网络信息检索

第五讲 文本处理与索引

董守斌
sbdong@scut.edu.cn

华南理工大学计算机学院
广东省计算机网络重点实验室
Communication & Computer Network Laboratory (CCNL)

主要内容

- 文本特性
- 文本操作
- 索引

文本的统计特性

- 文本的统计特征是？
- 不同词的频率是如何分布的？
- 当文档集增大时，词汇表的大小如何变化？
- 以上这些因素影响信息检索的性能，包括词的权重，以及检索系统的其他方面

信息论的观点

- 信息熵：设词汇表中有**σ**个词，文本中每个词出现的频率是p_i，那么该文本的熵可以定义为***E***：

$$E = -\sum_{i=1}^{\sigma} p_i \log_2 p_i$$

- 文本的信息量可以用熵来量化，熵的大小是对文本蕴含的信息量的一个度量，依赖于每个词出现的概率（频率）

信息熵的计算

- 例1：σ=2，如果两个词出现的频率相同时，熵为1；

$$E = -\sum_{i=1}^{\sigma} p_i \log_2 p_i$$

$$= -(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2})$$

$$= -\frac{1}{2}(\log_2 (\frac{1}{2} \times \frac{1}{2})) = 1$$

- 例2：如果只有一个词出现，熵为0。

齐普夫（Zipf）定律

- 哈佛大学语言学教授George Kingsley Zipf提出齐普夫定律（1948年）：如果将词汇表中的词按照出现的频率由高到低排序，那么某个词出现的频率$p(r)$与该词所对应的排序序号r满足幂律关系：

$$p(r) \propto \frac{1}{r^{\alpha}} \qquad \alpha \approx 1$$

- 因此，k是Zipf 参数

$$p(r) \times r = k$$

- 对很多文档集，有：

$$\text{常数}. k \approx 0.1$$

抽样的词频数据

Word	Frequency	$r \times p_r$	Word	Frequency	$r \times p_r$
the	1,130,021	0.059	by	118.863	0.081
of	547,311	0.058	as	109,135	0.080
to	516,635	0.082	at	101,779	0.080
a	464,736	0.098	mr	101,679	0.086
in	390,819	0.103	with	101,210	0.091
and	387,703	0.122	from	96,900	0.092
that	204,351	0.075	he	94,585	0.095
for	199,340	0.084	million	93,515	0.098
is	152,483	0.072	year	90,104	0.100
said	148,302	0.078	its	86,774	0.100
it	134,323	0.078	be	85,588	0.104
on	121,173	0.077	was	83,398	0.105

WSJ87 collection (46,449 articles, 19 million term occurrences, 132 MB)

真实数据是否满足 Zipf定律?

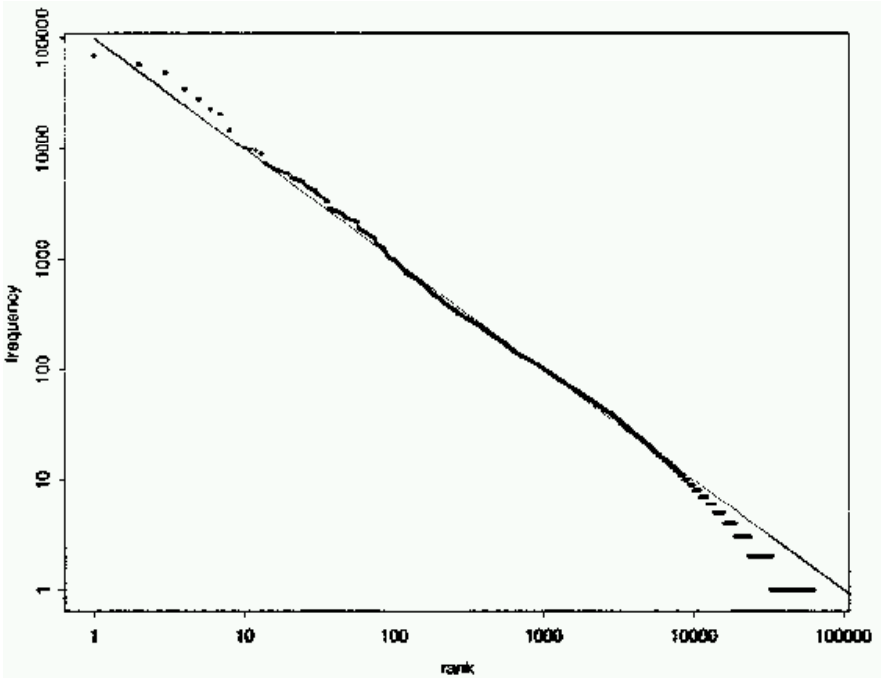
- 满足形如 $y = kx^c$ 的定律叫做幂律（**power law**）
- **Zipf**定律是一个幂律，其中$c = -1$
- 在**log-log**图中，**power laws**是一条斜率为c 的直线.

$$\log(y) = \log(kx^c) = \log k + c\log(x)$$

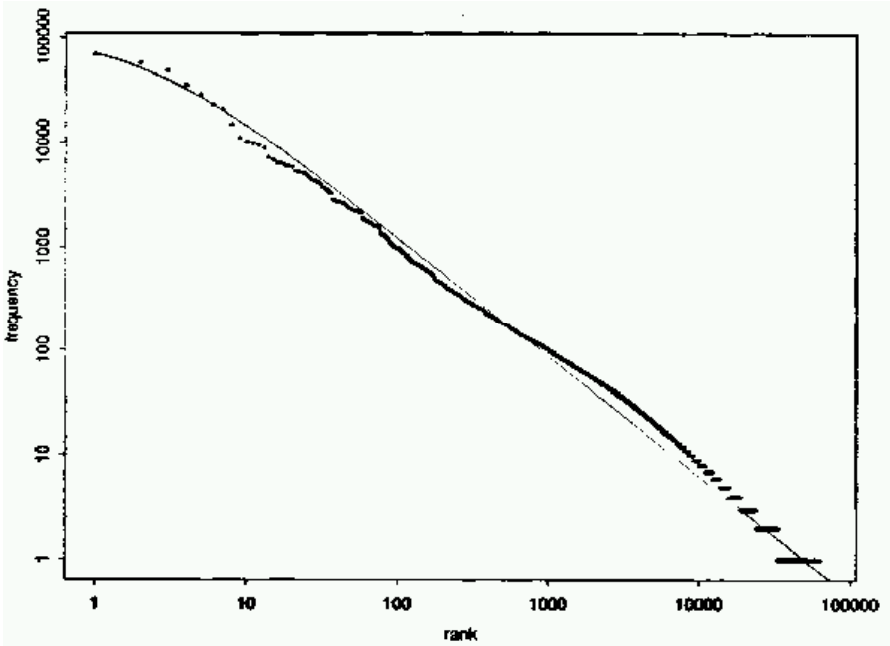
- 除了高频出现和低频出现的词，**Zipf**定律比较准确

例子：Brown Corpus

$f = P(r+\rho)^{-B}$ For constants P, B, ρ



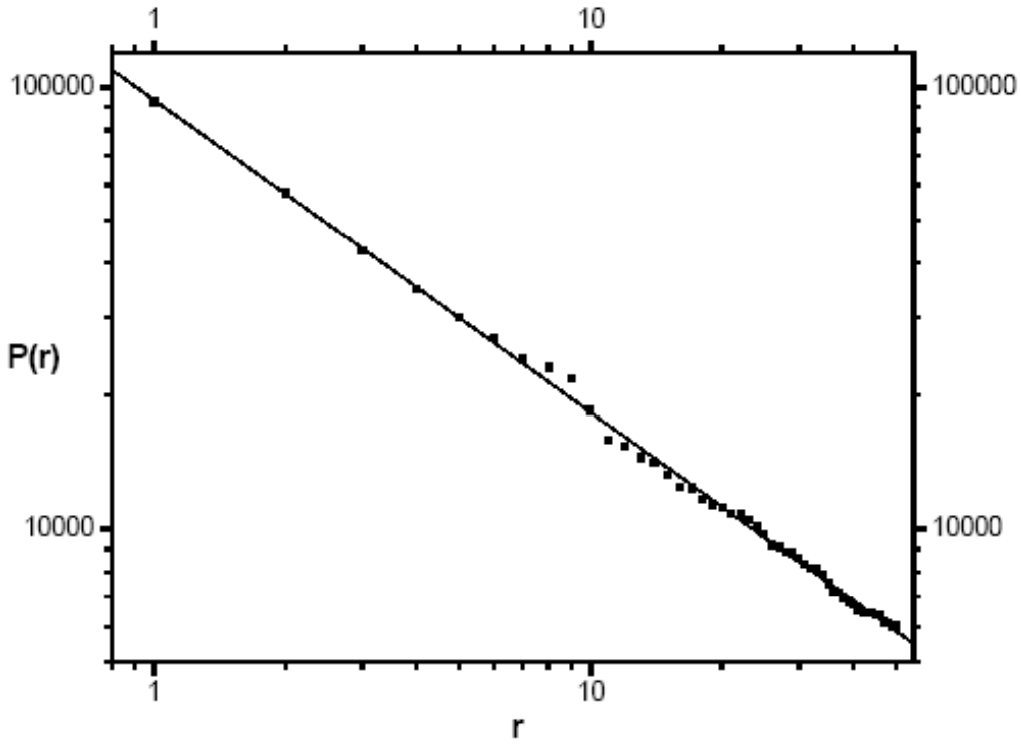
$k = 100{,}000$



Mandelbrot's function on Brown corpus

$P = 10^{5.4}$, $B = 1.15$, $\rho = 100$

例子：百度搜索词



百度搜索引擎2004年2月19日搜索综合排名前50个关键词的搜索词频分布_

Zipf指数一般在0.6到0.8之间_

http://www.qiji.cn/eprint/abs/840.html

例子

- 有一个英语单词表，包含单词**100,000**个，假设**zipf**常数$k = 0.1$，
  （1）出现了**50**次的单词，在排序词汇表中的排名第几？
  （2）出现了**50**次的词有多少个？

$$p(r) = \frac{出现次数}{总单词量} = \frac{n}{N} \qquad p(r) \times r = k$$

$$\frac{n}{N} \times r = k \qquad \Rightarrow r = k \times \frac{N}{n}$$

$$\Rightarrow r = 0.1 \times \frac{100{,}000}{50} = 200$$

$$I_n = r_n - r_{n+1} = \frac{kN}{n(n+1)}$$

$$= \frac{0.1 \times 100{,}000}{50 \times (50+1)}$$

$$= 3.9$$

- 所以，出现**50**次的单词在单词表中排名序号为**200**
- 所以，约有**4**个单词出现了**50**次

预测词频

- 出现***n***次的词的个数：

$$I_n = r_n - r_{n+1} = \frac{kN}{n} - \frac{kN}{n+1} = \frac{kN}{n(n+1)}$$

$$\frac{I_n}{D} = \frac{1}{n(n+1)}$$

- 如果排在末尾（序号最高）的单词只出现了一次，那它的排名序号为$r = kN/1 = D$，仅出现了一次的单词个数为$I_1 = r/2$，说明几乎有一半的单词只出现过一次

