集合的节点中转后再次返回本身，所以一定包含一条指向自身的有向边。节点 1 因为只有中转节点 2 使得其返回 Authority 子集合中的自身节点，所以只有指向自身的一条边，和其他节点没有边联系，所以例子中的 Authority 节点关系图由两个连通子图构成，一个只有节点 1，另外一个连通子图由剩余几个节点构成。

- 节点之间的转移概率

至于为何 Authority 节点关系图中，节点 3 到节点 5 的转移概率为 0.25，是因为前面介绍过，SALSA 的权值传播模型遵循随机游走模型。在图 2-16 的二分图中，从节点 3 转移到节点 5 的过程中，节点 3 有两条边可做选择来跳转到 Hub 子集合，所以每条边的选择概率为 $1/2$ ，可以选择其中一条边到达节点 6，同样，从节点 6 跳回到 Authority 子集合时，节点 6 也有两条边可选，选中每条边的概率为 $1/2$ 。所以从节点 3 出发，经由节点 6 跳转到节点 5 的概率为两条边权值的乘积，即为 $1/4$ 。

对于指向自身的有向边，其权重计算过程是类似的，我们仍然以节点 3 为例，指向自身的有向边代表从 Authority 子集合中的节点 3 出发，经由 Hub 子集合的节点再次返回节点 3 的概率。从图 2-16 的二分图可以看出，完成这个过程有两条路径可走，一条是从节点 3 到节点 1 返回；另外一条是从节点 3 经由节点 6 后返回；每一条路径的概率与上面所述计算方法一样，因为两条路径各自的概率为 0.25，所以节点 3 返回自身的概率为两条路径概率之和，即为 0.5。图中其他边的转移概率计算方式也是类此。

建立好 Authority 节点关系图后，即可在图上利用随机游走模型来计算每个节点的 Authority 权值。在实际计算过程中，SALSA 将搜索结果排序问题进一步转换为求 Authority 节点矩阵的主秩问题，矩阵的主秩即为每个节点的相应 Authority 得分，按照 Authority 得分由高到低排列，即可得到最终的搜索排序结果。

2.5.3 Authority 权值计算

经过数学推导，可以得出 SALSA 与求矩阵主秩等价的 Authority 权值计算公式。图 2-18 中的示意图表明了 SALSA 算法中某个网页节点的 Authority 权值是如何计算的。如图右上角公式所示，决定某个网页 i 的 Authority 权值涉及 4 个因子。

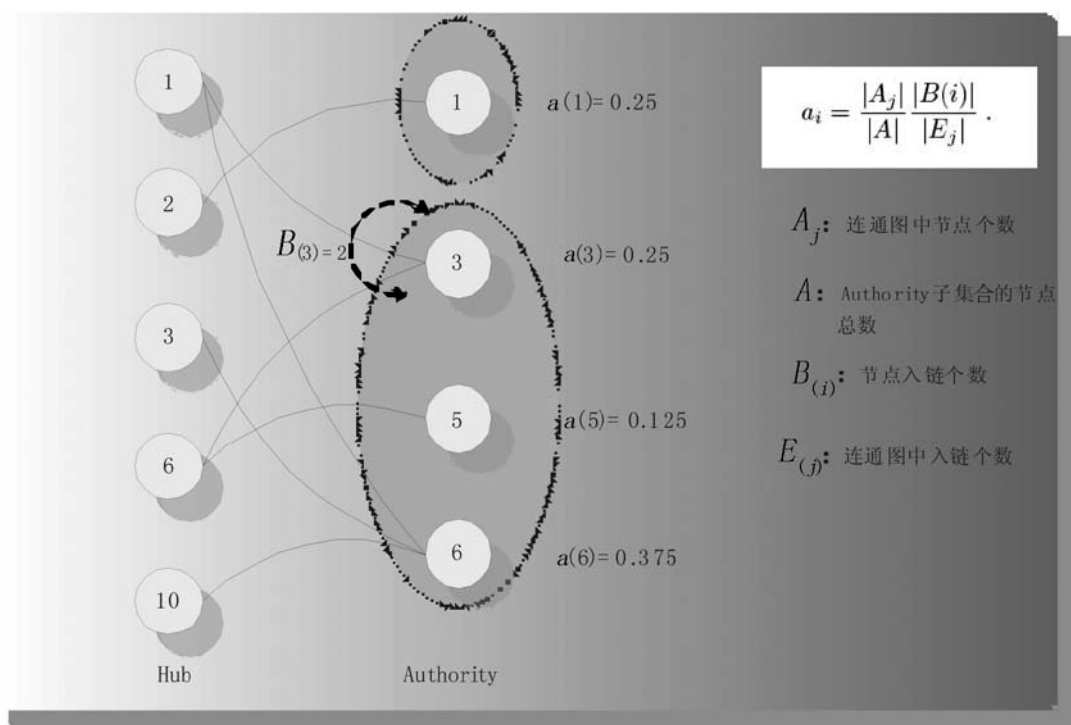


图 2-18 SALSA 节点权值计算公式

- Authority 子集合中包含的节点总数 $|A|$ 。其实这个因子对于 Authority 集合中任意节点来说都是相同的，所以对于最终的根据节点 Authority 权值进行的排序没有影响，只是起到保证权值得分在 0 到 1 之间，能够以概率形式表示权值的作用。
- 网页 i 所在连通图中包含的节点个数 $|A_j|$ 。网页所在的连通图包含的节点个数越多，则网页的 Authority 权值越大。
- 网页 i 所在连通图中包含的入链总数 $|E_j|$ 。网页所在的连通图包含的入链总数越少，则网页的 Authority 权值越大。
- 网页 i 的入链个数 $|B_i|$ 。节点入链越多，则 Authority 权值越大，这个因子是唯一一个和节点本身属性相关的。由此可见，SALSA 权值计算和节点入链个数成正比。

之前图 2-17 的“Authority 节点关系图”由两个连通子图组成，一个由唯一的节点 1 构成，另外一个由节点 3、5、6 3 个节点构成，两个连通子图在图 2-18 中也被分别圈出。

我们以节点 3 为例，看其对应的 4 个计算因素取值。

- Authority 子集共包括 4 个节点。
- 节点 3 所在连通图包含 3 个节点。
- 节点 3 所在连通图共有 6 个入链。

- 节点 3 的入链个数为 2。

所以，节点 3 的 Authority 权值为： $(3/4) \times (2/6) = 0.25$ 。其他节点权值的计算过程与此类似。SALSA 根据节点的 Authority 权值由高到低排序输出，即为搜索结果。

由上述权值计算公式可以推出：如果整个 Authority 子集合所有节点形成一个完整的连通图，那么在计算 Authority 权值的过程中，对于任意两个节点，4 个因子中除了节点入链个数外，其他 3 个因子总是相同的，即只有入链个数起作用，此时，SALSA 算法退化为根据节点入链个数决定排序顺序的算法。

从 SALSA 计算 Authority 得分过程中可看出，SALSA 算法不需要像 HITS 算法一样进行不断的迭代计算，所以从计算效率角度看要快于 HITS 算法。另外，SALSA 算法解决了 HITS 算法的计算结果主题漂移问题，所以搜索质量也优于 HITS 算法。SALSA 算法是目前效果最好的链接算法之一。

2.6 主题敏感 PageRank (Topic Sensitive PageRank)

主题敏感 PageRank 是 PageRank 算法的改进版本，该算法已被 Google 使用在个性化搜索服务中。

2.6.1 主题敏感 PageRank 与 PageRank 的差异

PageRank 算法基本遵循前面章节提到的随机游走模型，即用户在浏览某个网页时，如果希望跳转到其他页面，则随机选择本网页包含的某个链接，进入另一个页面。主题敏感 PageRank 则对该概念模型做出改进，引入了更符合现实的假设。一般来说用户会对某些领域感兴趣，同时当浏览某个页面时，这个页面也是与某个主题相关的（比如体育报道或者娱乐新闻），所以当用户看完当前页面，希望跳转时，更倾向于点击和当前页面主题类似的链接，即主题敏感 PageRank 是将用户兴趣、页面主题及链接所指向网页与当前网页主题的相似程度综合考虑而建立的模型。很明显，这更符合真实用户的浏览过程。

PageRank 是全局性的网页重要性衡量标准，每个网页会根据链接情况，被赋予一个唯一的 PageRank 分值。主题敏感 PageRank 在此点有所不同，该算法引入了 16 种主题类型，对于某个网页来说，对应某个主题类型都有相应的 PageRank 分值，即每个网页会被赋予 16 个主题相关 PageRank 分值。

在接收到用户查询后，两个算法在处理方式上也有较大差异。PageRank 算法与查询无关，

只能作为相似度计算的一个计算因子体现作用，无法独立使用。而主题敏感 PageRank 是查询相关的，可单独作为相似度计算公式使用。而且，在接收到用户查询后，主题敏感 PageRank 还需要利用分类器，计算该查询隶属于事先定义好的 16 个主题的隶属度，并在相似度计算时的排序公式中利用此信息。

2.6.2 主题敏感 PageRank 计算流程

主题敏感 PageRank 计算主要由两个步骤构成，第 1 步是离线的分类主题 PageRank 数值计算；第 2 步是在线利用算好的主题 PageRank 分值，来评估网页和用户查询的相似度，以按照相似度排序提供给用户搜索结果。下面以具体示例来了解主题敏感 PageRank 的计算流程。

● 分类主题 PageRank 计算

主题敏感 PageRank 参考 ODP 网站 (www.dmoz.org)，定义了 16 个大的主题类别，包括体育、商业、科技等。ODP (Open Directory Project) 是人工整理的多层级网页分类导航站点 (参见图 2-19)，在顶级的 16 个大分类下还有更细致的小粒度分类结构，在最底层目录下，人工收集了符合该目录主题的精选高质量网页地址，以供互联网用户导航寻址。主题敏感 PageRank 采用了 ODP 最高级别的 16 个分类类别作为事先定义的主题类型。

