信息检索与Web搜索

第6讲 索引压缩

Index compression

授课人：高曙明

*改编自“现代信息检索”网上公开课件（http://ir.ict.ac.cn/~wangbin）

索引压缩

什么是索引压缩?

- 是指用较少的比特存储原始索引数据

压缩分类:

- 无损压缩：压缩之后所有原始信息都被保留
- 有损压缩：压缩之后会丢掉一些信息

为什么要压缩?

- 大规模文档集的索引数据巨大，进行压缩具有以下优点:
  - 减少磁盘空间 (节省开销)
  - 加快从磁盘到内存的数据传输速度 (加快检索速度)
    - [读压缩数据到内存+在内存中解压]比直接读入未压缩数据要快很多
    - 前提: 解压速度很快
  - 增加内存存储内容 (加快检索速度)
    - 比如，可以尽可能将词典放入内存

词项的统计特性分析

N	文档数目	800,000
L	每篇文档的词条数目	200
M	词项数目(= 词类数目)	400,000
	每个词条的字节数 (含空格和标点)	6
	每个词条的字节数 (不含空格和标点)	4.5
	每个词项的字节数	7.5
T	无位置信息索引中的倒排记录数目	100,000,000

样本文档集 **Reuters RCV1** 原始统计数据

预处理后的统计数据

	不同词项			无位置信息倒排记录			词　　条①		
	数目	Δ%	T%	数目	Δ%	T%	数目	Δ%	T%
未过滤	484 494			109 971 179			197 879 290		
无数字	473 723	–2	–2	100 680 242	–8	–8	179 158 204	–9	–9
大小写转换	391 523	–17	–19	96 969 056	–3	–12	179 158 204	–0	–9
30个停用词	391 493	–0	–19	83 390 443	–14	–24	121 857 825	–31	–38
150个停用词	391 373	–0	–19	67 001 847	–30	–39	94 516 599	–47	–52
词干还原	322 383	–17	–33	63 812 300	–4	–42	94 516 599	–0	–52