例子：预测词频

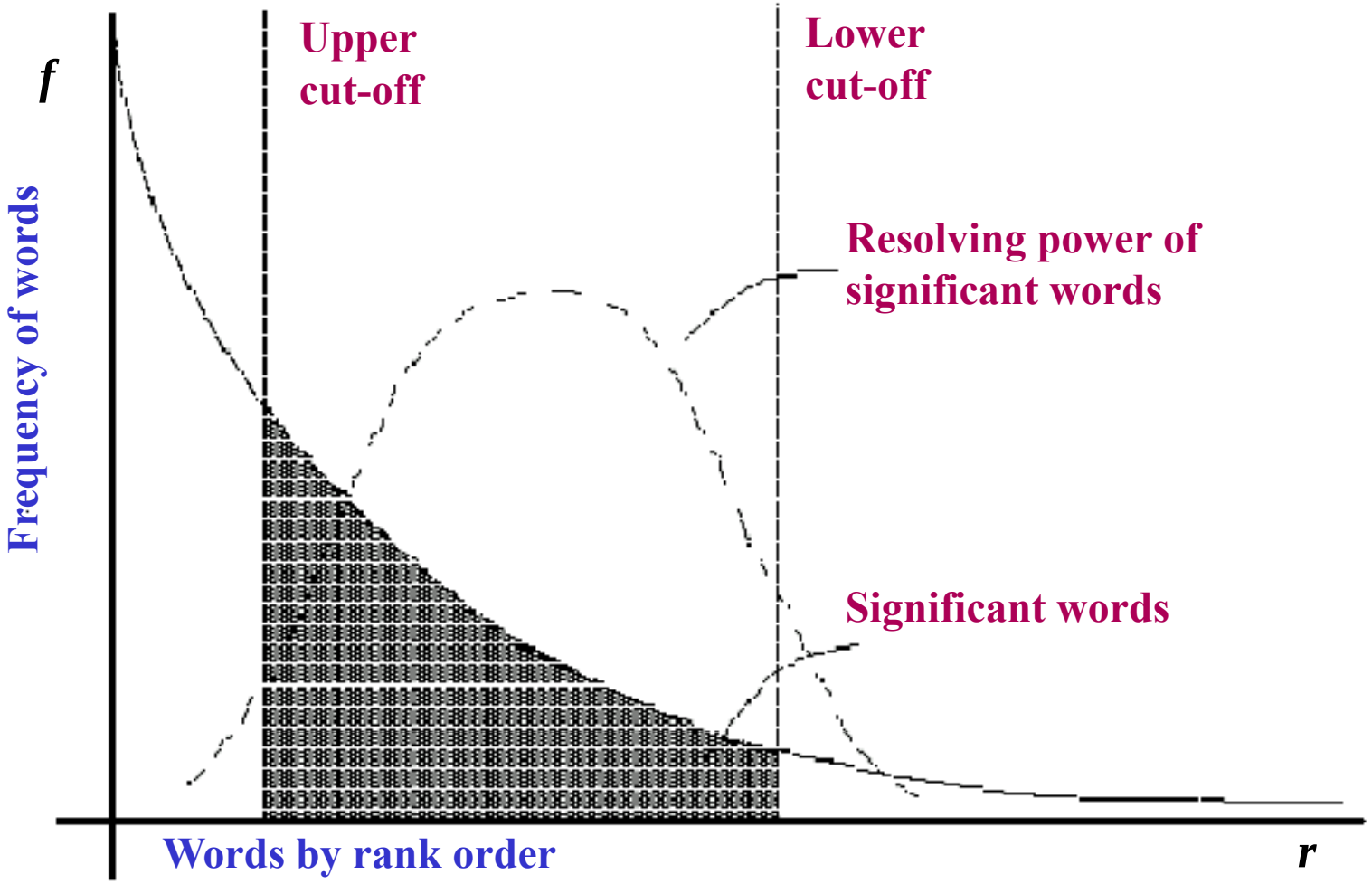
Number of Occurrences (n)	Predicted Proportion of Occurrences 1/n(n+1)	Actual Proportion occurring n times I_n/D	Actual Number of Words occurring n times
1	.500	.402	204,357
2	.167	.132	67,082
3	.083	.069	35,083
4	.050	.046	23,271
5	.033	.032	16,332
6	.024	.024	12,421
7	.018	.019	9,766
8	.014	.016	8,200
9	.011	.014	6,907
10	.009	.012	5,893

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus

125,720,891 total word occurrences; 508,209 unique words

N D

重尾分布



- 特别常用和特别不常用的词对索引没有太大用处

Zipf定律的解释

- 常用的词一般比较短
- 最常用的 **20%**的词汇覆盖了**60%**的使用率
- **Zipf**的解释：最小努力原则/省力原则（**principle of least effort** ）
  - **Gorege K.Zipf “Human Behavior and the Principleof Least Effort”**（人类行为和最小省力原则）：说明我们的许多认识行为都有使精力支出最小化的倾向
  - 例如：用最少量的词来传递最大量的信息

Zipf定律对检索系统的影响

- 好消息:
  - 停用词占文本中的很大一部分，因此删除停用词可以大量减少索引文档的存储空间
- 坏消息:
  - 对大多数词来说，进行词汇之间的相关分析并不容易，因为它们出现的次数比较少

词汇表

- 信息检索系统涉及到信息的存储，需要预留多大的存储空间呢？这个问题和文档中的词汇量相关
- 随着语料库的增长，词汇表以什么样的速度相应地增长？
- 由于专名的存在，词表实际上没有上限
  - 保先
  - 神五、神六

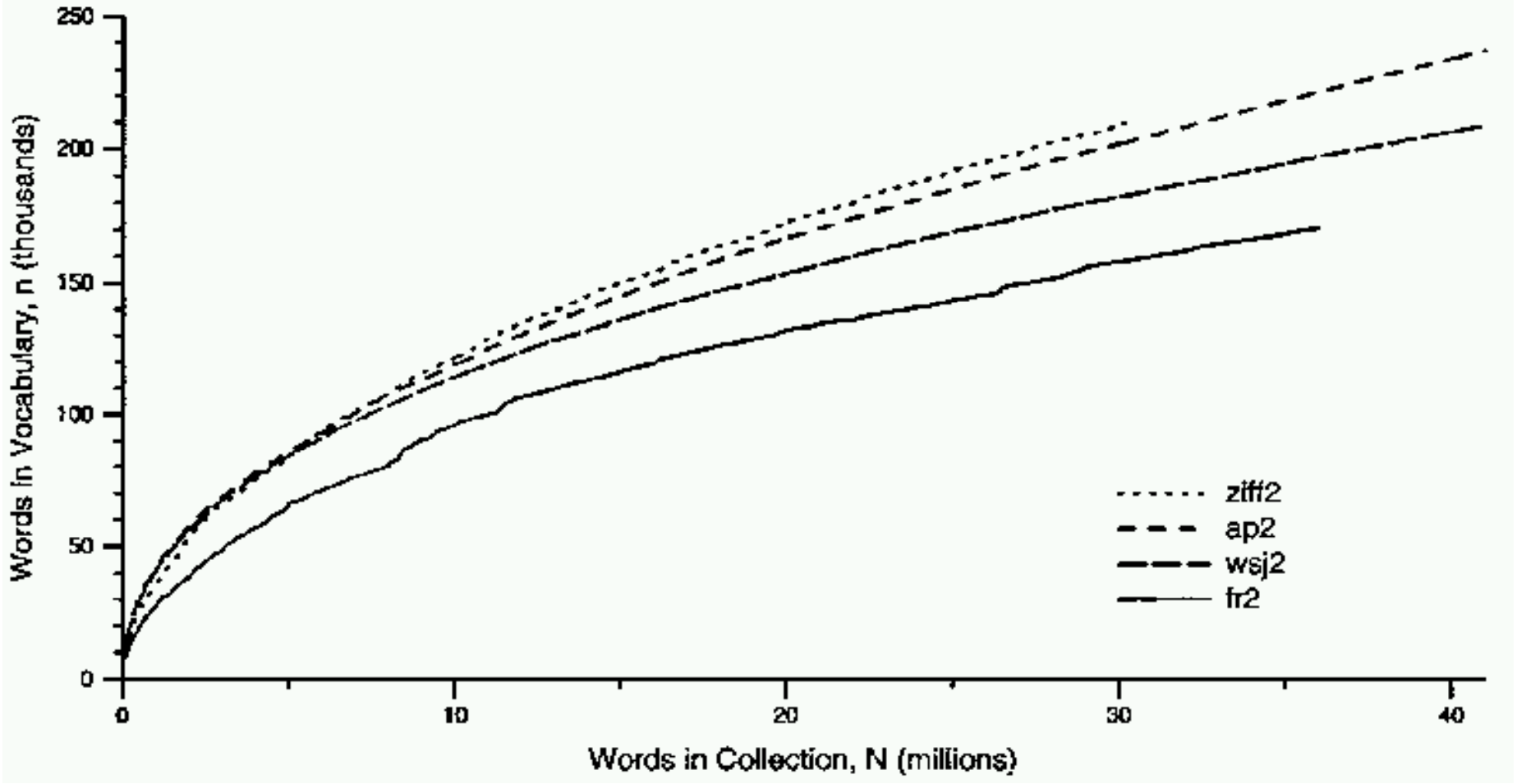
Heaps定律

- 如果***V*** 是词表大小（词汇量），***n*** 是文献集单词的个数

$$V = Kn^{\beta} \quad \text{with constants } K,\ 0 < \beta < 1$$

- 典型的常数
  - ***K*** ≈ **10–100**
  - β ≈ **0.4–0.6** （平方根指数）

例子：TREC-2文档集


Words in Vocabulary, n (thousands)
250
200
150
100
50
0
0
10
20
30
40
Words in Collection, N (millions)
ziff2
ap2
wsj2
fr2

课后练习

- **Suppose Zipf *k*=0.1, what is the fewest number of most common words that together account for more than 25% of word occurrences (i.e. the minimum value of *m* such as at least 25% of word occurrences are one of the *m* most common words)**

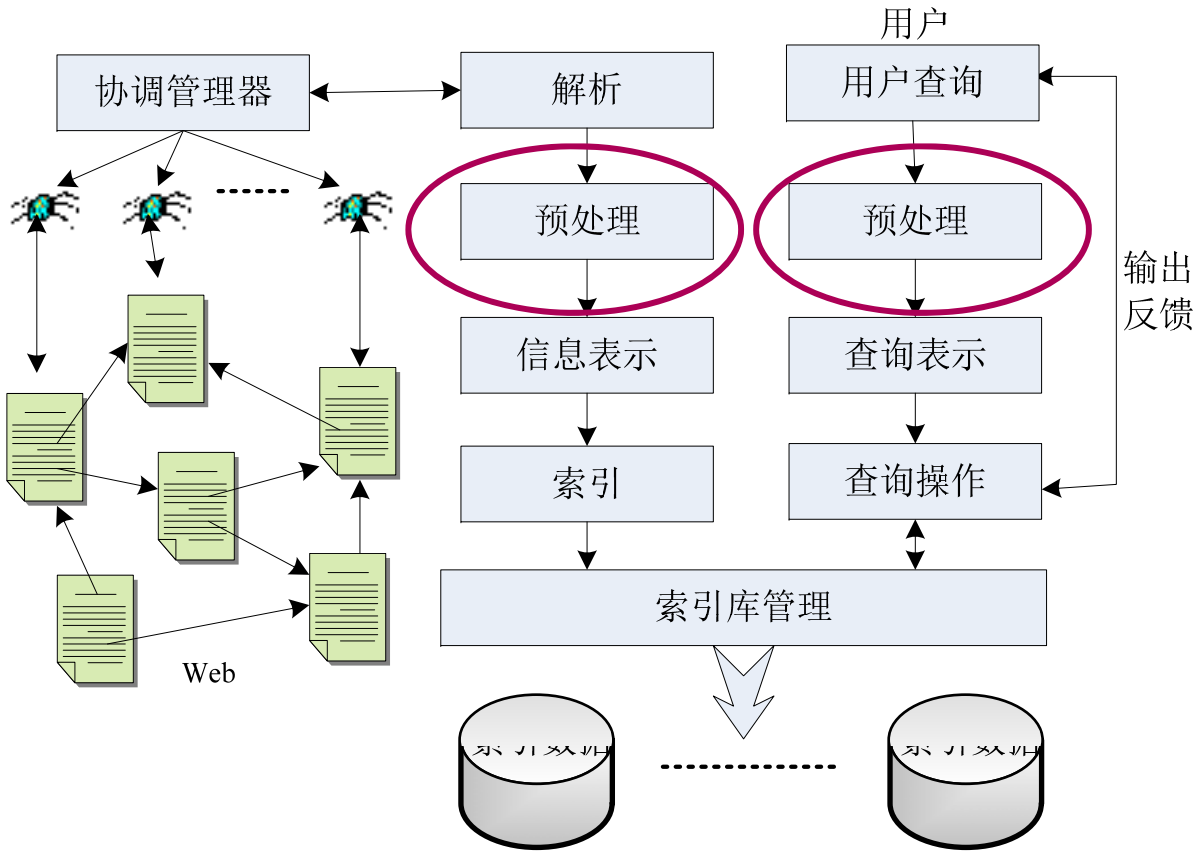
- **We want to estimate the size of the vocabulary for a corpus of 1,000,000 words. However, we only know statistics computed on smaller corpora sizes:**
  - **For 100,000 words, there are 50,000 unique words**
  - **For 500,000 words, there are 150,000 unique words**
  - **Estimate the vocabulary size for the 1,000,000 words corpus**
  - **How about for a corpus of 1,000,000,000 words?**

