


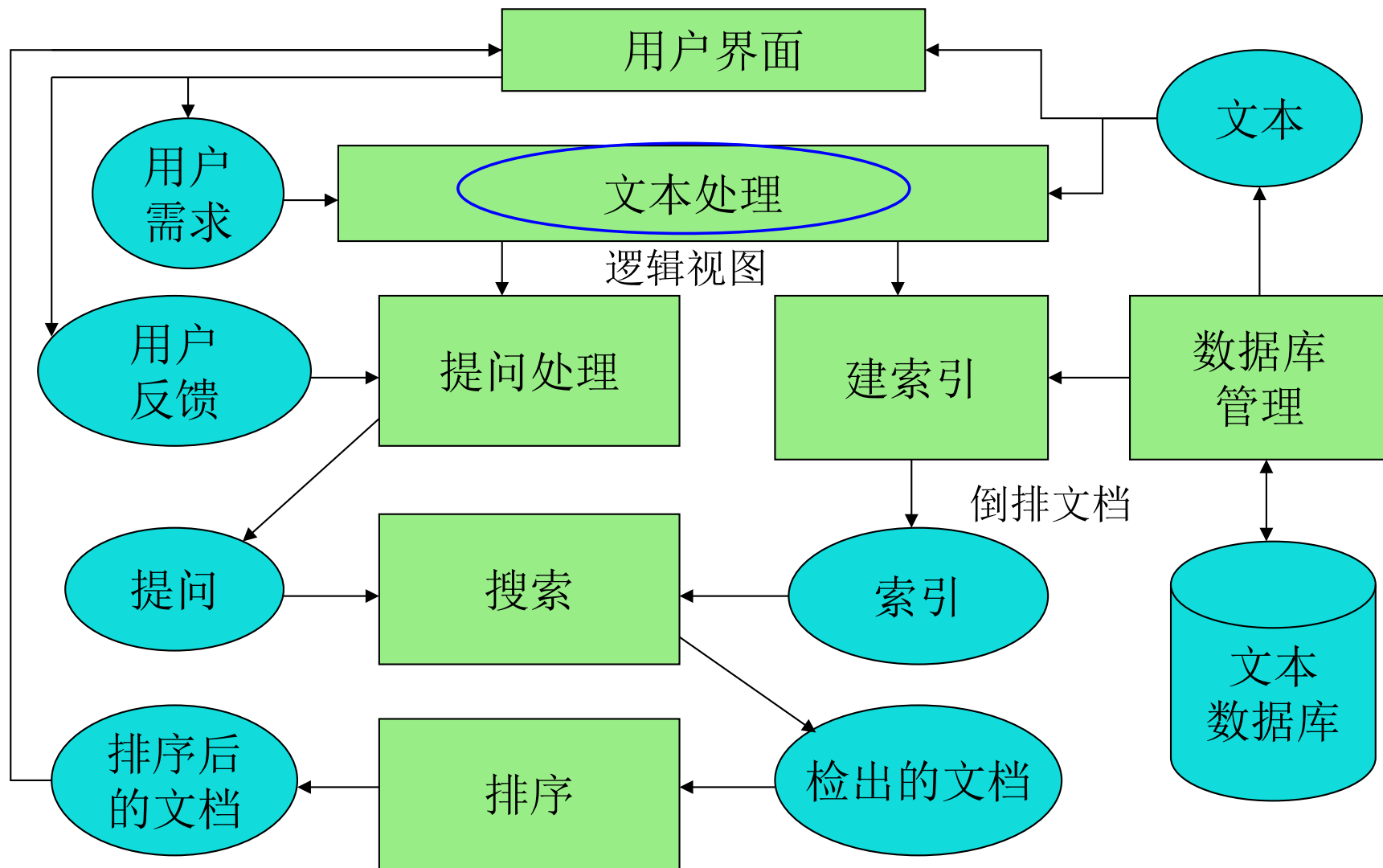
# 信息检索

# Information Retrieval



## 第四章 文本的操作

# 信息检索系统的体系结构



# 主要内容

---

- 文本的表示
- 文本的处理
  - 断词
  - 异文合并
  - 分词技术（中文）
- 文本的特性

# 文本的表示

- 文本可以表示为
  - 一个字符串
  - 词的集合
  - 语言单元 (例如：名词、短语)
- 简单的表示 (如：单个词项) 效果好
  - 以往的一些研究显示：基于短语的索引不如基于词的索引
  - 短语可能太特殊了

# 主要内容

---

- 文本的表示
- 文本的处理
  - 断词
  - 异文合并
  - 分词技术（中文）
- 文本的特性

# 文本处理的主要方法

---

- 英文

- 断词
- 形态还原
- 提取词干

- 中文

- 分词
- 简繁转换
- 其他.....

# 英文 — 断词

- 句点是英文中引起断词歧义最多的符号，也是最难处理的一个符号，如：

The experiments led by Dr. Alan achieved a precision of 90.7%.

