第二章：

文本特征提取技术

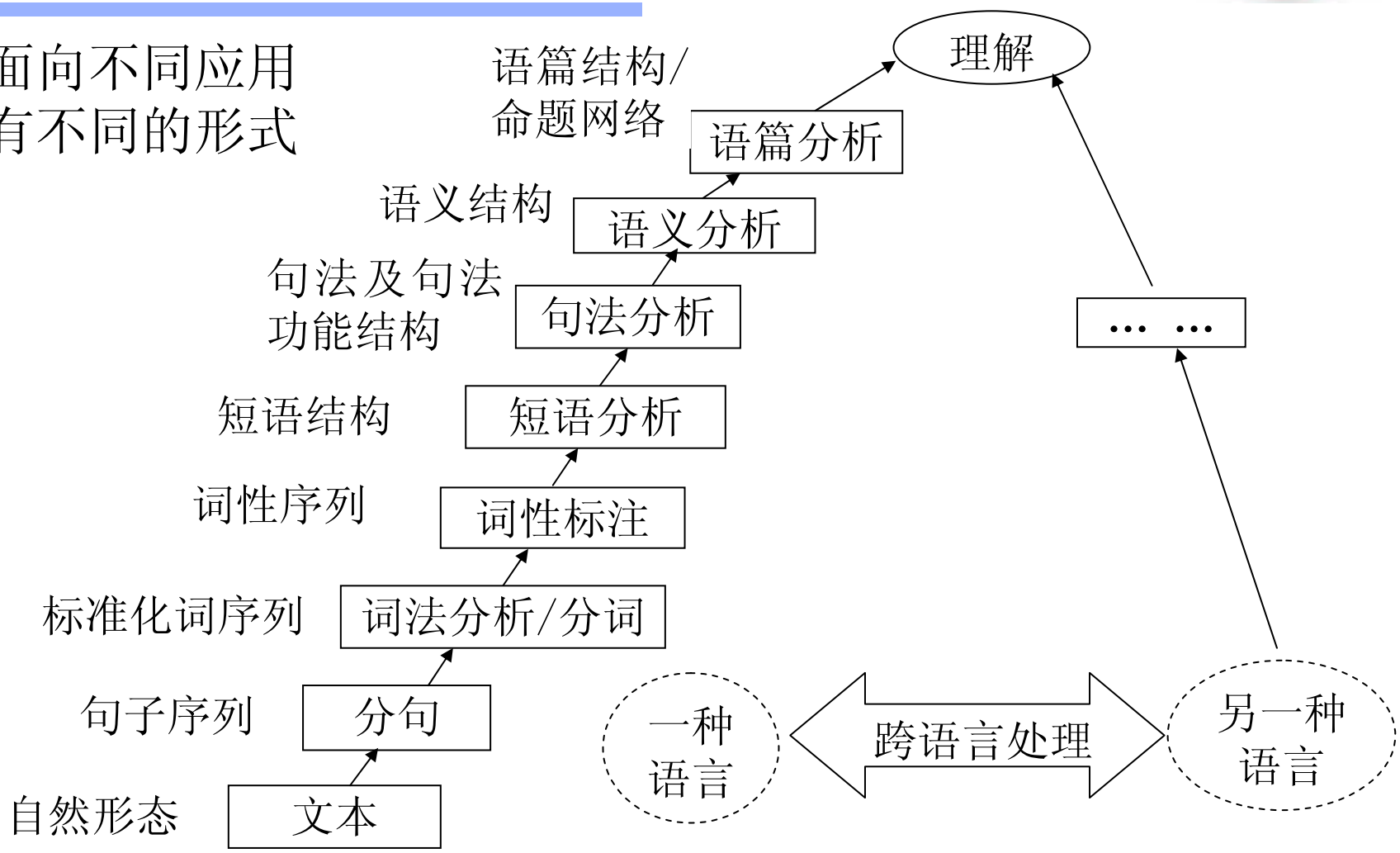
杨建武

北京大学计算机科学技术研究所

Email:yangjianwu@icst.pku.edu.cn

语言理解系统

面向不同应用
有不同的形式



分　词

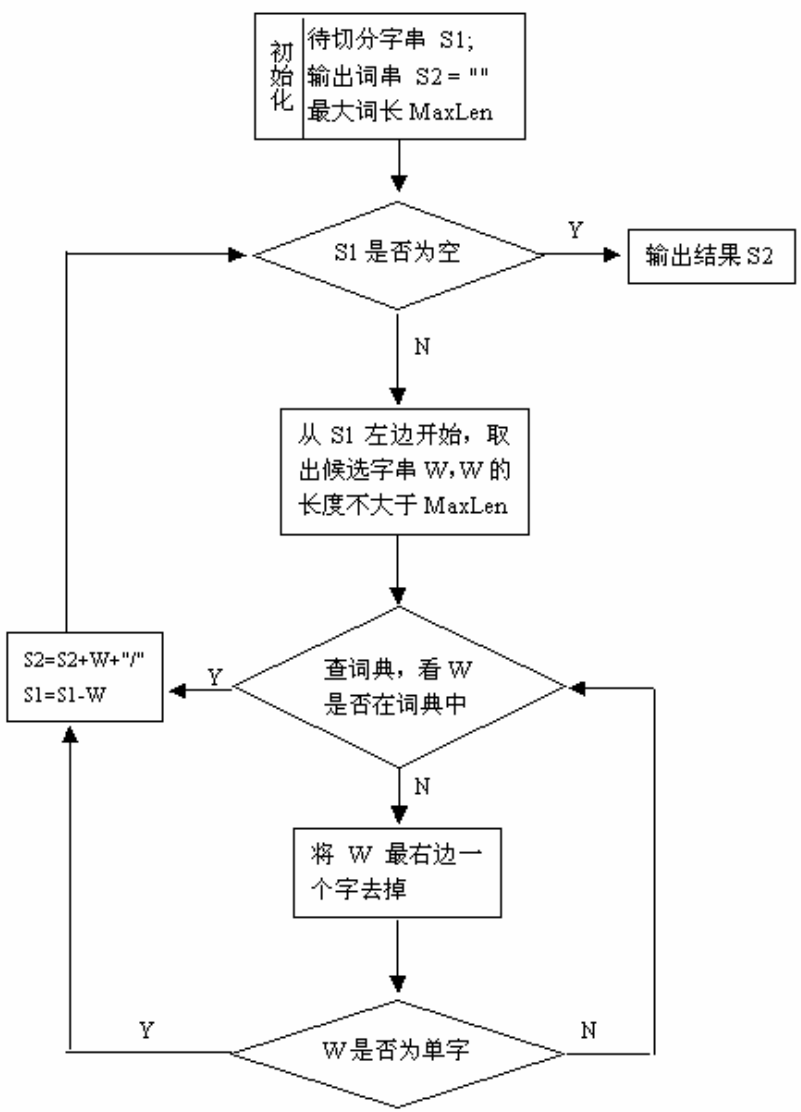
分词实例

- 和平民主
  - 和平、民主
  - 和、平民、主
- 提高人民生活水平
  - 提高、高人、人民、民生、生活、活水、水平
- 大学生活象白纸
  - 大学、生活、象、白纸
  - 大学生、活象、白纸

分词基本方法

- 最大匹配法
- 最大概率法分词
- 最短路径分词方法
- 难点：
  - 分词歧义
  - 未登录词识别

最大匹配法


初始化
待切分字串 S1;
输出词串 S2 = ""
最大词长 MaxLen
S1 是否为空
Y
输出结果 S2
N
从 S1 左边开始，取出候选字串 W，W 的长度不大于 MaxLen
查词典，看 W 是否在词典中
Y
S2=S2+W+"/"
S1=S1-W
N
将 W 最右边一个字去掉
W是否为单字
Y
N

最大匹配法示例

S1="计算语言学课程是三个课时"

设定最大词长
MaxLen = 5

S2= " "

词语
…
计算语言学
课程
课时
…

分词词表

最大匹配法示例（续）

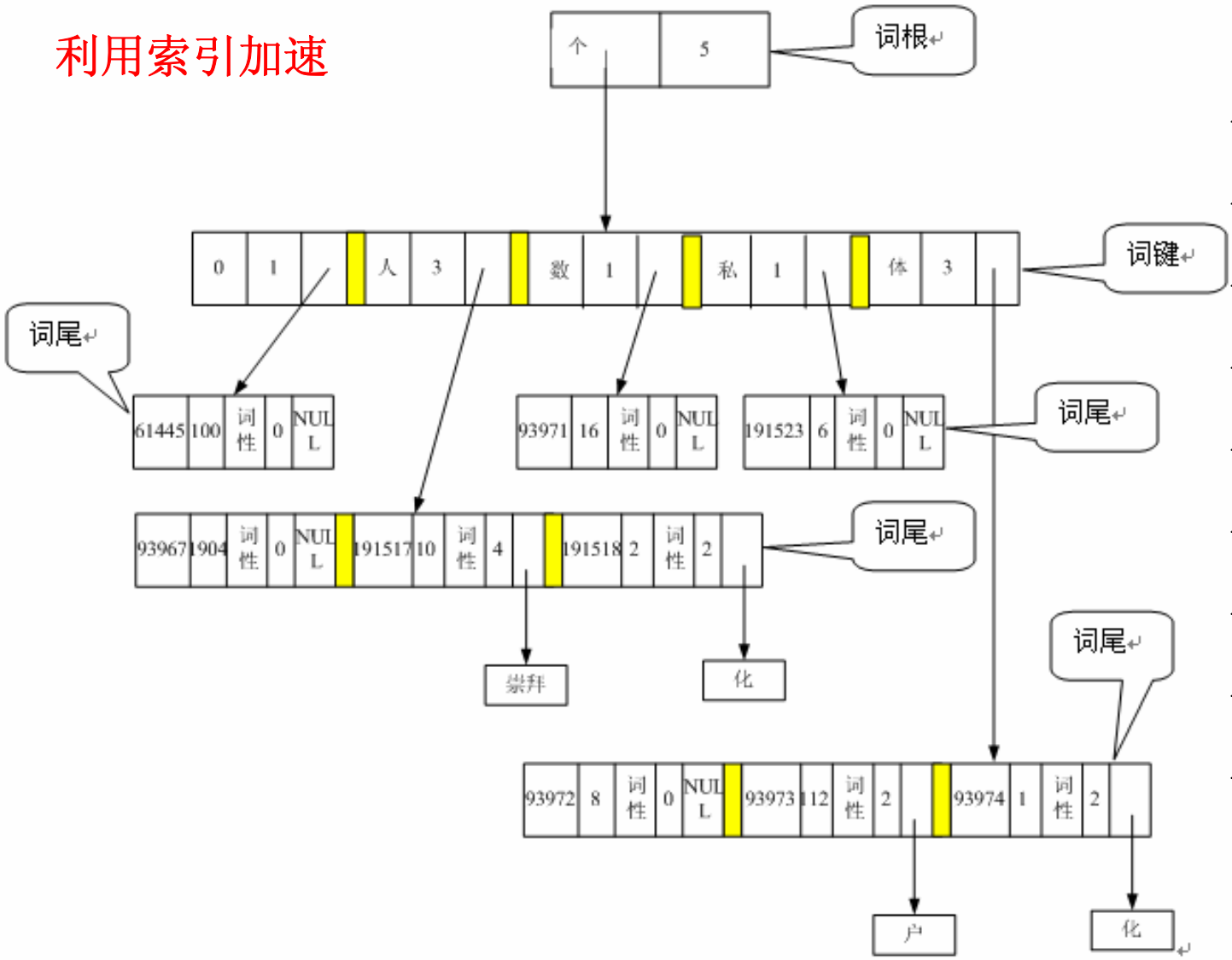
- （1）S2=“”；S1不为空，从S1左边取出候选子串
  W="计算语言学"；
- （2）查词表，“计算语言学”在词表中，将W加入到S2中，
  S2=“计算语言学/　”，并将W从S1中去掉，
  此时S1="课程是三个课时"；
- （3）S1不为空，于是从S1左边取出候选子串W="课程是三个"；
- （4）查词表，W不在词表中，将W最右边一个字去掉，得到
  W="课程是三"；
- （5）查词表，W不在词表中，将W最右边一个字去掉，得到
  W="课程是"；
- （6）查词表，W不在词表中，将W最右边一个字去掉，得到
  W="课程"
- （7）查词表，W在词表中，将W加入到S2中，
  S2=“计算语言学/　课程/　”，
  并将W从S1中去掉，此时S1="是三个课时"；

