

# 信息检索导论

## An Introduction to Information Retrieval

### 第16讲 隐性语义索引

### Latent Semantic Indexing

授课人：古晓艳

中国科学院信息工程研究所/国科大网络空间安全学院

# 提纲

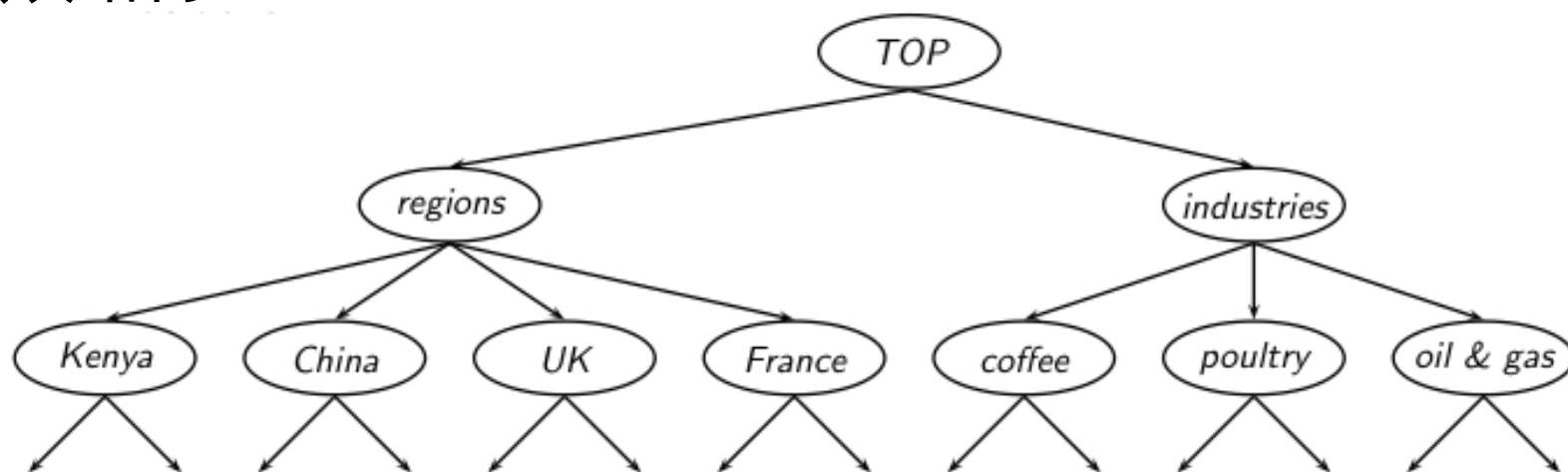
- ① 上一讲回顾
- ② 隐性语义索引
- ③ 空间降维处理
- ④ LSI 在IR中的应用