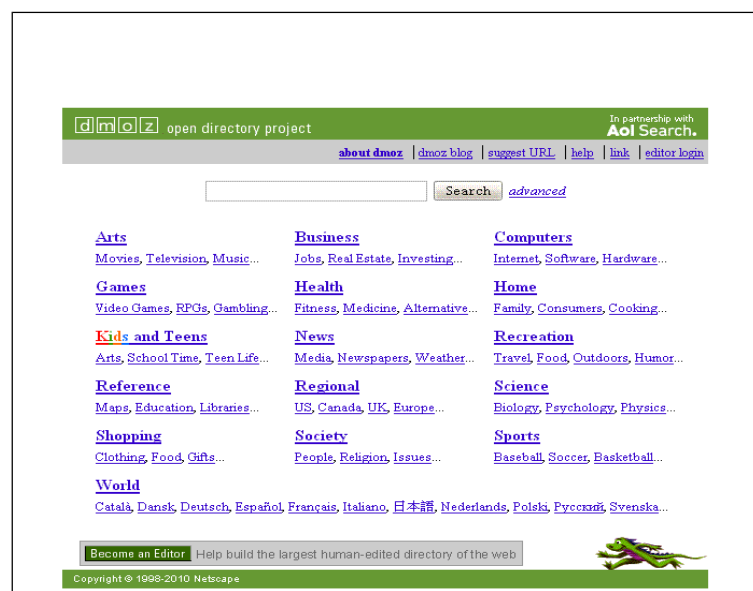


图 2-19 ODP 首页

主题敏感 PageRank 对 16 个类别的主题，依次计算该类别的 PageRank 分值，图 2-20 显示了其计算流程和基本思路，为了简化说明，示意图只表现出了 3 个分类类别。在计算某个类别的 PageRank 分值时，将所有网页划分为两个集合，一个集合是 ODP 对应分类

主题下所包括的所有网页，即人工精选的高质量网页，可以称之为集合 S ，剩下的网页放入另一个集合内，可称之为集合 T 。在计算 PageRank 时，由于集合 S 内的网页能够很好地表征分类主题，所以赋予较大的跳转概率值。通过这种设定，集合 S 内的网页根据链接关系向集合 T 中网页传递权值，因为直接有链接指向的往往主题类似，这样就将与该分类主题内容相似的网页赋予较高的 PageRank 值，而无关的网页则赋予较低权重的 PageRank 分值，以此方式达到对网页所包含主题的判断。

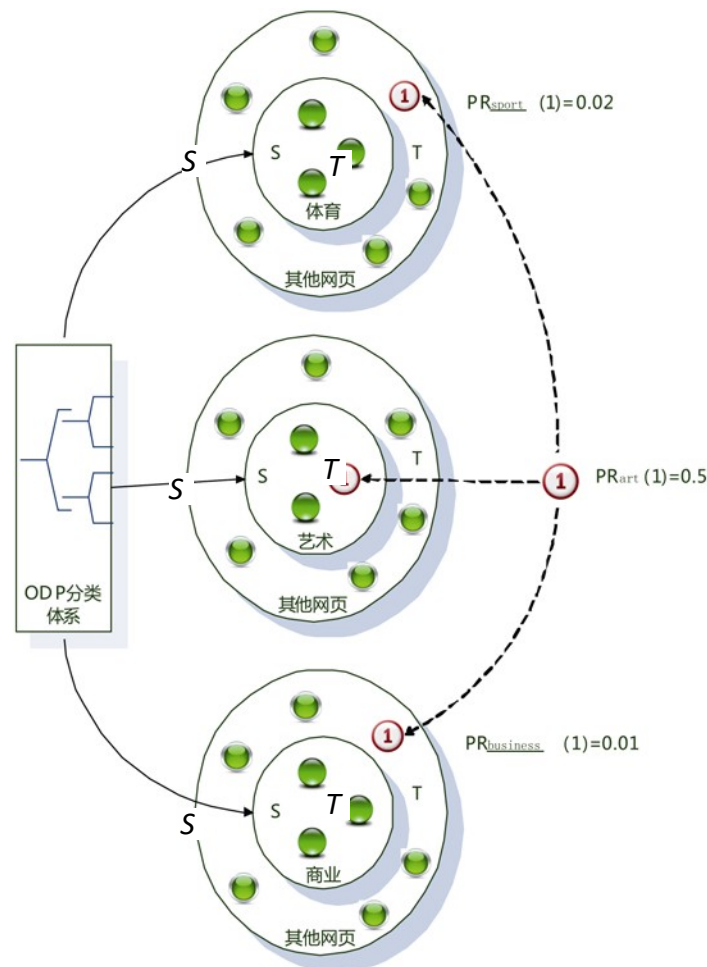


图 2-20 网页的分类主题 PageRank 计算

回到图 2-20，假设有个编号为 1 的网页，其被列到 ODP 目录中的艺术类别中，在对艺术类别进行 PageRank 计算时，1 号网页在集合 S 内，计算结束后，该网页获得的 PageRank 分值为 0.5。当计算体育和商业类别的主题 PageRank 分值时，1 号网页在集合 T 中，获得了相应的集合 S 中网页传递的权值，分别为 0.02 和 0.01。在所有类别计算结束后，1 号网页获得了 3 个不同主题对应的 PageRank 分值，组成一个主题 PageRank 向量。通过类似的方式，互联网内任意网页也可以获得相应的主题相关 PageRank 向量。通过以上过程可以看出，主题相关的 PageRank 分值向量其实代表了某个网页所讲述内容所属类

别的概率。

注意：在上述计算主题 PageRank 过程中，从集合 S 和集合 T 的划分及其权值传播方式中可以看出，该步骤计算过程也符合子集传播模型。但是由于本算法主框架及其出发点都是为了改进 PageRank，所以将其归入随机游走模型的衍生算法类别中。

- 在线相似度计算

图 2-21 给出了主题敏感 PageRank 在线计算用户查询与网页相似度的示意图。假设用户输入了查询请求“乔丹”，搜索系统首先利用用户查询分类器对查询进行分类，计算用户查询隶属于定义好的各个类别的概率分别是多少，在我们给出的例子里，“乔丹”隶属于体育类别的概率为 0.6，娱乐类别的概率为 0.1，商业类别的概率为 0.3。

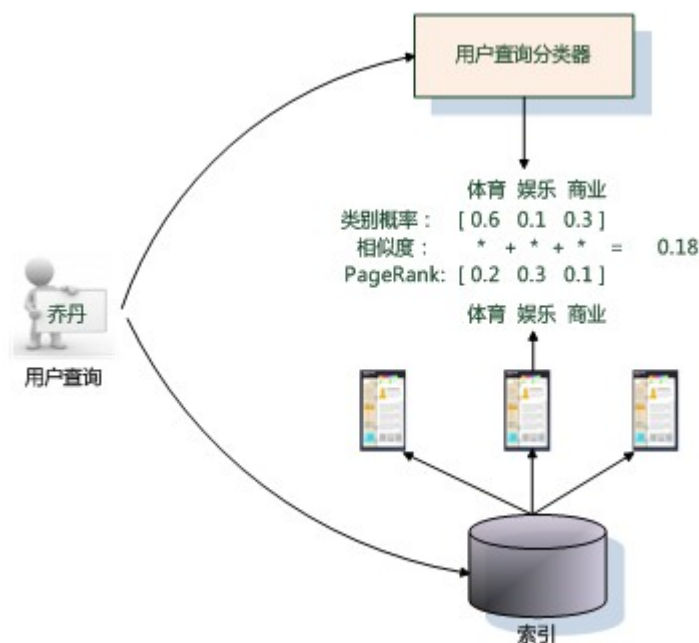


图 2-21 在线相似度计算

在进行上述用户查询分类计算的同时，搜索系统读取索引，找出包含了用户查询“乔丹”的所有网页，并获得上一步骤离线计算好的各个分类主题 PageRank 值，在图 2-21 的例子里，假设某个网页 A 的各个主题 PageRank 值分别为体育 0.2、娱乐 0.3 及商业 0.1。得到用户查询的类别向量和某个网页的主题 PageRank 向量后，即可计算这个网页和查询的相似度。通过计算两个向量的乘积就可以得出两者之间的相关性。在如图 2-21 所示的例子里，网页 A 和用户查询“乔丹”的相似度为：

$$\text{Sim}(\text{“乔丹”}, A) = 0.6 \times 0.2 + 0.1 \times 0.3 + 0.3 \times 0.1 = 0.18$$

对包含“乔丹”这个关键词的网页，都根据以上方法计算，得出其与用户查询的相似度后，

就可以按照相似度由高到低排序输出，作为本次搜索的搜索结果返回给用户。

2.6.3 利用主题敏感 PageRank 构造个性化搜索

以上内容介绍的是主题敏感 PageRank 的基本思想和计算流程，从其内在机制来说，这个算法非常适合作为个性化搜索的技术方案。

在如图 2-21 所示的例子中，计算相似度使用的只有用户当前输入的查询词“乔丹”，如果能够对此进行扩展，即不仅使用当前查询词，也考虑利用用户过去的搜索记录等个性化信息。比如用户之前搜索过“耐克”，则可以推断用户输入“乔丹”是想购买运动服饰，而如果之前搜索过“姚明”，则很可能用户希望获得体育方面的信息。通过这种方式，可以将用户的个性化信息和当前查询相融合来构造搜索系统，以此达到个性化搜索的目的，更精准地提供搜索服务。

2.7 Hilltop 算法

Hilltop 算法是 Torono 大学研发的链接分析算法，在 2003 年被 Google 公司收购，而 Google 在之后的排序算法大改版中引入了 Hilltop 算法。

Hilltop 算法融合了 HITS 和 PageRank 两个算法的基本思想。一方面，Hilltop 是与用户查询请求相关的链接分析算法，吸收了 HITS 算法根据用户查询获得高质量相关网页子集的思想，符合子集传播模型，是该模型的一个具体实例；同时，在权值传播过程中，Hilltop 算法也采纳了 PageRank 的基本指导思想，即通过页面入链的数量和质量来确定搜索结果的排序权重。

2.7.1 Hilltop 算法的一些基本定义

非从属组织页面（Non-affiliated Pages）是 Hilltop 算法的一个很重要的定义。要了解什么是非从属组织页面，先要搞明白什么是从属组织网站，所谓的从属组织网站，即不同的网站属于同一机构或者其拥有者有密切关联。具体而言，满足如下任意一条判断规则的网站会被认为是从属网站。

- 条件 1：主机 IP 地址的前 3 个子网段相同，比如：IP 地址分别为 159.226.138.127 和 159.226.138.234 的两个网站会被认为是从属网站。
- 条件 2：如果网站域名中的主域名相同，比如 www.ibm.com 和 www.ibm.com.cn 会被认

为是从属组织网站。

非从属组织页面的含义是：如果两个页面不属于从属网站，则为非从属组织页面。图 2-22 是相关示意图，从图中可以看出，页面 2 和页面 3 同属于 IBM 的网页，所以是从属组织页面，而页面 1 和页面 5、页面 3 和页面 6 都是非从属组织页面。由此也可看出，非从属组织页面代表的是页面的一种关系，单个页面是无所谓从属或者非从属组织页面的。