最大匹配法示例（续）

- （8）S1不为空，于是从S1左边取出候选子串W=″是三个课时″；
- （9）查词表，W不在词表中，将W最右边一个字去掉，得到W=″是三个课″；
- （10）查词表，W不在词表中，将W最右边一个字去掉，得到W=″是三个″；
- （11）查词表，W不在词表中，将W最右边一个字去掉，得到W=″是三″
- （12）查词表，W不在词表中，将W最右边一个字去掉，得到W=“是”，这时W是单字，将W加入到S2中，S2=“计算语言学/ 课程/ 是/ ”，并将W从S1中去掉，此时S1=″三个课时″；
- 。。。。。。
- 。。。。。。
- （21） S2=“计算语言学/ 课程/ 是/ 三/ 个/ 课时/ ”，此时S1=″″。
- （22）S1为空，输出S2作为分词结果，分词过程结束。

最大匹配法

利用索引加速



个

个人

个人崇拜

个人化

个数

个私

个体

个体户

个体化

其它基于匹配的分词方法

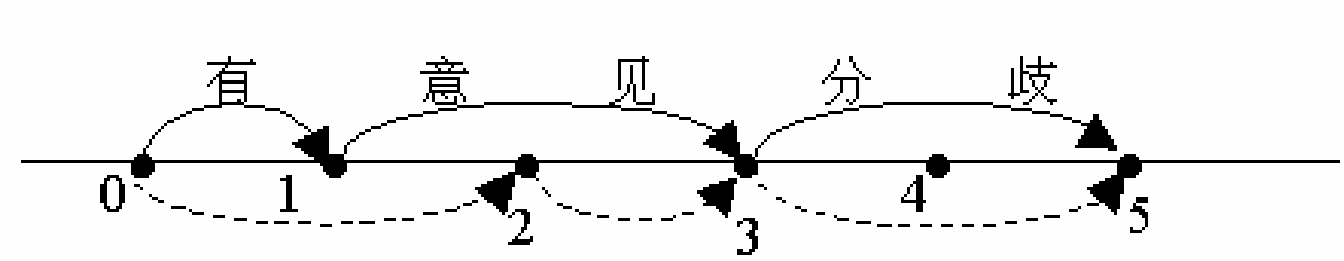
- 最大匹配法（Maximum Matching method）
  - 匹配的方向是从左向右。
- 逆向最大匹配法（Reverse Maximum method）
  - 匹配方向与MM法相反，是从右向左。
  - 实验表明：对于汉语来说，逆向最大匹配法比最大匹配法更有效。
- 双向匹配法(Bi-direction Matching method)
  - 比较MM法与RMM法的分词结果，从而决定正确的分词。
- 最佳匹配法（Optimum Matching method, OM法）
  - 将词典中的单词按它们在文本中的出现频度的大小排列，高频度的单词排在前，频度低的单词排在后，从而提高匹配的速度。
- 联想-回溯法（Association-Backtracking method）
  - 采用联想和回溯的机制来进行匹配。

最大概率法分词

基本思想是：

（1）一个待切分的汉字串可能包含多种分词结果

（2）将其中概率最大的那个作为该字串的分词结果



路径1： 0－1－3－5

路径2： 0－2－3－5

该走哪条路呢？

最大概率法分词

- S：有意见分歧
  - W1： 有/ 意见/ 分歧/
  - W2： 有意/ 见/ 分歧/

$$Max(P(W1|S), P(W2|S)) \ ?$$

$$P(W|S) = \frac{P(S|W) \times P(W)}{P(S)} \approx P(W)$$

$$P(W) = P(w_1, w_2, ..., w_i) \approx P(w_1) \times P(w_2) \times ... \times P(w_i)$$

独立性假设，一元语法

$$P(w_i) = \frac{w_i\text{在语料库中的出现次数}n}{\text{语料库中的总词数}N}$$

最大概率法分词示例

词语	概率
…	…
有	0.0180
有意	0.0005
意见	0.0010
见	0.0002
分歧	0.0001
…	…

$P(W1) = P(有) * P(意见) * P(分歧)$

$= 1.8 \times 10^{-9}$

$P(W2) = P(有意) * P(见) * P(分歧)$

$= 1 \times 10^{-11}$

$P(W1) > P(W2)$

最短路径分词方法

- 基本思想：
  - 在词图上选择一条词数最少的路径
- 优点：好于单向的最大匹配方法
  - 最大匹配：独立自主 和平 等 互利 的 原则 (6)
  - 最短路径：独立自主 和 平等互利 的 原则 (5)
- 缺点：
  同样无法解决大部分歧义
  - 结合 成分 子时
  - 他 说 的 确实 在理 （都是最短路径）
    他 说 的确 实在 理
    他 说 的确 实 在理

分词歧义分类（1）

- 交集型歧义
  - AB和BC都是词典中的词
  - 网球/　场/　：　网/　球场/
  - 链长：交集型歧义字段中含有交集字段的个数
- 组合型歧义
  - AB和A、B都是词典中的词
  - （我）　个人/　　：　　（三）　个/　人/
- 混合型歧义：　这样的人才能经受住考验
- 最大匹配法解决分词歧义的能力
  - 能发现部分交集型歧义
    - 增加歧义词表，规则等知识库
  - 无法发现组合型歧义

分词歧义分类（2）

- 分词歧义的四个层级（何克抗 等 1991，50883字语料）
  - 词法歧义 ： 84.1% （“用方块图形式加以描述”）
  - 句法歧义 ： 10.8% （“他一阵风似的跑了”）
  - 语义歧义 ： 3.4% （“学生会写文章”）
  - 语用歧义 ： 1.7% （“美国会采取措施制裁伊拉克”）
- 真假歧义
  - 真歧义 6%
    - 确实能在真实语料中发现多种切分形式
    - 比如“应用于”、“地面积”
  - 假歧义 94%
    - 虽然有多种切分可能性，但在真实语料中往往取其中一种切分形式
    - 如“挨批评”、“市政府”

未登录词识别

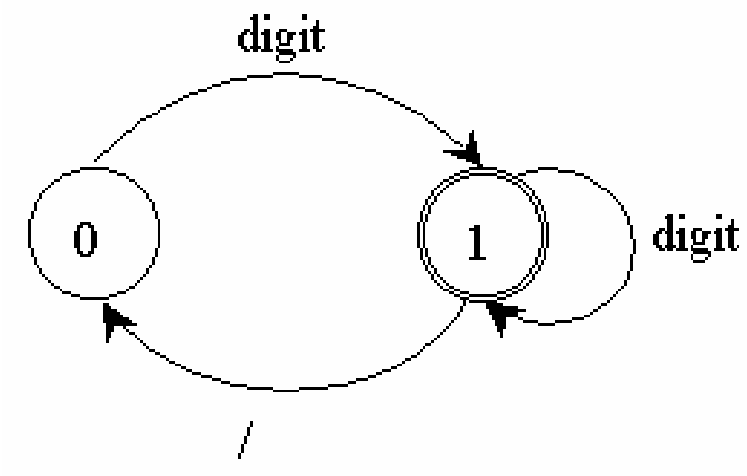
- 数字识别
- 命名实体识别
  - 人名
  - 地名
  - 机构名
  - 专业术语
- 形式词、离合词
  - 看看 看一看 打听打听 高高兴兴 乐呵呵
  - 游了一会儿泳 担什么心
- 未定义词识别的一般方法
  - 规则
  - 概率统计

数字的识别

- 正则表达式/regular expression
  - 识别分数，日期的正则表达式：
    [0-9] + (/ [0-9] +) +
    e.g. 12/21　5/13/2002
  - 识别百分数的正则表达式：
    [\+ | \ -] ? [0-9]+ . ? [0-9] * %
    e.g. - 5.9%　91%
  - 识别十进制数字的正则表达式：
    ([0-9] + , ?) + (. [0-9] + | [0-9] +) *
    e.g. 12,345

数字的识别

- 有限状态转移网络



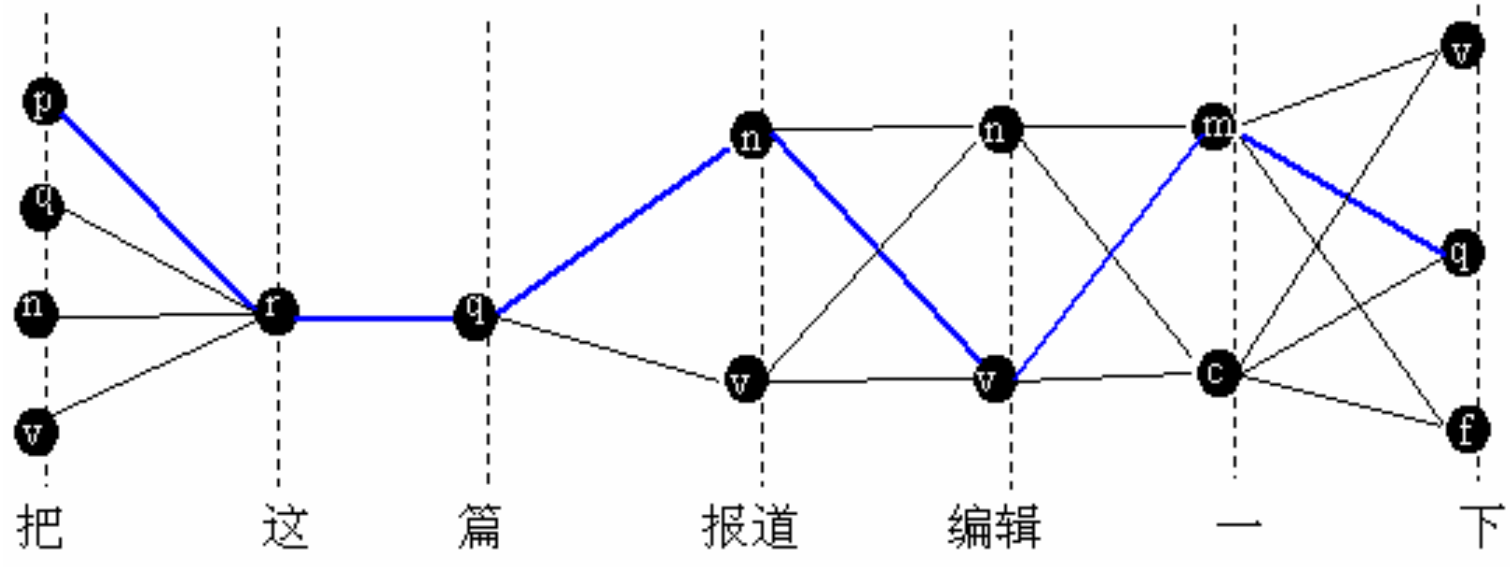
digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

中文姓名识别

序号	类型	属性	示例
1	姓氏用字	Surname	张，王，李，…
2	名字禁用字	Stop	死，吧，呢，…
3	姓名用词	Name	王学兵，…
4	普通用词	Common	非姓名用词　翻阅，浏览，…
5	非普通用词	None	
6	前称谓词	Left	经理王学兵，省长杨铮，…
7	后称谓词	right	王学兵经理，黄旭主席，…
⋮	⋮	⋮	⋮

词性标注

- 自动词性标注就是用计算机来自动地给文本中的词标注词性。
  - 兼类词：具有多种词性的词
  - 例如：领导（动词/名词）
  - 如何排除词性歧义？
- 标注技术路线：
  - 基于概率统计和基于规则



自动词性标注

- 早在60年代，国外学者就开始研究英语文本的自动词类标注问题。
- 1971年，美国布朗大学的格林（Greene）和鲁宾（Rubin）建立了TAGGIT系统，利用了3300条上下文框架规则，自动标注正确率达到77%.
- 1983年，里奇（G. Leech）和加塞德（R. Garside）等人建立了CLAWS系统，用概率统计的方法来进行自动词性标注，他们使用了133×133的词类共现概率矩阵，通过统计模型来消除兼类词歧义，自动标注的正确率达到了96%.