主要内容

- 文本特性
- 文本操作
- 索引

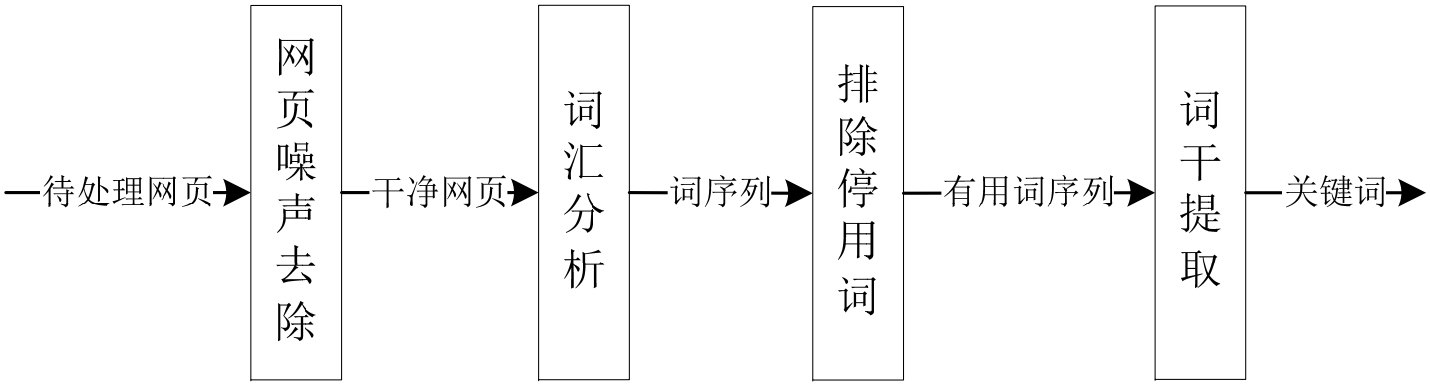
文本预处理的位置

- 在索引之前对文档集合进行预处理
- 在检索之前对查询进行预处理



文本预处理的步骤

- 网页噪声去除
- 词汇分析
- 排除停用词
- 词干提取
- 索引词的选择



主要内容

- 文本特性
- 文本操作
  - 网页去噪
  - 词汇分析
  - 排除停用词
  - 词干提取
- 索引

网站导航条

专题报告标题

精彩专题

推荐栏目或广告信息

搜索和评论

相关链接

热点新闻

精彩博文

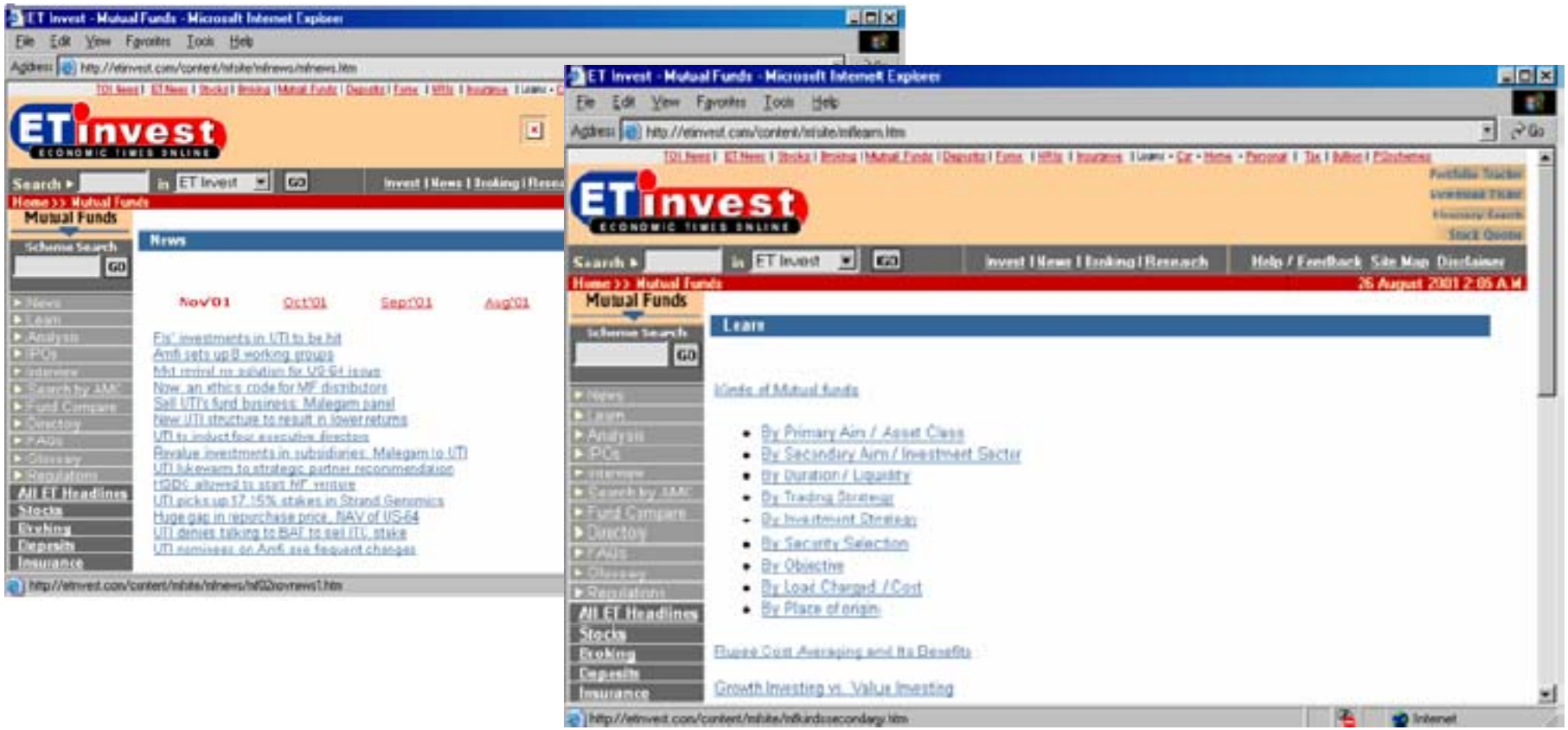
站点联系信息

网页噪声去除

- 网页中的内容除了主题内容外，还通常包含广告、版权声明、导航条等噪音信息，这些噪音内容会对基于网页内容的应用造成影响
- 网页噪声：主题内容无关的导航条、广告信息、版权信息以及调查问卷等内容，被称为“噪音”内容，或噪声网页
- 网页噪声干扰了网页原本的内容表现，影响了检索性能，必须去除
- 网页去噪过程就是保留网页中包含主题内容的内容块而去掉包含噪音内容的内容块

模板去噪

- 对于一个网站，很多网页是同一个模板生成的（基于底层的数据库），由模板生成的网页结构布局是基本一致的



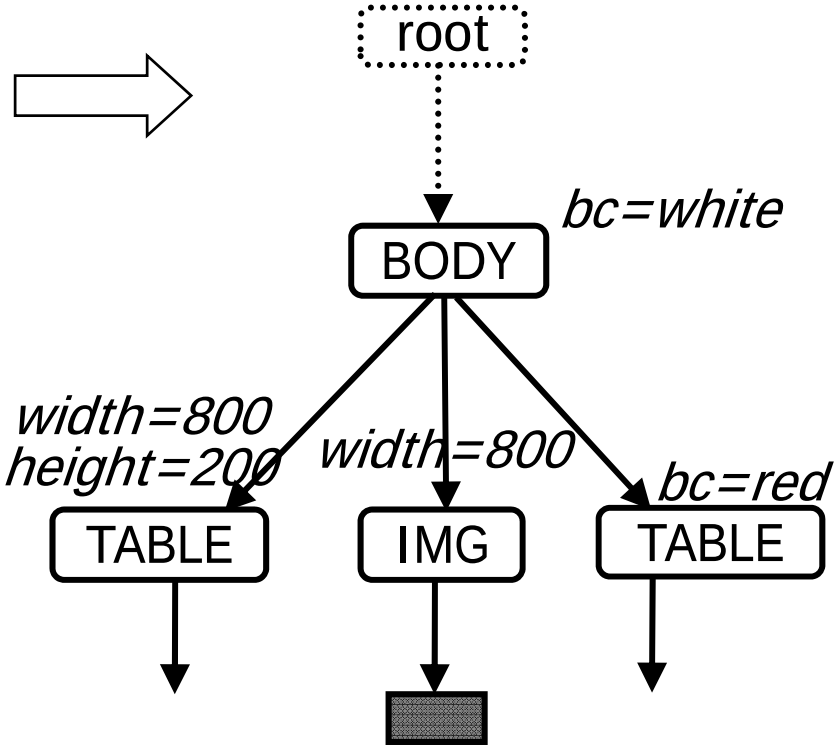
DSE算法

- **DSE（Data-rich Section Extraction）思想：同一个网站的页面的结构都是相似甚至是相同的。因此通过选择一个模板页面，目标页面和模板页面进行匹配，去掉相同部分（广告，导航信息等），剩下的就是主题内容。**
- **步骤：**
  - **第一步：确定模板页面**
    - **选择URL地址与目标页面的URL地址相似度比较大的页面作为模板页面**
  - **第二步：构建目标页面和模板页面的DOM树**
  - **第三步：匹配目标页面和模板页面的文档树，除去相同部分，提取主题**

J. Wang and Lochovsky, F.H., Data-rich section extraction from HTML pages, Web Information Systems Engineering (WISE) 2002.,Dec. 2002

网页的DOM树表示

```
<BODY bgcolor=WHITE>
   <TABLE width=800 height=200 >
      …
   </TABLE>
   <IMG src="image.gif" width=800>
   <TABLE bgcolor=RED>
      …
   </TABLE>
</BODY>
```



- 利用**JTidy（http://jtidy.sourceforge.net/）构造DOM树**

匹配过程

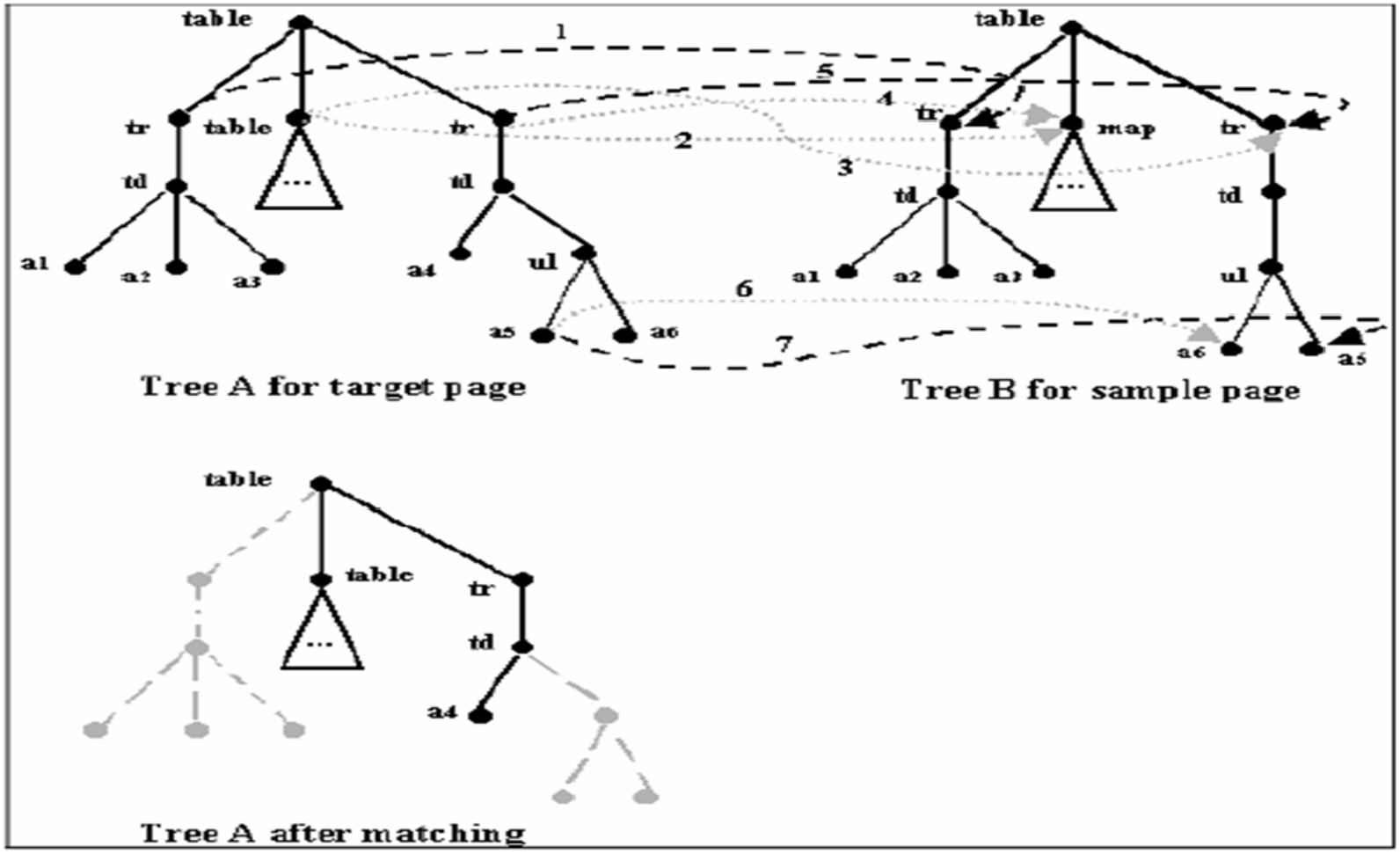


table
tr
table
td
a1
a2
a3
a4
ul
a5
a6
Tree A for target page
table
tr
map
td
a1
a2
a3
ul
a6
a5
Tree B for sample page
1
2
3
4
5
6
7
table
table
tr
td
a4
Tree A after matching

图12-6

主要内容

- 文本特性
- 文本操作
  - 网页去噪
  - 词汇分析
  - 排除停用词
  - 词干提取
- 索引

文本的词汇分析

- 把文献中的文本字符序列转换为单词（索引词的候选）序列的过程
  - 英文词法分析：书写时英文词之间通常通过空格或者标点进行区分，因此从英文字符串变成英文词是相对比较容易的。
  - 中文词法分析：书写时通常没有空格，需要分词

英文词法分析

- 数字的处理：一般可忽略
  - 如身份证号、手机号，年号（奥运2008）可能是非常好的索引项
- 连字符的处理：一般可去除
  - 如state-of-the-art变成state of the art
  - 但有些连字号中的词不宜分开，如B-49(一款机型号)
- 标点符号的处理：一般可去除
  - 但是当句点是词的一部分的时候，需要保留，如：510B.C 和 x.name
- 大小写的处理（英文）：一般进行统一
  - 但某些情况下，同一个单词的大小写含义不一样，如: China 和china
- 进行词法分析时需要考虑引入一些规则方法