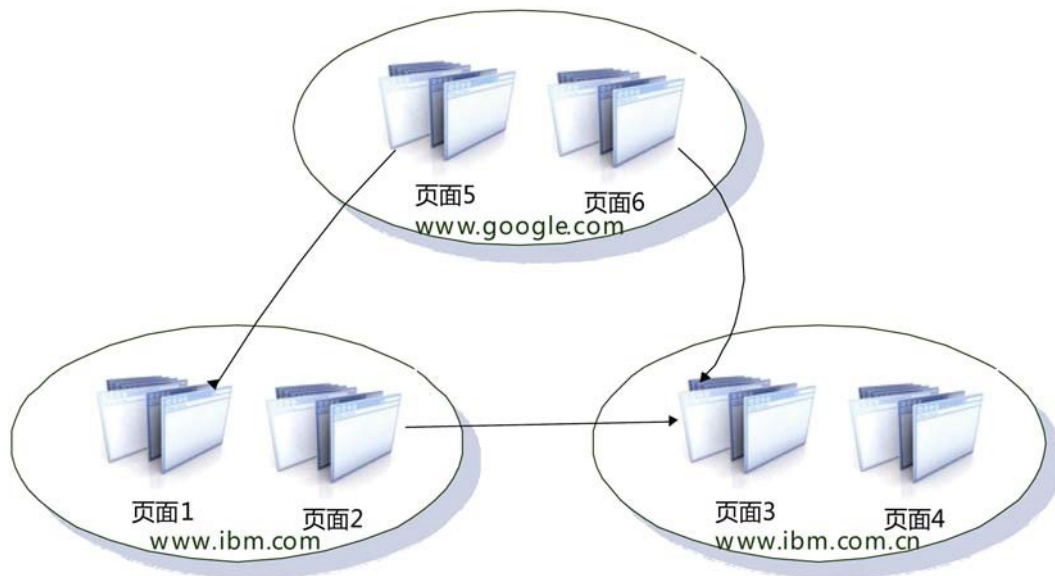


图 2-22 从属组织页面与非从属组织页面

专家页面 (Expert Sources) 是 Hilltop 算法的另外一个重要定义。所谓专家页面，即与某个主题相关的高质量页面，同时需要满足以下要求：这些页面的链接所指向的页面相互之间都是非从属组织页面，且这些被指向的页面大多数是与专家页面主题相近的。

Hilltop 算法将互联网页面划分为两类子集合，最重要的子集合是由专家页面构成的互联网页面子集，不在这个子集合里的剩下的互联网页面作为另外一个集合，这个集合称做目标页面集合 (Target Web Servers)。

2.7.2 Hilltop 算法

图 2-23 是 Hilltop 算法的整体流程示意。首先从海量的互联网网页中通过一定规则筛选出专家页面子集合，并单独为这个页面集合建立索引。Hilltop 在接收到用户发出的某个查询请求时，首先根据用户查询的主题，从专家页面子集合中找出部分相关性最强的专家页面，并对每个专家页面计算相关性得分，然后根据目标页面和这些专家页面的链接关系来对目标页面进行排序。基本思路遵循 PageRank 算法的链接数量假设和质量原则，将专家页面的得分通过链接关系传递给目标页面，并以此分数作为目标页面与用户查询相关性的排序得分。最后

系统整合相关专家页面和得分较高的目标页面作为搜索结果返回给用户。

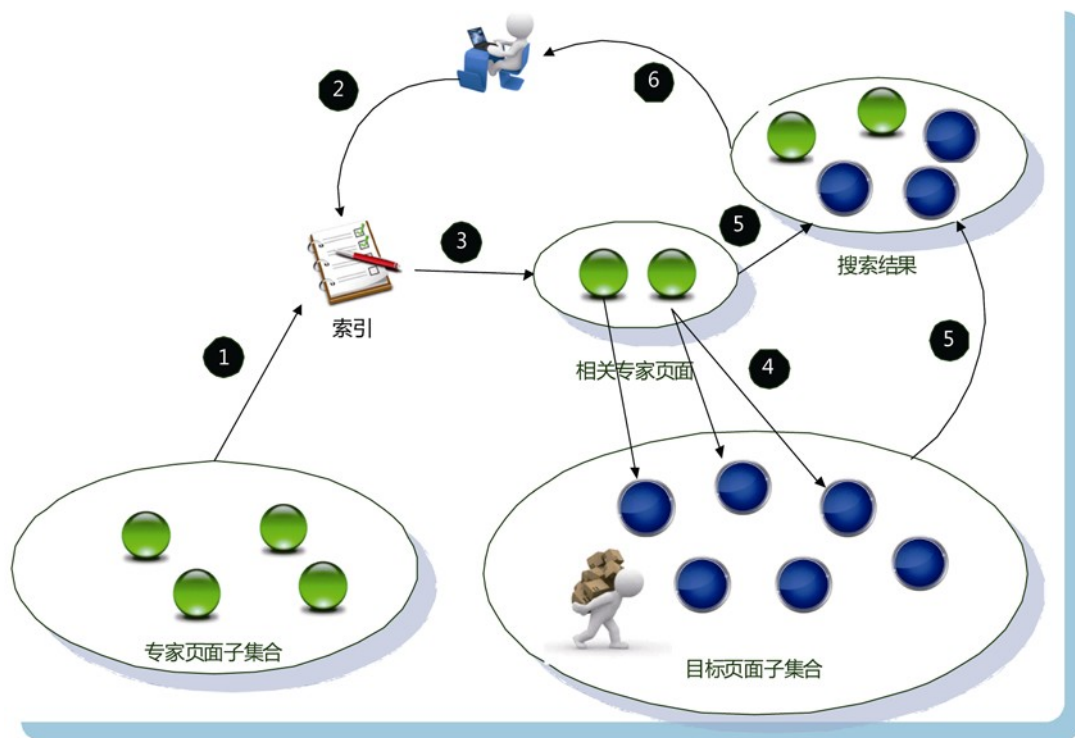


图 2-23 Hilltop 算法流程

若在上述过程中，Hilltop 算法无法得到一个足够大的专家页面集合，则返回搜索结果为空。由此可以看出，Hilltop 算法更注重搜索结果的精度和准确性，不太考虑搜索结果是否足够多或者对大多数用户查询是否都有相应的搜索结果，所以很多用户发出的查询的搜索结果为空。这意味着 Hilltop 算法可以与某个排序算法相结合，以提高排序准确性，但并不适合作为一个独立的网页排序算法来使用。

从上述整体流程描述可看出，Hilltop 算法主要包含两个步骤：专家页面搜索及目标页面排序。

● 步骤一：专家页面搜索

Hilltop 算法从 1 亿 4 千万网页中，通过计算筛选出 250 万规模的互联网页面作为专家页面集合。专家页面的选择标准相对宽松，同时满足以下两个条件的页面即可进入专家页面集合。

- 条件 1：页面至少包含 k 个出链，这里的数量 k 可人为指定。
- 条件 2： k 个出链指向的所有页面相互之间的关系都符合非从属组织页面的要求。

当然，在此基础上，可以设定更严格的筛选条件，比如要求这些专家页面所包含链接指向的页面中，大部分涉及的主题和专家页面的主题必须是一致或近似的。

根据以上条件筛选出专家页面后，即可对专家页面单独建索引，在此过程中，索引系统只对页面中的关键片段（Key Phrase）进行索引。所谓关键片段，在 Hilltop 算法里包含了网页的 3 类信息：网页标题、H1 标签内文字和 URL 锚文字。

网页的关键片段可以支配（Qualify）某个区域内包含的所有链接，支配关系代表了一种管辖范围，不同的关键片段支配链接的区域范围不同，具体而言，页面标题可以支配页面内所有出现的链接，H1 标签可以支配包围在<H1>和</H1>内的所有链接，而 URL 锚文字只能支配本身唯一的链接。图 2-24 给出了关键片段对链接支配关系的示意图，在以“奥巴马访问中国”为标题的网页页面中，标题支配了所有这个页面出现的链接，而 H1 标签的管辖范围仅限于标签范围内出现的两个链接，对于锚文字“中国领导人”来说，其唯一能够支配的就是本身的这个链接。之所以定义这种支配关系，对于第 2 阶段将专家页面的分值传递到目标页面时候会起作用。

系统接收到用户查询 Q，假设用户查询包含了多个单词，Hilltop 如何对专家页面进行打分呢？对专家页面进行打分主要参考以下 3 类信息。

- 关键片段包含了多少查询词，包含的查询词越多，则分值越高，如果不包含任何查询词，则该关键片段不计分。
- 关键片段本身的类型信息，网页标题权值最高，H1 标签次之，再次是链接锚文字。
- 用户查询和关键片段的失配率，即关键片段中不属于查询词的单词个数占关键片段总单词个数，这个值越小越好，越大则得分衰减越多。

Hilltop 综合考虑以上 3 类因素，拟合出打分函数来对专家页面是否与用户查询相关进行打分，选出相关性分值足够高的专家页面，以进行下一步骤操作，即对目标页面进行相关性计算。

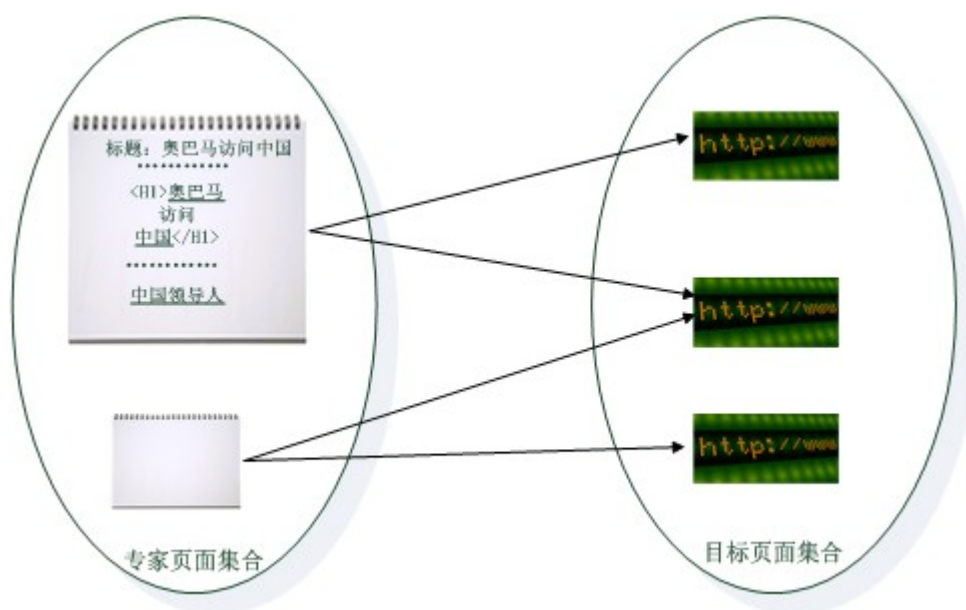


图 2-24 关键片段链接支配关系

● 步骤二：目标页面排序

Hilltop 算法包含一个基本假设，即认为一个目标页面如果是满足用户查询的高质量搜索结果，其充分必要条件是目标页面有高质量专家页面链接指向。然而，这个假设并不总是成立，比如有的专家页面的链接所指向的目标页面可能与用户查询并非密切相关。所以，Hilltop 算法在这个阶段需要对专家页面的出链仔细进行甄别，以保证选出那些和查询密切相关的目标页面。