基于规则的词性标注

- 主要依靠上下文来判定兼类词。
  - 这是一张白纸
    - “白”出现在名词”纸”之前，判定为形容词
  - 他白跑了一趟
    - “白”出现在动词“跑”之前，判定为副词
- 词性连坐：在并列的联合结构中，联合的两个成分的词类应该相同，如果其中一个为非兼类词，另一个为兼类词，则可把兼类词的词性判定为非兼类词的词性。
  - 我读了几篇文章和报告
    - “文章”为名词，是非兼类词，“报告”为动-名兼类词，由于处于联合结构中，故可判定“报告”为名词。

基于统计的词性标注

$$P(T|W)=\frac{P(T,W)}{P(W)}=\frac{P(T)P(W|T)}{P(W)}$$

T:一种词性标注的组合

找概率最大的T

$$P(T|W)\approx P(T)P(W|T)$$

$$P(T)=P(t_1|t_0)P(t_2|t_1,t_0)...P(t_i|t_{i-1},t_{i-2},...)$$

二元

$$P(T)\approx P(t_1|t_0)P(t_2|t_1)...P(t_i|t_{i-1})$$

$$P(W|T)=P(w_1|t_1)P(w_2|t_2,t_1,w_2,w_1)...P(w_i|t_i,t_{i-1},...t_1,w_i,w_{i-1},...,w_1)$$

一元

$$P(W|T)\approx P(w_1|t_1)P(w_2|t_2)...P(w_i|t_i)$$

$$P(t_i|t_{i-1})=\frac{\text{训练语料中}t_i\text{出现在}t_{i-1}\text{之后的次数}}{\text{训练语料中}t_{i-1}\text{出现的总次数}}$$

$$P(w_i|t_i)=\frac{\text{训练语料中}w_i\text{的词性被标记为}t_i\text{的次数}}{\text{训练语料中}t_i\text{出现的总次数}}$$

统计方法的缺陷

- 统计方法是根据同现概率来标注词性。但是，同现概率仅只是最大的可能而不是唯一的可能，以同现概率来判定兼类词，是以舍弃同现概率低的可能性前提的。
- 将统计方法和规则方法结合被认为是解决词性标注问题的最佳手段。

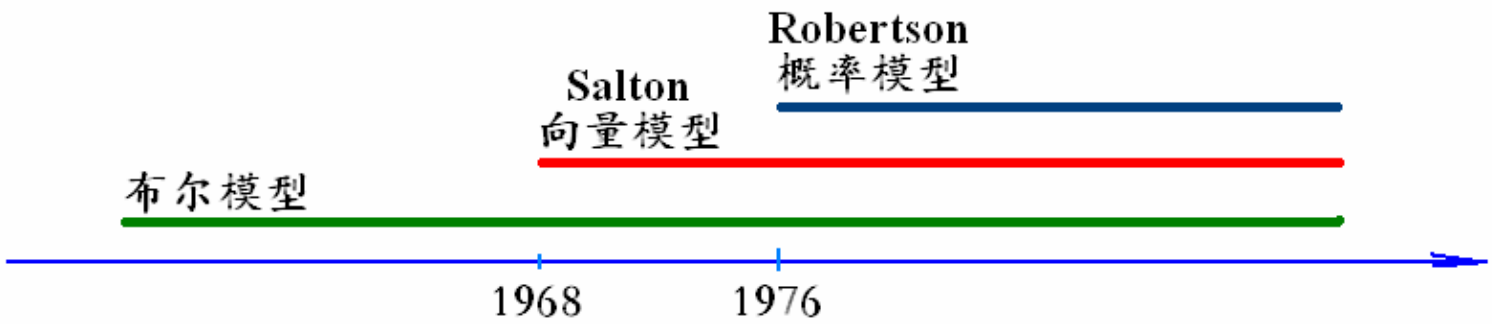
停用词与词形变化

- 停用词
  - 常用词：
    - 英文：“a,the,of,for,with,in,at，…”
    - 中文：“的，地，得，…”
  - 虚词：介词、连词等
  - 领域实词：数据库会议上的论文中的“数据库”一词，可视为停用词
- 词根问题
  - compute , computes , computed 同一词

文 档 模 型

文档模型

- 布尔模型
- 向量空间模型
- 概率模型

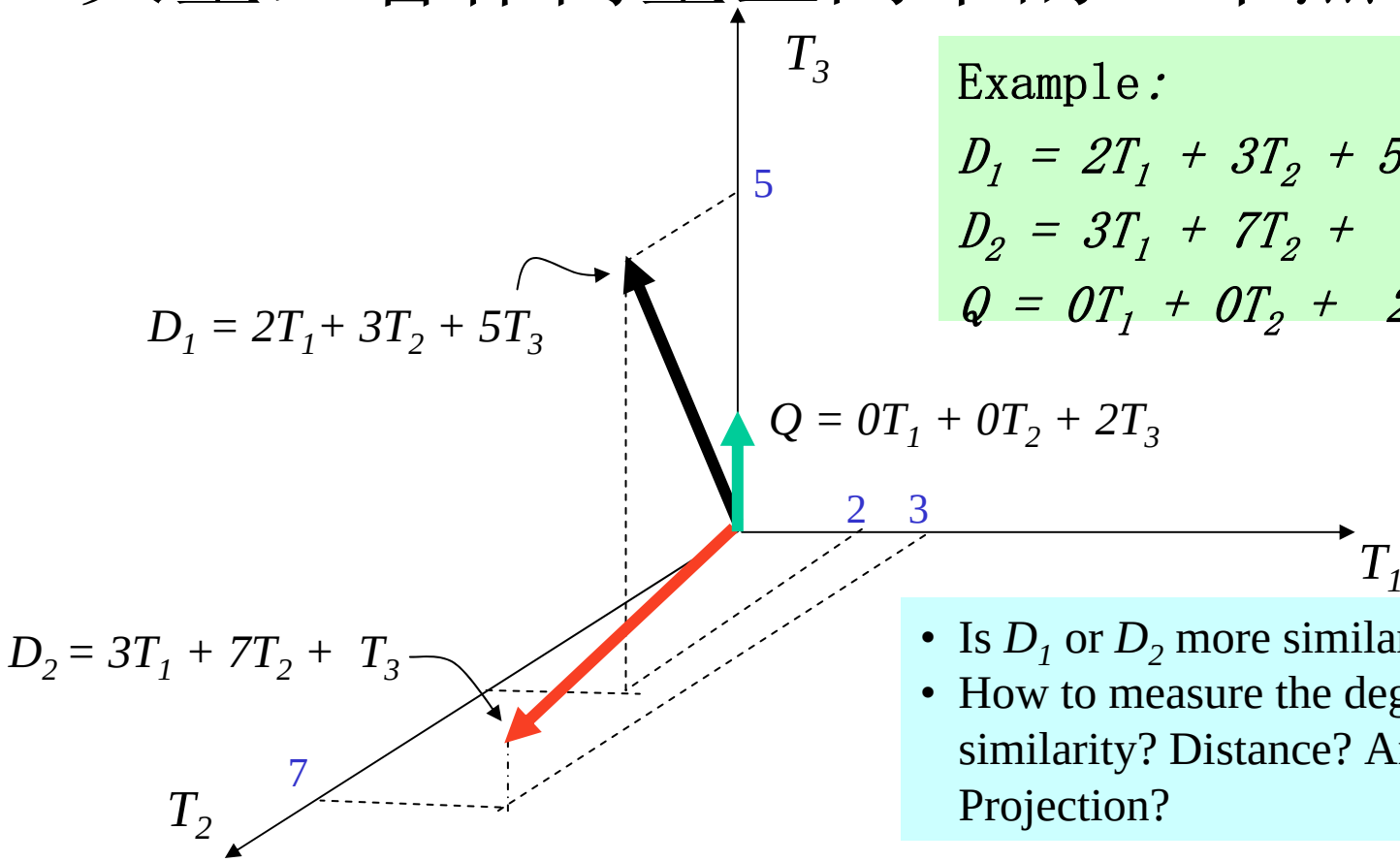


布尔模型

- 建立在经典的集合论和布尔代数的基础上
- 每个词在一篇文档中是否出现，对应权值为 0或1
- 文档检索→布尔逻辑运算
- 优点：
  - 简单、易理解、简洁的形式化。
- 缺点：
  - 准确匹配，信息需求的能力表达不足。

向量空间模型（VSM）

- 向量空间模型中将文档表达为一个矢量，看作向量空间中的一个点



Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$

- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

Term Weights

- The words of a text are not equally indicative of its meaning
  - "Most scientists think that butterflies use the position of the sun in the sky as a kind of compass that allows them to determine which way is north."
- Important:
  - butterflies, monarchs, scientists, compass
- Unimportant:
  - most, think, kind, sky,
- Term weights reflect the (estimated) importance of each term

Term frequency (tf)

- The more often a word occurs in a document, the better that term is in describing what the document is about
- Often normalized, e.g. by the length of the document

$$T = \frac{tf}{doc\_length} \qquad T = \frac{tf}{\max tf_d}$$

Inverse document frequency (idf)

- But terms that appear in many documents in the collection are not very useful for distinguishing a relevant document from a non-relevant one
- Terms that occur in many documents in the collection are less useful for discriminating among documents
- Document frequency (df): number of documents containing the term

idf often calculated as $I = \log\left(\frac{N}{df}\right) + 1$

Sometimes scaled to [0..1]

$$I = \frac{\log\left(\frac{N + 0.5}{df}\right)}{\log(N + 1.0)}$$

VSM Summary

- Based on occurrence frequencies only
- Consider both local and global occurrence frequencies
- Advantages
  - simplicity
  - able to handle weighted terms
  - easy to modify term vectors
- Disadvantages
  - assumption of term independence
  - lack the control of Boolean model (e.g. requiring a term to appear in a document)

文档概率模型

- 贝叶斯定理 $P(A|B)=\frac{P(B|A)\cdot P(A)}{P(B)}$
- 词条的独立假设：P(AB)= P(A) P(B) 当且仅当 A与B相互独立
- 由此对一篇文档而言，若文档中的各个词相互独立，则有 P(dj)=P(k1)…P(kt)

文档概率模型

文档dj与查询q的相似度定义为：

$$sim(d_j, q) = \frac{P(R \mid d_j)}{P(\overline{R} \mid d_j)}$$

R表示相关文档集；

$\overline{R}$ 表示R的补集；

$P(R|d_j)$ 表示文档dj与查询q相关的概率；

$P(\overline{R}|d_j)$ 表示文档dj与查询q不相关的概率；

文档概率模型

根据贝叶斯定理有

$$sim(d_j,q)=\frac{p(R|d_j)}{p(\overline{R}|d_j)}\Rightarrow\frac{p(d_j|R)\times p(R)}{p(d_j|\overline{R})\times p(\overline{R})}$$

假设标引词独立，则（*Ki*：所有的词）

$$sim(d_j,q)\sim\frac{(\prod_{g_i(d_j)=1}P(k_i|R))\times(\prod_{g_i(d_j)=0}P(\overline{k}_i|R))}{\prod_{g_i(d_j)=1}P(k_i|\overline{R}))\times(\prod_{g_i(d_j)=0}P(\overline{k}_i|\overline{R})}$$