# 提纲

- ① 上一讲回顾
- ② 隐性语义索引
- ③ 空间降维处理
- ④ LSI 在IR中的应用

# 层次聚类

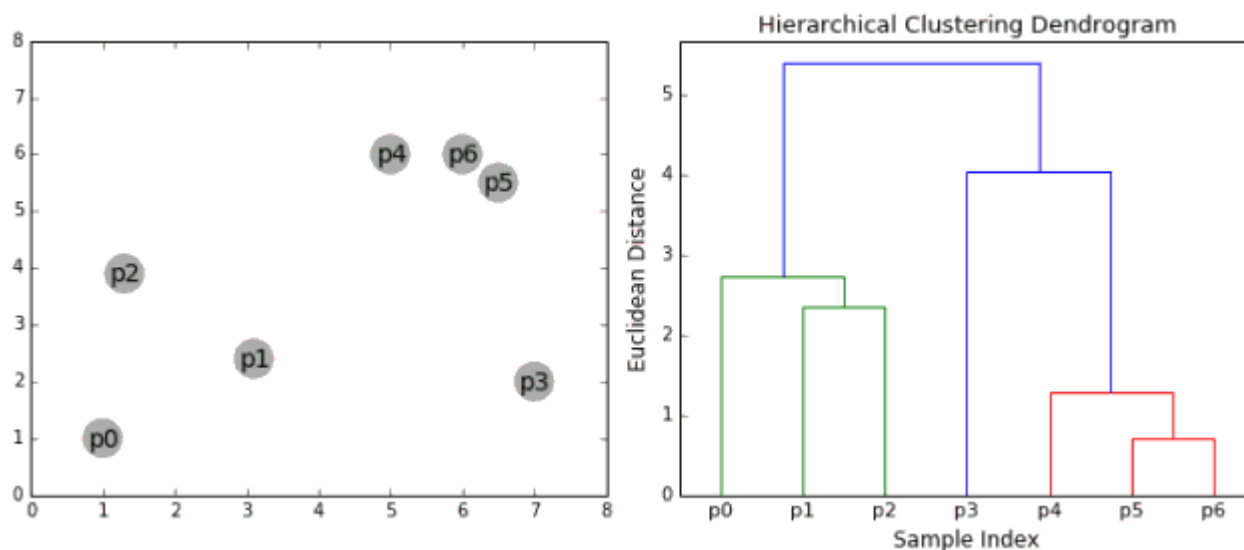
层次聚类的目标是生成类似于前面提到的Reuters目录的一个层次结构：



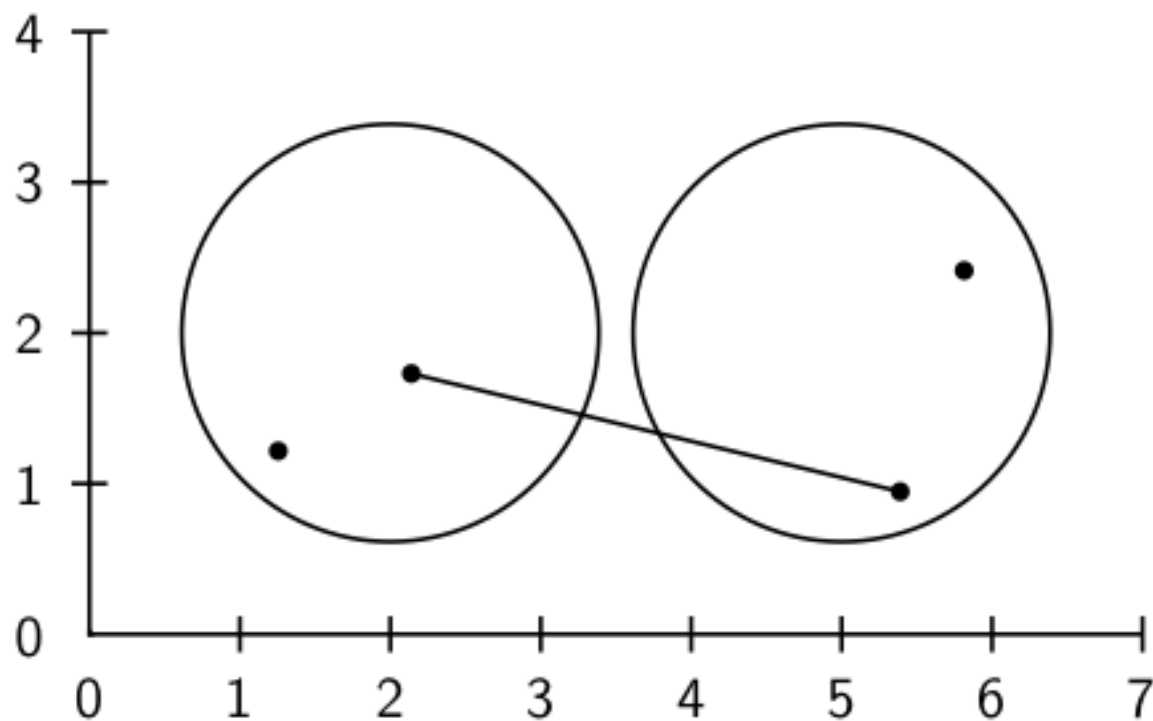
这个层次结构是自动创建的，可以通过自顶向下或自底向上的方法来实现。最著名的自底向上的方法是层次凝聚式聚类(hierarchical agglomerative clustering, HAC)。