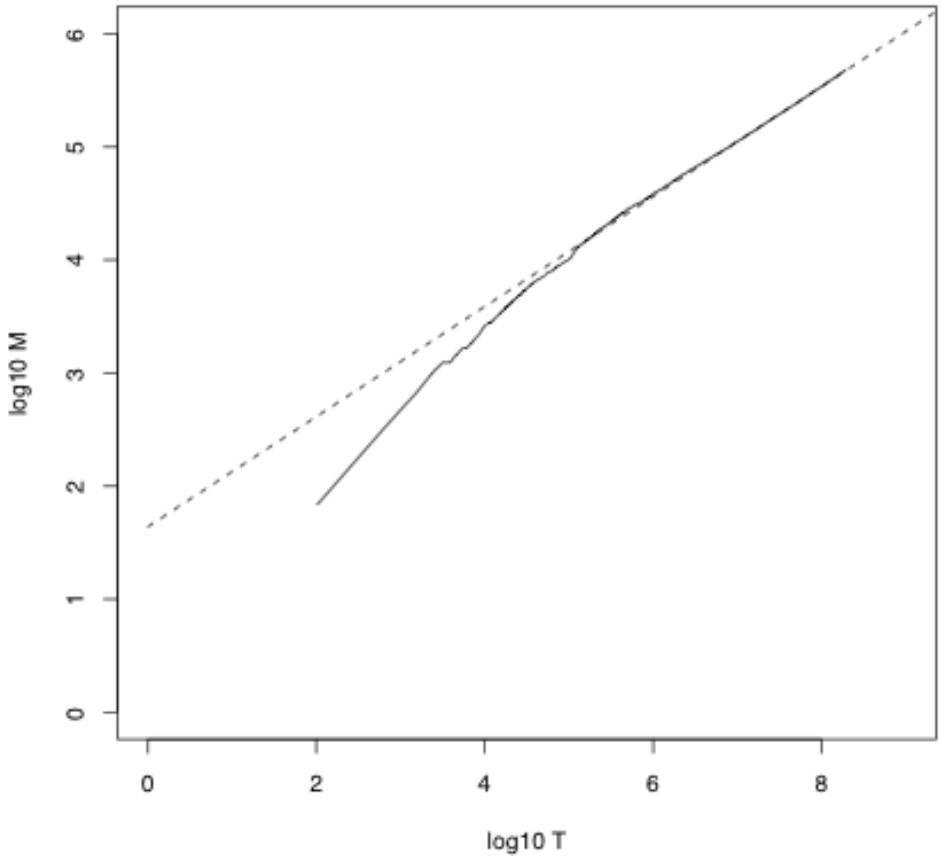
预处理对词典大小和无位置信息倒排记录数目影响很大！

词项数目的估计——Heaps定律

- Heaps定律: $M = kT^b$
- M 是词汇表大小, T 是文档集的大小(所有词条的个数)
- 参数k 和b 的一个经典取值是: $30 \leqslant k \leqslant 100$ 及 $b \approx 0.5$.
- Heaps定律在对数空间下是线性的

推论：词汇表大小会随着文档集的大小增长而增长!

Heaps定律在RCV1上的表现



- 图中通过最小二乘法拟合出的直线方程为：

$$\log_{10} M = 0.49 * \log_{10} T + 1.64$$

即：$M = 10^{1.64} T^{0.49}$

$$k = 10^{1.64} \approx 44$$

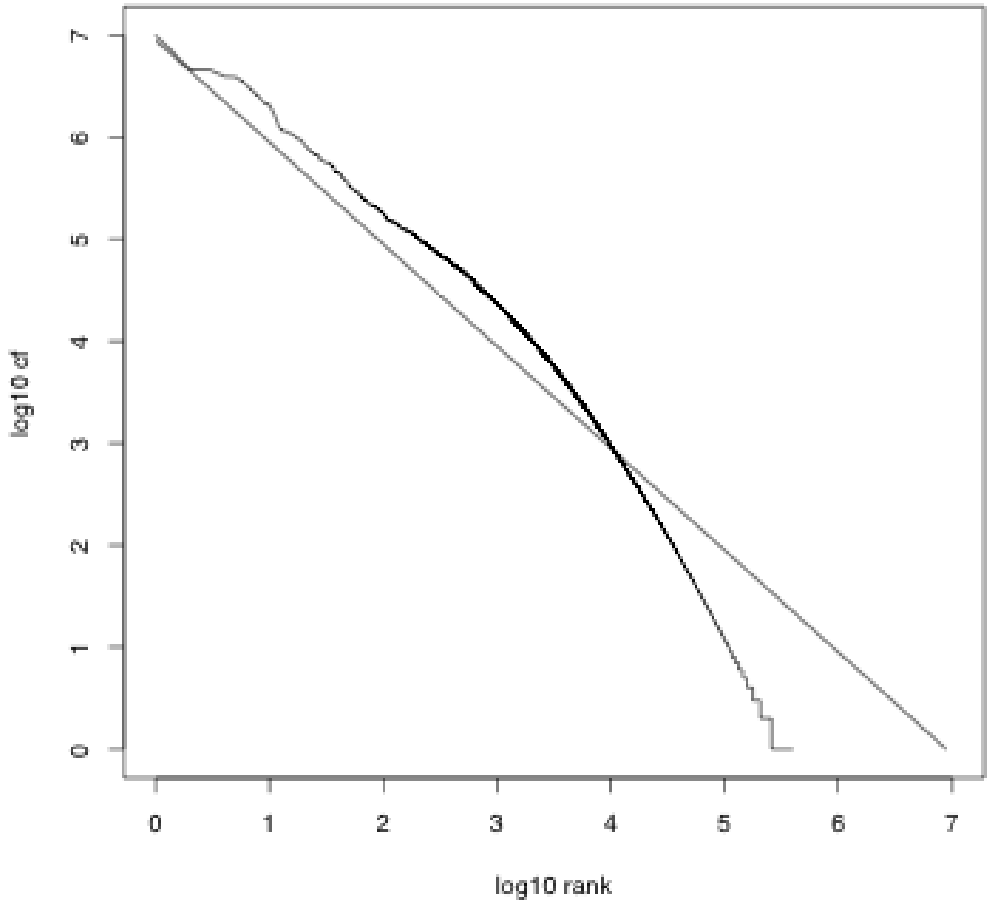
$$b = 0.49$$

词项的分布 — Zipf定律

- Zipf定律: $\mathrm{cf}_i \propto \frac{1}{i}$
- cf_i 是第 i 常见的词项 t_i 的文档集频率(collection frequency)，即词项 t_i 在所有文档中出现的次数
- 于是，如果最常见的词项(*the*)出现 cf_1 次，那么第二常见的词项 (*of*)出现次数为 $\mathrm{cf}_2 = \frac{1}{2}\mathrm{cf}_1 \quad \ldots$