取对数，在相同背景下，忽略恒定不变的因子：

$$sim(d_j,q)\sim\sum_{i=1}^{t}w_{iq}\times w_{ij}\times\left(\log\frac{P(k_i|R)}{1-P(k_i|R)}+\log\frac{1-P(k_i|\overline{R})}{P(k_i|\overline{R})}\right)$$

词权重 $w_{ij}\in\{0,1\},w_{iq}\in\{0,1\}$

文档概率模型

V：相关文档子集文档数，V_i：包含索引词k_i的文档数。

1).用相关文档中索引词k_i的分布来估计：

$$P(k_i \mid R) = \frac{V_i}{V}$$

2). 不相关文档中的索引词k_i的分布可以通过文档集中索引词的分布来估计

$$P(k_i \mid \overline{R}) = \frac{n_i - V_i}{N - V}$$

n_i表示包含索引词k_i的文档数，N表示集合中的文档总数。

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P(k_i \mid R)}{P(k_i \mid \overline{R})} \right)$$

$$W(t \text{ assigned}) = \log \frac{r(N - R - n + r)}{(R - r)(n - r)}$$

V→R Vi →r

文档概率模型

- 常用的变形形式：

$$w = \log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)}$$

(BM0) $w = 1$

(BM1) $$w = \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)}$$

(BM15) $$w = \frac{tf}{(k_1 + tf)} \times \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)} + k_2 \times nq \frac{(\Delta - d)}{(\Delta + d)}$$

(BM11) $$w = \frac{tf}{(\frac{k_1 \times d}{\Delta} + tf)} \times \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)} + k_2 \times nq \frac{(\Delta - d)}{(\Delta + d)}.$$

qtf:query term frequency; tf: term frequency
k1,k2,k3: 常量 (Okapi)

文本间相似性计算

Similarity Measure

- a similarity measure can represent the similarity between two documents, two queries, or one document and one query
  - it is possible to rank the retrieved documents in the order of presumed importance
- A similarity measure is a function which computes the *degree of similarity* between a pair of text objects
- There are a large number of similarity measures proposed in the literature, because the *best* similarity measure doesn't exist (yet!)

基于概率模型的相关度

- 查询与文档之间的相关性

$$sim(d_j,q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times \left(\log \frac{P(k_i \mid R)}{1-P(k_i \mid R)} + \log \frac{1-P(k_i \mid \overline{R})}{P(k_i \mid \overline{R})} \right)$$

- Okapi系统
  - 伦敦城市大学开发,
  - 20世纪80年代末问世
  - 经过10多年的发展，Okapi系统越来越健壮，检索精确度也越来越高
  - 在TREC比赛中，有不少参加者采用Okapi系统取得了很好的成绩。
  - 不过，Okapi系统不是免费的，并且不提供源代码
  - http://www.soi.city.ac.uk/~andym/OKAPI-PACK/index.html

基于概率模型的相关度

(BM0) $w = 1$

(BM1) $w = \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)}$

(BM15) $w = \frac{tf}{(k_1 + tf)} \times \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)} + k_2 \times nq \frac{(\Delta - d)}{(\Delta + d)}$

(BM11) $w = \frac{tf}{(\frac{k_1 \times d}{\Delta} + tf)} \times \log \frac{N - n + 0.5}{n + 0.5} \times \frac{qtf}{(k_3 + qtf)} + k_2 \times nq \frac{(\Delta - d)}{(\Delta + d)}.$

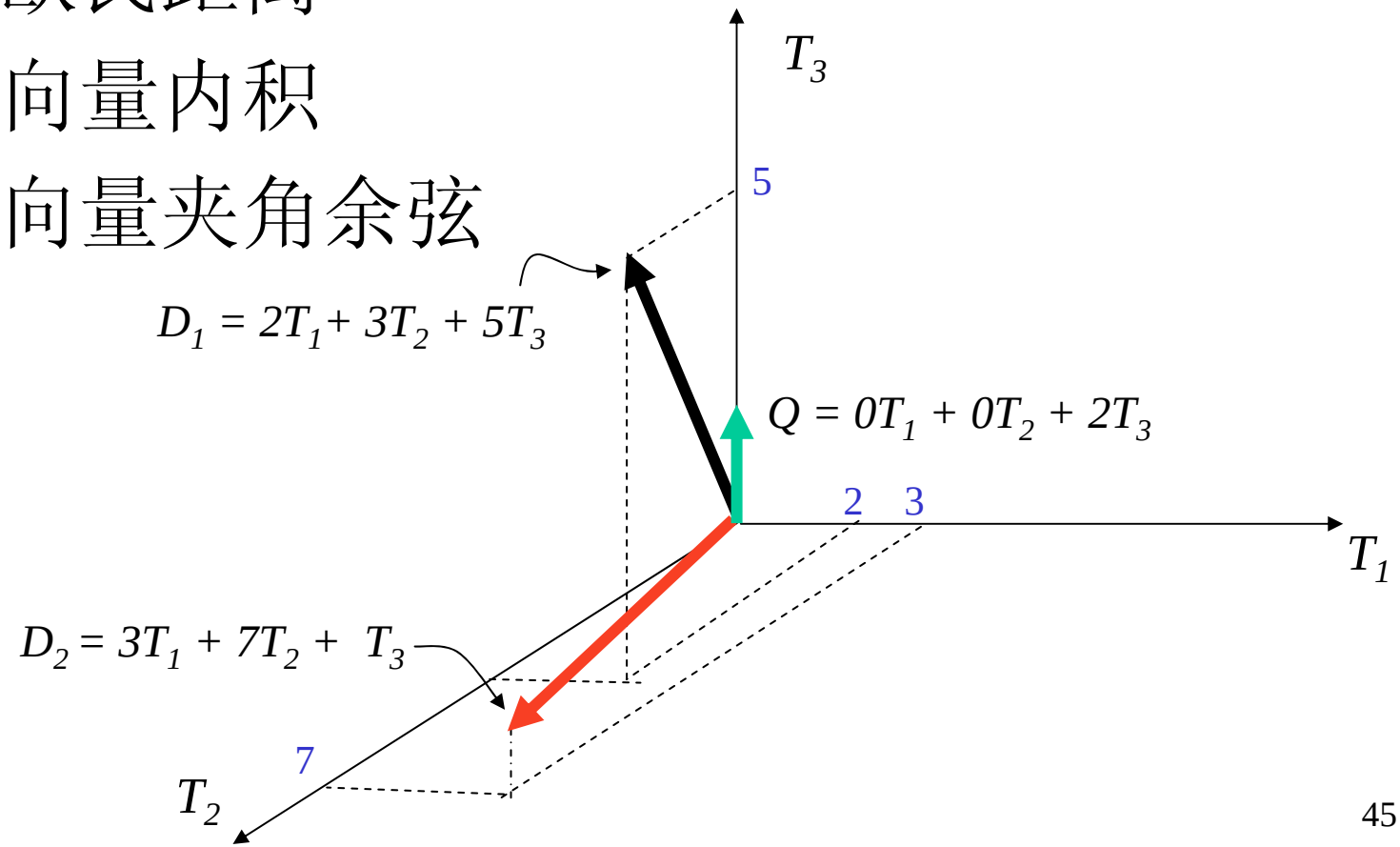
BM2500 $$\sum_{T \in Q} w^1 \frac{(k_1 + 1) tf \, (k_3 + 1) qtf}{(K + tf)(k_3 + qft)}$$

$$w^1 = \log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)}$$

- qtf:query term frequency; tf: term frequency; ki: 常量

基于VSM的相关度计算方法

- 基于向量空间模型的常用方法
  - 欧氏距离
  - 向量内积
  - 向量夹角余弦



欧氏距离

- 空间两点的欧氏距离：

$$Dis(x,y) = |x-y| = \sqrt{\sum_{k=1}^{t}(x_k - y_k)^2}$$

Weighted

$$D_1 = 2T_1 + 3T_2 + 5T_3$$

$$D_2 = 3T_1 + 7T_2 + T_3$$

$$\text{sim}(D_1, D_2) = \sqrt{(1 + 16 + 16)} = \sqrt{33} = 5.74$$

该方法很少使用

向量内积相似度

$$Sim(x, y) = x \bullet y = \sum_{k=1}^{t} (x_k \cdot y_k)$$

Binary:

	retrieval	database	architecture	computer	text	management	information
D =	1,	1,	1,	0,	1,	1,	0
Q =	1,	0 ,	1,	0,	0,	1,	1

sim(D, Q) = 3

Weighted

$D_1 = 2T_1 + 3T_2 + 5T_3 \quad D_2 = 3T_1 + 7T_2 + T_3$

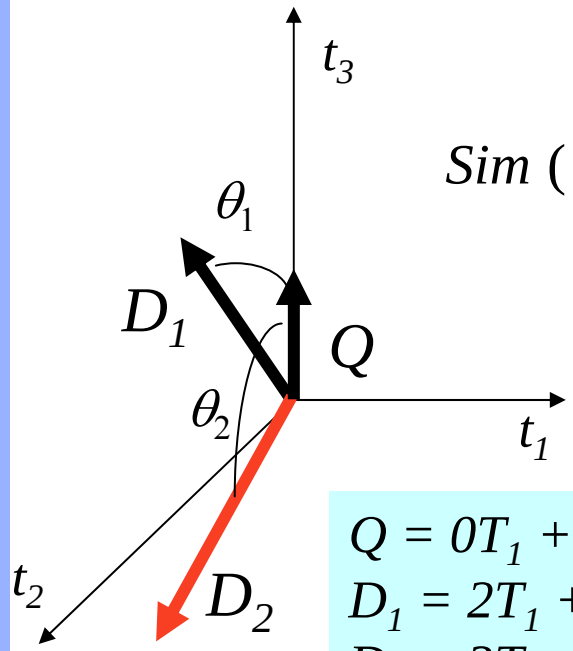
$Q = 0T_1 + 0T_2 + 2T_3$

sim(D_1 , Q) = 2*0 + 3*0 + 5*2 = 10

sim(D_2 , Q) = 3*0 + 7*0 + 1*2 = 2

余弦相似度

- 两个向量间夹角的余弦值作为文档间的相似度



$$Sim(x, y) = \frac{x \bullet y}{|x| \cdot |y|} = \frac{\sum_{k=1}^{t} (x_k \cdot y_k)}{\sqrt{\sum_{k=1}^{t} x_k^2} \cdot \sqrt{\sum_{k=1}^{t} \cdot y_k^2}}$$

$Q = 0T_1 + 0T_2 + 2T_3$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10/ √ (38*4) = 0.81

$D_2 = 3T_1 + 7T_2 + T_3$ CosSim(D_2 , Q) = 2 / √ (59*4) = 0.13

余弦相似度与内积相似度的差别？

Inner product normalized by the vector lengths

余弦相似度

$$Sim(x,y)=\frac{x\bullet y}{|x|\cdot|y|}=\frac{x}{|x|}\bullet\frac{y}{|y|}$$

$$x'=\frac{x}{|x|}=x\Big/\sqrt{\sum_{k=1}^{t}x_k^{\,2}}$$

$$Sim(x,y)=\sum_{k=1}^{t}(x'_k\cdot y'_k)$$

效率：大量计算两两文档间相似都时，为降低计算量，先对文档进行向量进行单位化

余弦相似度的应用

- Smart retrieval system
- http://ftp.cs.cornell.edu/pub/smart/
- G. Salton ed. The SMART Retrieval System—Experiments in Automatic Document RetrievalEnglewood Cliff, NJ: Prentice Hall Inc.; 1971
- G. Salton, M.J. McGill Introduction to Modern Information Retrieval McGraw-Hill Book Co. New York, NY, USA 1986

$$Sim(Q,D) = \frac{\sum_{k=1}^{t}(w_{qk} \cdot w_{dk})}{\sqrt{\sum_{k=1}^{t}(w_{qk})^2 \cdot \sum_{k=1}^{t}(w_{dk})^2}}$$

余弦相似度的应用

- G. Salton, C. Buckley, Term weighting approaches in automatic text retrieval. InformationProcessing and Management 24(5):513-523, 1988