# 层次凝聚式聚类 (HAC)

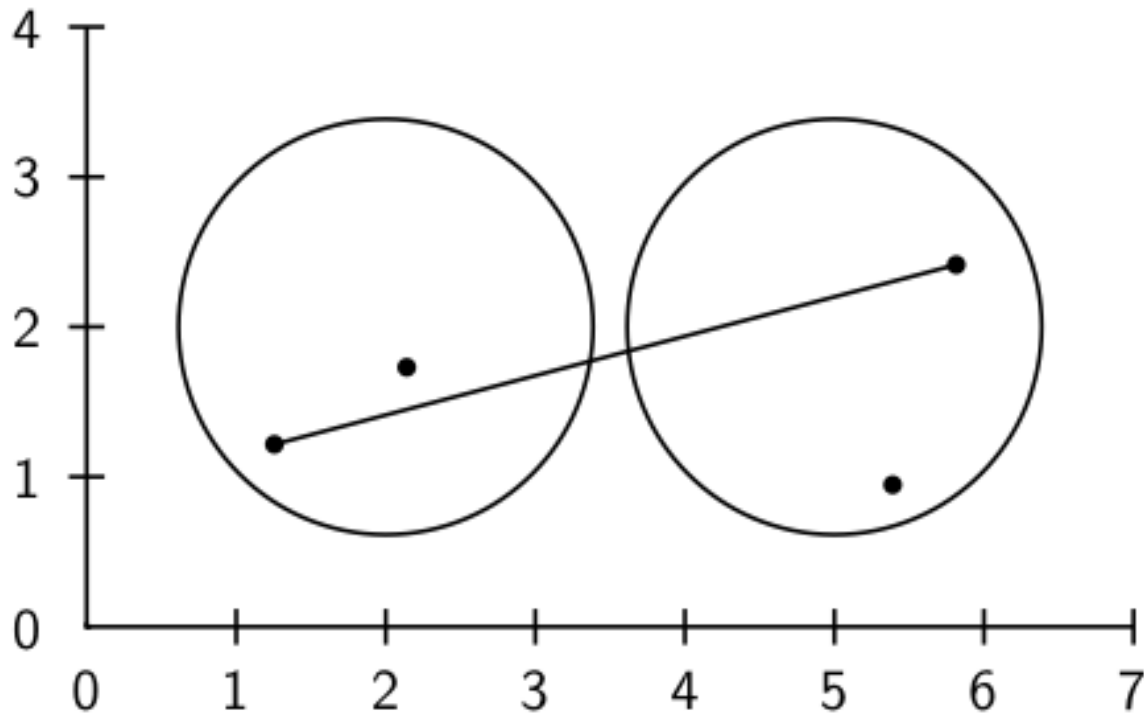
- 一开始每篇文档作为一个独立的簇
- 然后，将其中**最相似**的两个簇进行合并
- 重复上一步直至仅剩一个簇
- 整个合并的历史构成一个二叉树
- 一个标准的描述层次聚类合并历史的方法是采用树状图