■ 解决方案：规则+词表

- 撇号主要用于构成英文的动词缩写式和名词所有格，如：

I'm, won't, children's, parents'

■ 解决方案：整体标注（Brown语料），分开（Penn Treebank）

Brown语料库，20世纪60年代，第一个标准语料库，100万词规模。

Penn Treebank，宾州大学语料库，1993年完成约300万词次英语句子的语法结构标注，2000年完成第一版中文树库，约10万词次。

## 英文 — 断词

- 连字符可以用来构成合成词，用连字符构成的合成词有两类：
  - 一类已经固定成词，如：e-mail, co-operate;
  - 另一类是根据特定用法或语言环境生成的词，如 four-year, 1983-1987, All-In-One
  - 解决方案：主要通过词表解决



# 英文 — 异文合并

---

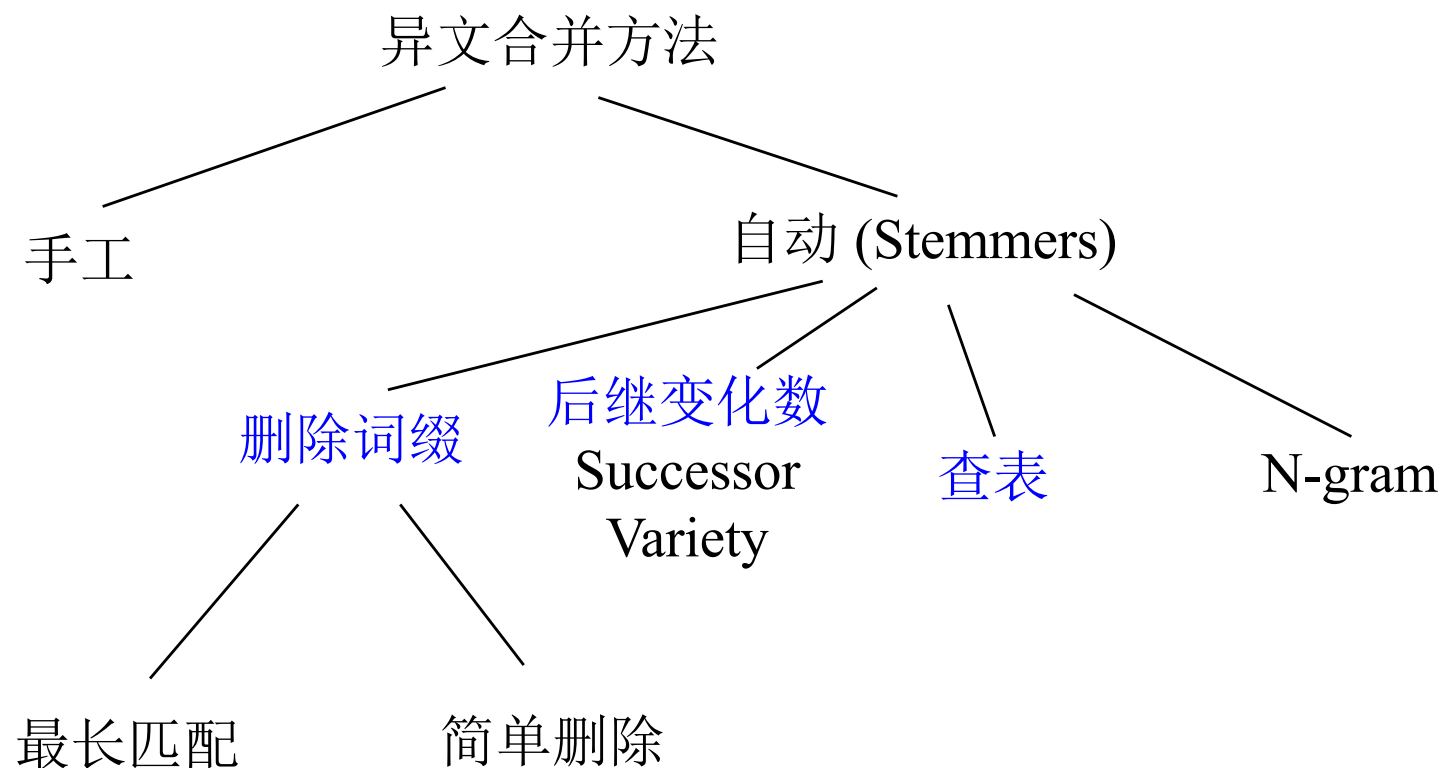
- 英文的异文合并主要体现在提取词干及形态还原
- 中文的异文合并主要体现在繁简转化

# 英文 — 异文合并

## ● 异文合并的特点

- 克服词形的变化，把所有同根词转变为单一形式
  - *RECOGNIZE, RECOGNISE, RECOGNIZED, RECOGNIZATION*
- 优点：
  - 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
- 缺点：
  - 正确率显然达不到100%
  - 不正确的stemming算法可能改变词的含义
  - 需要避免过分的截断
    - MEDICAL和MEDIA被识别为MED\*

