目 录

张庆国

第二章　基于倒排档的检索系统

2.1　倒排档检索技术发展简史

设我们已有文献——语词矩阵 D

$$D=\begin{array}{c} \\ D_1 \\ D_2 \\ \vdots \\ D_N \end{array}\begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \cdots\cdots & T_M \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1M} \\ a_{21} & a_{22} & \cdots\cdots & a_{2M} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ a_{N1} & a_{N2} & \cdots\cdots & a_{NM} \end{pmatrix} \end{array} \qquad (2.1)$$

其中 a_{ij} 表示标引词 T_j 对文献 D_i 的权值。a_{ij} 取离散值 0 和 1。容易理解，对于检索同时具有标引词 T_{j1} 和 T_{j2} 的文献的提问来说，文献 D_i 是命中文献的充分必要条件是：a_{ij1} 和 a_{ij2} 同时为 1。

这个思想一直指导着倒排档检索技术的发展。

首先值得介绍的是 Batten 在四十年代提出的“重叠比孔”检索方式。在重叠比孔检索系统中，文献——语词矩阵（2.1）中的每一列被制成一张卡片，即一张卡片代表一个主题类目——标引词，并将该标引词记在卡片的顶部。卡片上的其余部分分成若干小格，每一小格供写某一文献号。小格可以是矩阵排列的。文献 D_i 用标引词 T_j 标引后，便从系统中抽出卡片 T_j，并在这张卡片上的第 i 个小格穿上一个孔。

一张普通的卡片可以容纳一千个孔位（当然，这并不意味着它被实际地穿了一千个孔）。显然，一般的检索系统都不会只有一千篇文献，因此一个标引词往往需要若干张穿孔卡，这若干张穿孔卡

在[illegible]个[illegible]下。

检索时，设需要检索同时具有标引词 T_{j1} 和 T_{j2} 的文献，抽出一张 T_{j1} 卡片和一张 T_{j2} 卡片（当然，这两张卡片是要在同一文献号范围之内的，例如，从1到1000，从1001到2000，等等），将它们重叠在一起，如果第i篇文献同时被 T_{j1} 和 T_{j2} 标引过，则在第i个孔位上有重叠的孔，可以用一个同光箱相连的透明小格读出文献号。

如下图所示

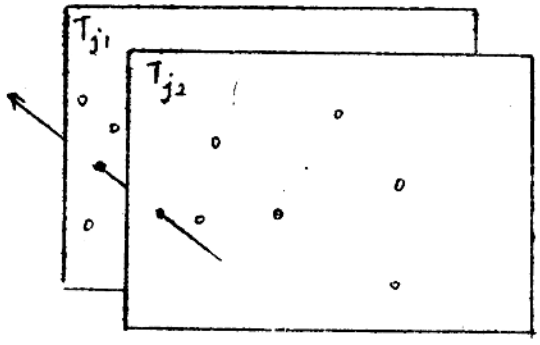

图 2.1　重叠比孔卡

标引时，若某篇文献用若干个标引词同时标引，则收集相应的若干张卡片，使用高精度的[illegible]孔器，在每张这样的卡片的相同位上穿出一个孔。

重叠比孔卡当然只是一种机械化的匹配方式，目前，这种原理已被应用到缩微检索中。将代表不同文献标识的卡片图案制成缩微胶片或胶卷，用光学的方法进行匹配，一旦产生光电效应，则得到命中文献号。