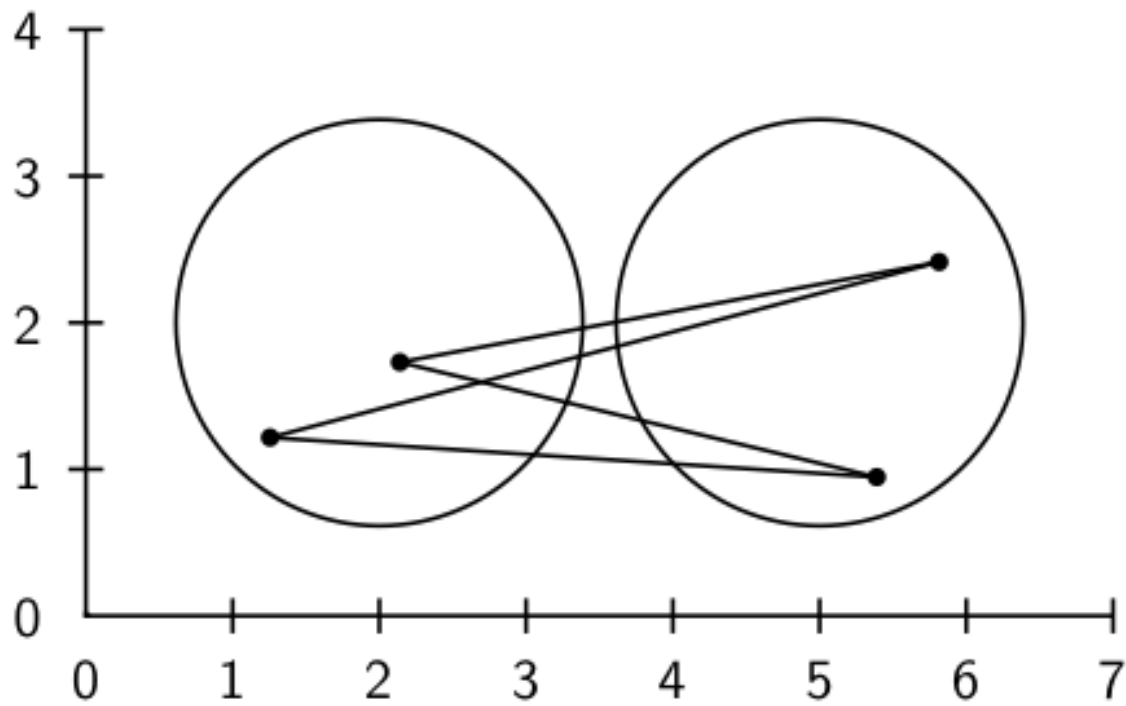
# 单连接: 最大相似度(最短距离)