  第三常见的词项 (*and*) 出现次数为 $\mathrm{cf}_3 = \frac{1}{3}\mathrm{cf}_1$
- 另一种表示方式: $\mathrm{cf}_i = ci^k$ 或 $\log \mathrm{cf}_i = \log c + k \log i \quad (k = -1)$

Zipf定律在RCV1上的表现



- 拟合度不是非常高
- 但可以发现：

高频词项很少，

低频罕见词项很多

词典压缩

□ 进行词典压缩的必要性：

- 最好能将词典放入内存，以提高查询效率
- 满足一些特定领域特定应用的需要，如手机、机载计算机上的应用
- 保证快速启动
- 与其他应用程序共享资源

定长数组方式下的词典存储

词项	文档频率	指向倒排记录表的指针
a	656 265	→
aachen	65	→
…	…	…
zulu	221	→

空间需求：　　20 字节　　4 字节　　4 字节

对Reuters RCV1语料: (20+4+4)*400,000 = 11.2 MB

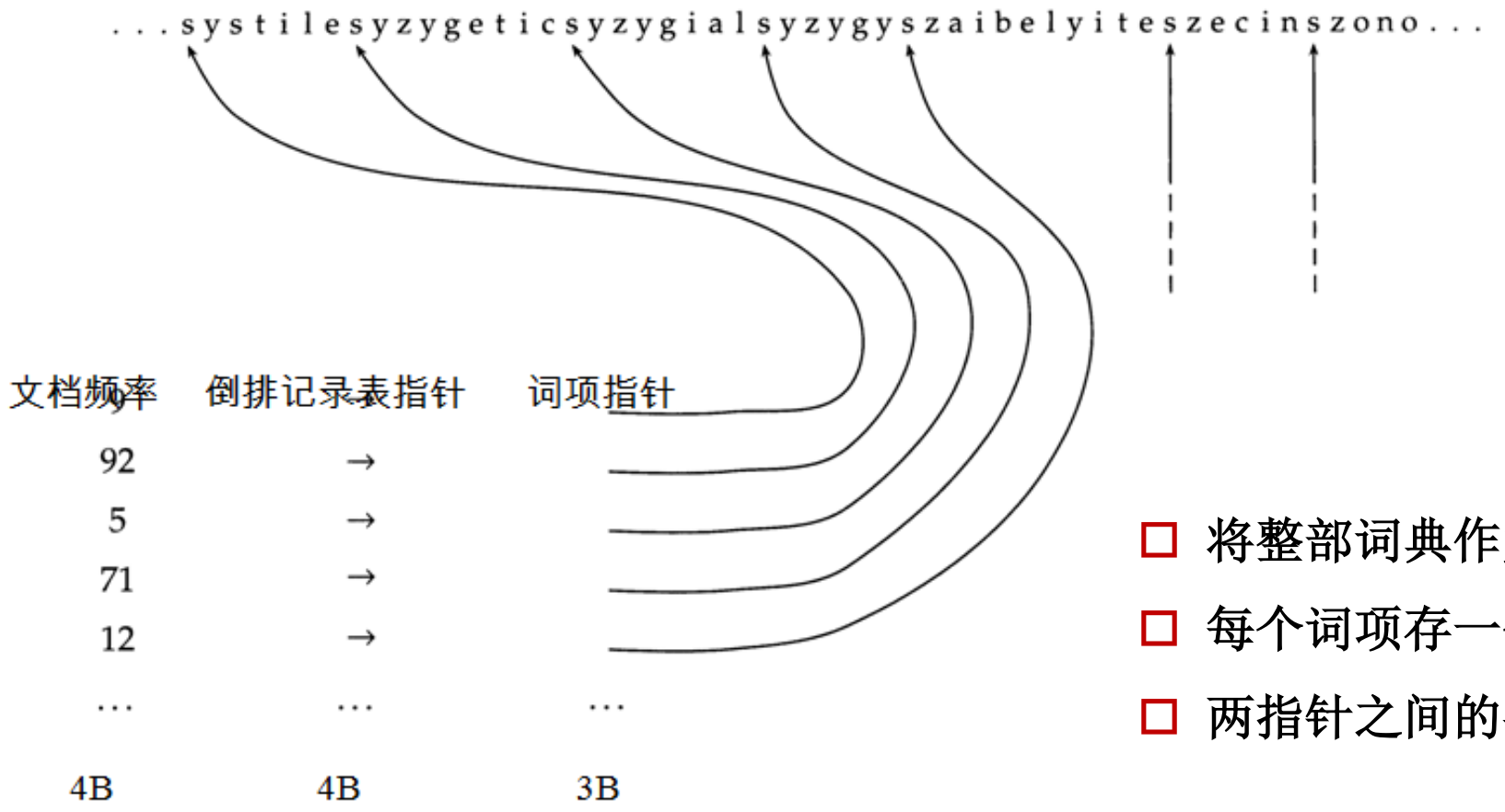