五十年代出现的一种类似于重叠比孔卡的检索方法是元词卡片。它不是穿孔，而是在卡片的不同格位上书写文献号，匹配时靠人工比较判断，如下图所示：

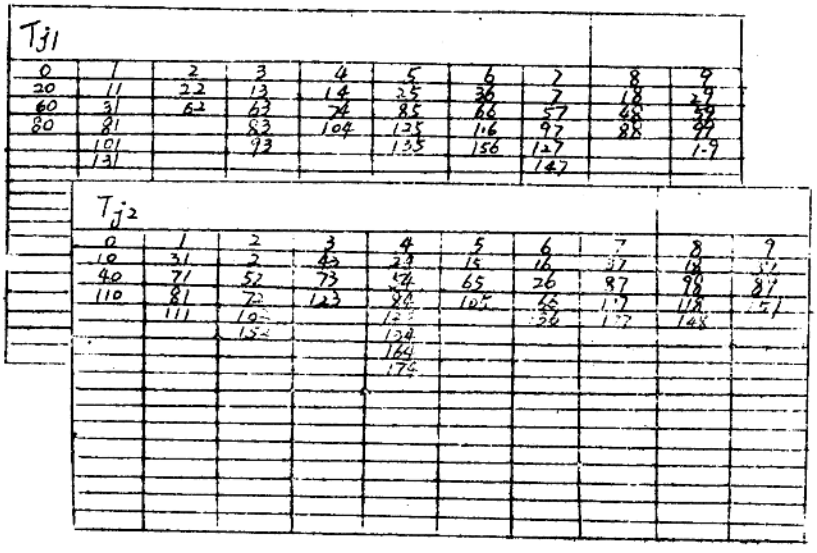

图 2.2　元词卡片

元词卡片比重叠比孔卡要节省一些卡片，但它的人工判读过程要困难一些，因此它实际上比重叠比孔卡后退了一步。

与元词卡片方法相似的还有双套字典法。

重叠比孔卡和元词卡片都是以标引词作为卡片单位的。另一种不同的方法是以文献作为卡片单位，典型的是 Mooers 的“边缘开口卡片”，图 2．3 是这种卡片的形式。每张卡片代表一篇文献，在卡片的边缘穿上孔，不同的孔位表示不同的标引词，当该篇文献被某一标引词标引后，便轧开相应的孔。

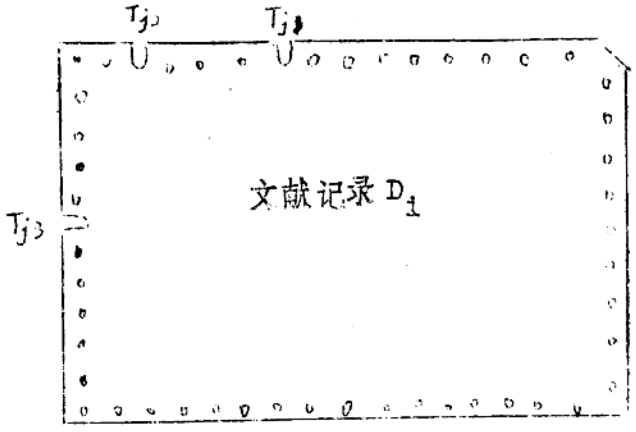

图 2.3 边缘开口卡片

检索时，设需要检索有标引词 T_{j1} 的文献，用穿条穿进所有卡片上的 T_{j1} 这个孔，提起穿条时，落下的文献卡片则全是被 T_{j1} 标引过的。设需要检索有标引词 T_{j1} 和 T_{j2} 的文献，对上述穿孔得到的卡片再穿孔 T_{j2}，便得到所需文献。

需要指出的是，尽管重叠比孔卡和边缘穿孔卡的形式不同——一个是标引词卡片，一个是文献卡片——但它们都是从语词出发而不是从文献出发进行检索的，因而都是倒排检索。对于同样的检索提问，这两种检索方式所得到的检索结果也完全相同。

Batten 和 Mooers 在四十年代研制出来的上述文档组织方式——语词款目和文献款目——仍然是现代计算机情报检索系统文档组织的基本方式。现有的全部系统，都可以简单地看作是 Batten 和 Mooers 系统的日益成熟和完善的自动化形式。在本章和本书的其他部分，我们将较为详细地讨论这种自动化检索方式。

2.2 布尔逻辑

用户的提问当然可以仅由一个检索词组成，如“Videodisc”，但在现在社会里，即使是这样一个专指度很高的词，也有成千上万篇文献，用户不可能都需要它们，因此，用户在提问式中往往用较多的词对检索提问的主题对象加以限制，这就涉及到各提问词之间的关系问题。