# 英文异文合并 (*Conflation*) 的方法



# 异文合并方法之一：查表

- 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

## 异文合并方法之二：词缀删除

- 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：

*computability* --> *comput*      *singing* --> *sing*

避免: *ability* -> *NULL*, *sing*->*s*

### ■ 迭代式最长匹配

- WILLINGNESS --> 删除NESS --> 删除ING

# 异文合并方法之二：词缀删除

- 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

- 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 (1980)

- 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:  $sses \rightarrow ss, ies \rightarrow i, s \rightarrow NULL$
  - 上下文有关规则:  $(*v*) : ed \rightarrow NULL, ing \rightarrow NULL$

$(*v*)$ 的含义是：词根必须包含一个元音       $plastered \rightarrow plaster$   
 $bled \rightarrow bled$       删除词缀后，剩下的词干里没有元音

## *Porter*算法 (1980)

---

- 存在的问题:
  - 需要大量的语言知识来定义规则
  - 由于人类语言的复杂性，规则无法覆盖全部情况
  - 规则依赖于语言



# 异文合并方法之三：后继变化数

- 基于对文本集合的统计分析

- 给定一个足够大的语料库，可以通过统计的方法获得词干
- 这种方法是自动的，和语言关联性不大的

- 后继变化数的定义：

- 语料库中跟在某个字符串后的不同字符的数

- 考虑英文词典

- *pr?* -> 后继变化数是多少？
- *pro?* -> ?
- *pr* 和 *pro* 谁更像一个词根？

## 异文合并方法之三：后继变化数

Corpus
ABLE, APE
BEATABLE
FIXABLE
READ
READABLE
READING
READS
RED
ROPE, RIPE

Prefix	Successor Variety	Letters
R	3	E,I,O
RE	2	A,D
REA	1	D
READ	3	A,I,S
READA	1	B
READAB	1	L
READABL	1	E
READABLE	1	BLANK

# 异文合并方法之三：后继变化数

- 利用后继变化数信息切分词
  - 通过后继变化数的*cut off*值识别边界
    - 当后继变化数 $\geq$ 阈值时，进行切分
    - 考虑阈值 = 2     *R|E|AD|ABLE*
  - 尖峰和高地
    - 在后继变化数比前后都大，出现尖峰的位置切开
      - *READ|ABLE*
  - 切出来的词必须完整
    - 如：READ

# 中文 — 分词技术

---

- 中文检索系统主要有两种检索方案：
  - 基于字的检索按单字建立索引，需要在检索时进行逻辑运算
  - 基于词的检索按词建立索引，检索时直接命中
    - 基于词的方法具有检索速度快、准确率高的优点，目前的中文检索系统多数都支持基于词的检索

# 分词的歧义问题

## ● 交集型歧义

- 汉字串  $AJB$  被称作交集型切分歧义，如果满足  $AJ$ 、 $JB$  同时为词 ( $A$ 、 $J$ 、 $B$  分别为汉字串)。此时汉字串  $J$  被称作交集串

➤ [例] “结合成分子”

□ 结合|成分|子|      结合|成|分子|      结 |合成 |分子|

➤ [例] “美国会通过 $\text{会对台售武法案}$ ”

➤ [例] “乒乓球 $\text{拍}$ 卖完了”

# 分词的歧义问题

## ● 组合型歧义

- 汉字串 $AB$ 被称作组合型切分歧义，如果满足条件： $A$ 、 $B$ 、 $AB$ 同时为词

➤ [例] 组合型切分歧义：“起身”

➤ 他站 | 起 | 身 | 来。

➤ 他明天 | 起身 | 去北京。

# 正向最大匹配分词

## ● 基本思想：

1. 设自动分词词典中最长词条所含汉字个数为 $N$ ；
2. 取被处理语句当前字符串序数中的 $N$ 个字作为匹配字段，查找分词词典。若词典中有这样的一个 $N$ 字词，则匹配成功，匹配字段作为一个词被切分出来，转6；
3. 如果词典中找不到这样的一个 $N$ 字词，则匹配失败；
4. 匹配字段去掉最后一个汉字， $N--$ ；
5. 重复2-4，直至切分成功为止；
6.  $N$ 重新赋初值，转2，直到切分出所有词为止。

## 正向最大匹配分词 – 分析

- “市场/中国/有/企业/才能/发展/”
- 对交叉歧义和组合歧义没有什么好的解决办法
- 错误切分率为1 / 169
- 往往不单独使用，而是与其它方法配合使用



## 逆向最大匹配分词

- 分词过程与正向最大匹配分词方法相同，不过是从句子(或文章)末尾开始处理，每次匹配不成功时去掉的是前面的一个汉字
- “市场/中/国有/企业/才能/发展/
- 实验表明：逆向最大匹配法比最大匹配法更有效，错误切分率为1 / 245

# 其他的分词方法

---

- 基于统计的分词方法
- 未登录词识别方法
- .....