Hilltop 在本阶段是基于专家页面和目标页面之间的链接关系来进行的，在此基础上，将专家页面的得分传递给有链接关系的目标页面。传递分值之前，首先需要对链接关系进行整理，能够获得专家页面分值的目标页面需要满足以下两点要求：

- 条件 1：至少需要两个专家页面有链接指向目标页面，而且这两个专家页面不能是从属组织页面，即不能来自同一网站或相关网站。如果是从属组织页面，则只能保留一个链接，抛弃权值低的那个链接。
- 条件 2：专家页面和所指向的目标页面也需要符合一定要求，即这两个页面也不能是从属组织页面。

在步骤一，给定用户查询，Hilltop 算法已经获得相关的专家页面及其与查询的相关度得分，在此基础上，如何对目标页面的相关性打分？上面列出的条件1指出，能够获得传递分值的目标页面一定有多个专家页面链接指向，所以目标页面所获得的总传播分值是每个有链接指向的专家页面所传递分值之和。而计算其中某个专家页面传递给目标页面

权值的时候是这么计算的：

- 找到专家页面中那些能够支配目标页面的关键片段集合 S 。
- 统计 S 中包含用户查询词的关键片段个数 T ， T 越大传递的权值越大。
- 专家页面传递给目标页面的分值为： $E \times T$ ， E 为专家页面本身在第一阶段计算得到的相关得分， T 为 b 步骤计算的分值。

我们以图 2-25 的具体例子来说明。假设专家页面集合内存在一个网页 P ，其标题为“奥巴马访问中国”，网页内容由一段<H1>标签文字和另一个单独的链接锚文字组成。该页面包含 3 个出链，其中两个指向目标页面集合中的网页 www.china.org，另外一个指向网页 www.obama.org。出链对应的锚文字分别为“奥巴马”、“中国”和“中国领导人”。

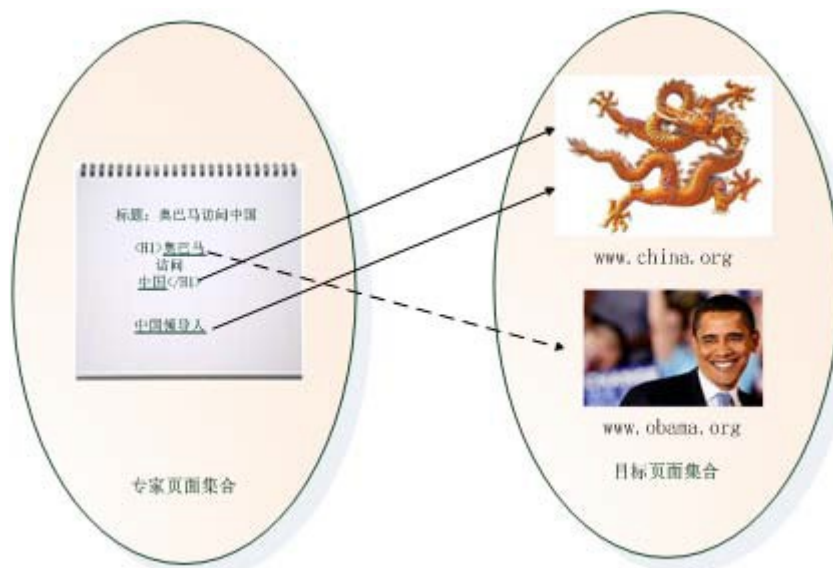


图 2-25 Hilltop 算法分值传递

从图示的链接关系可以看出，网页 P 中能够支配 www.china.org 这个目标页面的关键片段集合包括： $\{\text{中国领导人，中国，}<\text{H1}>\text{奥巴马访问中国}</\text{H1}>，\text{标题：奥巴马访问中国}\}$ ，而能够支配 www.obama.org 目标页面的关键片段集合包括： $\{\text{奥巴马，}<\text{H1}>\text{奥巴马访问中国}</\text{H1}>，\text{标题：奥巴马访问中国}\}$ 。

接下来我们分析专家页面 P 在接收到查询时，是怎样将分值传递给与其有链接关系的目标页面的。假设系统接收到的查询请求为“奥巴马”，在接收到查询后，系统首先根据上述章节所述，找出专家页面并给予分值，而网页 P 作为专家页面中的一个页面，并获得了相应的分值 s ，我们重点关注分值传播步骤。

对于查询“奥巴马”来说，网页 P 中包含这个查询词的关键片段集合为： $\{\text{奥巴马，}<\text{H1}>$

奥巴马访问中国</H1>，标题：奥巴马访问中国}，如上所述，这 3 个关键片段都能够支配 www.obama.org 页面，所以网页 P 传递给 www.obamba.org 的分值为 $S \times 3$ 。而对于目标页面 www.china.org 来说，这 3 个关键片段中只有{<H1>奥巴马访问中国</H1>，标题：奥巴马访问中国}这两个能够支配目标页面，所以网页 P 传递给 www.china.org 的分值为 $S \times 2$ 。

对于包含多个查询词的用户请求，则每个查询词单独如上计算，将多个查询词的传递分值累加即可。

Hilltop 算法存在与 HITS 算法类似的计算效率问题，因为根据查询主题从专家页面集合中选取主题相关的页面子集也是在线运行的，这与前面提到的 HITS 算法一样会影响查询响应时间。随着专家页面集合的增大，算法的可扩展性存在不足之处。

2.8 其他改进算法

上述章节详述了 5 个非常重要的链接分析算法，在此基础上，学术界提出了很多改进方法，本节简述其中几个相对重要方法的基本思路和本质思想。

2.8.1 智能游走模型 (Intelligent Surfer Model)

智能游走模型是对 PageRank 算法的改进，算法提出者认为 PageRank 所遵循的随机游走模型不符合真实情况，浏览者在浏览一个页面并选择下一步点击对象时，并非随机的，而是会对链接进行甄别，如果某个链接的网页内容能够表达浏览者向搜索引擎发出的查询词主题，则用户选择点击这个链接的概率会更大。智能游走模型即是对这种情况建立模型，会判断网页包含的链接所指向的网页内容和用户查询的相关性，以此来改善链接分析效果。

智能游走模型考虑到了网页内容和用户输入查询词的相关性，所以是查询相关的链接分析算法。但是需要在线计算某个查询和网页内容的相关性，所以计算速度较慢是其重要缺陷。

2.8.2 偏置游走模型 (Biased Surfer Model)

偏置游走模型也是针对 PageRank 的随机游走模型的改进，其基本思路 and 智能游走模型很接近，不同点在于：智能游走模型考虑的是网页内容和用户查询的相关性，而偏置游走模型考虑的是链接指向的网页内容和当前浏览网页内容之间的相似性，即认为如果链接指向网页内容和当前浏览网页内容越相似，则用户越可能点击这个链接。

偏置游走模型由于没有考虑查询,而只需要计算网页内容之间的相似性,可以离线进行计算,并在线使用,所以计算速度要优于智能游走模型。

2.8.3 PHITS 算法 (Probability Analogy of HITS)

PHITS 算法是对 HITS 算法的直接改进。HITS 算法在通过链接进行权值传递时,会将 Hub 或者 Authority 节点的权值全部传递给所有相连链接,而 PHITS 算法认为不同链接其传递权值的能力应该是不同的,这是其基本思想。基于此,PHITS 需要计算两个页面 S 和 T 之间链接的连接强度。

PHITS 算法在页面 S 和页面 T 之间引入主题隐变量,而两个页面之间链接的强度取决于两个页面和这个隐变量的主题耦合关系,所以可以将这个算法看做 PLSI 算法的链接分析版本。虽然看上去复杂,但是 PHITS 算法的本质思想和偏置游走模型是一样的,即考虑的其实是页面 S 和页面 T 之间的内容相似性,内容越相似其链接强度越强,传递权值的能力越强。PHITS 算法和直接计算页面内容相似性的区别,仅仅是加入一个隐藏主题层,通过网页和隐藏层的关系来计算相似性。

2.8.4 BFS 算法 (Backward Forward Step)

BFS 算法作为 SALSA 算法的改进提出,而其本质既是对 SALSA 算法的扩展,也是对 HITS 算法的限制。SALSA 算法在计算网页的 Authority 分值时,仅仅考虑了通过 Hub 节点能够直接建立联系的网页之间的关系,而 HITS 算法考虑的是链接图的整个结构,任意两个网页 S 和 T,如果网页 S 能够通过中间节点链接到网页 T,则网页 S 就对网页 T 有影响,而影响的大小取决于从 S 到 T 的不同路径走法的个数,与 S 到 T 的距离远近没有关系。

BFS 则是对两者的一个折中考虑,解除了 SALSA 算法只允许直接相邻网页才能有影响的限制,只要网页 S 可以通达 T,即可对网页 T 施加影响,但是又对 HITS 的网页影响方式做了限制,如果网页 S 距离网页 T 距离越远,那么网页 S 的影响就随着距离增大而呈指数衰减,即距离越远,影响越小。通过这种折中,来实现对 SALSA 算法的扩充和对 HITS 算法的限制。

本章提要

- 链接分析在搜索引擎搜索结果排序中起到非常重要的作用。
- 绝大部分链接分析算法建立在随机游走模型和子集传播模型基础之上。

- PageRank 和 HITS 算法是最重要且基础的两种链接分析算法，很多链接分析算法是对这两种方法的改进。
 - SALSA 算法是目前效果最好的链接分析算法之一，其融合了 HITS 算法与查询相关的特点，以及 PageRank 算法的随机游走模型。
 - 主题敏感 PageRank 是对 PageRank 算法的改进，可以应用于个性化搜索中。
 - Hilltop 算法除了可以用来改善搜索系统的精确性，还可以用来当做网页反作弊的技术手段。

本章参考文献

- [1] Page, L., Brin, S., Motwani, R. and Winograd, T. (1998). The PageRank citation ranking: bringing order to the Web. Stanford Digital Library Technologies.
- [2] Borodin, A., Roberts, G.O., Rosenthal, J.S., and Tsaparas, P. (2001). Finding authorities and hubs from link structures on the World Wide Web, WWW10 Proceedings, Hongkong.
- [3] Lempel, R. and Moran S. (2000). The stochastic approach for link-structure analysis (SALSA) and the TKE effect. In the 9th International World Wide Web Conference.
- [4] Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. 668–677.
- [5] Bharat, K. and Mihail, G. A. (2001). When experts agree: Using non-affiliated experts to rank popular topics. In Proceedings of the 10th International World Wide Web Conference.
- [6] Desikan, P., Nishith, P., Srivastava, J. and Kumar, V. (2005). “Incremental PageRank Computation on evolving graphs”, Poster Paper at Fourteenth International World Wide Web Conference on May 10-14, 2005, in Chiba, Japan.
- [7] Taher, H. Haveliwala. (2002). Topic-Sensitive PageRank, in Proceedings of the Eleventh International World Wide Web Conference.
- [8] Xing, W. and Ghorbani, A. (2004). “Weighted PageRank algorithm”. Proc. of the 2nd Annual Conference on Communication Networks & Services Research, Fredericton, Canada, pp. 305-314.
- [9] Richardson, M. and Domingos, P. (2002). The intelligent surfer: Probabilistic combination of link and content information in PageRank. In Advances in Neural Information Processing Systems (NIPS) 14.