用户提问中各要素之间的关系基本上是布尔逻辑关系，我们先对布尔逻辑作一简单介绍。

布尔逻辑由英国[illegible]布尔[illegible]，布尔代数是一种二值代数，其基本运算符有三个：与（*），或（+），非（ ）。运算规则如下：

$$\begin{cases} 1 * 1 = 1 \\ 1 * 0 = 0 \\ 0 * 1 = 0 \\ 0 * 0 = 0 \end{cases} \qquad (2.2)$$

$$\begin{cases} 1 + 1 = 1 \\ 1 + 0 = 1 \\ 0 + 1 = 1 \\ 0 + 0 = 0 \end{cases} \qquad (2.3)$$

$$\begin{cases} \overline{1} = 0 \\ \overline{0} = 1 \end{cases} \qquad (2.4)$$

运用到集合论中，设A，B是两个集合，则

$$A * B = \{x \mid x \text{属于} A \text{并且} x \text{属于} B\}$$

$A+B=\{x \mid x$属于A，或者x属于B，或者x同时属于A和B$\}$

$\overline{A}=\{x \mid x$不属于A$\}$

用文氏图表示如下

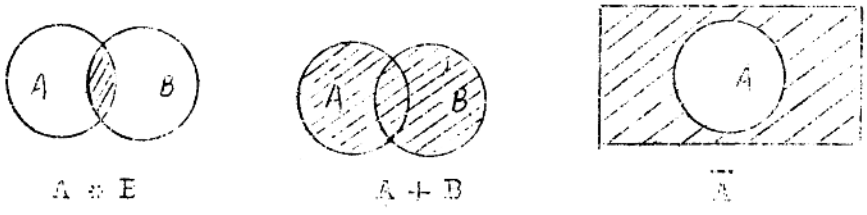

图2.4 集合的布尔运算

对于情报检索，可以将许多检索词用布尔逻辑关系符号进行组合以构成检索式。例如，作为主题词，如果用A代表天线（Aerial），B代表广播（Broadcast），C代表通信（Communication），D代表抛物天线（Dish），则可以构成以下提问式。

$Q_1 = A + B + C + D$

$Q_2 = (A + B) * C$

$Q_3 = B * C * \overline{D}$

上述讨论是把检索词都作为主题词看待的。实际检索中，往往对主题词之外的其他文献属性也有要求，如作者、语种等等，这些要求应该反映到检索提问式中。例如，我们可以用下面的符号作为属性指示符：

Tit （Title，篇名）

Au （Author，作者）

Sub　（Subject，主题词）

Da　（Date，出版年）

La　（Language，语种）

等等。于是可以构成以下提问式

$$Q_4 = B_{(Sub)} * C_{(Sub)} * Smith_{(Au)} * 1977_{(Da)}$$

对于年代和其他数值性属性值，还可以规定比较条件

GT　（Great Than，大于）

LT　（Litter Than，小于）

EQ　（Equal，等于）

GE　（Great Equal，大于等于）

LE　（Litter Equal，小于等于）

NE　（Not Equal，不等于）

BT　（Between，介于）

于是可以有下列检索式

$$Q_5 = (A + B) * GE\ 1981_{(Da)}$$

2.3　典型的文档结构

在计算机文献情报检索系统中，文档结构占有极其重要的地位。这首先是因为，计算机情报检索系统中的文档不同于一般计算机系统中的文件系统。后者中的多数内容是暂时性存放的，前者则不仅巨大，而且永久性存放，是整个检索系统赖以存在和运行的重要物质基础；其次，计算机情报检索系统的主要任务就是从各文档中查找所需的情报数据，文档结构在很大程度上影响甚至决定着检索过程乃至整个系统的效率。

从最基本的意义上讲，倒排档检索系统只要有两个文档——

倒排索引档和顺排资料档就够了。其中倒排索引档相当于按列存贮文献——语词矩阵，用来查找用户提问中的各检索词，以及与它们相关的文献号；顺排资料档相当于按行存贮文献——语词矩阵，只是不仅给出标引词，而且给出整个的二次文献记录，用来向用户输出。但是，由于现在系统中这两种文档的巨大规模，一般不可能直接在其中查找，因此它们又各自需要有索引文档，其中倒排档的索引文档通称为“词典”。