# 中文 — 简繁转换

- 一种语言，两种写法
  - 中国进行了语言文字改革，数以千计的汉字被大大地简化了（总表1986）
    - 以简化形式书写的中文称作简体中文(SC)
  - 台湾、香港以及大多数海外华人仍沿用传统的复杂形式，称作繁体中文(TC)

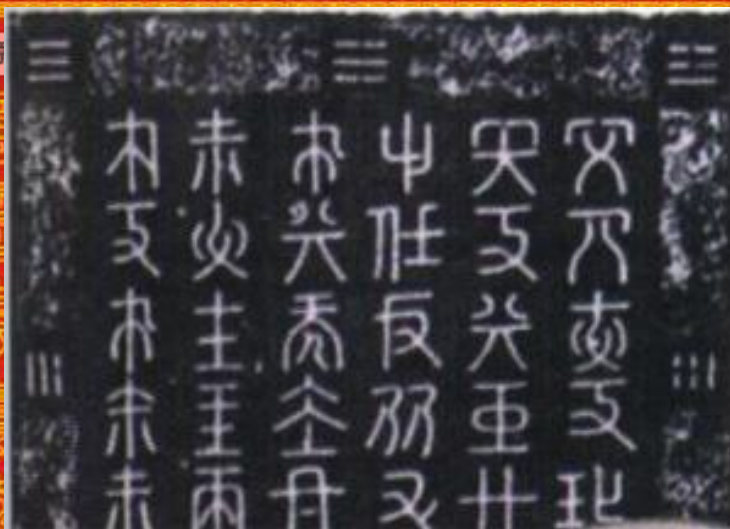
# 中文 — 简繁转换

- 简体中文与繁体中文的自动转换过程
  - 码对转换：转换的失败率很高
  - 字对转换：被转化的是有意义的语言单位，特别是多字词
  - 词对转换：这种汉字简繁转换不是按照拼写，而是按照语义进行的。例如，简体中的“信息”转换成繁体语义对应词时，就变成了“资讯”

# 中文—简繁转换

发：髮、發  
台：臺、檯、颱

点击查看源



殷懃懃殷

西夏 譯文 知足 常樂 歲在丙 戌年 於銀川 氏立書

女真字飲水思源

愛新覺羅啟琮



女 夫 太 卓 氣 余  
东 亏 太 中

# 其他文本的处理

- 词性标注

- 采用*HMM*的算法
- 参加下页词性标记表

- 词义消歧

- 采用贝叶斯等算法
- 《同义词词林》，参见后文

- 停用词表

- 英文: the, a, and, .....
- 中文: 的, 了, 和, .....

# 词性标注表

普通名词 n	时间名词 nt	方位名词 nd	处所名词 nl	人名 nh	地名 ns
团体、机构、组织的专名 ni	其它专名 nz	动词 v	形容词 a	区别词 b	副词 d
数词 m	量词 q	代词 r	介词 p	连词 c	叹词 e
拟声词 o	助词 u	前接成分 h	后接成分 k	习用语 i	简称 j
语素字 g	非语素字 x	标点 wp	字符串 ws		

# 主要内容

---

- 文本的表示
- 文本的处理
  - 断词
  - 异文合并
  - 分词技术（中文）
- 文本的特性



# 文本中词频的分布

- 大规模文档集，词的频率是怎样分布的？
- 极少的词是非常常见的
  - 英文中最常用的两个词是：“*the*”, “*of*”，他们的出现频率占全部英文词的10%
- 大多数词很少出现
  - 语料库中的一半词只出现一次
  - 称为 “*heavy tailed*” 分布, 因为大多数的概率值都是 “*tail*”

# 文本的特点

- 有些词在文本中出现的频率非常高，而且对文本所携带的信息基本不产生影响
  - 英文中的 “*a, the, of*”
  - 中文的 “的，了，着”
  - 字符串 “*http*”、“*.com*” 以及各种标点符号
- 文本经过词法分析之后，停用词通常被过滤掉，不参加文件的索引
- 在检索的时候，用户的查询中如果含有停用词，检索系统一般也将其过滤掉

# 样本词频数据(*from B. Croft, UMass*)

## TREC Volume 3 语料库 (1GB) 中词频统计

文档数: 336,310; 所有词的个数: 125,720,891; 不重复的词数目: 508,209

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

# Zipf(齐普夫) 's定律

Rank(R)	Term	Frequency(F)	$R * F * (10^{-6})$
1	the	69,971	$1 * 69971 * 10^{-6} = 0.070$
2	of	36,411	$2 * 36411 * 10^{-6} = 0.073$
3	and	28,852	$3 * 28852 * 10^{-6} = 0.086$
4	to	26,149	$4 * 26149 * 10^{-6} = 0.104$
5	a	23,237	$5 * 23237 * 10^{-6} = 0.116$
6	in	21,341	$6 * 21341 * 10^{-6} = 0.128$
7	that	10,595	$7 * 10595 * 10^{-6} = 0.074$
8	is	10,009	$8 * 10009 * 10^{-6} = 0.081$
9	was	9,816	$9 * 9816 * 10^{-6} = 0.088$
10	he	9,543	$10 * 9543 * 10^{-6} = 0.095$