汉语里的单词（Word）？

- 语素是最小的语音语义结合体，是最小的语言单位
- 词是代表一定的意义，具有固定的语音形式，可以独立运用的最小的语言单位
  - 独立地运用于造句之中的语素也是词—单纯词
- 短语是两个或以上的实词的组合，是大于词而小于句子的语言单位
  - 泛珠三角、磁悬浮列车、按劳分配
- 汉语单词例子
  - 天，葡萄，牙齿，中华民族，共青团员
- **26.7%单字词（unigram）, 69.8%双字词（bigram）, 2.7%的三字词（trigrams）**
  （现代汉语词典）

中文分词技术

- 统计方法
  - 互信息计算
  - 前提：大的文档集
- 基于规则的方法
  - 最长正向匹配
  - 前提：好的词典
- 混合方法
  - 规则+统计

统计方法

- 计算中文双字词的互信息（**mutual information**）

$$I(x,y)=\log_2(p(x,y)/(p(x)*p(y)))$$
$$=\log_2((f(x,y)/N)/((f(x)/N)*(f(y)/N)))$$
$$=\log_2((f(x,y)*N)/(f(x)*f(y)))$$

$f(x)$：双字词的第一个词出现的频率

$f(y)$：双字词的第二个词出现的频率

$f(x,y)$：双字词出现的频率

- 紧密相关： $I(x,y)$的值较大
- 不相关： $I(x,y)$接近于0
- 不相关： $I(x,y)$ 为负

例子：统计方法

文字：中国大陆新发现的油田

Bigrams	*f*(*x*)	*f*(*y*)	*f*(*x*,*y*)	*I*(*x*,*y*)
中国	615,222	925,353	228,090	4.69
国大	925,353	417,826	6,791	0.18
大陆	417,826	15,331	6,946	6.13
陆新	15,331	256,559	22	-1.46
新发	256,559	328,500	1,058	-0.30
发现	328,500	139,630	11,946	4.07
现的	139,630	2017,406	4,340	-0.00
的油	2017,406	26,690	676	-0.30
油田	26,690	24,869	2,412	7.87

中国 大陆 新 发现 的 油田

基于 TREC-5的中文文档集

基于规则的匹配

- 基于词典的方法：给出一部词典，根据这部词典进行匹配基于
- 最大匹配（**Maximum matching**）：将匹配词典条目的最大序列做为词
- 最小匹配（**Minimum matching**）：将匹配词典条目的最短序列作为词
- 正向（**Forward**）：从句子的头开始
- 反向（**Backward**）：从句子的末端开始

正向最大匹配：

中国 大陆 新 发现 的 油田

例子：基于规则匹配

0	1	2	3	4	5	6
他	说	的	确	实	在	理

- 正向最大匹配

指针位置	剩余词串	首字	最大匹配词条
0	他说的确实在理	他	他
1	说的确实在理	说	说
2	的确实在理	的	的确
4	实在理	实	实在
6	理	理	理

- 逆向最大匹配

指针位置	剩余词串	尾字	最大匹配词条
6	他说的确实在理	理	在理
4	他说的确实	实	确实
2	他说的	的	的
1	他说	说	说
0	他	他	他

中文分词例子

华南理工大学是直属教育部的全国重点大学，坐落在南方名城广州。校园分为两个校区。北校区湖光山色交相辉映，绿树繁花香飘四季，民族式建筑与现代化楼群错落有致，环境优美清新，文化底蕴深厚，是教育部命名的"文明校园"。南校区是一个环境优美、设施先进、管理完善、制度创新的现代化校园，是莘莘学子求学的理想之地。

华南/ns 理/n 工/j 大学/n 是/v 直属/b 教育部/n 的/u 全国/n 重点/n 大学/n ，/w 坐落/v 在/p 南方/s 名城/n 广州/ns 。/w 校园/n 分为/v 两/m 个/q 校/ng 区/n 。/w 北/vg 校/ng 区/n 湖光山色/i 交相辉映/i ，/w 绿/a 树/n 繁花/n 香/n 飘/v 四季/n ，/w 民族/n 式/k 建筑/n 与/c 现代化/vn 楼群/n 错落有致/i ，/w 环境/n 优美/a 清新/a ，/w 文化/n 底蕴/n 深厚/a ，/w 是/v 教育部/nt 命名/v 的/u "/w 文明/a 校园/n "/w 。/w 南/j 校/ng 区/n 是/v 一个/m 环境/n 优美/a 、/w 设施/n 先进/a 、/w 管理/vn 完善/v 、/w 制度/n 创新/v 的/u 现代化/vn 校园/n ，/w 是/v 莘莘学子/n 求学/v 的/u 理想/n 之/u 地/n 。/w

By ICTCLAS: http://www.nlp.org.cn/project/project.php?proj_id=6

中文分词存在的问题

- 中文分词歧义
  - “在海湾大学生活象白纸”
    - 真/伪歧义
  - “部分居民生活水平”
    - 分居、居民、民生、生活、
    - 交集型
  - “老人家在天津”
    - 老人、老人家
    - 组合型
- 未登录词
  - “非典”
  - 专有名词（人名、地名、机构名、译名、术语等）、新词

在Web上可能会有更多的问题

- 字符集与编码（**charset and encoding**）
  - **GB2312,GBK,BIG5,HZ →UNICODE**
    - 很难识别真正的**web**页面 **charset/encoding**集 （**WEB!**）
- 简体繁体转换
  - 乾杯，乾坤

主要内容

- 文本特性
- 文本操作
  - 网页去噪
  - 词汇分析
  - 排除停用词
  - 词干提取
- 索引

去除停用词

- 文献集中频频出现（如大于80%）的单词
  - 冠词、介词、连词通常是停用词
  - 可能其他词，构造停用词表
- 优点：降低了索引空间
- 缺点：可能降低查全率
  - 有时消除的停用词对检索是有意义的，如：“的士”中的“的”“to be or not to be”，因此有些搜索引擎直接采用全索引（full index）
- 消除方法：
  - 查表法：建立一个停用词表，通过查表的方式去掉停用词
    - SMART’s commonword list
    - WordNet stopword list（http://wordnet.princeton.edu）
  - 基于DF的方法：统计每个词的DF，如果超过总文档数目的某个百分比(如80%)，则作为停用词去掉。

停止词例子

- 英文停止词:
  - ***a, about, above, across, after, again, against, all, almost, alone, along, already, also, although, always, among, an, and, another, any, anybody, anyone, anything, anywhere, are, area, areas, around, as, ask, asked, asking, asks, at, away, b, back, backed, backing, backs, be, because, became…..***
  - ***and, an, by, from, of, the, with***

- 中文停止词:
  - 一 不 了 也 就 人 都 所 我们 你 我 他 她 他们 要 会 很 大 能 着 还 可以 最 自己 来 所 两 可 为 好 又 将 更 才 已 ...

主要内容

- 文本特性
- 文本操作
  - 网页去噪
  - 词汇分析
  - 排除停用词
  - 词干提取
- 索引

词干提取（stemming）

- 克服词形的变化，把所有同根词转变为单一形式
  - **RECOGNIZE, RECOGNISE, RECOGNIZED, RECOGNIZATION**
- 词干提取的用处:
  - 减少不同term的数量
  - 识别相似的词
  - 改进了检索性能，但不采用语言分析的方法

查表法

- 创建一个term和stem的对应表

TERM	STEM
engineering	engineer
engineered	engineer
engineer	engineer

- 表可以被索引起来，以便加快查找速度
- 创建这样的表很困难
- 存储空间的开销较大

词缀删除算法

- 词缀删除算法将**term**的前缀和（或）后缀删除，留下词干
- 大多数算法删除后缀，例如：**-SES, -ATION, -ING等等**
  - 最长匹配
    - 从词中删除最长匹配的后缀：
    - 如：**computability --> comput；singing --> sing**
    - 避免：**ability ->NULL, sing->s**
  - 迭代式最长匹配
    - 重复最长匹配的过程：
    - **WILLINGNESS --> 删除NESS --> 删除ING**

上下文有关和上下文无关

- 上下文无关
  - 根据后缀表删除后缀（或基于规则集）
- 上下文有关
  - 考虑词的其它性质，例如：**happily →happi →happy**
    - 定义一个上下文敏感的转换规则：如果一个词根以i结尾，i前面是p，那么将i转换为y
- 需要控制许多例外规则，如
  - 从**TABLE**中删除**-ABLE**不行，从**GAS**中删除**-S**也不行
  - 有时需要删除“双写字母” **FORGETTING →FORGET**

Porter算法

- 基于规则的词干提取（1980年提出）
- 每一步有一组上下文无关或有关的规则用来删除后缀，或者将其转换为其它形式
  - 上下文无关规则：**sses → ss, ies → i, s → NULL**
  - 上下文有关规则：**(*v*) : ed → NULL, ing → NULL;**
  - **(*v*)**的含义是：词根必须包含一个元音
    - **plastered → plaster**
    - **bled → bled** 删除词缀后，剩下的词干里没有元音
- 参考：
  - **http://tartarus.org/~martin/PorterStemmer/index.html**

Porter算法例子

- 原文本

for example compressed and compression are both accepted as equivalent to compress.

- 处理后

for exampl compres and compres are both accept as equival to compres.

Porter算法的缺点

- 问题：基于大量的语言知识来定义规则。但由于人类语言的复杂性，规则无法覆盖全部情况
- 可能产生无意义的词干
  - **“computer”, “computational”, “computation”** 都被简化为 **“comput”**
- 可能将实际上不同的词变为相同的
  - **organization, organ→organ**
  - **police, policy → polic**
  - **arm, army →arm**
- 可能无法识别所有的派生形态
  - **cylinder, cylindrical**
  - **create, creation**
  - **Europe, European**

中文重叠词还原

- 汉语的某些形容词有重叠式用法。这些重叠式用法是词典里所没有的，所以必须通过还原算法从重叠式用法变回到基本形式上
- 也可以看成是一种“词干”提取

形容词(AB)	ABAB 式	AABB 式	A 里 AB 式
高兴	高兴高兴	高高兴兴	
明白	明白明白	明明白白	
热闹	热闹热闹	热热闹闹	
潇洒	潇洒潇洒	潇潇洒洒	
糊涂		糊糊涂涂	糊里糊涂
流气			流里流气
粘乎	粘乎粘乎	粘粘乎乎	
凉快	凉快凉快	凉凉快快	

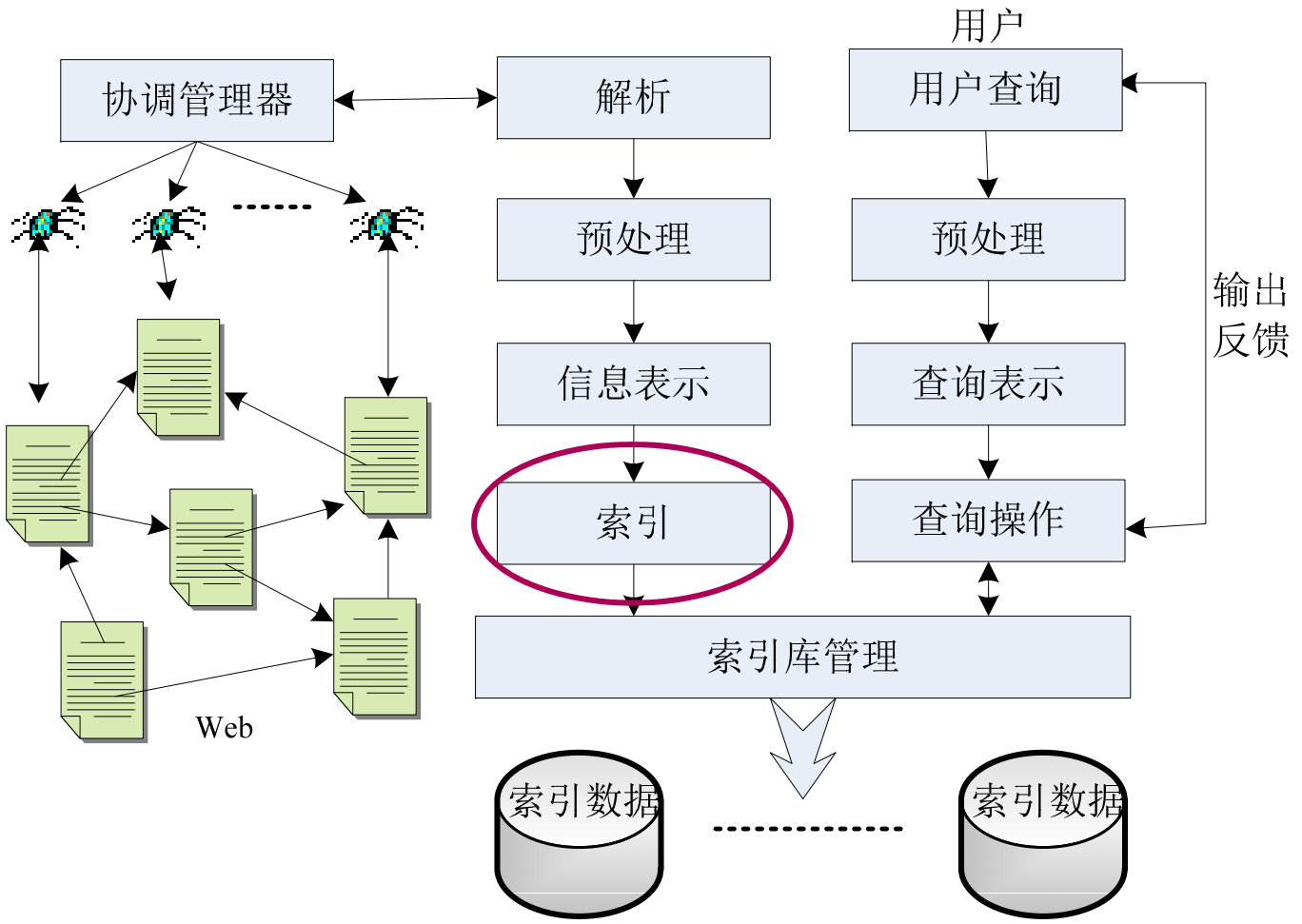
总结：文本预处理

- 目标：选择更有意义的词或者概念来表示文档
  - 选择名词
  - 选择名词和名词短语（**computer science**）
  - 选择一组组的名词（每个组内的名词比较相似，一个组可以称为一个概念）
- 有些步骤不是必须的
  - **Porter**算法对于中文网页
  - 中文分词与编码转换对于英文网页
- 但良好的处理会极大地提高检索的性能和效果

主要内容

- 文本特性
- 文本操作
- 索引

索引在信息检索系统中的位置



为什么需要索引?