定长方式的不足

- 英语中每个词项的平均长度为8个字符
- 因此，对所有词项采用固定的20个字节存储造成空间浪费
  - 即使是长度为1的词项，我们也分配20个字节
- 不能处理长度大于20字节的词项，如

  HYDROCHLOROFLUOROCARBONS

  SUPERCALIFRAGILISTICEXPIALIDOCIOUS
- **理想方案：**对每个词项平均只使用8个字节来存储

基于单一字符串的压缩方法

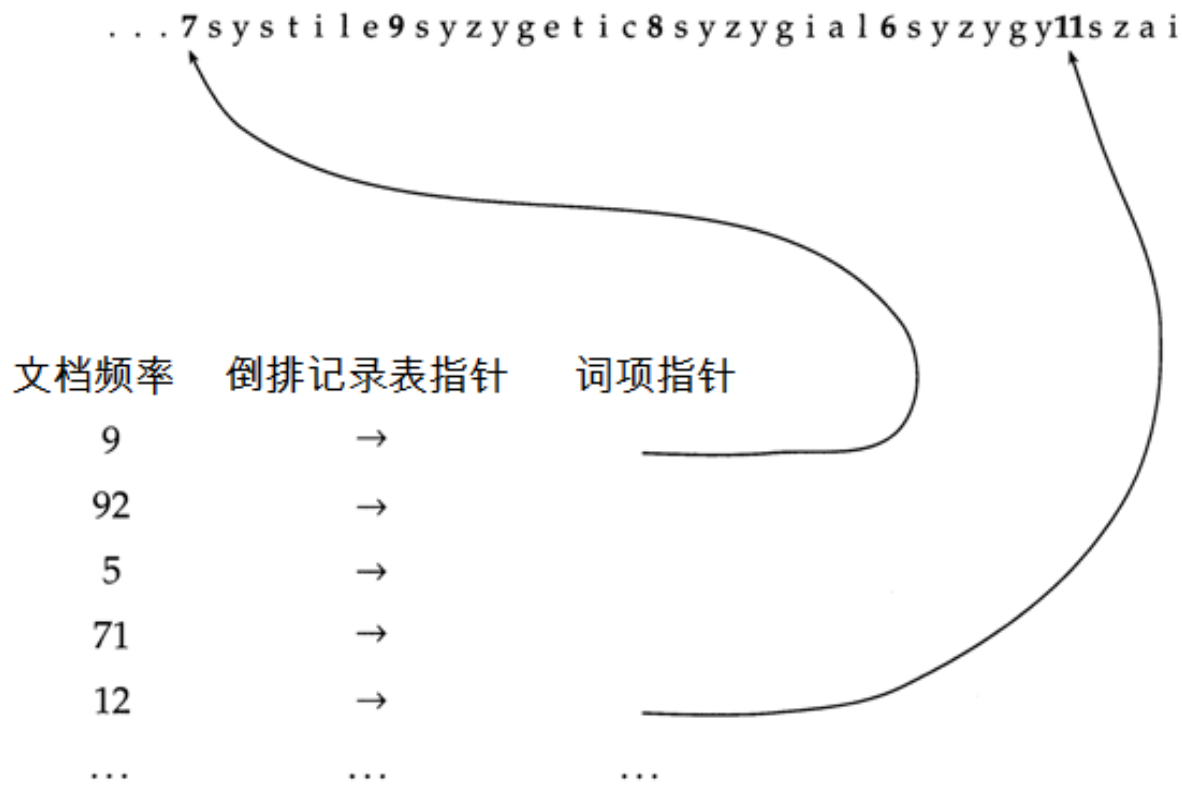


- 将整部词典作为一个字符串存储
- 每个词项存一个定位指针
- 两指针之间的字符构成一个词项

单一字符串方式下的空间消耗

- 每个词项的词项频率需要4个字节
- 每个词项指向倒排记录表的指针需要4个字节
- 每个词项平均需要8个字节
- 指向字符串的指针需要3个字节 (8*400000个位置需要log2（8 * 400000） < 24 位来表示)
- 空间消耗: 400,000 × (4 +4 +3 + 8) = 7.6MB （而定长数组方式需要11.2MB)