# Zipf(齐普夫)'s定律

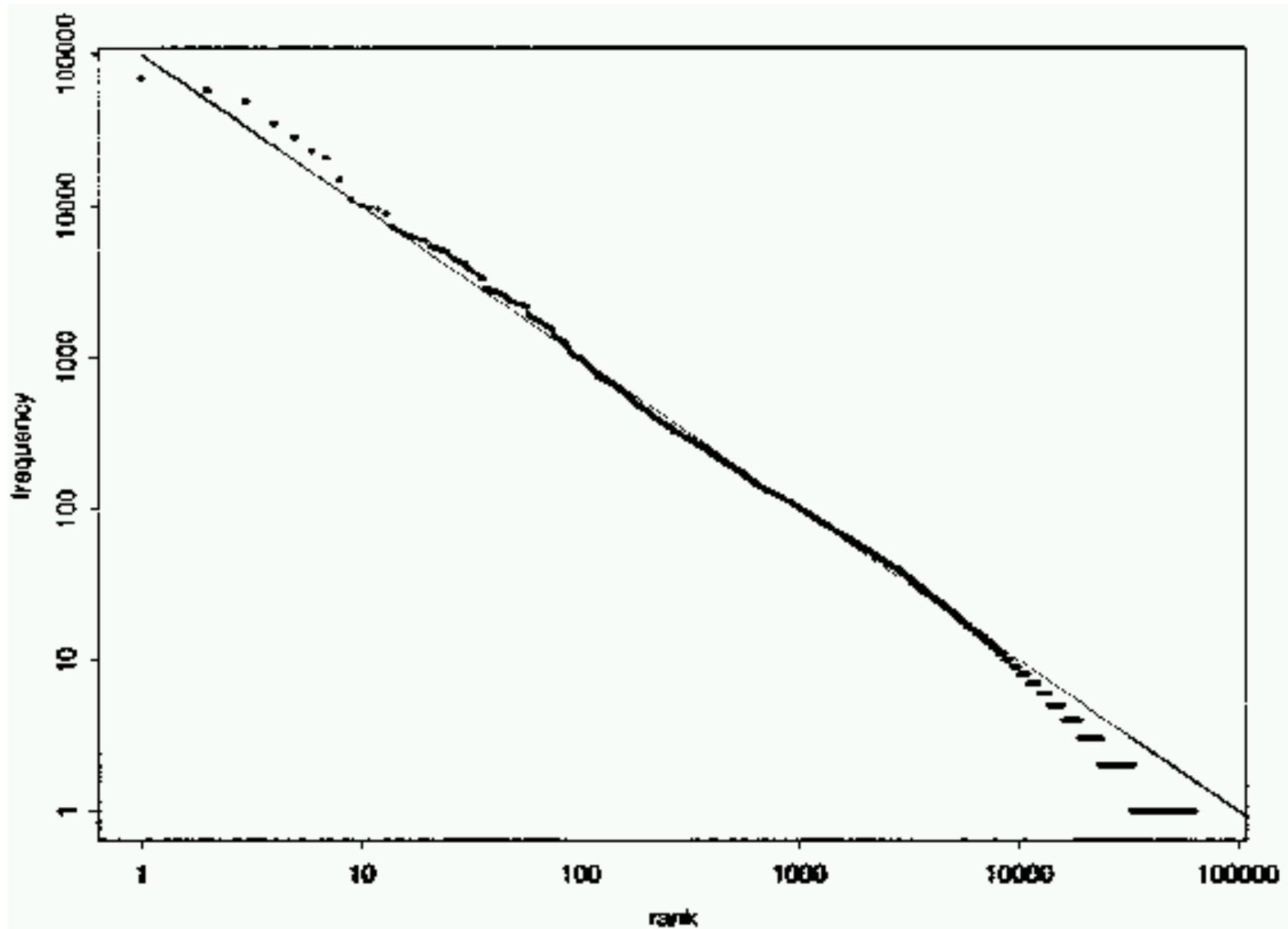
- *Rank (r)*: 在按词频(*f*)降序排列的词表中所处的位置
- *Zipf (1949)*“发现”:

$$f \propto \frac{1}{r} \quad f * r = k \text{ (for constant } k)$$

- 如果*rank*为*r*的词的概率为*pr*, *N* 是所有词出现的次数:

$$pr = \frac{f}{N} = \frac{A}{r} \quad (\text{for constant } A \approx 0.1)$$

# *Brown*语料库验证*Zipf*定律



# Zipf定律对IR的影响

## ● 正面作用

- 停用词在文本中占的比重很大，排除停用词可以极大地节省索引文件的磁盘空间
- 有的检索系统中，这种空间的节省甚至能达到40%以上，目前的检索系统，基本都使用过滤停用词的策略

## ● 负面作用

- 对大多数词来说，进行词汇之间的相关分析并不容易，因为它们出现的比较少

# 词表的增长

---

- 随着语料库的增长，词表以什么样的速度相应地增长
- 这决定了随着语料库规模的增长，倒排文件需要怎样增长
- 由于专名的存在，词表实际上没有上限



## *Heap's Law*

- $V$  是词表大小,  $n$  是语料库的长度 (词数)