Table 1. Term-weighting componerts

Term Frequency Component		
b	1.0	binary weight equal to 1 for terms present in a vector (term frequency is ignored)
t	tf	raw term frequency (number of times a term occurs in a document or query text)
n	$0.5 + 0.5 \frac{\text{tf}}{\max \text{tf}}$	augmented normalized term frequency (tf factor normalized by maximum tf in the vector, and further normalized to lie between 0.5 and 1.0)
Collection Frequency Component		
x	1.0	no change in weight; use original term frequency component (*b*, *t*, or *n*)
f	$\log \frac{N}{n}$	multiply original tf factor by an inverse collection frequency factor (*N* is total number of documents in collection, and *n* is number of documents to which a term is assigned)
p	$\log \frac{N-n}{n}$	multiply tf factor by a probabilistic inverse collection frequency factor
Normalization Component		
x	1.0	no change; use factors derived from term frequency and collection frequency only (no normalization)
c	$1 \Big/ \sqrt{\sum_{\text{vector}} w_i^2}$	use cosine normalization where each term weight *w* is divided by a factor representing Euclidian vector length

余弦相似度的应用

Table 2. Typical term-weighting formulas

Weighting System	Document term weight	Query Term weight
Best fully weighted system *tfc·nfx*	$\frac{\mathrm{tf}\cdot\log\frac{N}{n}}{\sqrt{\sum_{\text{vector}}\left(\mathrm{tf}_i\cdot\log\frac{N}{n_i}\right)^2}}$	$\left(0.5+\frac{0.5\ \mathrm{tf}}{\max\ \mathrm{tf}}\right)\cdot\log\frac{N}{n}$
Best weighted probabilistic weight *nxx·bpx*	$0.5+\frac{0.5\ \mathrm{tf}}{\max\ \mathrm{tf}}$	$\log\frac{N-n}{n}$
Classical idf weight *bfx·bfx*	$\log\frac{N}{n}$	$\log\frac{N}{n}$
Binary term independence *bxx·bpx*	1	$\log\frac{N-n}{n}$
Standard tf weight: *txc·txx*	$\frac{\mathrm{tf}}{\sqrt{\sum_{\text{vector}}(\mathrm{tf}_i)^2}}$	tf

余弦相似度的应用

Table 4. Performance results for eight term-weighting methods averaged over 5 collections

Term-weighting methods	Rank of method and ave precision	CACM 3204 docs 64 queries	CISI 1460 docs 112 queries	CRAN 1397 docs 225 queries	INSPEC 12,684 docs 84 queries	MED 1033 docs 30 queries	Averages for 5 collections
1. Best fully weighted (*tfc·nfx*)	Rank	1	14	19	3	19	11.2
	P	0.3630	0.2189	0.3841	0.2626	0.5628	
2. Weighted with inverse frequency *f* not used for docs (*txc·nfx*)	Rank	25	14	7	4	32	16.4
	P	0.3252	0.2189	0.3950	0.2626	0.5542	
3. Classical tf × idf No normalization (*tfx·tfx*)	Rank	29	22	219	45	132	84.4
	P	0.3248	0.2166	0.2991	0.2365	0.5177	
4. Best weighted probabilistic (*nxx·bpx*)	Rank	55	208	11	97	60	86.2
	P	0.3090	0.1441	0.3899	0.2093	0.5449	
5. Classical idf without normalization (*bfx·bfx*)	Rank	143	247	183	160	178	182
	P	0.2535	0.1410	0.3184	0.1781	0.5062	
6. Binary independence probabilistic (*bxx·bpx*)	Rank	166	262	154	195	147	159
	P	0.2376	0.1233	0.3266	0.1563	0.5116	
7. Standard weights cosine normalization (original Smart) (*txc·txx*)	Rank	178	173	137	187	246	184
	P	0.2102	0.1539	0.3408	0.1620	0.4641	
8. Coordination level binary vectors (*bxx·bxx*)	Rank	196	284	280	258	281	260
	P	0.1848	0.1033	0.2414	0.0944	0.4132	

Jaccard相似度

Jaccard Coefficient:

$$Sim(x, y) = \frac{x \bullet y}{|x| + |y| - x \bullet y}$$

$$= \frac{\sum_{k=1}^{t} (x_k \cdot y_k)}{\sum_{k=1}^{t} x_k^{\,2} + \sum_{k=1}^{t} y_k^{\,2} - \sum_{k=1}^{t} (x_k \bullet y_k)}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ Sim(D_1 , Q) = 10 / (38+4-10) = 10/32 = 0.31
$D_2 = 3T_1 + 7T_2 + T_3$ Sim(D_2 , Q) = 2 / (59+4-2) = 2/61 = 0.04
$Q = 0T_1 + 0T_2 + 2T_3$

- Difference between Jaccard and CosSim?

基于向量内积的几种方法的对比

Inner Product: $\sum_{k=1}^{t}(x_k \bullet y_k)$

Cosine: $$\frac{\sum_{k=1}^{t}(x_k \bullet y_k)}{\sqrt{\sum_{k=1}^{t} x_k^2 \bullet \sum_{k=1}^{t} y_k^2}}$$

Jaccard : $$\frac{\sum_{k=1}^{t}(x_k \bullet y_k)}{\sum_{k=1}^{t} x_k^2 + \sum_{k=1}^{t} y_k^2 - \sum_{k=1}^{t}(x_k \bullet y_k)}$$

基于集合计算的几种方法

$\lvert X \cap Y \rvert$	Simple matching
$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	Dice's Coefficient
$\frac{\lvert X \cap Y \rvert}{\lvert X \cup Y \rvert}$	Jaccard's Coefficient
$\frac{X \bullet Y}{\lvert X \rvert^{1/2} \times \lvert Y \rvert^{1/2}}$	Cosine Coefficient
$\frac{\lvert X \cap Y \rvert}{\min(\lvert X \rvert, \lvert Y \rvert)}$	Overlap Coefficient

几种方法的对比

Inner Product: $\sum_{k=1}^{t}(x_k \bullet y_k)$ $\quad |x \cap y|$

Cosine: $\dfrac{\sum_{k=1}^{t}(x_k \bullet y_k)}{\sqrt{\sum_{k=1}^{t} x_k^2 \bullet \sum_{k=1}^{t} y_k^2}}$ $\quad \dfrac{|x \cap y|}{\sqrt{|x|} \cdot \sqrt{|y|}}$

Jaccard : $\dfrac{\sum_{k=1}^{t}(x_k \bullet y_k)}{\sum_{k=1}^{t} x_k^2 + \sum_{k=1}^{t} y_k^2 - \sum_{k=1}^{t}(x_k \bullet y_k)}$ $\quad \dfrac{|x \cap y|}{|x| + |y| - |x \cap y|}$

x and *y* here are vectors

x and *y* here are sets of keywords

文本序列

文本序列

- 前面的方法（除分词外）均没有考虑文本的顺序
- 文本顺序在表达文本语义具有重要意义
  - 是近年的研究热点

序列比较

- 序列比较可以分为四种基本情况：
  - （1）两条长度相近的序列相似找出序列的差别
  - （2）一条序列是否包含另一条序列(子序列)
  - （3）两条序列中是否有非常相同的子序列
  - （4）一条序列与另一条序列逆序相似
- **相似度**：它是两个序列的函数，其值越大，表示两个序列越相似
- **距离**：距离越大，则两个序列的相似度就越小

海明距离

s =	AAT	AGCAA	AGCACACA
t =	TAA	ACATA	ACACACTA
Hamming Distance(s,t)=	2	3	6

直接距离计算的不足?

反序、子序列、漂移等关系不能发现

编辑距离

- 编辑距离
  - 用来计算从原串（s）转换到目标串(t)所需要的最少的插入，删除和替换的数目
- 举例说明：
  - 原序列：　她是剧院的一明星。
  - 目标序列：她是京剧团的明星。
  - 编辑距离：3
- 插入：京
- 替换：团→院
- 删除：一

编辑操作

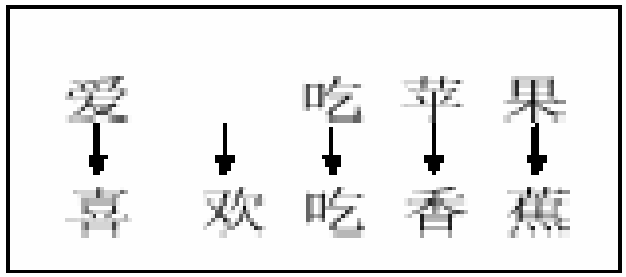
- 字符编辑操作可将一个序列转化为一个新序列
  - Match（a，a）
  - Delete（a，-）
  - Replace（a，b）
  - Insert（-，b）

编辑操作的代价

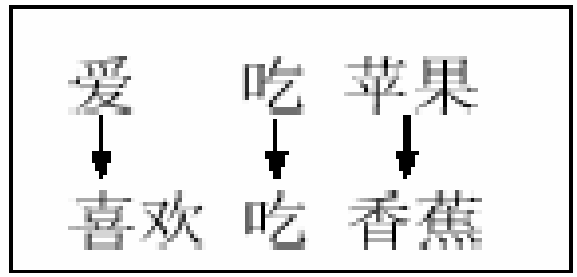
- 为编辑操作定义函数w，它表示“代价（cost）”或“权重（weight）”
- 对字母表A中的任意字符a、b，定义
  - w (a, a) = 0
  - w (a, b) = 1　a ≠ b
  - w (a, -) = w (-, b) = 1
- 不同编辑操作的代价不同，使用得分（score）函数来评价编辑操作
  - p (a, a) = 1
  - p (a, b) = 0　a ≠ b
  - p (a, -) = w (-, b) = - 1

编辑操作的代价

- (a)“爱吃苹果”与“喜欢吃香蕉”之间的编辑距离为4，如四条虚线所显示;
- (b)“爱吃苹果”与“喜欢吃香蕉”之间的改进编辑距离为1.1，
  - 其中“爱”→“喜欢”代价为0.5;
  - “苹果” →“香蕉”代价为0.6

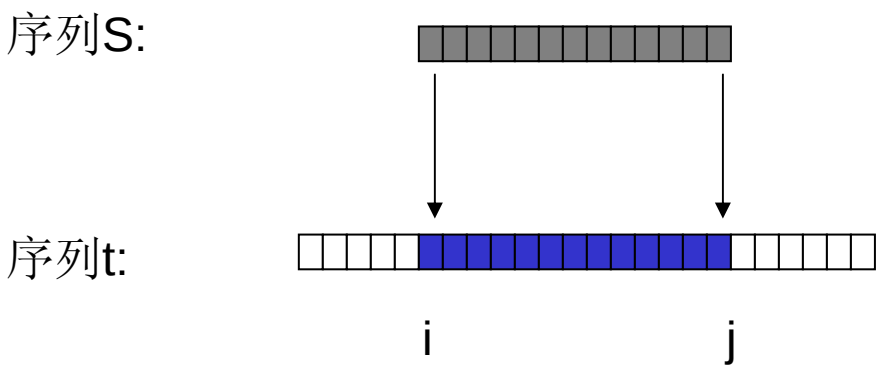


(a)



(b)

子序列与完整序列的比对



不计前缀的得分，也不计删除后缀的得分

编辑距离算法

- 编辑距离的算法是首先由俄国科学家Levenshtein提出的，故又叫Levenshtein Distance

Step	Description
1	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing 0..m rows and 0..n columns.
2	Initialize the first row to 0..n. Initialize the first column to 0..m.

编辑距离算法

3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If s[i] equals t[j], the cost is 0. If s[i] doesn't equal t[j], the cost is 1.
6	**Set cell d[i,j] of the matrix equal to the minimum of:** **a. The cell immediately above plus 1: d[i-1,j] + 1.** **b. The cell immediately to the left plus 1: d[i,j-1] + 1.** **c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.**
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

编辑距离算法例子

S = "GUMBO" T = "GAMBOL".