按块存储的压缩方法

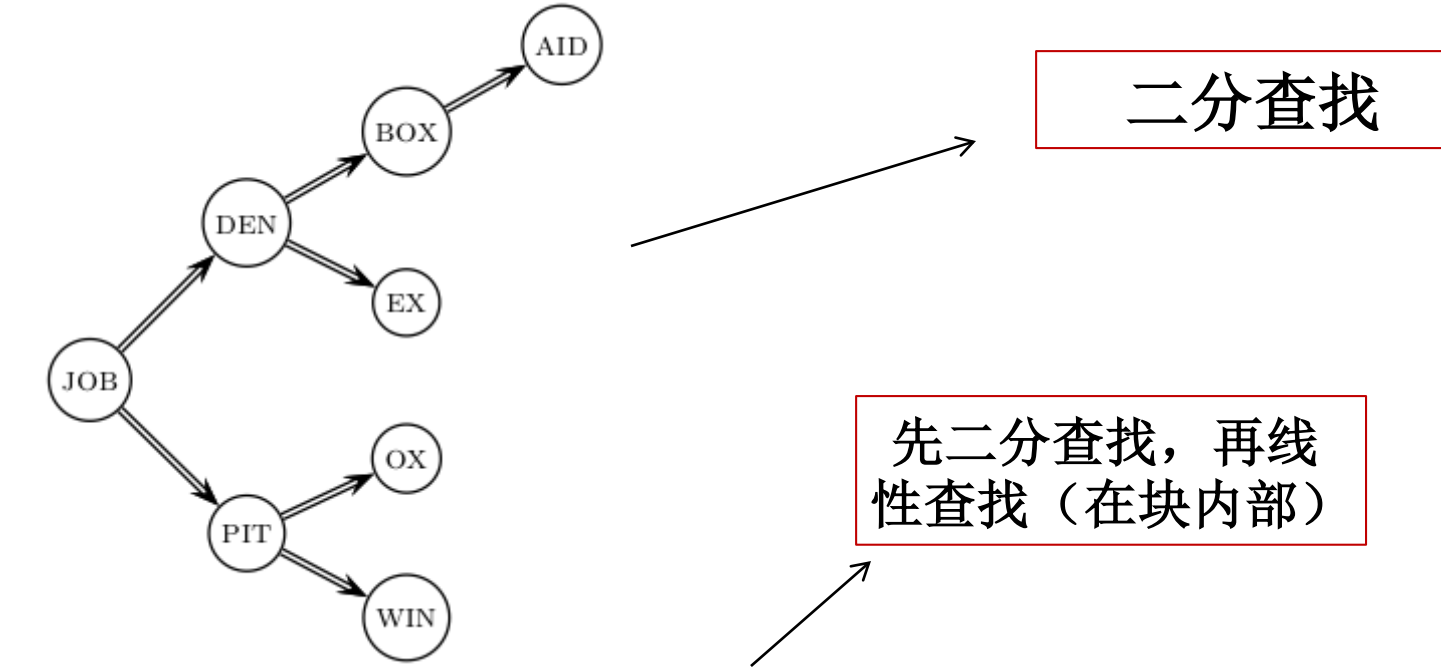


- 将长字符串中的词项进行分组变成大小为k的块(即k个词项一组)