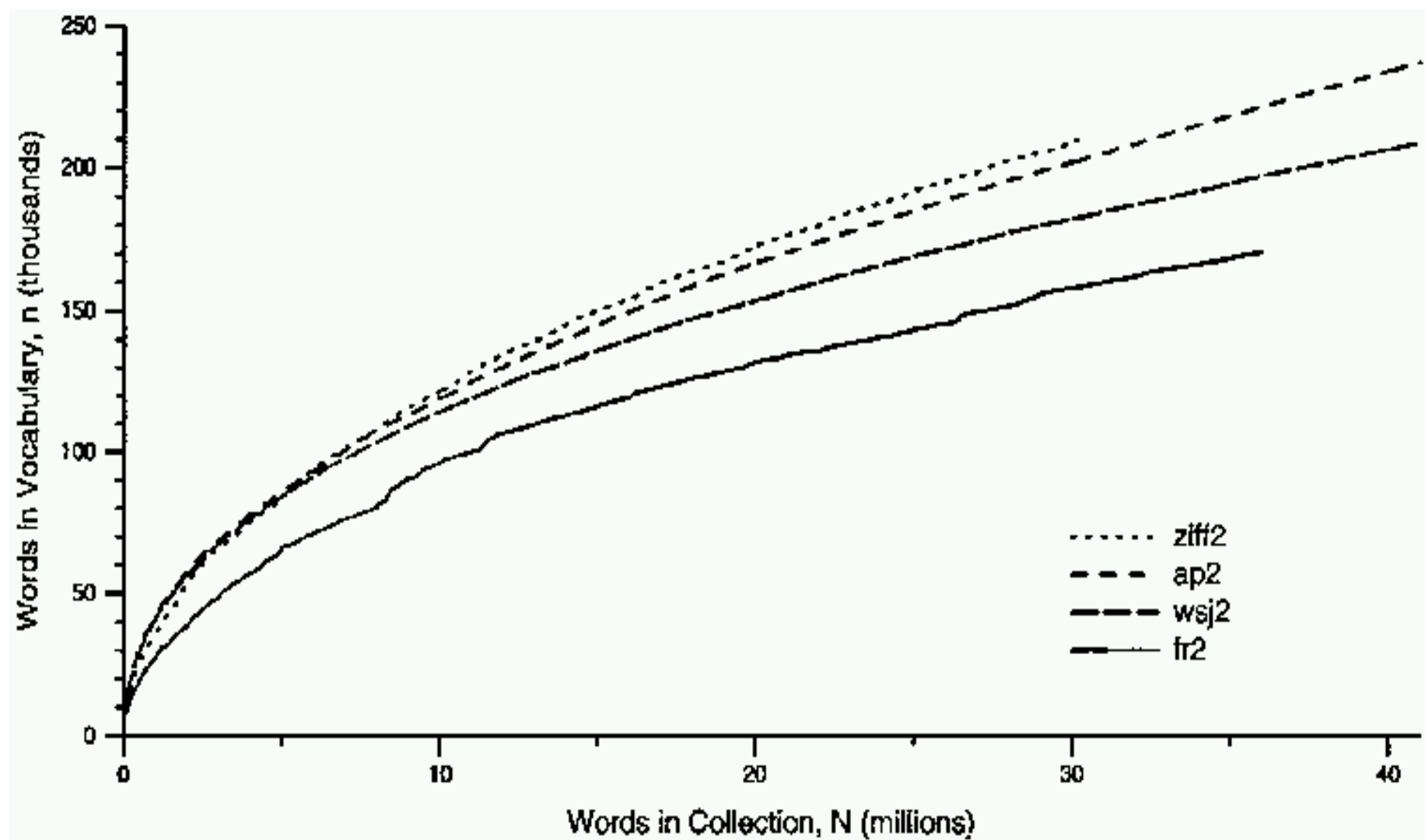
$$V = Kn^{\beta} \quad \text{with constants } K, 0 < \beta < 1$$

- 典型的常数:

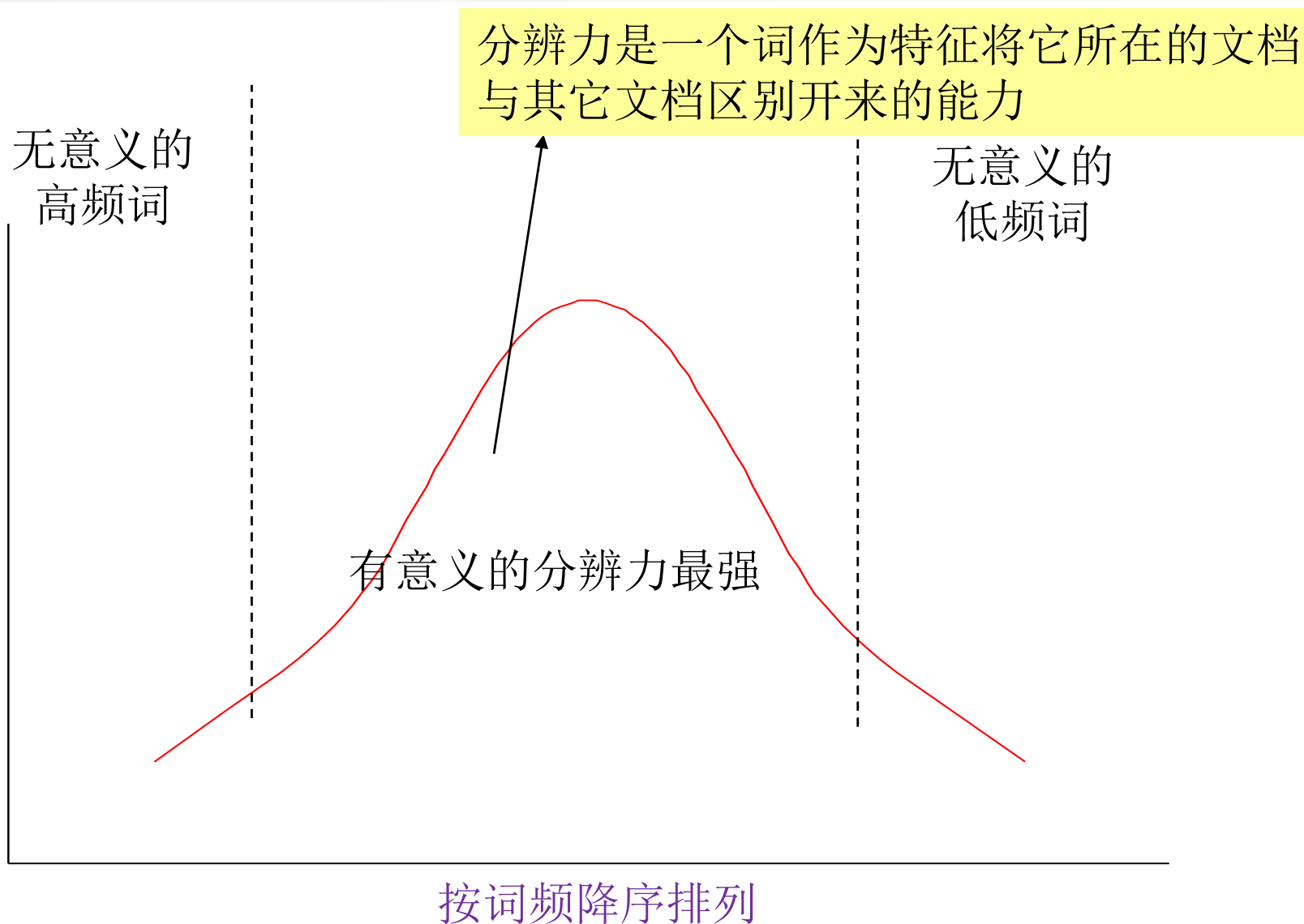
- $K \approx 10-100$

- $\beta \approx 0.4-0.6$  (*approx. square-root*)

# *Heap's Law*

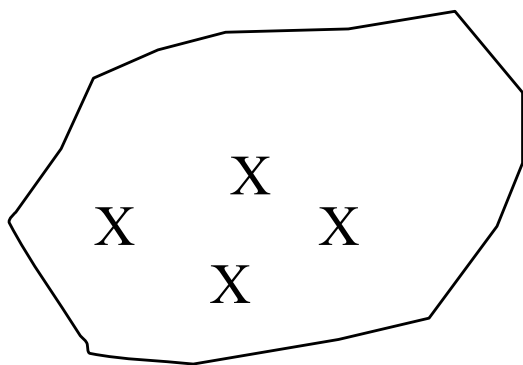


# 选择分辨力强的索引项

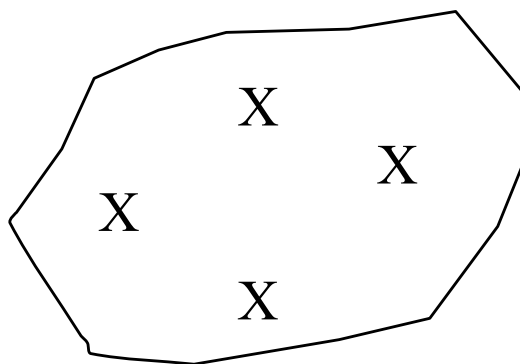


# 索引项的分辨力

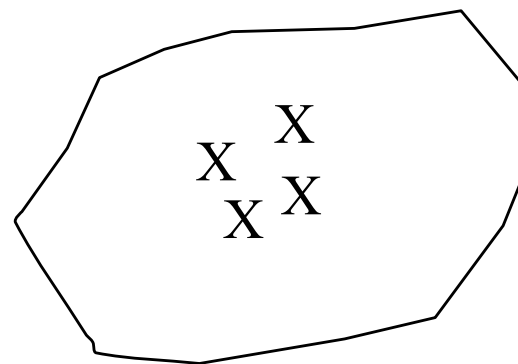
- 好的索引项能够将文档尽可能地离散开
  - 例如：关于“计算机科学”的文档集合中