Steps 1 and 2

		G	U	M	B	O
	0	1	2	3	4	5
G	1					
A	2					
M	3					
B	4					
O	5					
L	6					

Steps 3 to 6 When i = 1

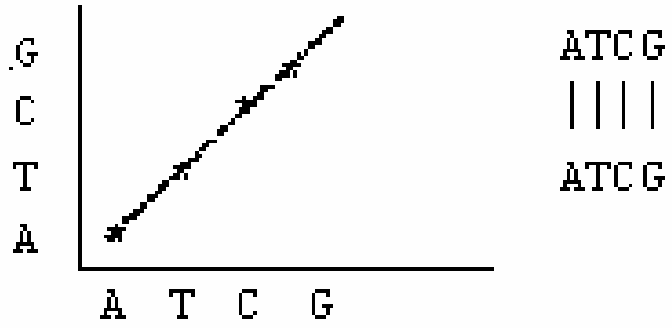
		G	U	M	B	O
	0	1	2	3	4	5
G	1	0				
A	2	1				
M	3	2				
B	4	3				
O	5	4				
L	6	5				

Steps 3 to 6 When i = 5

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

通过点矩阵进行序列比较

- “矩阵作图法” 或 “对角线作图”



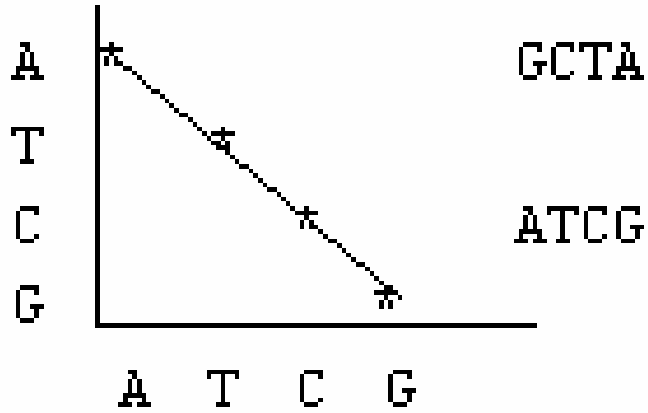
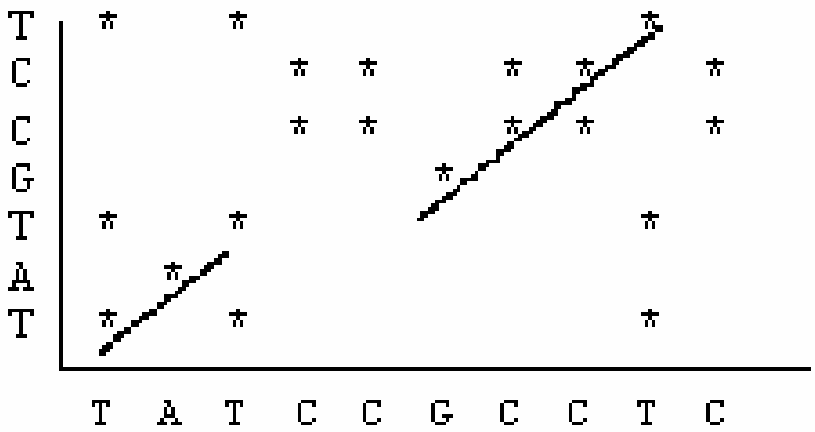


图3.2 序列比较矩阵标记图



TAT--GCCT
||| |||| 3.4 反向序列矩阵标记图
TATCCGCCTC

图 3.5 多个相同子序列矩阵标记图

TextTiling 简介

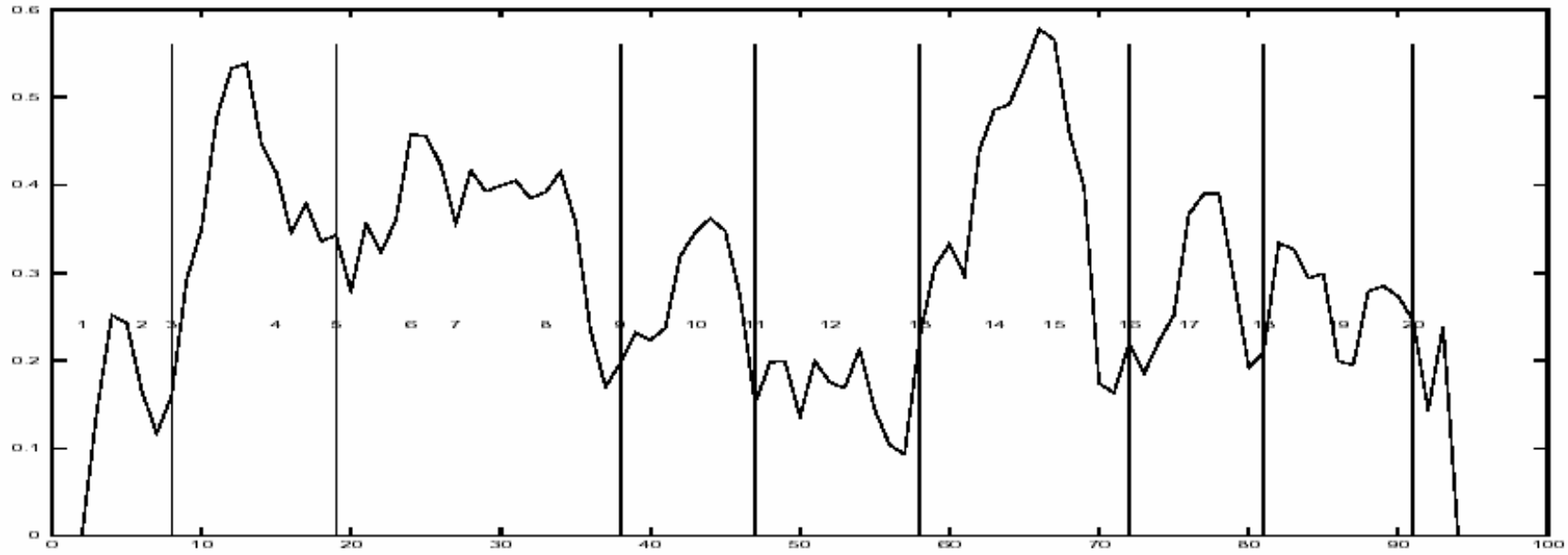
- TextTiling (Hearst and Plaunt 1993) is one of the most famous system for Topic Segmentation .(对文档分段)
- The basic idea of this algorithm is to search for parts of a text where the vocabulary shifts from one **subtopic** to another. These points are then interpreted as the boundaries of multi-paragraph units.

TextTiling method

- 先把文档分成块（tile），每块k个句子。
- 计算相邻块之间的相似度，用标准的文档相似度计算公式。
- 用结果绘图。非常相似处会出现波峰，很不相似的地方出现波谷。选择波谷处作为分界线，把块组成段，这些段很可能是有关同一个Sub topic的。

TextTiling: Cohesion

The similarity calculation can be illustrated as the following figure where the x-axis is the gap number.



特征空间的变化

隐语义分析(LSA)

- Latent Semantic Analysis (LSA)
- Latent Semantic Indexing (LSI)
- 问题提出：一词多义和同义词
- 中心思想：用概念（或特征）代替词
- 基本方法
  - 利用矩阵理论中的“奇异值分解（singular value decomposition,SVD）”技术，将词频矩阵转化为奇异矩阵（K×K）

隐语义分析(LSA)

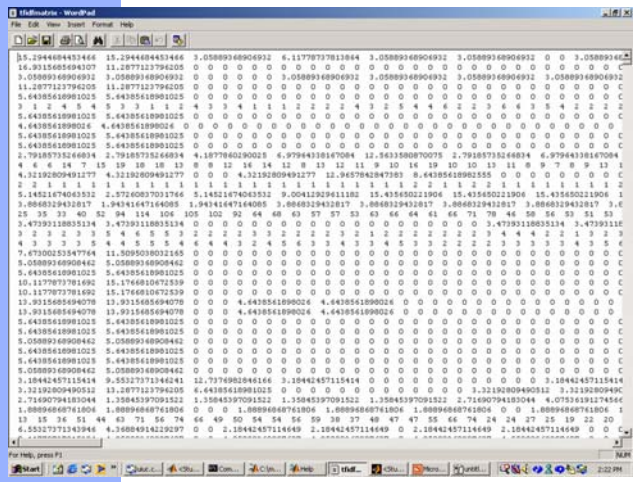
- LSI思想方法最初应用于文本信息检索领域有效地解决了同义词和多义词的问题，通过识别文本中的同义词，LSI将信息检索精度提高了10%--30%.（查询扩展）
- 随着应用领域的不断拓展，LSI在信息过滤、信息分类/聚类、交叉语言检索、信息理解、判断和预测等众多领域中得到了广泛的应用。（语义，降维）

隐语义分析(LSA)

- Introduced in 1990; improved in 1995
- S. Deerwester, S. Dumas, G. Furnas, T. Landauer, R. Harsman: *Indexing by latent semantic analysis,* J. American Society for Information Science, 41, 1990, pp. 391-407
- M. W. Berry, S.T. Dumas, G.W. O'Brien: *Using linear algebra for intelligent information retrieval,* SIAM Review, 37, 1995, pp. 573-595
- Based on spectral analysis of term-document matrix

隐语义分析(LSA)

Weighted Frequency Matrix



Query Terms:
- **Insulation**
- **Joint**

DOCUMENTS:

'CM031.txt'

'CM046.txt'

'CM001.txt'

'CM029.txt'

'CM040.txt'

K>> return

TERMS:

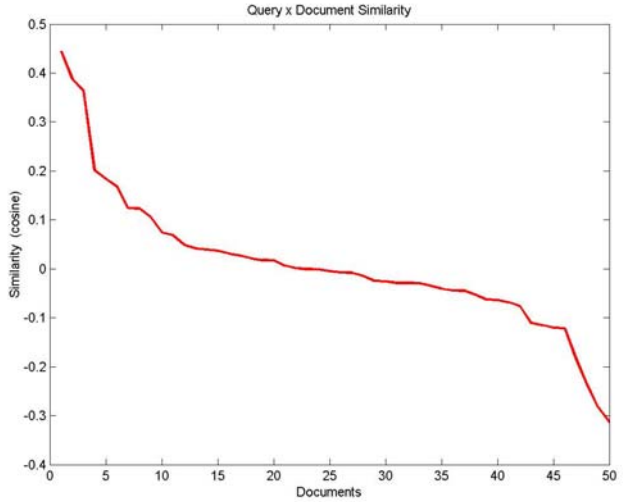
'joint'

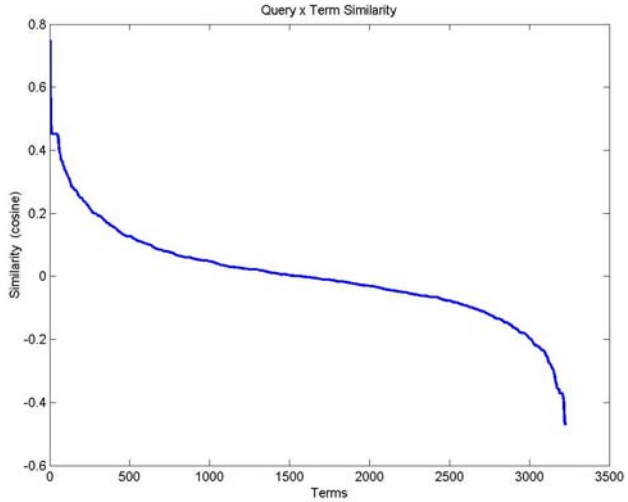
'insulation'

'roofing'

'expansion'

'saw'





隐语义分析(LSA)

- 输入：term-by-document matrix
- 输出：
  - U: concept-by-term matrix
  - V: concept-by-document matrix
  - S: elements assign weights to concepts