倒排档又称主题词文档，因为它的每一个记录相应于一个主题词款目，并且只被一个主题词唯一地标识。在每一个主题词记录中，包含着用该主题词标引过的全部文献的文献号。每一个主题词记录的形式如下：

主题词	文献号个数	文献号，文献号，……

在主题词文档中，各主题词记录之间的排列可以按某种指定顺序（如主题词的字顺），但由于主题词文档索引文档（即词典）的存在，主题词文档一般并不直接用于查找，所以各记录也可以是任意次序的。

除了主题词文档这种最主要的倒排档之外，还可以为作者等其他可检项目建立倒排档。在有的系统中，各倒排档是分立的；在有的系统中，则把多个对应不同检索项目的倒排档混合排成一个倒排档（如按字顺）。

“词典”这个名称比较形象地说明了倒排档索引的作用。由于倒排档中每个记录往往较大，因此整个倒排档也是很大的，要在这个很大的倒排档中检索某个主题词记录，往往要求系统付出较大的时间和空间开销，因此专门设一个索引（词典），每检索一个主题词记录时，首先在词典中找到该主题词记录在倒排档中的位置，以

及该主题词记录的大小（记录中的文献号个数），然后到倒排档中直接调取此记录。

词典中每个记录的形式通常如下：

主题词	文献号个数	地址

我们看到，记录中对主题词和文献号个数的存贮与主题词是重复的，因此，后者中的这两个字段可以省略。

词典是专为检索而设置的，因此，它的结构（各记录之间的排列次序）必须能提供检索手段。在大多数情况下，词典被组织成索引顺序文件，但如利用Hash方法检索，也可以将词典组织成散列文件。

词典的查找技术问题具有一般性，因为全部检索问题实际上都是查找问题。对于查找技术，我们以后将专门讨论。

顺排文档又称主资料档，它的每个记录是一篇文献的二次文献，即记载了该篇文献的用户可能需要的信息（篇名，作者，出版项，主题词，文摘等）。各记录的排列可以是有序的（如按照文献号），但在存在索引文档时，各记录之间也可以是无序的。

主资料档是系统中最大的文档，也是用户要最终检索的文档，其他各文档都是为便利主文档的检索而建立的。

主文档的索引文档的功能也是显然的。由于在倒排档记录中给出的是文献号而不是文献记录的实际物理地址，因此在主文档中调出命中文献记录之前，需将文献号码转换成文献地址。主文档索引文档提供了这一功能，从而避免了对主文档直接查找。不过，在有的系统中，也不设这一索引文档。