# 全连接: 最小相似度

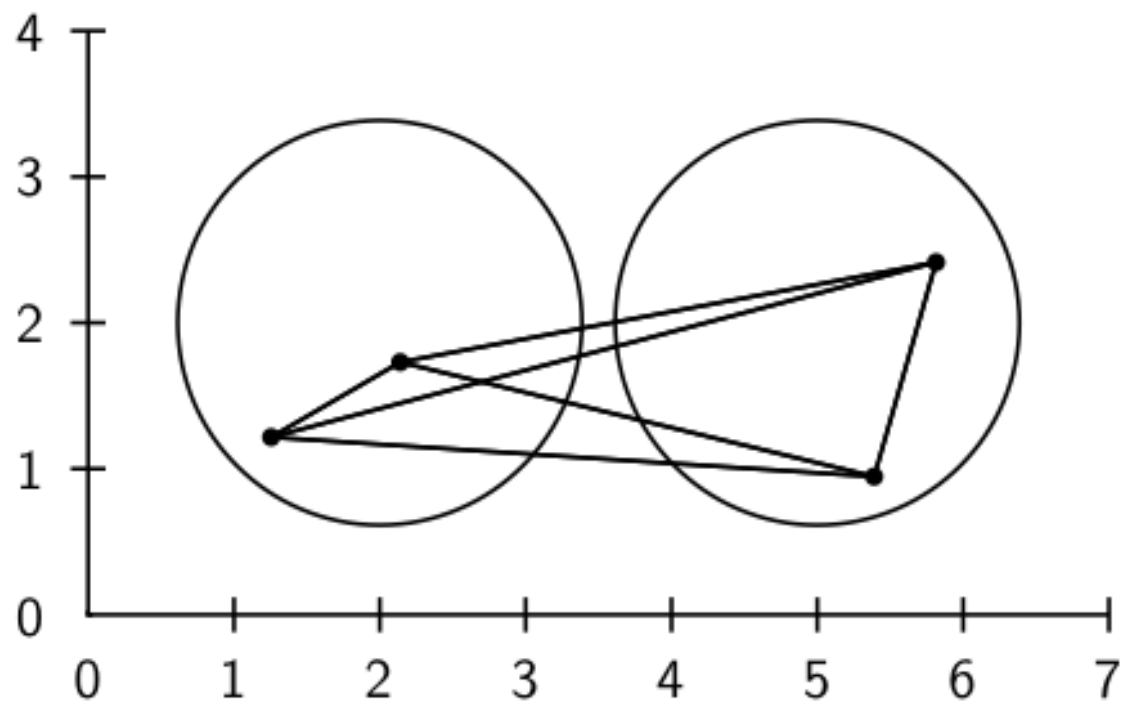


# 质心法

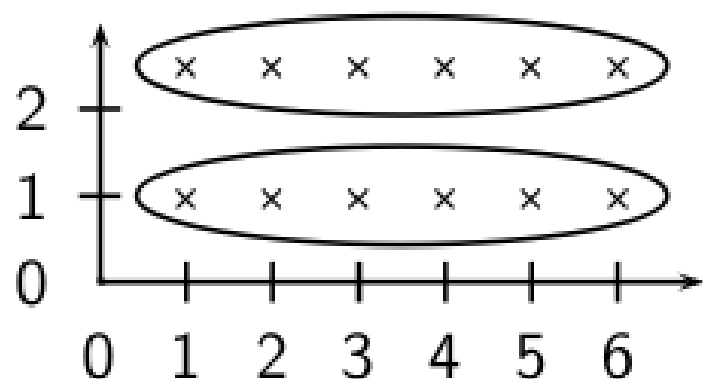




# 组平均

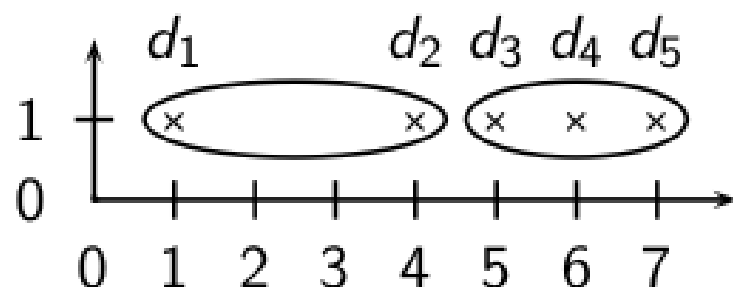


# 单连接方法的链化(Chaining)现象



单连接聚类算法往往产生长的、凌乱的簇结构。对大部分应用来说，这些簇结构并不是所期望的。

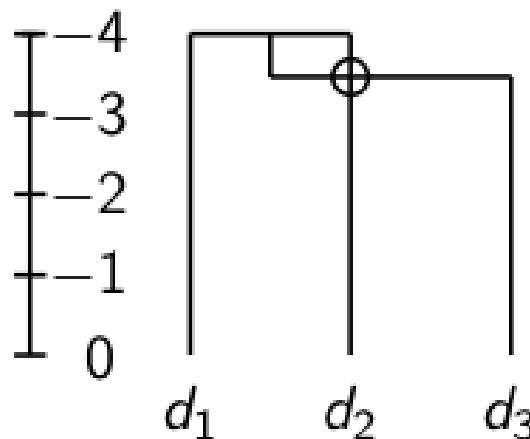
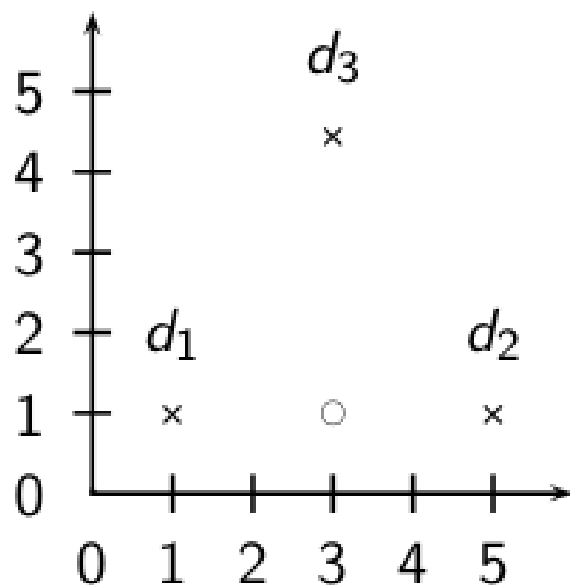
# 全连接法: 对离群点非常敏感