- 每个块存一个指针
- 字符串中存词项的长度
- 以k=4个词项为一块，每块节省

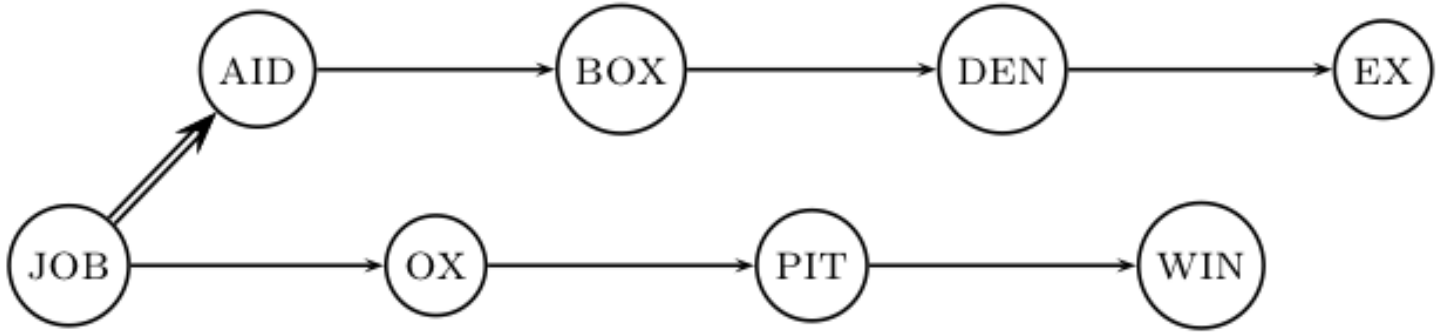
  9B-4B=5B
- 整个词典节省0.5MB

两种方式下的词项查找



二分查找

先二分查找，再线性查找（在块内部）



前端编码技术(Front coding)

- **基本思想：**通过省略词项之间公共前缀实现压缩
- **举例**
  - 按块存储压缩后的某个块 (k = 4)

    8automata8automate9automatic10automation

    ⇓

  - . . . 可以采用前端编码方式继续压缩为：

    8automat*a1◊e2◊ic3◊ion

Reuters RCV1词典压缩情况总表

数据结构	压缩后的空间大小（单位：MB）
词典，定长数组	11.2
词典，长字符串+词项指针	7.6
词典，按块存储，k=4	7.1
词典，按块存储+前端编码	5.9

倒排记录表压缩

□ 问题分析

- 主要存储内容：doc ID
- 需要采用多少位表示 doc ID?
- 对于Reuters RCV1，可以采用$\log_2$ 800,000 ≈ 19.6 < 20 位来表示每个docID
- 如何压缩 doc ID 的表示?

预处理：docID -> docID间隔

- 倒排记录表的一个特点：

$$\text{doc ID}_{i+1} = \text{doc ID}_i + (\text{doc ID}_{i+1} - \text{doc ID}_i)$$

- 自第二个记录开始，可以只存储间隔
- 通常间隔较小（特别是高频词）
- 显然，存储doc ID 间隔有利于压缩空间

	编码对象	倒排记录表					
the	文档ID	…	283 042	283 043	283 044	283 045	…
	文档ID间距			1	1	2	…
computer	文档ID	…	283 047	283 154	283 159	283 202	…
	文档ID间距			107	5	43	…
arachnocentric	文档ID	252 000	500 100				
	文档ID间距	252 000	248 100				

变长编码

- 目标
  - 对于 ARACHNOCENTRIC 及其他罕见词项，对每个间隔仍然使用20比特
  - 对于THE及其他高频词项，每个间隔仅仅使用很少的比特位来编码
- 为了实现上述目标，需要设计一个变长编码(variable length encoding)
- 可变长编码对于小间隔采用短编码而对于长间隔采用长编码

可变字节(VB)码

- **基本思想：** 利用整数个字节对间距编码，字节的数目根据具体的间距确定，一个表中的所有倒排记录作为一字节流存放
- **举例**

文档ID	824	829	215406
间距		5	214577
VB编码	00000110 10111000	10000101	00001101 00001100 10110001

- 被很多商用/研究系统所采用

22

VB 编码算法

- 将字节的第一位设置为延续位，用于标识间距编码的结束
- 如果间隔表示少于7比特，那么c置 1，将间隔编入一个字节的后7位中
- 否则：将低7位放入当前字节中，并将c置 0，剩下的位数采用同样的方法进行处理，最后一个字节的c置1（表示结束）

```
VBENCODENUMBER(n)
1  bytes ← ⟨⟩
2  while true
3  do PREPEND(bytes, n mod 128)
4     if n < 128
5        then BREAK
6     n ← n div 128
7  bytes[LENGTH(bytes)] += 128
8  return bytes
```

```
VBENCODE(numbers)
1  bytestream ← ⟨⟩
2  for each n ∈ numbers
3  do bytes ← VBENCODENUMBER(n)
4     bytestream ← EXTEND(bytestream, bytes)
5  return bytestream
```