我们影响有影响力的人

北京 上海 广州 大连 西安 太原 成都 杭州 武汉 南京 深圳 -



第3章 网页反作弊

“唯之与阿，相去几何？美之与恶，相去若何？人之所畏，不可不畏。荒兮，其未央哉！众人熙熙，如享太牢，如春登台。我独泊兮，其未兆；沌沌兮，如婴儿之未孩；累累兮，若无所归。众人皆有馀，而我独若遗。我愚人之心也哉！俗人昭昭，我独昏昏。俗人察察，我独闷闷。众人皆有以，而我独顽且鄙。我独异于人，而贵食母。”

 老子·《道德经》

网页反作弊是目前所有商业搜索引擎需要解决的重要难点，出于商业利益驱使，很多网站站长会针对搜索引擎排名进行分析，并采取一些手段来提高网站排名，这种行为本身无可厚非，很多优化行为是符合搜索引擎排序规则的，但是也存在一些恶意的优化行为，通过特殊手段将网页的搜索排名提高到与其网页质量不相称的位置，这样会严重影响搜索引擎用户的搜索体验。而搜索引擎为了保证排名的公正性，也需要对作弊行为进行识别和处罚。所谓“道高一尺，魔高一丈”，只要这种经济利益存在，作弊与反作弊会一直作为搜索引擎领域的斗争而存在下去。

本章主要讲解目前常见的一些互联网网页作弊方法及搜索引擎公司对应的反制措施。从大的分类来说，比较常见的作弊方法包括：内容作弊、链接作弊、隐藏作弊及最近几年兴起的Web 2.0作弊方法。学术界和搜索引擎公司也有针对性地提出了各种反作弊算法，本章将介绍比较典型的各类反作弊算法思路，并抽象出了几种反作弊算法的框架。

3.1 内容作弊

内容作弊的目的是通过精心更改或者调控网页内容，使得网页在搜索引擎排名中获得与其网页不相称的高排名。搜索引擎排名一般包含了内容相似性和链接重要性计算，内容作弊主要针对的是搜索引擎排序算法中的内容相似性计算部分，通过故意加大目标词词频，或者在网页重要位置引入与网页内容无关的单词来影响搜索结果排名。

3.1.1 常见内容作弊手段

比较常见的内容作弊方式包括如下几种。

1. 关键词重复

对于作弊者关心的目标关键词，大量重复设置在页面内容中。因为词频是搜索引擎相似度计算中必然会考虑的因子，关键词重复本质上是通过提高目标关键词的词频来影响搜索引擎内容相似性排名的。

2. 无关查询词作弊

为了能够尽可能多地吸引搜索流量，作弊者在页面内容中增加很多和页面主题无关的关键词，这本质上也是一种词频作弊，即将原先为 0 的单词词频增加到非 0 词频，以此吸引更多搜索引擎流量。

比如有的作弊者在网页的末端以不可见的方式加入一堆单词词表，也有作弊者在正文内容插入某些热门查询词，甚至有些页面内容是靠机器完全随机生成或者利用其他网页的页面内容片段随机拼凑而成的。

3. 图片 alt 标签文本作弊

alt 标签原本是作为图片描述信息来使用的，一般不会在 HTML 页面显示，除非用户将鼠标放在图片上，但是搜索引擎会利用这个信息，所以有些作弊者将 alt 标签的内容以作弊词汇来填充，达到吸引更多搜索流量的目的。

4. 网页标题作弊

网页标题作为描述网页内容的综述性信息，对于判断一个网页所讲述的主题是非常重要的启发因素，所以搜索引擎在计算相似性得分时，往往会增加标题词汇的得分权重。作弊者利用这一点，将与网页主题无关的目标词重复放置在标题位置来获得好的排名。

5. 网页重要标签作弊

网页不像普通格式的文本，是带有 HTML 标签的，而有些 HTML 标签代表了强调内容重要性的含义，比如加粗标记、，段落标题<h>、</h>，字体大小标记等。搜索引擎一般会利用这些信息进行排序，因为这些标记因素能够更好地体现网页的内容所表现的主题信息。作弊者通过在这些重要位置插入作弊关键词也能影响搜索引擎排名结果。

6. 网页元信息作弊

网页元信息，比如网页内容描述区（Meta Description）和网页内容关键词区（Meta Keyword）是供制作网页的人对网页主题信息进行简短描述的，同以上情况类似，作弊者往往也会通过在其中插入作弊关键词来影响网页排名。

通过以上几种常见作弊手法的描述，我们可以看出，作弊者的作弊意图主要有以下几类。

1. 增加目标作弊词词频来影响排名。
2. 增加主题无关内容或者热门查询吸引流量。
3. 关键位置插入目标作弊词影响排名。

3.1.2 内容农场 (Content Farm)

Google 在 2011 年 2 月高调宣布针对低质量网页内容调整排序算法，据报道此算法影响了大约 11.8% 的网页排名，而这项调整措施是专门针对以 Demand Media 网站为代表的內容农场作弊手法的。

图 3-1 是内容农场运作模式的示意图，内容农场运营者廉价雇佣大量自由职业者，支持他们付费写作，但是写作内容普遍质量低下，很多文章是通过复制稍加修改来完成的，但是他们会研究搜索引擎的热门搜索词等，并有机地将这些词汇添加到写作内容中。这样，普通搜索引擎用户在搜索时，会被吸引进入内容农场网站，通过大量低质量内容吸引流量，内容农场可以赚取广告费用。



图 3-1 内容农场运营模式

与传统的内容作弊方式相比，内容农场不采用机器拼接内容等机械方式，而是雇佣人员写作，但是由于写作者素质等原因决定了其发布内容质量低下，这种作弊方式搜索引擎往往难以给出是否作弊的明确界定，但是又严重影响搜索结果质量，所以是一种很难处理的作弊手法。

3.2 链接作弊

所谓链接作弊，是网站拥有者考虑到搜索引擎排名中利用了链接分析技术，所以通过操纵页

面之间的链接关系，或者操纵页面之间的链接锚文字，以此来增加链接排序因子的得分，并影响搜索结果排名的作弊方法。常见的链接作弊方法众多，此节简述几种比较流行的作弊方法。

1. 链接农场 (Link Farm)

为了提高网页的搜索引擎链接排名，链接农场构建了大量互相紧密链接的网页集合，期望能够利用搜索引擎链接算法的机制，通过大量相互的链接来提高网页排名。链接农场内的页面链接密度极高，任意两个页面都可能存在互相指向的链接。图 3-2 展示了一个精心构建的链接农场。

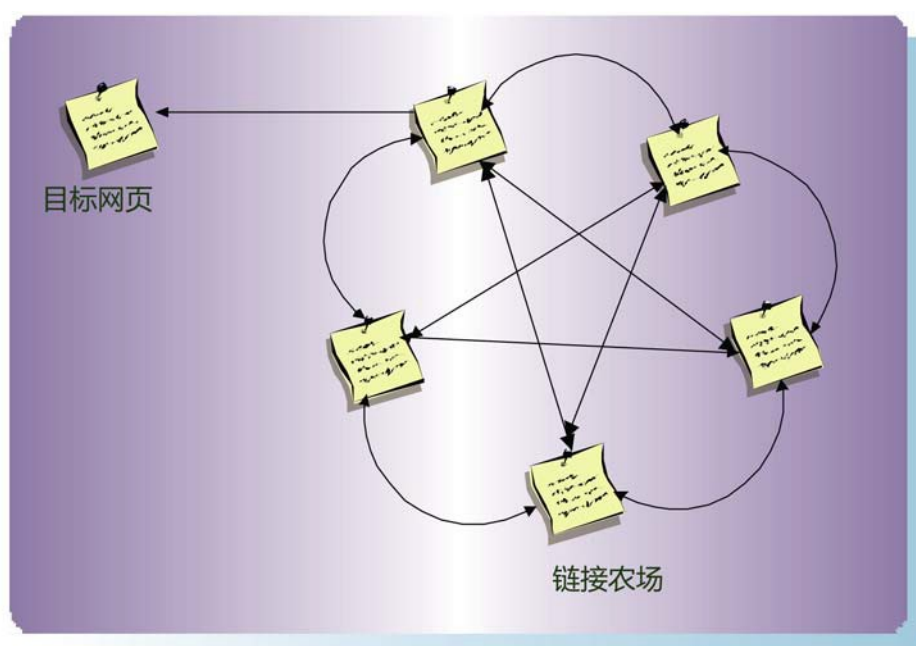


图 3-2 链接农场

2. Google 轰炸 (Google Bombing)

锚文字是指向某个网页的链接描述文字，这些描述信息往往体现了被指向网页的内容主题，所以搜索引擎往往会在排序算法中利用这一点。

作弊者通过精心设置锚文字内容来诱导搜索引擎给予目标网页较高排名，一般作弊者设置的锚文字和目标网页内容没有什么关系。

几年前曾经有个著名例子，采用 Google 轰炸来操控搜索结果排名。当时如果用 Google 搜索“miserable failure”，会发现排在第 2 位的搜索结果是美国时任总统小布什的白宫页面，这就是通过构建很多其他网页，在页面中包含链接指向目标页面，其链接锚文字包含“miserable failure”关键词（参考图 3-3 和图 3-4）所达到的效果。通过这种方式就导致