原始文档空间：  
system



添加了好索引项：  
system, database



添加了不好的索引项：  
system, computer

# 索引项分辨力举例

	all terms	indexed	bad	good
d1	(a,b,c,d,r)	(b,c,d)	(a,b,c,d)	(b,c,d,r)
d2	(a,b,n,d,r)	(b,n,d)	(a,b,n,d)	(b,n,d,r)
d3	(a,m,p,q)	(m,p,q)	(a,m,p,q)	(m,p,q)
d4	(a,x,p,q)	(x,p,q)	(a,x,p,q)	(x,p,q)

- ◆ *a*就不是一个好的索引项，因为各个文档都包含*a*
- ◆ *r*可以使*d1*和*d2* 靠近，并使它们远离 *d3*和*d4*

# 索引项分辨力的计算

- 好的索引空间能够使文档集中各文档对之间的相似度的总和尽可能小
- $N$ 个文档之间的平均相似度为：

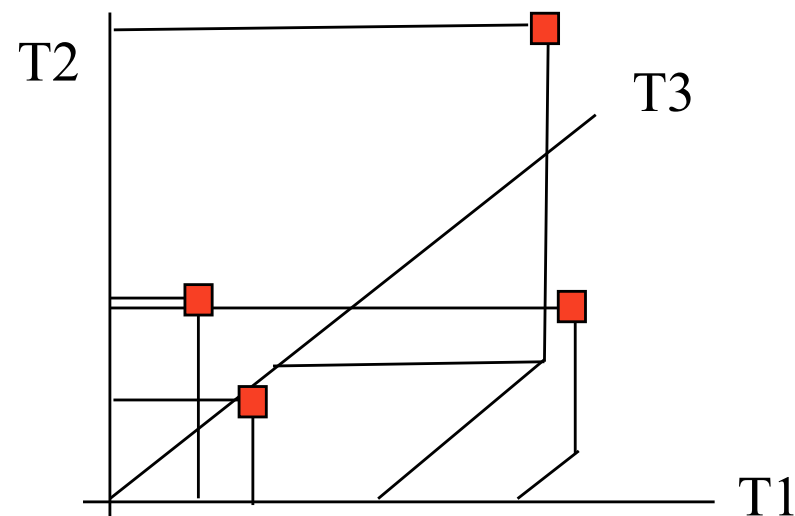
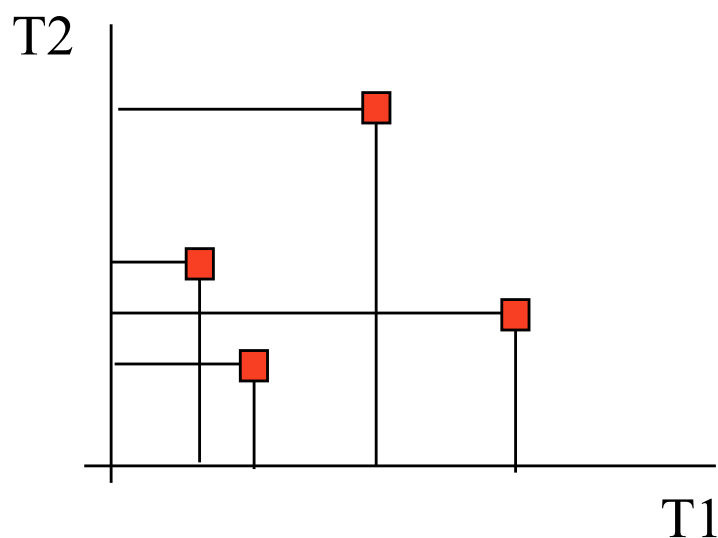
$$Q = \frac{1}{N(N-1)} \sum_{i=1}^N \sum_{\substack{k=1 \\ k \neq i}}^N \text{sim}(D_i, D_k)$$

- 假如 $Q$ 和 $Q_j$ 是加入 $term\ j$ 之前和之后的平均相似度，这 $term\ j$ 的分辨力为

$$dv_j = Q - Q_j$$

## 索引项分辨力 (图示)

- 加一个索引项就加了一维空间，这将使包含该索引项的文档和不包含该索引项的文档拉开距离



# 文本的质心

- $dv_j$  的计算开销太大
  - 需要针对每个term，计算文档对的相似度
  - 文档更新后，需要重新计算
- 折中方案：
  - 找到文档空间的质心(centroid)
  - $Q$ 是每个文档和质心之间的相似度

$$C = (c_1, c_2, \dots, c_t) \quad c_j = \frac{1}{N} \sum_{k=1}^N d_{kj}$$

$$Q = \frac{1}{N} \sum_{k=1}^N \text{sim}(C, D_k)$$



# 本章小结

---

- 文本的表示
- 文本的处理
  - 断词
  - 异文合并
  - 掌握基本的分词技术（中文），正向、逆向最大匹配法
- 掌握文本的特性，为下一步索引的建立打好基础