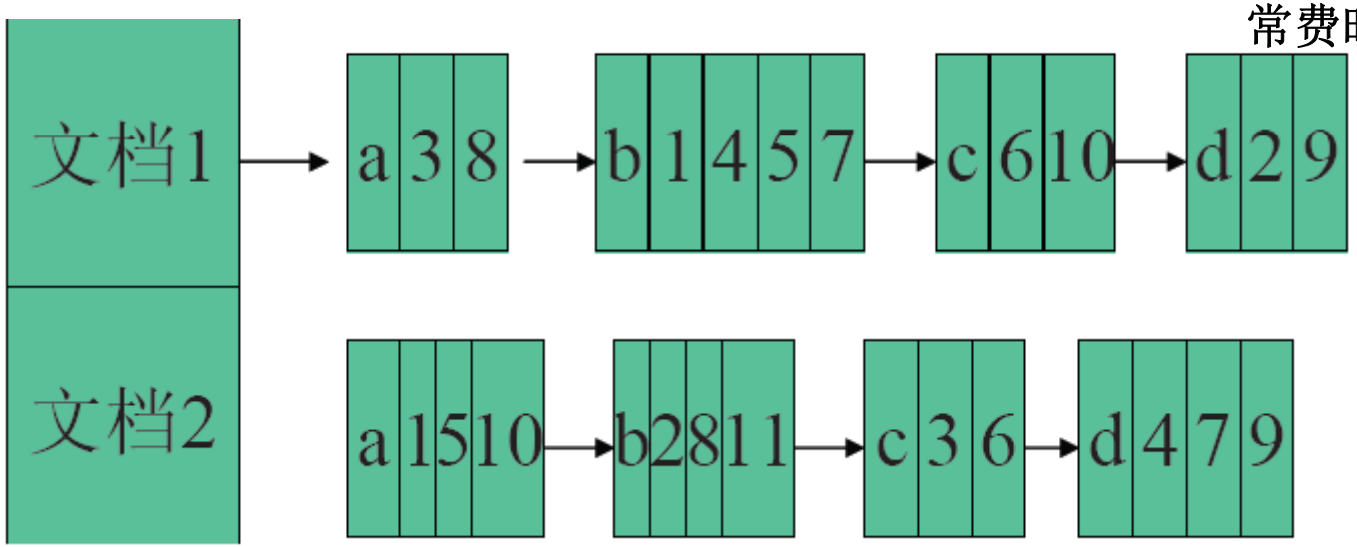
- 信息检索问题
  - 不能预测人们在查询中使用的键值
  - 文档中的每个词都是潜在的搜索词
- 在线文本搜索：
  - 最初始的想法：顺序地扫描文字，字符串匹配
- 解决方案：建立索引
  - 在文本上建立数据结构以加速搜索
  - 半静态结合（**Semi-static collections**）：在合理的期限内更新

什么样的数据结构是合适的？

- 将每篇文档表示成DocID及其文本内容组成的类向量模式—前向索引
- 例如：文档1：b d a b b c b a d c

  文档2：a b c d a c d b d a b

依次扫描每个文档，但是对于一个字符串的多次出现不需要一一扫描，如果文档数目较多，仍非常费时费力。



倒排索引的思想

- 都不在查询和文档中出现的词一般不影响**cosine**相似度
- 通常查询很短，因此查询矢量是非常稀疏（**sparse**）的
- 利用倒排索引来发现包含至少一个查询词的有限文档

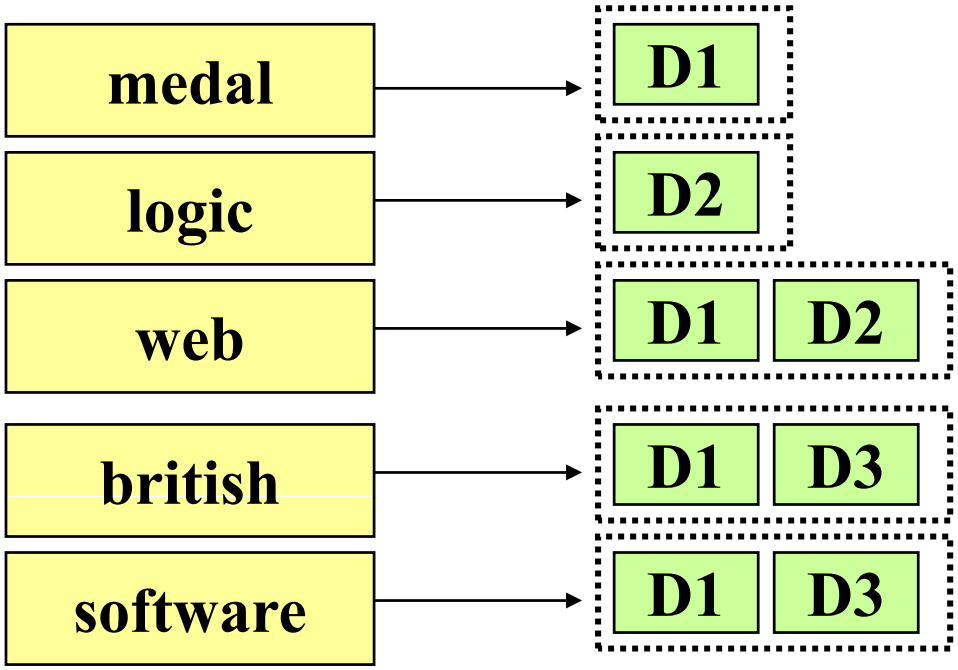
原始的文档视图

词 / 文档	medal	logic	web	british	software
D_1	1	0	1	1	1
D_2	0	1	1	0	0
D_3	0	0	0	1	1

反向视图

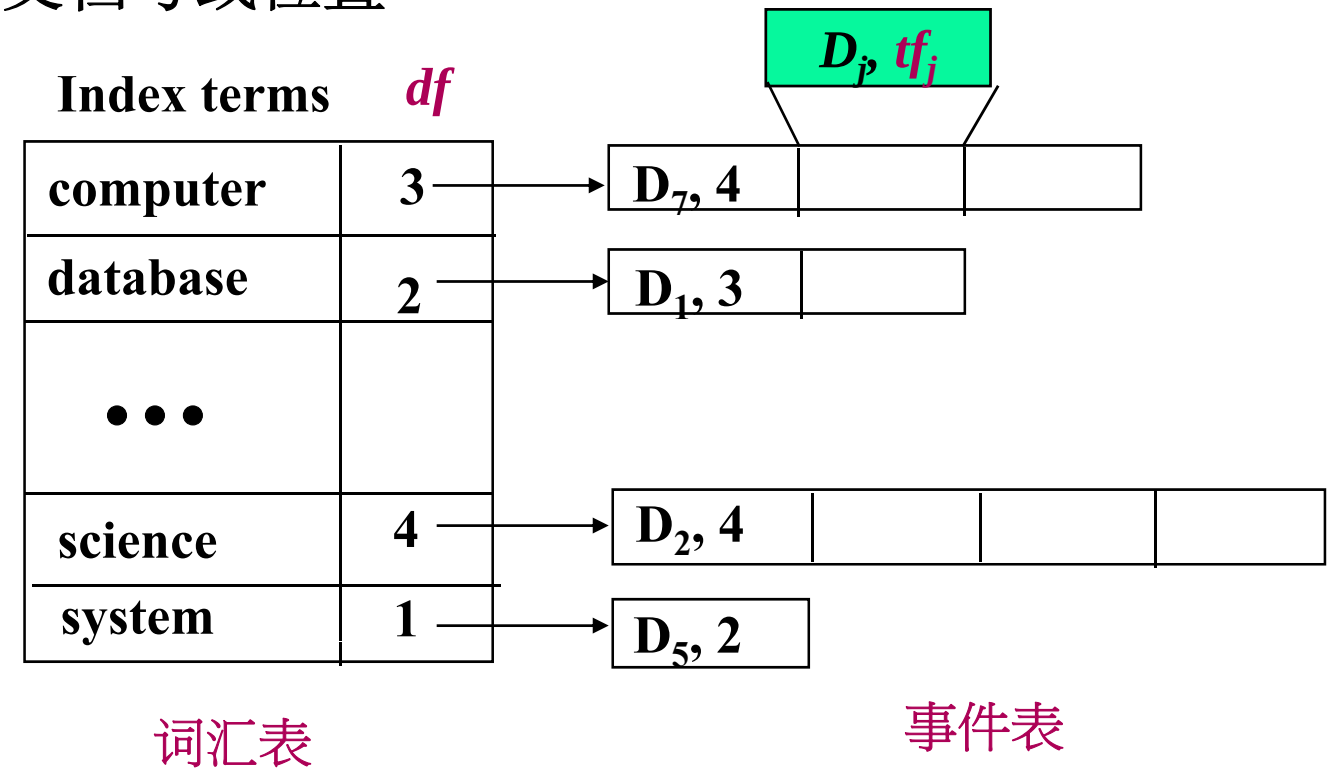
文档 / 词	D_1	D_2	D_3
medal	**1**	**0**	**0**
logic	**0**	**1**	**0**
web	**1**	**1**	**0**
british	**1**	**0**	**1**
software	**1**	**0**	**1**

反向索引，倒排文件

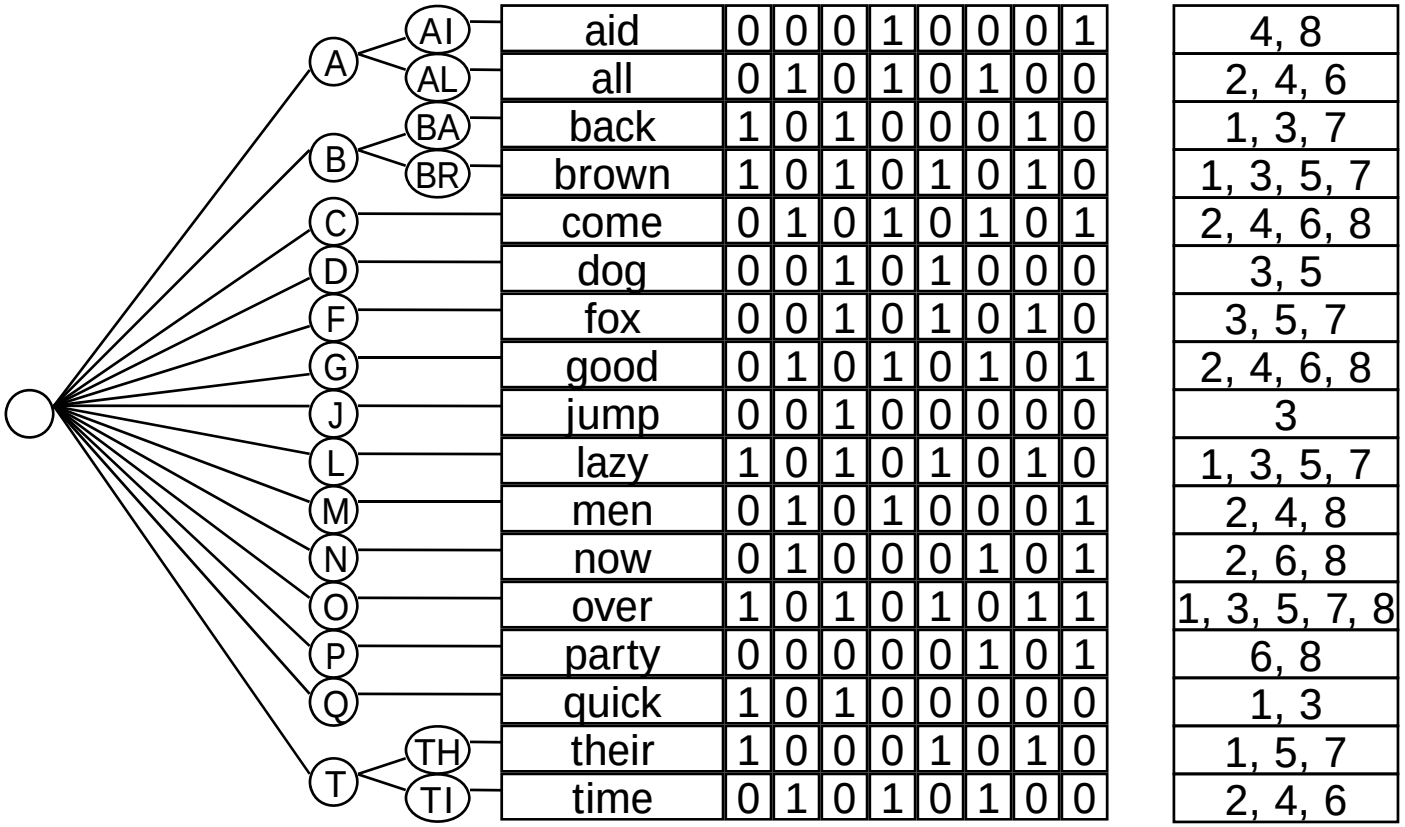


倒排文档

- 面向词的机制（**Word-oriented mechanism**）
- 由两部分组成：
  - 词汇表：文档集合中的所有不同词汇
  - 事件表（置入文件，**Posting List**）：对应单词所在的文档号或位置