了人们看到的搜索结果。

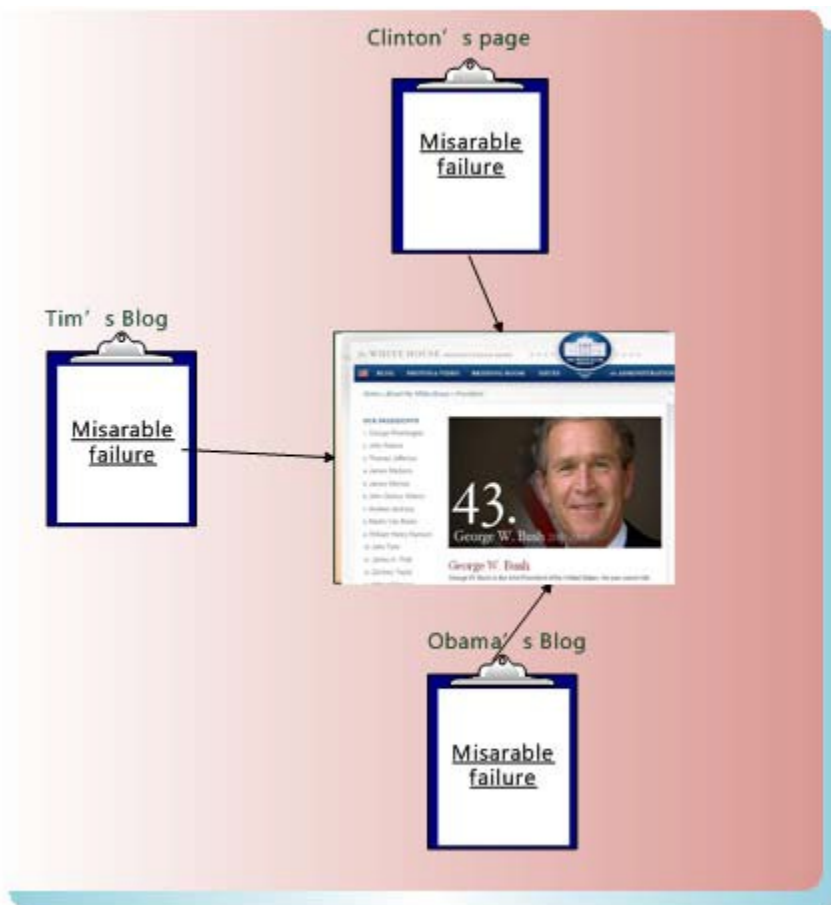


图 3-3 Google 轰炸的原理



图 3-4 Google 轰炸后的效果

3. 交换友情链接

作弊者通过和其他网站交换链接，相互指向对方的网页页面，以此来增加网页排名。很多作弊者过分地使用此手段，但是并不意味着使用这个手段的都是作弊网站，交换友情链接的做法也是正常网站的常规措施。

4. 购买链接

有些作弊者会通过购买链接的方法，即花钱让一些排名较高的网站的链接指向自己的网页，以此来提高网站排名。

5. 购买过期域名

有些作弊者会购买刚刚过期的域名，因为有些过期域名本身的 PageRank 排名是很高的，通过购买域名可以获得高价值的外链。

6. “门页”作弊 (Doorway Pages)

“门页”本身不包含正文内容，而是由大量链接构成的，而这些链接往往会指向同一网站内的页面，作弊者通过制造大量的“门页”来提升网站排名。

3.3 页面隐藏作弊

页面隐藏作弊通过一些手段瞒骗搜索引擎爬虫，使得搜索引擎抓取的页面内容和用户点击查看看到的页面内容不同，以这种方式来影响搜索引擎的搜索结果。常见的页面隐藏作弊方式有如下几种。

1. IP 地址隐形作弊 (IP Cloaking)

网页拥有者在服务器端记载搜索引擎爬虫的 IP 地址列表，如果发现是搜索引擎在请求页面，则会推送给爬虫一个伪造的网页内容，而如果是其他 IP 地址，则会推送另外的网页内容，这个页面往往是有商业目的的营销页面。

2. HTTP 请求隐形作弊 (User Agent Cloaking)

客户端和服务端在获取网页页面的时候遵循 HTTP 协议，协议中有一项叫做用户代理项 (User Agent)。搜索引擎爬虫往往会在这一项有明显的特征 (比如 Google 爬虫此项可能是: Googlebot/2.1)，服务器如果判断是搜索引擎爬虫则会推送与用户看到的不同的页面内容。

图 3-5 是一个 HTTP 请求隐藏作弊的例子，作弊网站服务器推送给搜索引擎爬虫的页面是讲述减肥食品的内容，而推送给页面访问者的则是减肥产品销售推广页面。这样当用户在搜索减肥知识的时候就会直接访问减肥产品页面，从而达到作弊者的商业目的。



图 3-5 HTTP 请求隐藏作弊

3. 网页重定向

作弊者使搜索引擎索引某个页面内容，但是如果是用户访问则将页面重定向到一个新的页面。

4. 页面内容隐藏

通过一些特殊的 HTML 标签设置，将一部分内容显示为用户不可见，但是对于搜索引擎来说是可见的。比如设置网页字体前景色和背景色相同，或者在 CSS 中加入不可见层来隐藏页面内容。将隐藏的内容设置成一些与网页主题无关的热门搜索词，以此增加被用户访问到的概率。

3.5 反作弊技术的整体思路

如上所述，目前搜索引擎作弊手段五花八门、层出不穷，作为应对方的搜索引擎，也相应调整技术思路，不断有针对性地提出反作弊的技术方案，所以如果整理反作弊技术方案，会发现技术方法很多，理清思路不易。

尽管如此，如果对大多数反作弊技术深入分析，会发现在整体技术思路还是有规律可循的。

从基本的思路角度看，可以将反作弊手段大致划分为以下3种：信任传播模型、不信任传播模型和异常发现模型。其中前两种技术模型可以进一步抽象归纳为“链接分析”一章提到的子集传播模型，为了简化说明，此处不再赘述，而是直接将这两个子模型列出。将具体算法和这几个模型建立起关系，有助于对反作弊算法的宏观思路和相互联系建立起清晰的概念。

3.5.1 信任传播模型

图3-6展示了信任传播模型的示意图。所谓信任传播模型，基本思路如下：在海量的网页数据中，通过一定的技术手段或者人工半人工手段，从中筛选出部分完全值得信任的页面，也就是肯定不会作弊的页面（可以理解为白名单），算法以这些白名单内的页面作为出发点，赋予白名单内的页面节点较高的信任度分值，其他页面是否作弊，要根据其和白名单内节点的链接关系来确定。白名单内节点通过链接关系将信任度分值向外扩散传播，如果某个节点最后得到的信任度分值高于一定阈值，则认为没有问题，而低于这一阈值的网页则会被认为是作弊网页。

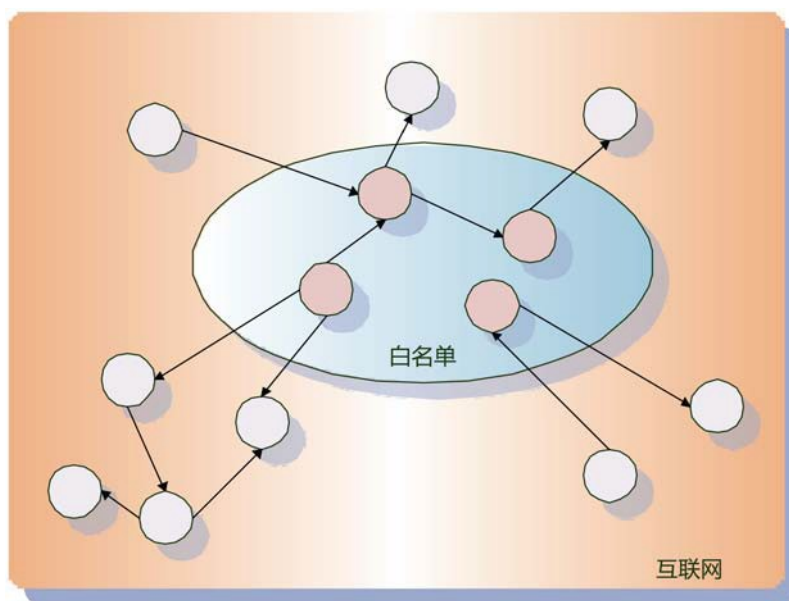


图 3-6 信任传播模型示意图

很多算法在整体流程和算法框架上遵循如上描述，其区别点往往体现在以下两方面。

- 如何获得最初的信任页面子集合，不同的方法手段可能有差异。
- 信任度是如何传播的，不同的方法可能有细微差异。

3.5.2 不信任传播模型

图 3-7 展示了不信任传播模型的整体框架示意图。从大的技术框架上来讲，其和信任传播模型是相似的，最大的区别在于：初始的页面子集合不是值得信任的页面节点，而是确认存在作弊行为的页面集合，即不值得信任的页面集合（可以理解为黑名单）。赋予黑名单内页面节点不信任分值，通过链接关系将这种不信任关系传播出去，如果最后页面节点的不信任分值大于设定的阈值，则会被认为是作弊网页。

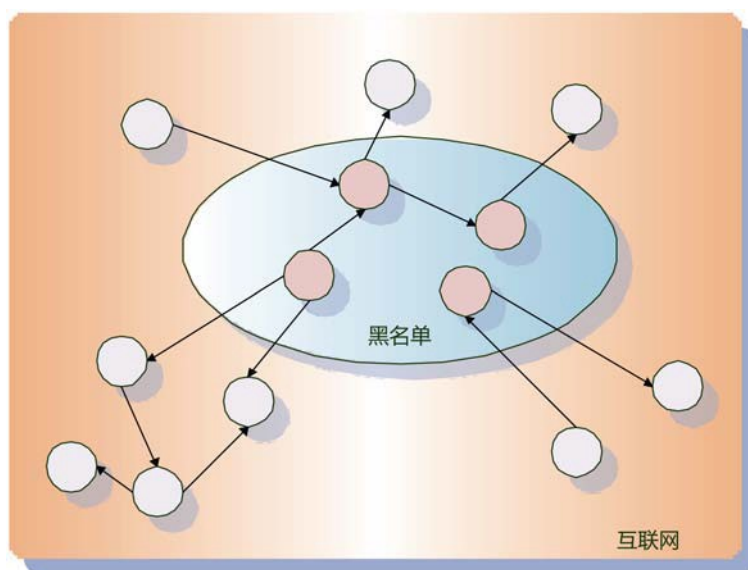


图 3-7 不信任传播模型的整体框架示意图

同样，很多算法可以归入这一模型框架，只是在具体实施细节方面有差异，整体思路基本一致。

3.5.3 异常发现模型

异常发现模型也是一个高度抽象化的算法框架模型，其基本假设认为：作弊网页必然存在有异于正常网页的特征，这种特征有可能是内容方面的，也有可能是链接关系方面的。而制定具体算法的流程往往是先找到一些作弊的网页集合，分析出其异常特征有哪些，然后利用这些异常特征来识别作弊网页。