基本步骤

- 1.建立词频矩阵，frequency matrix
- 2.计算frequency matrix的奇异值分解
  - 分解frequency matrix成3个矩阵U，S，V。U和V是正交矩阵（$U^TU=I$），S是奇异值的对角矩阵（K×K）
- 3.对于每一个文档 d，用排除了SVD中消除后的词的新的向量替换原有的向量
- 4.用转换后的文档索引和相似度计算

词频矩阵

- 词频矩阵：矩阵表示一组文档
  - 行对应关键词t，列对应文档d向量
  - 将每一个文档视为空间的一个向量
  - 向量值反映单词t与文档d的关联度

表示文档词频的词频矩阵

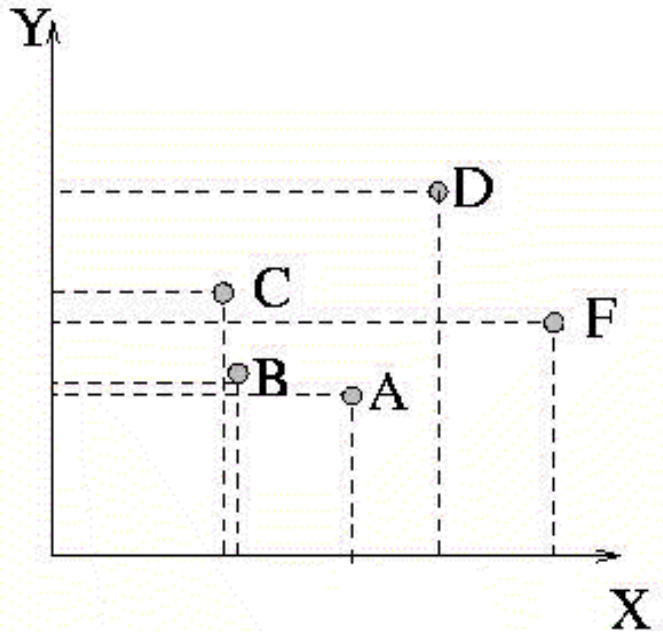
	d_1	d_2	d_3	d_4	d_5	d_6
t_1	322	85	35	69	15	320
t_2	361	90	76	57	13	370
t_3	25	33	160	48	221	26
t_4	30	140	70	201	16	35

SVD基本思想

- 直接选维带来的问题：

A的两个最近邻：

- 二维（XY）
  - A的2-nn：B和C
- 一维（X）
  - A的2-nn：B和D

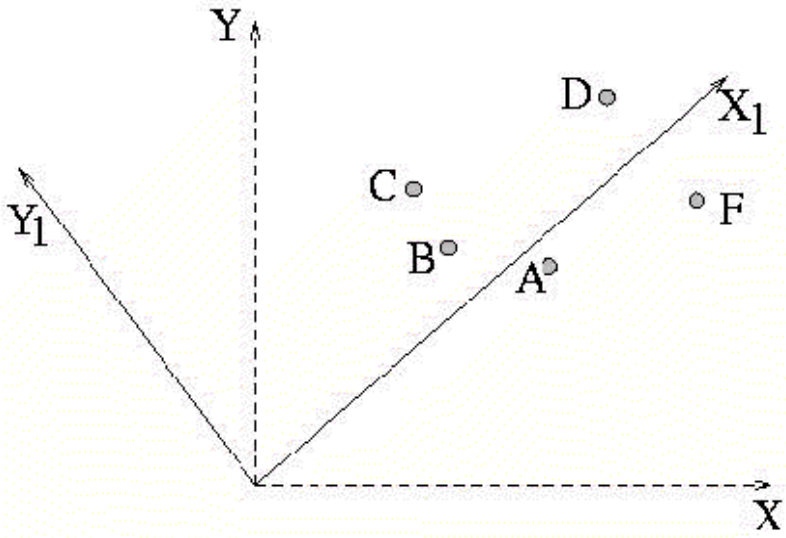


SVD基本思想

- 先旋转坐标轴在进行选维：

$XY \rightarrow X_1 Y_1$

- 二维（XY）
  - A的2-nn： B和C
- 一维（X_1）
  - A的2-nn： B和C



SVD理论基础

Definition 2 A matrix $\boldsymbol{A} = (a_{ij})_{m\times n}$ is orthogonal[a] if

$$\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} = \mathbf{1}_{n\times n} \qquad \text{正交矩阵} \qquad (1)$$

where $\boldsymbol{A}^{\mathsf{T}}$ is the transpose of $\boldsymbol{A}$.

Definition 3 $\boldsymbol{x} \in \mathbb{R}^n$ is an eigenvector of $\boldsymbol{A}_{n\times n}$ if $\lambda \in \mathbb{R}$ such that

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \qquad \text{特征向量} \qquad (2)$$

where λ is called an eigenvalue of $\boldsymbol{A}$. 特征值

Defi nition 6 Given a matrix $\boldsymbol{A}_{m\times n}$ whose rank is r, then the eigenvalues of $\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}$ are

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0$$

$\sigma_i = \sqrt{\lambda_i}$ is called the singular value of $\boldsymbol{A}$ where $i = 1, 2, \cdots, n$.

奇异值

SVD理论基础

Theorem 2 Given a matrix $\boldsymbol{A}_{m\times n}$ whose rank is r and $m \geq n$, there exist two orthogonal matrixes $\boldsymbol{U}_{m\times n} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n)$ (term vectors) and $\boldsymbol{V}_{n\times n} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n)$ (document vectors) s.t.

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} \\ &= \sum_{i=1}^{r} \boldsymbol{u}_i \cdot \sigma_i \cdot \boldsymbol{v}_i^{\mathsf{T}} \end{aligned} \tag{9}$$

可分解

where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n) \in \mathbb{R}^{n\times n}$ and σ_i is the singular value of $\boldsymbol{A}$. (9) is called the Singular Value Decomposition (SVD) of $\boldsymbol{A}$.

SVD理论基础

Theorem 3 (Eckart-Young) Let the SVD of A be given by (9) with $r = \mathrm{rank}(A) \leq p = \min\{m, n\}$ and define

$$A_k = U_k \Sigma_k V_k^{\mathsf{T}} = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^{\mathsf{T}} \tag{10}$$

then A_k is the optimal approximation of A in the view of

可裁减

$$\begin{aligned} \min_{\mathrm{rank}(B)=k} \|A - B\|_F &= \|A - A_k\|_F \\ &= \sqrt{\sum_{i=k+1}^{p} \sigma_i^2} \\ \min_{\mathrm{rank}(B)=k} \|A - B\|_2 &= \|A - A_k\|_2 \\ &= \sigma_{k+1} \end{aligned} \tag{11}$$

隐语义分析(LSA)

- For every $m \times n$ matrix A there is **singular value decomposition (SVD)**

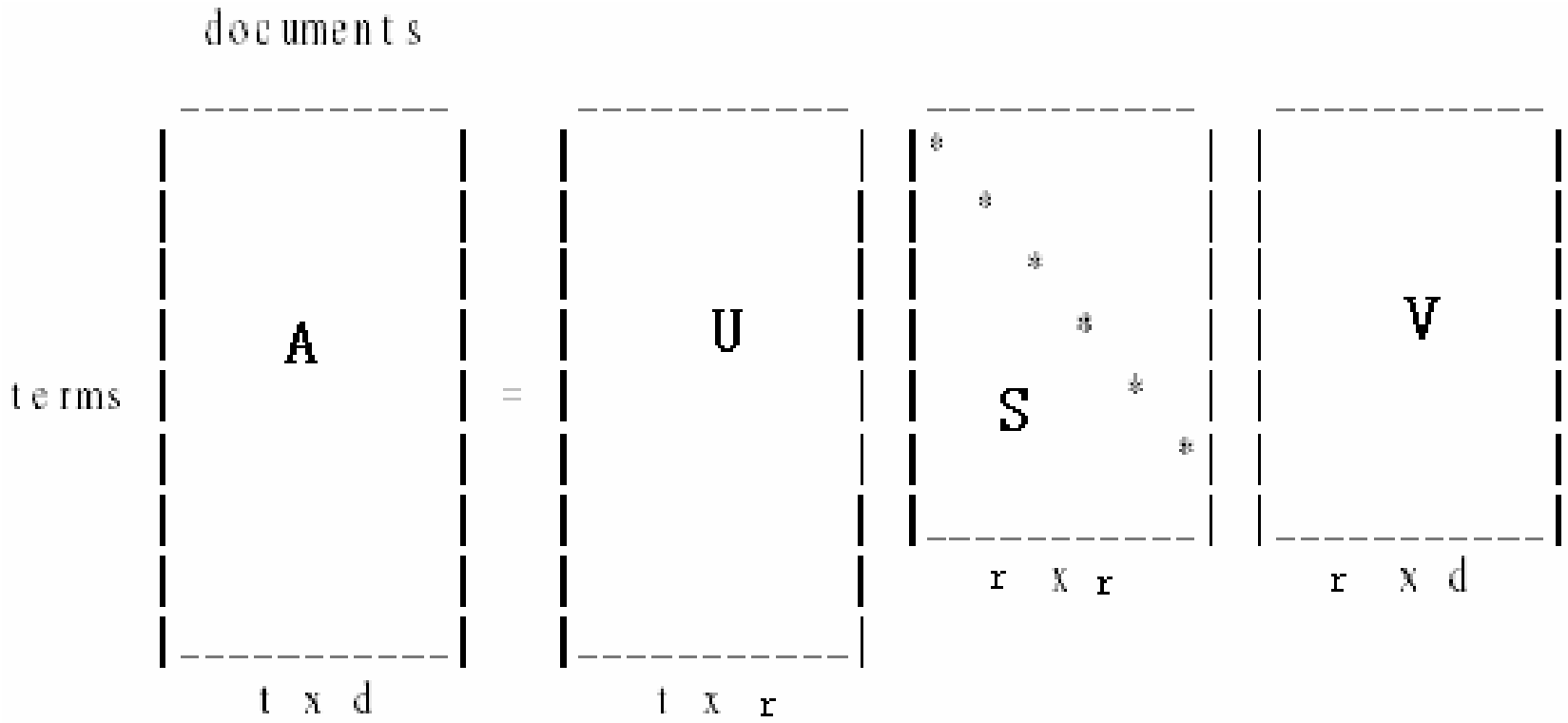
$$A = U\Sigma V^T$$

U orthogonal $m \times r$ matrix whose columns are left singular vectors of A
(正交矩阵 $U^TU=I$, U→ terms)

Σ diagonal matrix on whose diagonal are singular values of matrix A in descending order (对角矩阵，奇异值矩阵)

V orthogonal $r \times n$ matrix whose columns are right singular vectors of A
(正交矩阵 $V^TV=I$, V→documents)

隐语义分析(LSA)



t is the number of rows of X

d is the number of columns of X

r is the rank of X ($\leqslant \min(t, d)$)

隐语义分析(LSA)