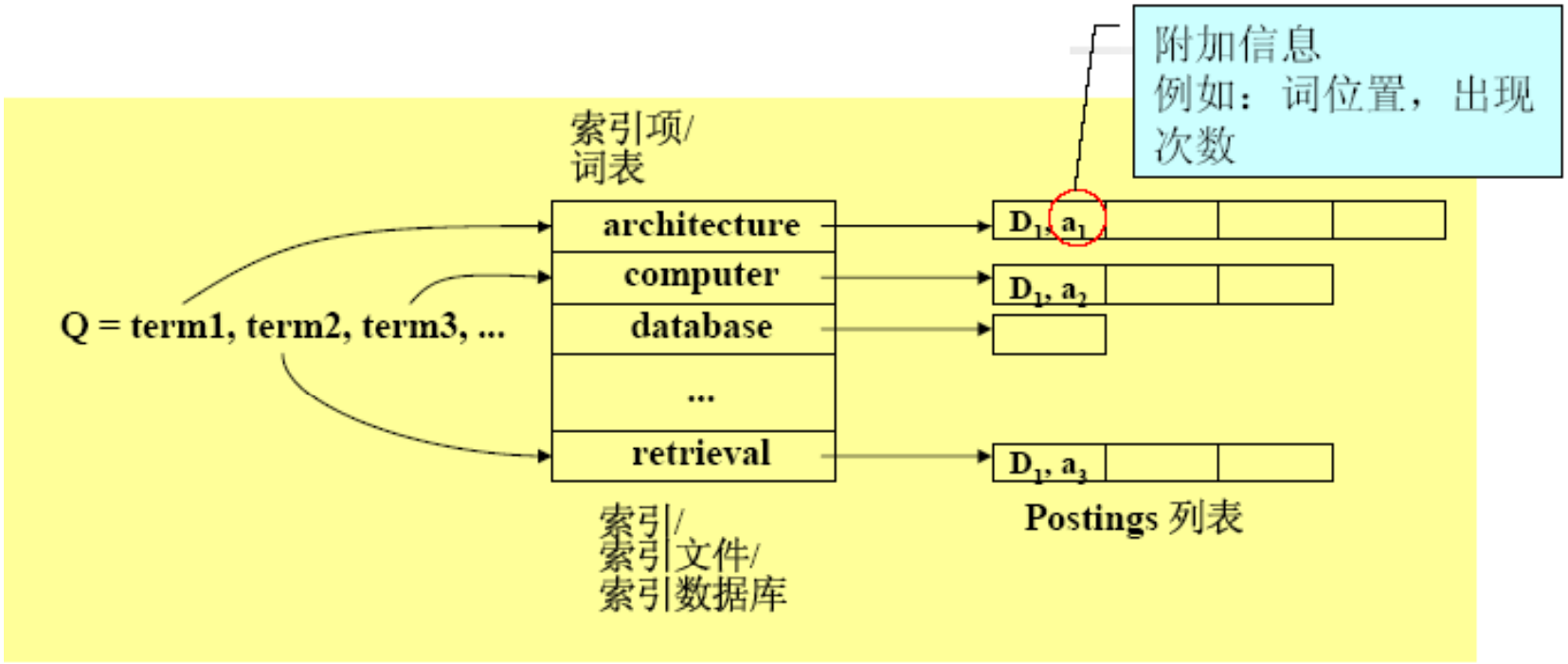


例子：倒排文档


Inverted File
Term
Doc 1
Doc 2
Doc 3
Doc 4
Doc 5
Doc 6
Doc 7
Doc 8
Postings
A
AI
AL
B
BA
BR
C
D
F
G
J
L
M
N
O
P
Q
T
TH
TI
aid 0 0 0 1 0 0 0 1 4, 8
all 0 1 0 1 0 1 0 0 2, 4, 6
back 1 0 1 0 0 0 1 0 1, 3, 7
brown 1 0 1 0 1 0 1 0 1, 3, 5, 7
come 0 1 0 1 0 1 0 1 2, 4, 6, 8
dog 0 0 1 0 1 0 0 0 3, 5
fox 0 0 1 0 1 0 1 0 3, 5, 7
good 0 1 0 1 0 1 0 1 2, 4, 6, 8
jump 0 0 1 0 0 0 0 0 3
lazy 1 0 1 0 1 0 1 0 1, 3, 5, 7
men 0 1 0 1 0 0 0 1 2, 4, 8
now 0 1 0 0 0 1 0 1 2, 6, 8
over 1 0 1 0 1 0 1 1 1, 3, 5, 7, 8
party 0 0 0 0 0 1 0 1 6, 8
quick 1 0 1 0 0 0 0 0 1, 3
their 1 0 0 0 1 0 1 0 1, 5, 7
time 0 1 0 1 0 1 0 0 2, 4, 6

一般的倒排索引

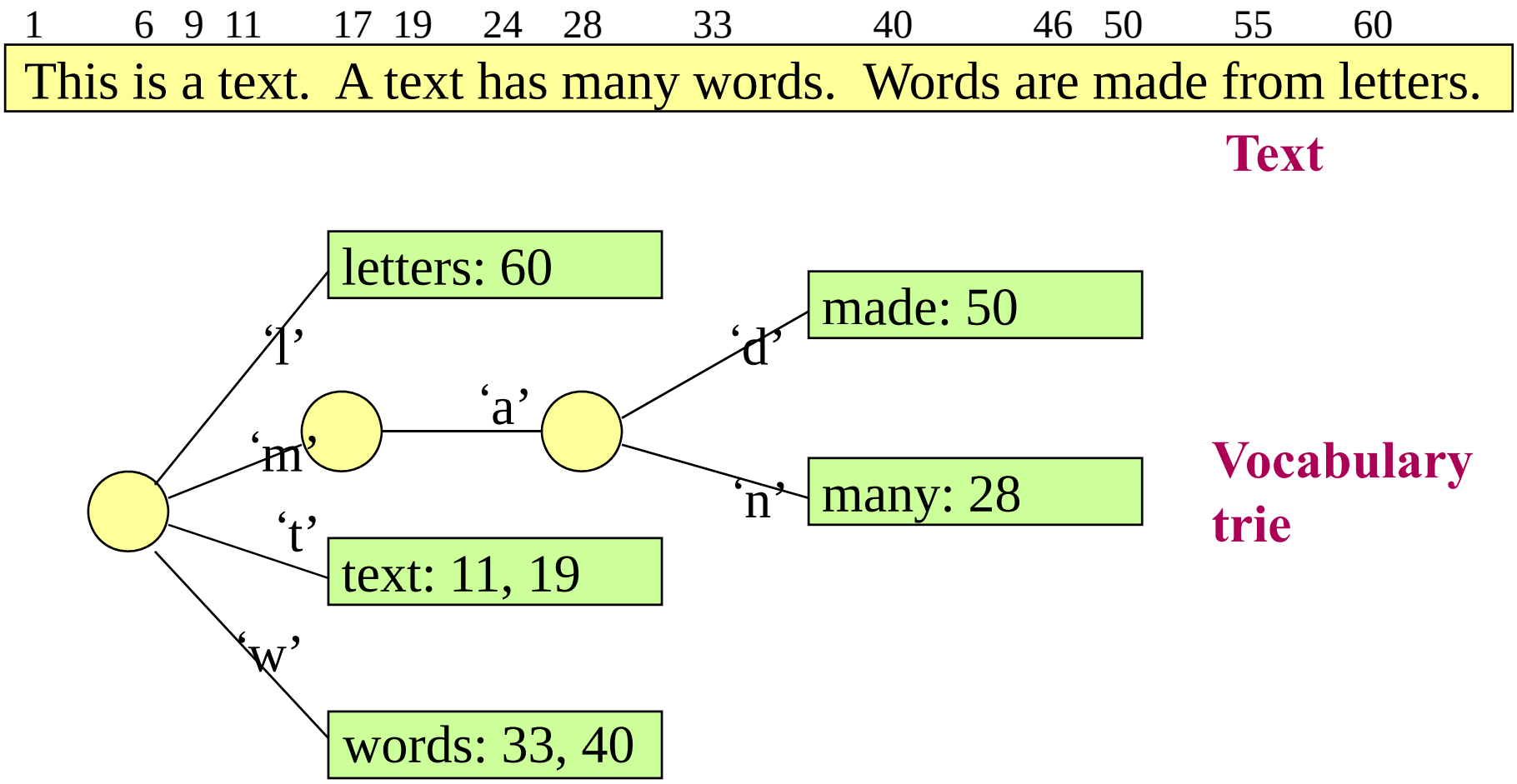
- 索引文件可以用任何文件结构来实现
- 索引文件中的词项是文档集合中的词表



词汇表结构

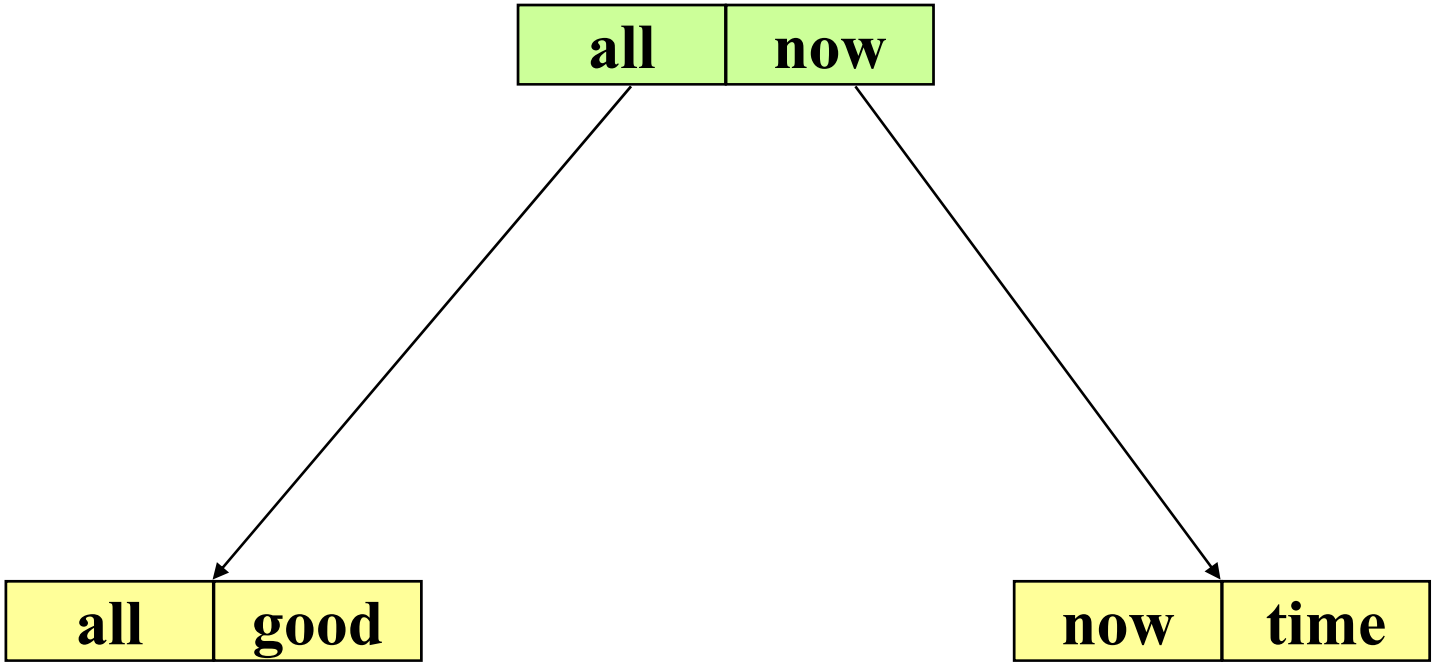
- 排序数组（**Sorted Arrays**）：采用字典序，查找采用二分法。空间消耗小，查找较快，但是插入删除麻烦
- 哈希表：通过**Hash**函数直接把词映射到地址，空间消耗和**Hash**函数设计有关。较快，插入删除容易
- 二叉搜索树、**B**树、**Trie**树等