具体来说，这个框架模型又可细分为两种子模型，这两种子模型在如何判断异常方面有不同的考虑角度。一种考虑角度比较直观，即直接从作弊网页包含的独特特征来构建算法（参见图 3-8）；另外一种角度则认为不正常的网页即为作弊网页，也就是说，是通过统计等手段分析正常的网页应该具备哪些特征，如果网页不具备这些正常网页的特征，则被认为是作弊网

页（参见图 3-9）。图 3-8 和图 3-9 体现了这两种不同的思路。

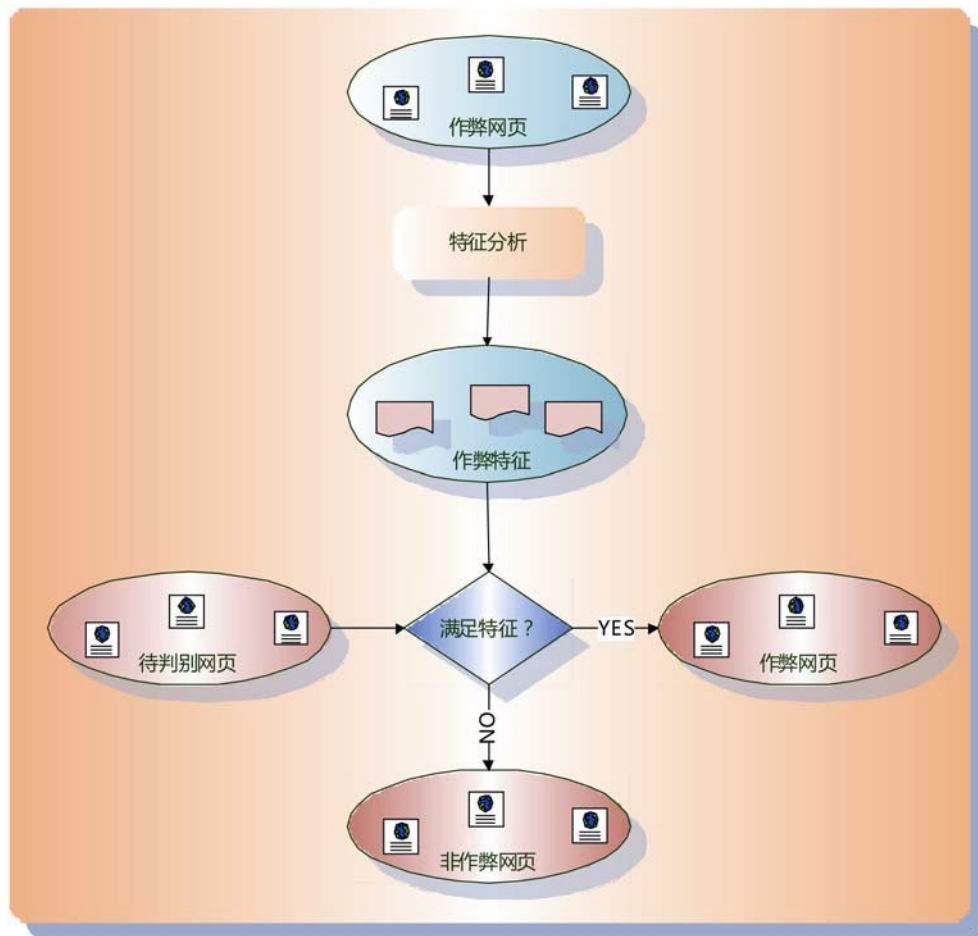


图 3-8 异常发现模型一

尽管反作弊算法五花八门，但是不论采取哪种具体算法，其实都包含了一些基本假设，经常被反作弊算法使用的基本假设有：

- a) 尽管作弊网页喜欢将链接指向高质量网页，但是很少有高质量网页将链接指向作弊网站。
- b) 作弊网页之间倾向于互相指向。
- c) 很多算法的基本思路都是从这些基本假设出发来构造的。

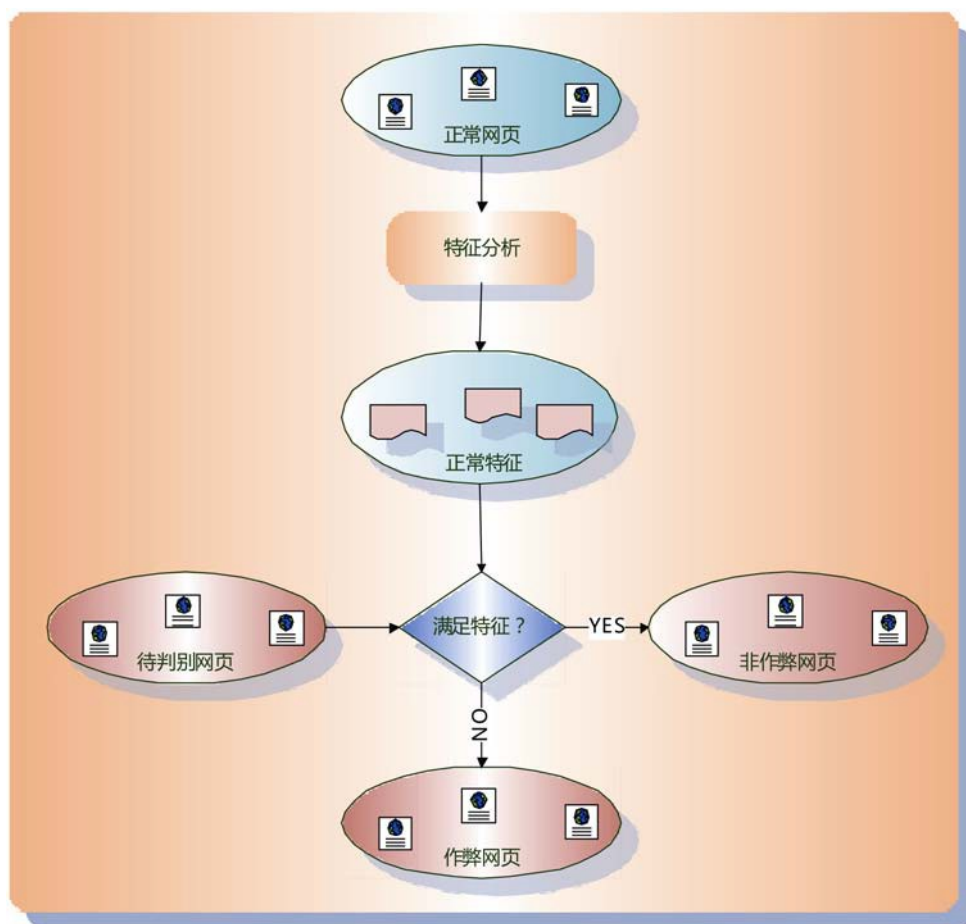


图 3-9 异常发现模型二

3.7 专用链接反作弊技术

上一节所述的通用链接反作弊技术与具体作弊方法无关，具有通用性，只要作弊手段采用了链接分析，一般都会有一定的识别作用。但是通用性的代价是针对某些具体的链接作弊方法的，其识别效果因为没有针对性，所以可能不会太好，专用的链接反作弊技术则是非常有针对性的设计算法，往往效果较好。本节简述针对链接农场和 Google 轰炸的专用反作弊技术。

3.7.1 识别链接农场

链接农场是作弊者精心构建起来的页面链接关系，和正常的链接必然有不同之处。很多研究通过比较正常网页之间链接关系的统计规律，同时研究链接农场网页之间的链接关系分布规律，通过比较两者之间的差异来识别链接农场。

识别算法比较常用的统计特征包括如下几条。

1. 网页出链的统计分布规律，正常网页的出链满足 Power-law 分布，作弊网页的出链违反该分布。
2. 网页入链的统计分布规律，正常网页的入链也满足 Power-law 分布，作弊网页则违反该分布。
3. URL 名称统计特征，作弊网页的网址倾向于较长，包含更多的点画线和数字等。
4. 很多作弊网页的 URL 地址尽管不同，但是常常会对应同一个 IP 地址。
5. 网页特征会随着时间变化，比如入链的增长率、出链的增长率等，正常网页和作弊网页在这些变化模式上是不同的。

除了对比统计特征外，还可以利用链接农场的结构特征。链接农场的结构特征是农场内的网页之间链接关系非常紧密，这也是可以直接用来进行作弊识别的特征。使用一些紧密链接子图自动发现算法，可以识别出这些紧密链接的页面子图，研究表明这种紧密链接子图中很大比例确实是由作弊网页构成的。

3.7.2 识别 Google 轰炸

Google 轰炸利用了指向目标网页的锚文字来操纵搜索结果排名，而锚文字很可能和被指向的页面没有任何语义关系，所以一个直观的判断方式即为判断锚文字是否和被指向页面有语义关系，如果有语义关系存在，则被判断为正常链接，否则可被判断为作弊链接。

但是事实上由于锚文字都比较短小，如果在字面上和被指向页面内容没有直接关系也是很正常的，所以自动判断 Google 轰炸作弊具有较大难度。

3.8 识别内容作弊

上述章节是针对链接作弊方法的一些可能反制方法，本节叙述针对内容作弊的一些反制方法。针对内容作弊，往往可以采用一些启发规则或者内容统计分析的方式进行识别。

比如对于重复出现关键词这种作弊方式，可以判断文本内一定大小的窗口中是否连续出现同一关键词，如果是的话则消除掉重复出现的内容。

比如对于标题关键词作弊，可以判断标题词汇在文本正文出现的比例和权重，如果达到一定条件则可判断为标题关键词作弊。

也可以采用一些统计手段来进行内容作弊识别，比如统计正常网页中句子长度的规律、停用

词的分布规律或者词性分布规律等, 通过比较页面内容统计属性是否异常来识别内容作弊的情况。

3.9 反隐藏作弊

常见的隐藏作弊方式包括页面隐藏和网页重定向, 下面介绍一些技术思路来识别隐藏作弊网页。

3.9.1 识别页面隐藏

页面隐藏的本质特征是向搜索引擎爬虫和用户推送不同内容的页面。所以一个直观地识别这种作弊方式的方法就是对网页做两次抓取, 第 1 次是正常的搜索引擎爬虫抓取, 第 2 次抓取则以模拟人工访问网页的方式抓取。如果两次抓取到的内容有较大差异, 则会认为是作弊页面。很明显, 这种方法虽然有效, 但是对所有页面做多次抓取的成本显然非常高。

考虑到以上方法的效率问题, 研究人员希望将识别范围缩小。因为作弊者大都具有商业动机, 所以他们认为包含一些热门查询, 以及具有商业价值查询词的页面更可能会采取隐藏作弊。可以从查询日志中挖掘最热门的查询, 同时挖掘出能够引发搜索结果中出现“赞助商链接”的商业性词汇。经过分别使用搜索引擎爬虫和模拟人工访问, 多次抓取排在搜索引擎结果前列的网页, 并比较两次下载页面的单词重叠度。研究人员发现包含商业性词汇的页面中, 如果网站采取了页面隐藏, 则有 98% 的内容是作弊页面, 而在包含热门查询词的网页中, 这个比例是 73%。