主文档索引文档中每个记录的一般形式如下：

文献号	地址

主文档索引文档中各记录的排列一般是有序的（如按照文献号）除非它有其他的检索手段（如Hash法）。

下面给出这种典型的文档结构示意图。

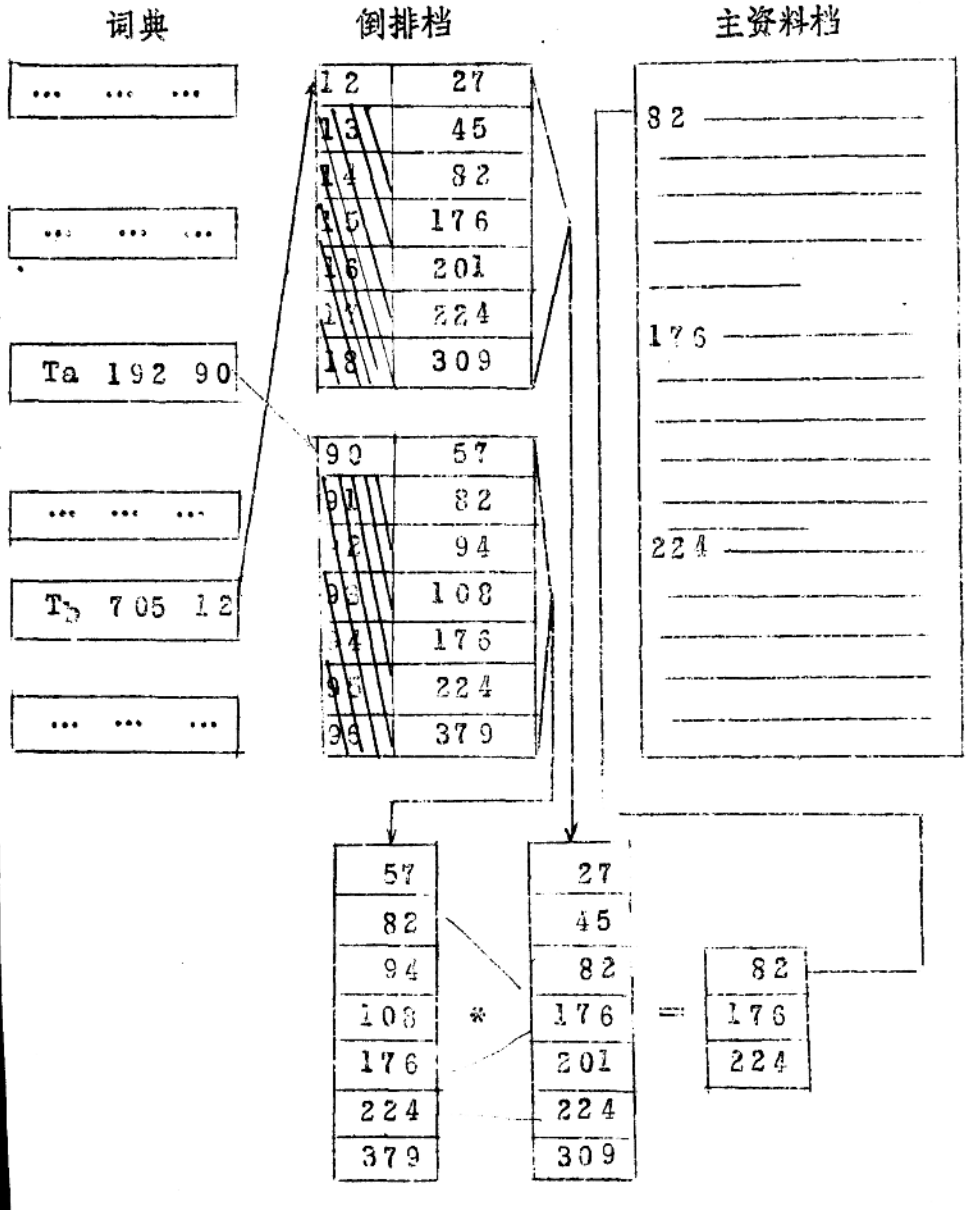

图2.5 倒排档检索的文档结构

在上图中，我们未给出主文档的索引文档。

2.4 检索过程

基于倒排档的联机检索过程一般为

1. 检索者输入检索提问式；

2. 系统分解出检索提问式中的各提问因子，提问因子主要有主题词，也可以有作者，出版项等；

3. 主词典和倒排档中检索，分别得到对应每个检索词的文献集合，这时可以告知检索者对应于各个检索词的命中文献数；

4. 按照原检索提问式中各检索因子之间的逻辑关系，对上述检出的各文献集合进行逻辑运算，得到满足用户提问的命中文献集合；

5. 根据命中文献集合中的文献号，在主文档中找到对应的命中文献记录，输出。

上述各步骤的具体实现方法当然很多，下面简单介绍比较典型的一种。

1. 输入检索提问式

输入可以是键盘输入，也可以是卡片或低带输入，但无论怎样输入，都要先和系统“挂勾”，取得联系。联机情报检索系统一般都是多用户系统，系统的硬、软件资源是各用户共享的，系统在接到用户的检索请求，并验证了用户的合法身份后，由操作系统中的作业调度程序接受这一作业，并为该用户开辟一工作区。检索式输入之后，存放在工作区之中。

作业被接受之后，由操作系统按时间片将中央处理机（CPU）及其他硬、软件资源（内、外存，通道，系统程序和检索应用程序