Trie树


1 6 9 11 17 19 24 28 33 40 46 50 55 60
This is a text. A text has many words. Words are made from letters.
Text
letters: 60
made: 50
‘l’
‘d’
‘a’
‘m’
‘n’
many: 28
‘t’
text: 11, 19
‘w’
words: 33, 40
Vocabulary trie

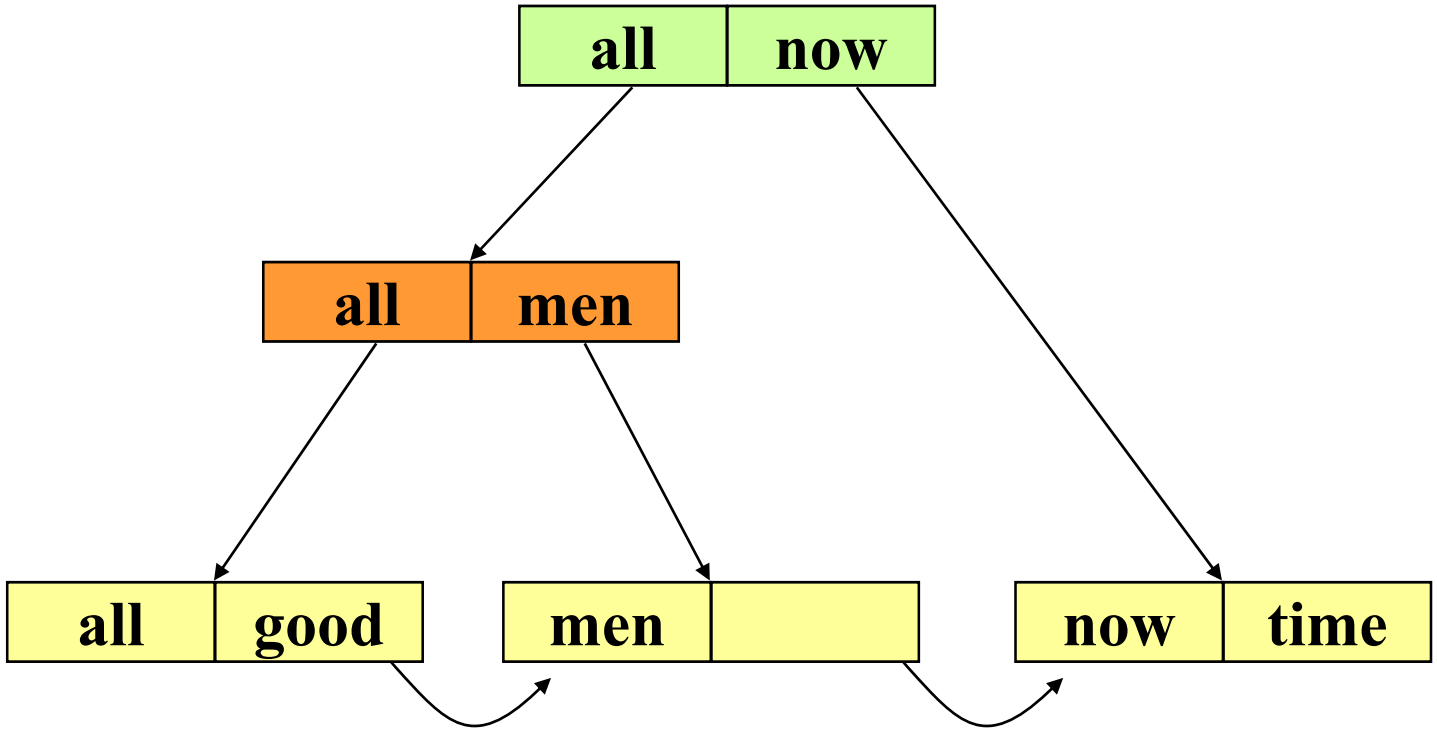
B+ 树

Now is the time for all good …



加一个新词

Now is the time for all good men …



事件表的内容

- 内容依赖于检索模型
- 特征
  - 布尔量
  - 统计值：***tf, df, ctf, doclen, maxtf)***
- 位置信息
  - 特征在文档中的位置
  - 粒度（**Granularities**）可以是单词、句子或段落 等
  - 粗粒度的索引（分块寻址），可能准确度较低，但是占用较低的空间
  - 单词水平粒度可以有约 **20-30%** 的压缩空间

事件表

- 布尔查询
  - 只需记录文档号
- 排序查询
  - 文档号和权重（TF*IDF, ...）
- 模糊查询及短语查询
  - 每个词出现的文字

D4, D8
D2, D4, D6
D1,D3,D7

D4:2, D8:3
D2:1, D4:3, D6:2
D1:1, D3:1, D7:4

D4(17,36), D8(3,45,200)
D2(44), D4(10,20), D6(8,37)
D1(18), D3(6), D7(5,9,31,45)

如果存储了足够多的信息，则可以支持复杂的检索操作

倒排文档的一般性质

- 它是面向单词的索引技术，为文本集建立索引，加快检索的速度
- 由两种元素构成：词汇表和事件表
- 词汇表的空间占用遵循**Heaps**定律（如**1G**的**TREC-2**文献集占用**5M**）
- 事件表占用的空间相对较大，为**30~40%**

例子

- 假设有一个中文文档集，包括4个文本，分别是：

T_1=南边来了个哑巴，腰里别了个喇叭

T_2=北边来了个喇嘛，手里提了个獭犸

T_3=喇嘛回家炖獭犸

T_4=哑巴嘀嘀哒哒吹喇叭

请建立倒排索引

例子：文本编号及单词位置示意

编号	文本内容和单词位置
T_1的单词序号	1 2 3 4 5 6 7 8 9 10
文本1：	南边 来 了 个 哑巴，腰里 别 了 个 喇叭
T_2的单词序号	1 2 3 4 5 6 7 8 9 10
文本2	北边 来 了 个 喇嘛，手里 提 了 个 獭狗
T_3的单词序号	1 2 3 4
文本3	喇嘛 回家 炖 獭狗
T_4的单词序号	1 2 3 4
文本4	哑巴 嘀嘀哒哒 吹 喇叭

例子：多文本的倒排索引

词汇表	事件表（文本号: 词位置）
北边	（2 : 1）
别	（1 : 7）
吹	（4 : 3）
滴滴答答	（4 : 2）
炖	（3 : 3）
回家	（3 : 2）
来	（1 : 2），（2 : 2）
喇叭	（1 : 10），（4 : 4）
喇嘛	（2 : 5），（3 : 1）
南边	（1 : 1）
手里	（2 : 6）
獭犸	（2 : 10），（3 : 4）
提	（2 : 7）
哑巴	（1 : 5），（4 : 1）
腰里	（1 : 6）

扩展—布尔检索（1）

- 一个布尔检索包含n个用布尔操作符连接的词项，例如：“**computer AND news AND NOT newsgroup**”
- 每个**term**从倒排索引中返回一个**postings list**
  - 如果**term**不在任何文档中出现，则**postings list**为空
- 检索结果根据逻辑关系相结合：
  - **AND:** 集合做交运算
  - **OR:** 集合做并运算
  - **NOT:** 集合做差运算

扩展—布尔检索（2）

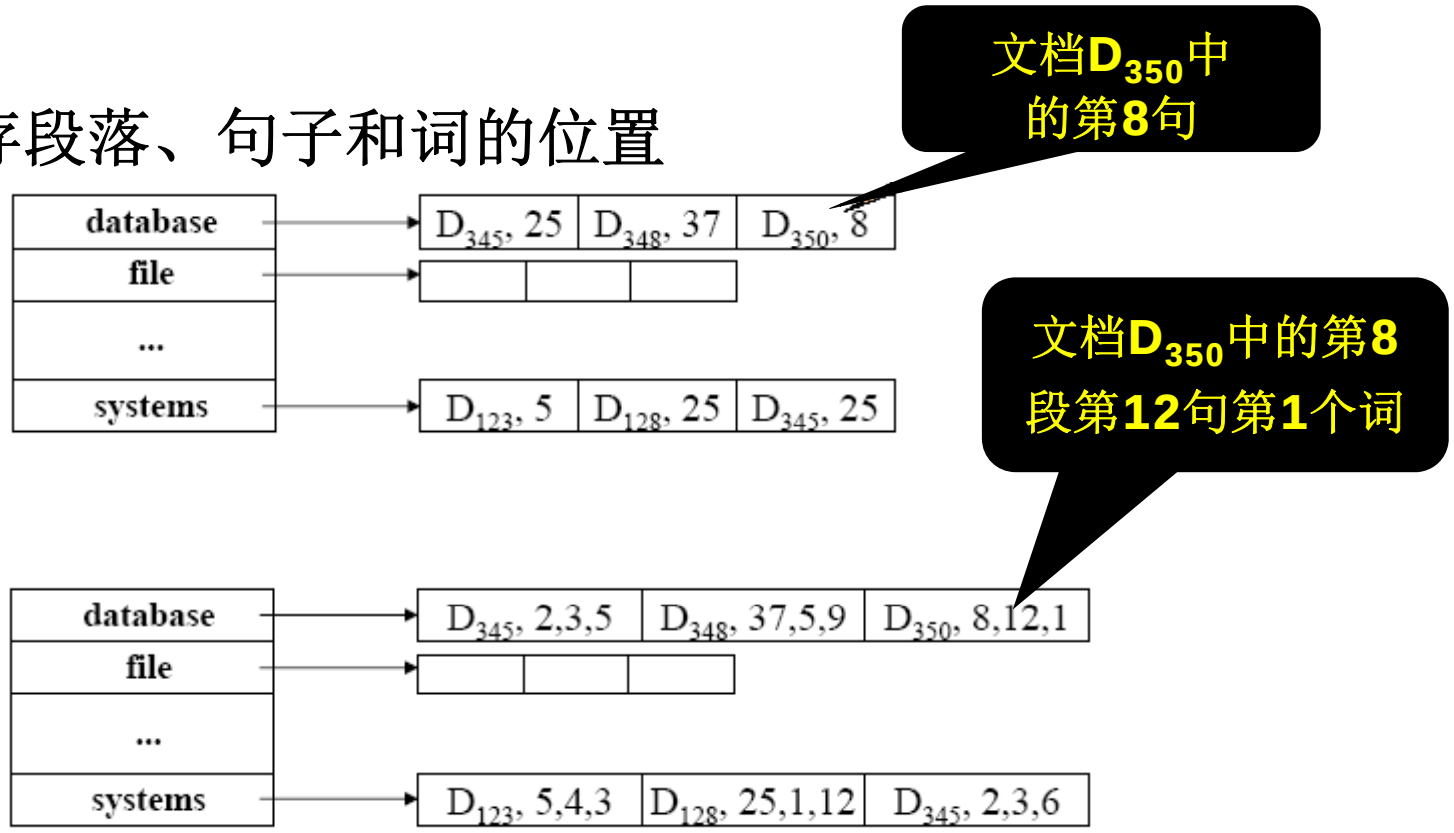
- 从最短的事件表开始做“与”操作，保证中间结果越小越好
- “网络” AND “病毒” AND “蠕虫”
  - 从哪个词项开始做交运算呢？
  - 显然是：“病毒”和“蠕虫”

扩展—距离约束（1）

- 常常需要知道邻接条件，例如：
  - “database” 后面紧跟着 “systems”
    - 例如：短语搜索 “database systems”
  - “database”和 “systems”之间不能间隔超过3个词
  - “database”和 “architecture”在同一个句子里
- 需求扩展：
  - 倒排索引中保存着关键词在文档中的位置，文档的组成单元(标题, 小标题, 句子分割标记等)
  - 检索算法和位置信息相关联，并需检查文档的组成单元