3.9.2 识别网页重定向

网页重定向是很容易识别的, 目前大部分搜索引擎对于采取了重定向的网页都会有相应的降权惩罚。但是, 采取了重定向的网页未必一定就是作弊网站, 如何更精确地识别此类作弊方式是个值得探讨的问题。

Strider 系统给出了根据网页重定向来识别到底哪些是作弊网页的解决方案。这个系统首先收集一批作弊页面, 然后根据这批作弊网页进行扩展, 如果有在论坛中和这些作弊 URL 经常一起出现的网页链接, 会逐步将其扩充进可疑页面集合。之后, 依次访问这些可疑 URL, 并记录下访问时是否做了重定向及重定向到哪个页面, 如果某个页面被很多可疑 URL 重定向指向, 则认为这个重定向地址是作弊网页, 反过来, 那些重定向到这个作弊网页的可疑 URL 也被认为是作弊网页, 其他可疑 URL 则可以被认为是正常网页。

3.10 搜索引擎反作弊综合框架

只要操纵搜索引擎搜索结果能够带来收益，那么作弊动机就会始终存在，尤其是在网络营销起着越来越重要宣传作用的时代尤其如此。作弊与反作弊是相互抑制同时也是相互促进的一个互动过程，“道高一尺，魔高一丈”的故事不断重演。

本章前述内容主要是以技术手段来进行反作弊，而事实上纯粹技术手段目前是无法彻底解决作弊问题的，必须将人工手段和技术手段相互结合，才能取得较好的反作弊效果。技术手段可以分为相对通用的手段和比较特殊的手段，相对通用的手段对于可能新出现的作弊手法有一定的预防能力，但是因为其通用性，所以针对性不强，对特殊的作弊方法效果未必好。而专用的反作弊方法往往是事后诸葛亮，即只有作弊行为已经发生并且比较严重，才可能归纳作弊特征，采取事后过滤的方法。人工手段则与技术手段有很强的互补性，可以在新的作弊方式一出现就被人发现，可以看做一种处于作弊进行时的预防措施。所以从时间维度考虑对作弊方法的抑制来说，通用反作弊方法重在预防，人工手段重在发现，而专用反作弊方法重在事后处理，其有内在的联系和互补关系存在。

一个有效的搜索引擎反作弊系统一定是一个综合系统，有机融合了人工因素、通用技术手段和专用技术手段。图 3-12 给出了一个综合反作弊系统的框架，用户可以在浏览搜索结果甚至是上网浏览时随时举报作弊网页，比如 Google 推出了浏览器插件来方便用户举报，搜索引擎公司内部会有专门的团队来审核与主动发现可疑页面，经过审核确认的网页可以放入黑名单或者白名单中。

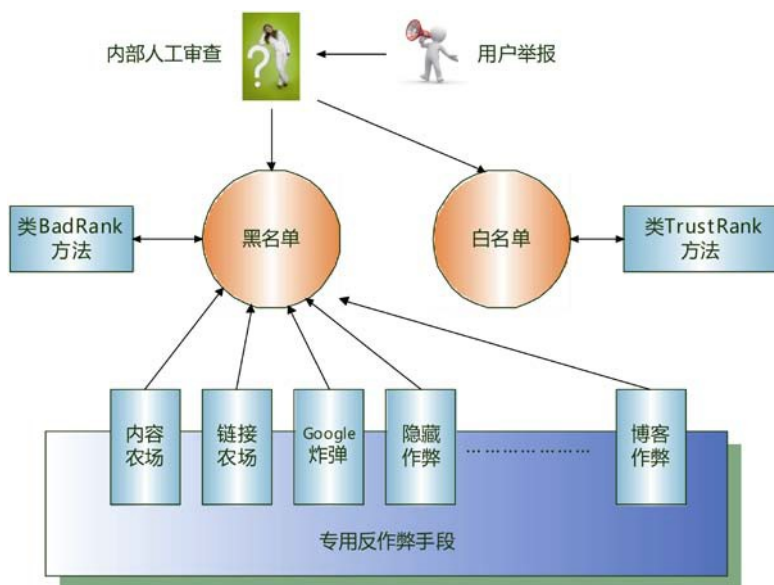


图 3-12 综合反作弊框架

通用的反作弊方法大体有两类，一种类似于 BadRank 的思路，即从黑名单出发根据链接关系探寻哪些是有问题的网页；另外一种类似于 TrustRank 的思路，即从白名单出发根据链接关系排除掉那些没有问题的网页。两者显然有互补关系，通过两者搭配可以形成有效的通用反作弊屏障。这种通用方法的好处是具有预防性，哪怕是新出现的作弊方式，只要作弊网页需要通过链接关系进行操纵，那么通用方法就能在一定程度上起到作用。但是正是因为通用方法的通用性，所以其反作弊思路没有针对性，对于一些特殊的作弊手段无法有效发现。此时，针对特殊作弊手段的方法形成了第 3 道屏障，即搜索引擎公司针对具体作弊方法采取专用技术手段来进行识别，因为有针对性所以效果较好，但是缺点在于一类反作弊方法只能识别专门的作弊手段，对于新出现的作弊方法往往无能为力，而且在时间上往往滞后于作弊现象。综上所述，这几种反作弊方法是有互补关系存在的，有效融合三者才能够获得较好的反作弊效果。

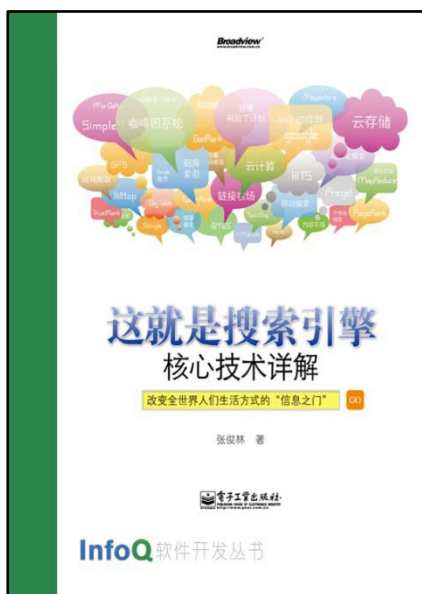
本章提要

- 作弊与反作弊相生相克，只要作弊存在经济利益，两者斗争一定会持续。
- 常见的作弊方法包括：内容作弊、链接作弊、隐藏作弊和 Web 2.0 作弊。
- 通用反作弊手段大致划分为以下 3 种类型：信任传播模型、不信任传播模型和异常发现模型。
- 纯粹用技术手段目前无法彻底解决作弊问题，必须将人工手段和技术手段相互结合，才能取得较好的反作弊效果。

本章参考文献

- [1] Gyongyi, Z. and Garcia-Molina, H. (2005). Web spam taxonomy. In First International Workshop on Adversarial Information Retrieval on the Web.
- [2] Wu, B. and Davison, B. (2005). Cloaking and redirection: a preliminary study. In First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05).
- [3] Fetterly, D., Manasse, M. and Najork, M. (2004). Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In S. Amer-Yahia and L. Gravano, editors, WebDB, 1–6.
- [4] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D. (2006). Detecting spam web pages through content analysis. In Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland). WWW '06. ACM Press, New York, NY, 83-92.

- [5] Gyöngyi, Z., Garcia-Molina, H., and Pedersen, J. (2004). Combating web spam with trustrank. In Proceedings of the Thirtieth international Conference on Very Large Data Bases - Volume 30. 576-587.
- [6] Krishnan, V. and Raj, R. (2006). Web Spam Detection with Anti-Trust-Rank. In the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb) .
- [7] Becchetti, L., Castillo¹, C., Donato¹, D., Leonardi, S., and Baeza-Yates, R. (2006). Using Rank Propagation and Probabilistic Counting for Link Based Spam Detection. In Proc. of WebKDD'06.
- [8] Baoning, W. and Brian, D. (2005). Identifying link spam farm pages. In *WWW*, 2005.
- [9] Gyongyi, Z., Garcia-Molina, H., Berkhin, P., and Pedersen, J. (2006). Link spam detection based on mass estimation. In *VLDB*.
- [10] Bencz, A. and Uher, M. (2005). Spamrank: fully automatic link spam detection. In Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web, Chiba, Japan.
- [11] Bencz, A. and Uher, M. (2006). Detecting nepotistic links by language model disagreement. In *WWW*, 939–940.
- [12] Paul, H., Georgia, K., and Hector, G. (2007) "Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges," *IEEE Internet Computing*, vol. 11, pp. 36-45.



这就是搜索引擎

原书责任编辑：付睿

原书美术编辑：李玲

迷你书责任编辑：郑柯

迷你书美术编辑：胡伟红

本迷你书主页为

<http://www.infoq.com/cn/minibooks/this-is-search-engine>

《这就是搜索引擎：核心技术详解》一书由电子工业出版社 2012 年 1 月出版，电子工业出版社对该作品享有专有出版权，未经授权，不得以任何方式复制或抄袭书中之部分或全部内容。版权所有，侵权必究。

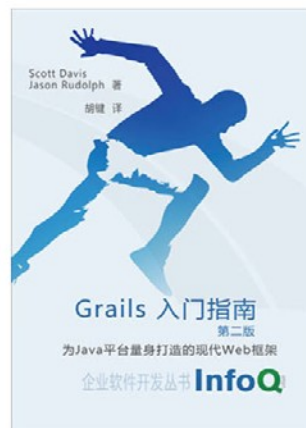
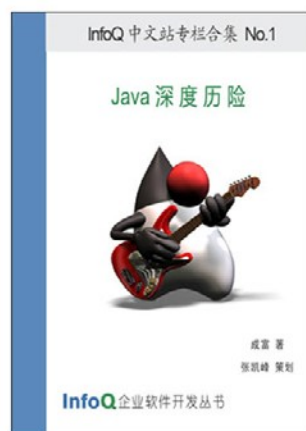
本电子迷你书摘自《这就是搜索引擎：核心技术详解》一书第 3、6、8 章。由 InfoQ 制作，属于 InfoQ 软件开发丛书。如果您打算订购本书完整的纸质版，请登录[豆瓣图书相关页面](#)。本书提到的公司产品或者使用到的商标为产品公司所有。

如果读者要了解具体的商标和注册信息，应该联系相应的公司。

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译等，请联系 editors@cn.infoq.com。

InfoQ 软件开发丛书

欢迎免费下载



商务合作: sales@cn.infoq.com

读者反馈/内容提供: editors@cn.infoq.com