- For LSI truncated SVD is used (k<r)

$$A_k = U_k \Sigma_k V_k^T$$

where

U_k is $m \times k$ matrix whose columns are first k left singular vectors of A

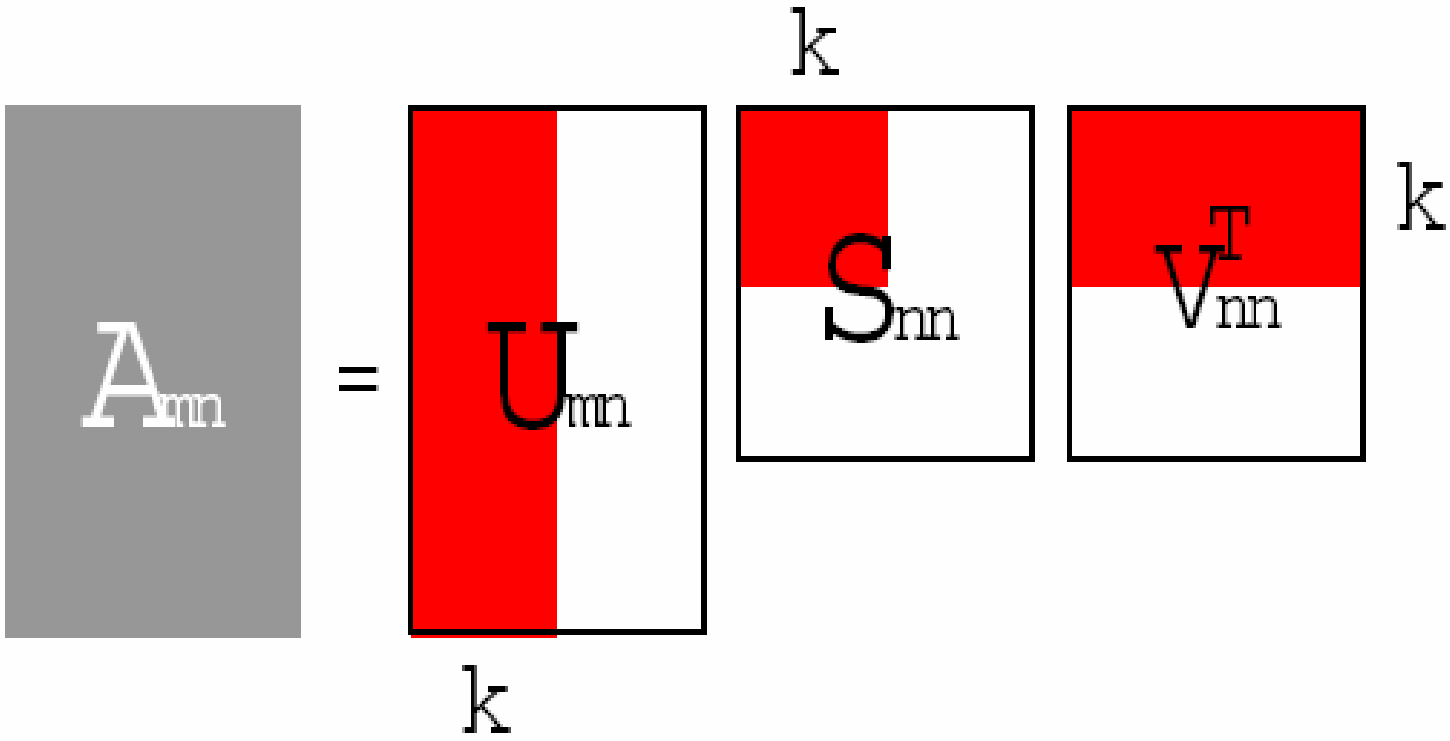
Σ_k is $k \times k$ diagonal matrix whose diagonal is formed by k leading singular values of A

V_k is $n \times k$ matrix whose columns are first k right singular vectors of A

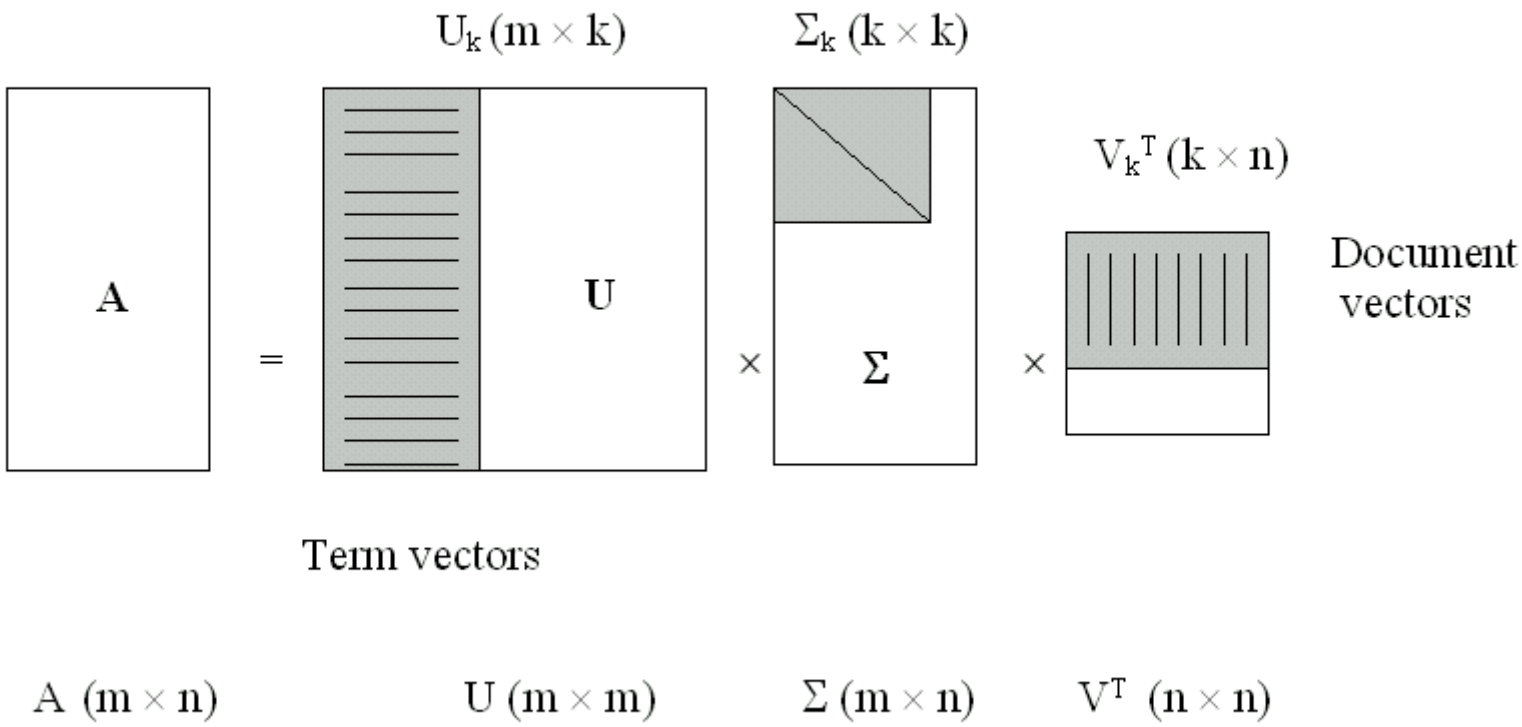
- Rows of U_k = terms
- Rows of V_k = documents

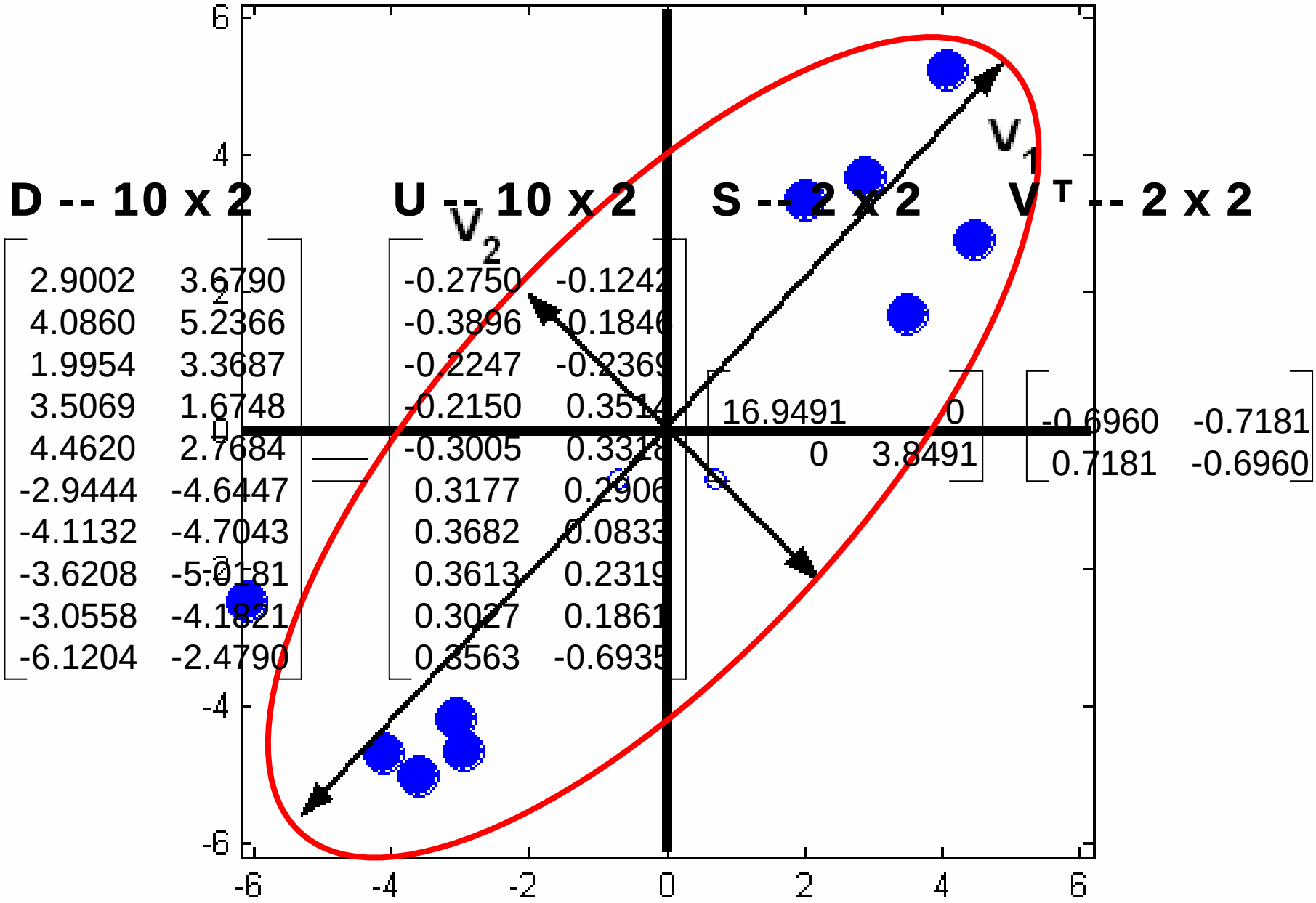
隐语义分析(LSA)

The red part renews $A_{m\times n}$ with rank $k \leq r$ the best.



隐语义分析(LSA)


U_k (m × k)
Σ_k (k × k)
V_k^T (k × n)
A
=
U
×
Σ
×
Document vectors
Term vectors
A (m × n)
U (m × m)
Σ (m × n)
V^T (n × n)


D -- 10 x 2
2.9002 3.6790
4.0860 5.2366
1.9954 3.3687
3.5069 1.6748
4.4620 2.7684
-2.9444 -4.6447
-4.1132 -4.7043
-3.6208 -5.0181
-3.0558 -4.1821
-6.1204 -2.4790
U -- 10 x 2
-0.2750 -0.1242
-0.3896 0.1846
-0.2247 -0.2369
-0.2150 0.3514
-0.3005 0.3318
0.3177 0.2906
0.3682 0.0833
0.3613 0.2319
0.3027 0.1861
0.3563 -0.6935
S -- 2 x 2
16.9491 0
0 3.8491
V T -- 2 x 2
-0.6960 -0.7181
0.7181 -0.6960
V1
V2
6
4
2
0
-2
-4
-6
-6 -4 -2 0 2 4 6

SVD工具

- SVD: matlab实现
- 分别取矩阵a为如下两个值时作SVD分解，即在matlab中键入后，得到：s,u,v的结果（注意s的取值）

```
>a=[1 1; 1 1; 0 0];
>[u,s,v]=svd(a);
>a=[1 0 0 0 1; 0 1 0 1 0; 0 0 1 1 1;0 0 0 0 0];
>[u,s,v]=svd(a);
```

SVD工具

- Svdpack
  - http://www.netlib.org/svdpack/
  - http://scicomp.ewha.ac.kr/netlib/svdpack/
  - This software package implements Lanczos and subspace iteration-based methods for determining several of the largest singular triplets (singular values and corresponding left- and right-singular vectors) for large sparse matrices.

Any Question?