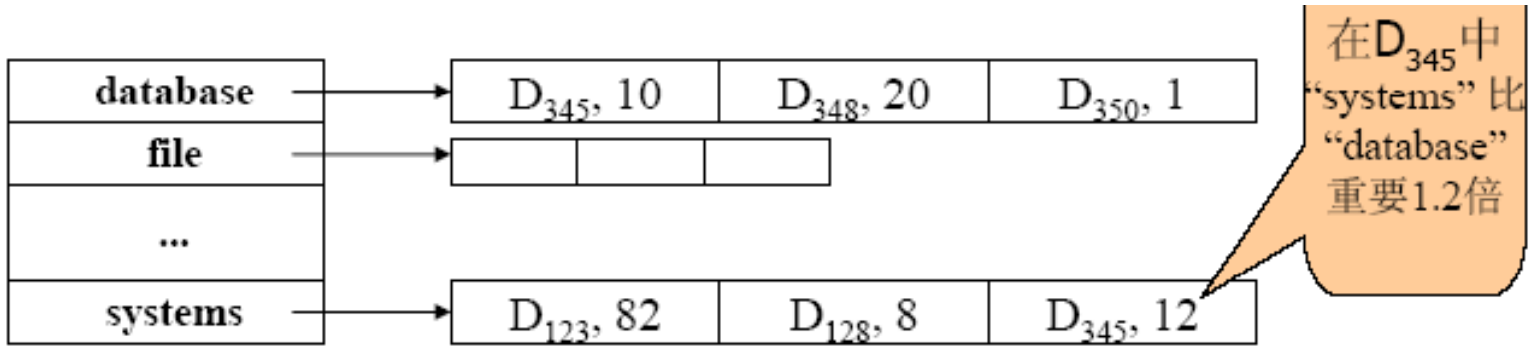
扩展—距离约束（2）

- 保存在倒排表中的位置信息
  - 保存句子位置
  - 保存段落、句子和词的位置



扩展– Term的权重

- 可保存出现频率，以便支持基于统计的检索
- 事件表中的第二个单元可以是该term的权重（例如，可以被归一化在0和1之间），或者是该term的出现频率



扩展—同义词

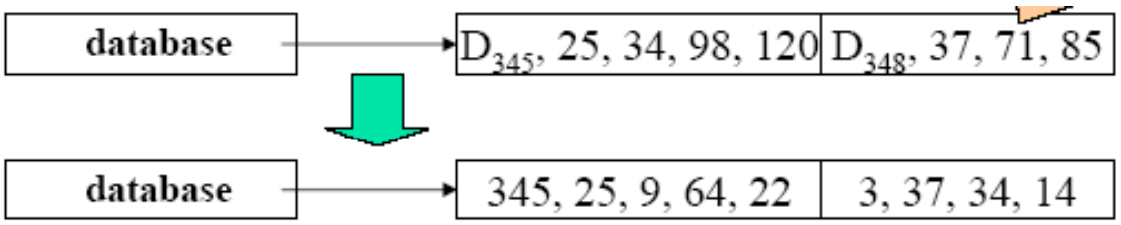
- 同义词对于提高召回率很有意义
- 同义词可以通过指针指向同一个事件表

扩展– Term截断

- 后缀截断是stemming的简单形式:
  - comput*
    - computer, computing, computation, etc.
- 如果词汇表用trie来实现，则很容易进行后缀截断
- 在B+树上进行这种操作就有点难度（例如：可能需要使用一个映射表将comput*映射为compute, computer, computing等）
- 如果使用hash，则不会太灵活

倒排文档的压缩

- 词汇表和事件表的压缩问题
  - 分配给文档ID和词位置的字节数
  - **16 bits不够，32 bits又浪费空间**
  - 仅记录相邻文档ID和词位置之间的差异
    - 类似视频压缩中仅记录本帧和上一帧的差异

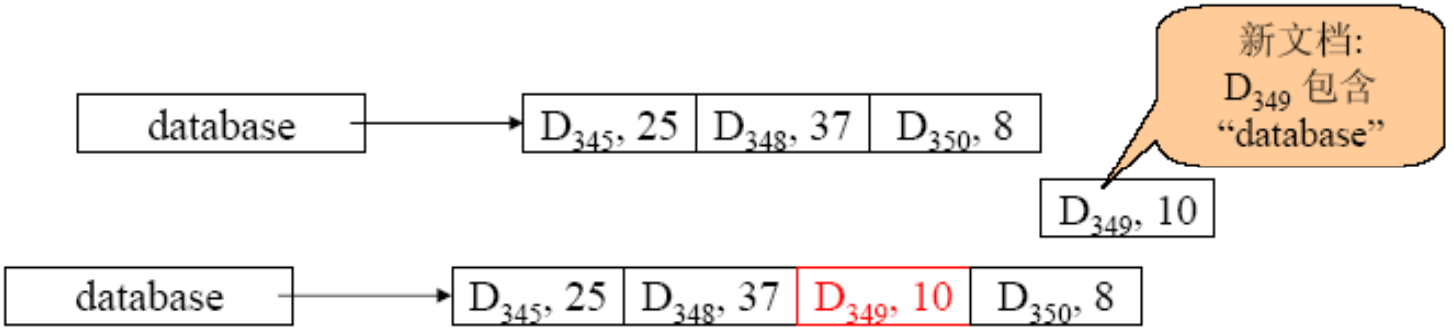


例子：索引压缩

- 事件表 (需要 7 bytes = 56 bits)
  - 37, 42, 43, 48, 97, 98, 243
- 求差
  - 37, 5, 1, 5, 49, 1, 145
- 优化霍夫曼编码（Optimal Huffman Code）
  - 0:1, 10:5, 110:37, 1110:49, 1111: 145
- 压缩后（17 bits）
  - 11010010111001111

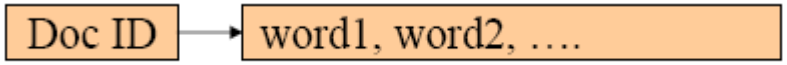
倒排索引的维护开销

- 一个文档插入时最坏的情况：
  - 当文档包含n个词，并且每个词都不重复，插入时需要更新n个事件表
- 对于事件表中的每个更新操作：
  - 如果事件表没有排序，新的事件项可以被追加到表的末端，更新操作很快，但是检索无序的事件表很慢
  - 如果事件表是排序了的，那么插入一个新的事件项需要很大的开销



删除更新

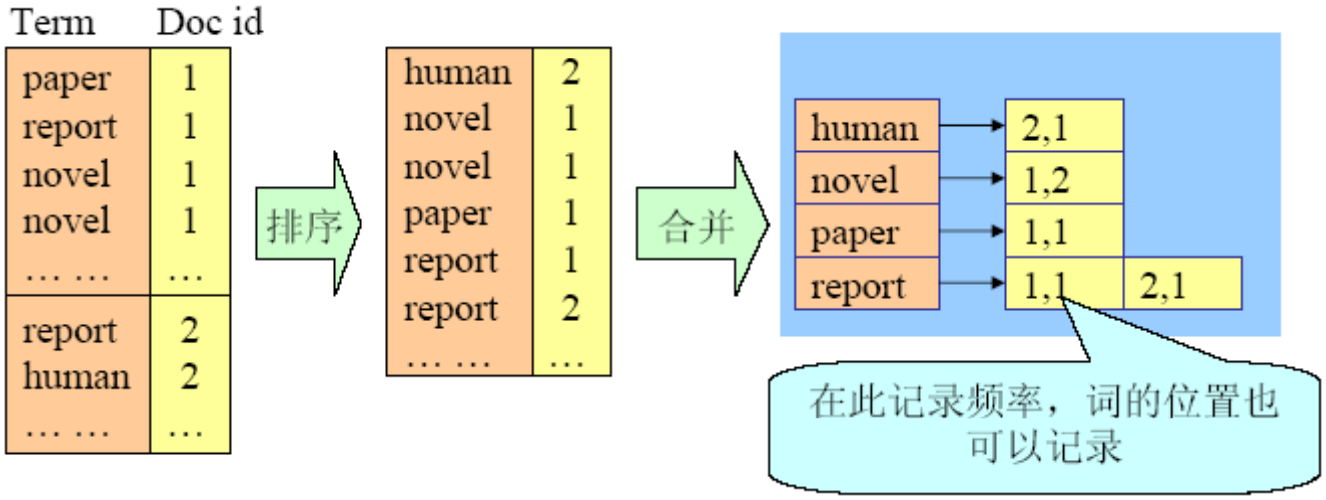
- 删除更新就是一个删除操作，后面跟着一个插入操作
- 为了支持删除操作，需要维护一个前向索引(forward index)来记录文档中包含的词



- 找到将要被删除的文档ID
- 从前向索引中获得该文档中包含的词
- 根据该文档中的词定位倒排索引中的事件项，并在这些事件项中将该文档ID删除
- 删除开销很大; 为了降低删除开销，可以：
  - 维护一个表，表中存放要删除的文档ID号（先不在倒排表上实施）
  - 在检索过程中，忽略那些被登记在表中的文档ID
  - 周期性地对倒排文件进行更新

通过排序进行批插入

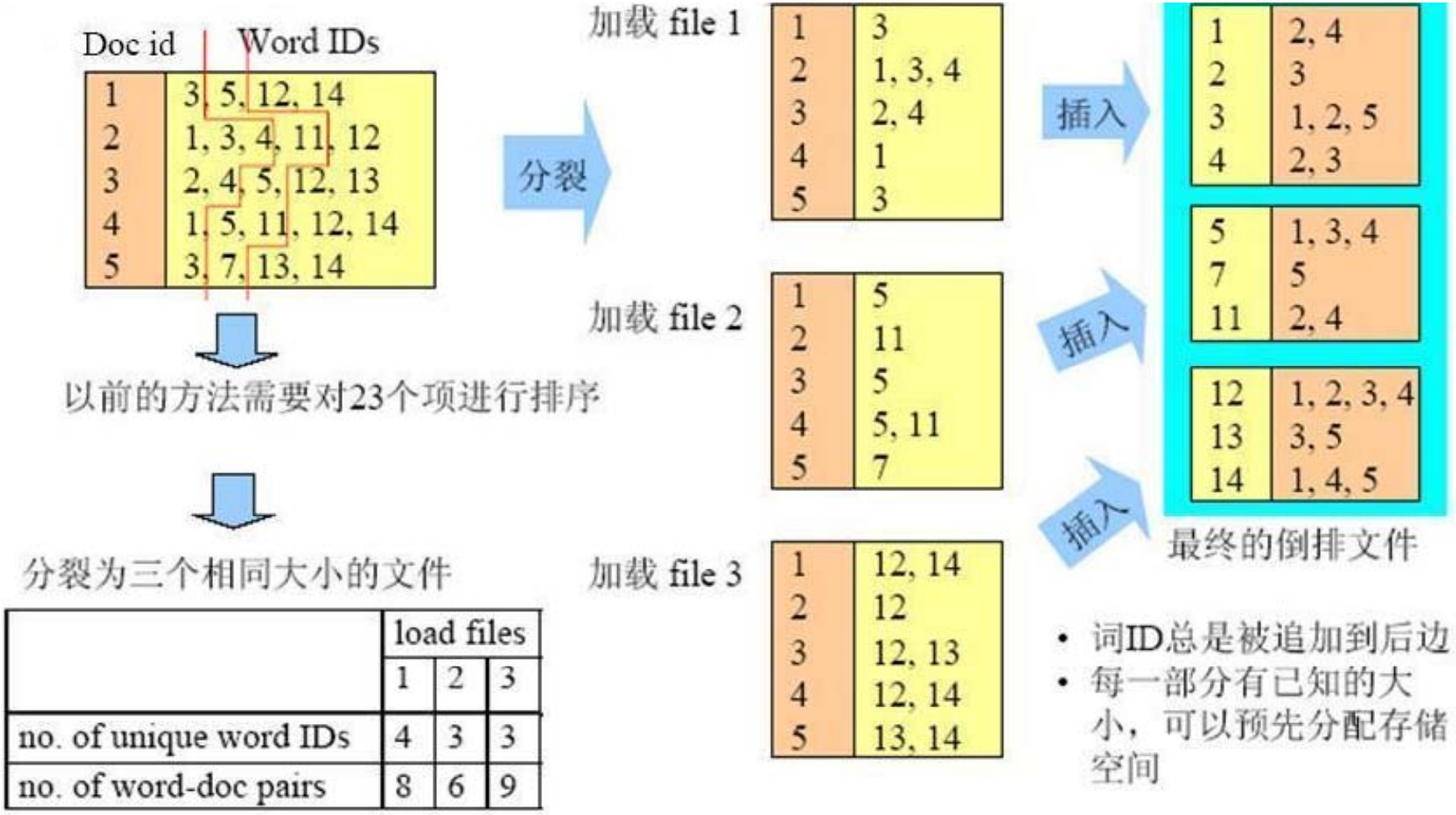
- 收集所有将被插入索引中的新文档
- 从每个文档中提取词干，并准备一个批倒排文档



快速倒排算法

- 大量文档的倒排操作
- 每批多大合适？
  - 批的大小受到主存的限制，因为需要对倒排文件进行排序
- 怎样降低倒排文档的随机扩展呢？
  - 给倒排文件分配磁盘空间

快速倒排算法实例



	load files		
	1	2	3
no. of unique word IDs	4	3	3
no. of word-doc pairs	8	6	9

索引与检索

- 索引
  - 遍历倒排文档， 看是否需要分裂
  - 按排序顺序查询事件表
  - 对大数据集可能需要几个小时或几天
- 查询处理
  - 遍历，寻找查询词
  - 读入事件表
  - 基于查询操作事件表
  - 对极大的数据集，响应时间为秒或毫秒

本讲小结

- 文本特性
  - 文本的统计特性 （4.1）
- 文本预处理技术
  - 网页去噪（4.2.1，12.2.3）
  - 词汇分析（4.2.2）
  - 排除停用词（ 4.2.2 ）
  - 词干提取（ 4.2.2 ）
- 文本索引方法
  - 倒排文件 （Inverted File），（4.3.4）
  - 后缀树及后缀数组（Suffix trees and arrays），（4.3.2）
  - 签名档（Signature File），（4.3.3）

推荐阅读和网站

- 《网络信息检索》第四章
- **An Introduction to Information Retrieval**
  - **Ch4: Index construction**
  - **Ch5: Index compression**

- **Lucene in action. Luceneinaction.pdf**
- **J. Wang and Lochovsky, F.H., Data-rich section extraction from HTML pages, Web Information Systems Engineering (WISE) 2002.,Dec. 2002**

- 计算所汉语词法分析系统**ICTCLAS:**
  http://www.nlp.org.cn/project/project.php?proj_id=6
- **Lucene: http://lucene.apache.org/**