23

VB编码的解码算法

```
VBDecode(bytestream)
 1  numbers ← ⟨⟩
 2  n ← 0
 3  for i ← 1 to Length(bytestream)
 4  do if bytestream[i] < 128
 5        then n ← 128 × n + bytestream[i]
 6        else n ← 128 × n + (bytestream[i] − 128)
 7             Append(numbers, n)
 8             n ← 0
 9  return numbers
```

当延续位为1，bytestream[i]>128，因此 if bytestream[i]<128
判断的是数字之间界线

其它编码

- 除字节外，还可以采用不同的对齐单位：比如32位(word)、16位及4位(nibble)等等
- 如果有很多很小的间隔，那么采用可变字节码会浪费很多空间，而此时采用4位为单位将会节省空间
- 最近一些工作采用了32位的方式 – 参考讲义末尾的参考材料

γ编码

- 是一种基于位的变长编码
- 最简单的位编码：一元码
  - 将 n 表示成 n 个1和最后一个0
  - 比如： 3的一元码是 1110
  - 40的一元码是
    1111111111111111111111111111111111111110
  - 70的一元码是：
    11111111111111111111111111111111111111111111111111
    11111111111111111111110

γ编码方法

- 将G 表示成长度(length)和偏移(offset)两部分
- 偏移对应G的二进制编码，只不过将首部的1去掉
- 长度部分给出的是偏移的位数
- 长度部分采用一元编码: 1110
- 举例: 13的γ编码
  - 13 → 1101 → 101 = 偏移
  - G=13 (偏移为 101), 长度部分为 3
  - 13的整个γ编码是1110101

γ编码的例子

数　字	一元编码	长　度	偏　移	γ 编　码
0	0			
1	10	0		0
2	110	10	0	10,0
3	1110	10	1	10,1
4	11110	110	00	110,00
9	1111111110	1110	001	1110,001
13		1110	101	1110,101
24		11110	1000	11110,1000
511		111111110	11111111	111111110,11111111
1025		11111111110	0000000001	11111111110 ,0000000001

课堂练习

- 计算130的可变字节码
- 计算130的γ编码

γ编码的长度

- 偏移部分是 $\lfloor \log_2 G \rfloor$ 比特位
- 长度部分需要 $\lfloor \log_2 G \rfloor + 1$ 比特位
- 因此，全部编码需要$2\lfloor \log_2 G \rfloor + 1$比特位
- γ 编码的长度均是奇数
- γ 编码在最优编码长度的2倍左右

γ 编码的性质

- γ 编码是前缀无关的，从而保证了解码的唯一性。
- 编码在最优编码的2或3倍之内
- 上述结果并不依赖于间隔的分布，是通用性(universal)编码
- γ 编码是无参数编码，不需要通过拟合得到参数

γ 编码的对齐问题

- 机器通常有字边界 – 8, 16, 32 位
- 按照位进行压缩或其他处理可能会较慢
- 可变字节码通常按字边界对齐，因此可能效率更高
- 除去效率高之外，可变字节码虽然额外增加了一点开销，但是在概念上也要简单很多

Reuters RCV1索引压缩总表

数据结构	压缩后的空间大小（单位：MB）
词典，定长数组	11.2
词典，长字符串+词项指针	7.6
词典，按块存储，k=4	7.1
词典，按块存储+前端编码	5.9
文档集（文本、XML标签等）	3 600.0
文档集（文本）	960.0
词项关联矩阵	40 000.0
倒排记录表，未压缩（32位字）	400.0
倒排记录表，未压缩（20位）	250.0
倒排记录表，可变字节码	116.0
倒排记录表，γ编码	101.0

参考资料

- 《信息检索导论》第5章
- http://ifnlp.org/ir
  - 有关字对齐二元编码的原文Anh and Moffat (2005); 及 Anh and Moffat (2006a)
  - 有关可变字节码的原文Scholer, Williams, Yiannis and Zobel (2002)
  - 更多的有关压缩 (包括位置和频率信息的压缩)的细节参考Zobel and Moffat (2006)

课后作业

- 见课程网页:

  http://10.76.3.31