- 全连接聚类将 $d_2$ 和它的正确邻居分开----这显然不是我们所需要的
- 出现上述结果的最主要原因是存在离群点  $d_1$ .
- 这也表明单个离群点的存在会对全连接聚类的结果起负面影响
- 单连接聚类能够较好地处理这种情况

# 质心法聚类过程中的相似度颠倒(Inversion)现象

- 在相似度颠倒过程中，合并过程中相似度会增加，导致“颠倒”的树状图
- 下图中，第一次合并  $(d_1 \cup d_2)$  的相似度是-4.0，第二次合并的相似度  $((d_1 \cup d_2) \cup d_3) \approx -3.5$ .



# 高效的单连接聚类算法

SINGLELINKCLUSTERING( $d_1, \dots, d_N$ )

```

1  for  $n \leftarrow 1$  to  $N$ 
2  do for  $i \leftarrow 1$  to  $N$ 
3      do  $C[n][i].sim \leftarrow \text{SIM}(d_n, d_i)$ 
4       $C[n][i].index \leftarrow i$ 
5       $I[n] \leftarrow n$ 
6       $NBM[n] \leftarrow \arg \max_{X \in \{C[n][i]: n \neq i\}} X.sim$ 
7   $A \leftarrow []$ 
8  for  $n \leftarrow 1$  to  $N - 1$ 
9  do  $i_1 \leftarrow \arg \max_{\{i: I[i]=i\}} NBM[i].sim$ 
10      $i_2 \leftarrow I[NBM[i_1].index]$ 
11      $A.APPEND(\langle i_1, i_2 \rangle)$ 
12     for  $i \leftarrow 1$  to  $N$ 
13     do if  $I[i] = i \wedge i \neq i_1 \wedge i \neq i_2$ 
14         then  $C[i_1][i].sim \leftarrow C[i][i_1].sim \leftarrow \max(C[i_1][i].sim, C[i_2][i].sim)$ 
15         if  $I[i] = i_2$ 
16         then  $I[i] \leftarrow i_1$ 
17      $NBM[i_1] \leftarrow \arg \max_{X \in \{C[i_1][i]: I[i]=i \wedge i \neq i_1\}} X.sim$ 
18  return  $A$ 
    
```

# 四种HAC算法的比较

方 法	结合相似度	时间复杂度	是否最优?	注 释
单连接	簇间文档的最大相似度	$\Theta(N^2)$	yes	链化效应
全连接	簇间文档的最小相似度	$\Theta(N^2 \log N)$	no	对离群点敏感
质心法	所有簇间相似度的平均值	$\Theta(N^2 \log N)$	no	相似度颠倒
组平均	所有文档相似度的平均值	$\Theta(N^2 \log N)$	no	大部分应用中的最佳选择

# 簇标签生成的例子

	文档 数目	簇标签生成方法		
		质心	互信息	标题
4	622	oil plant mexico production crude <b>power</b> <b>000 refinery gas</b> bpd	plant oil production <b>barrels</b> crude bpd mexico <b>dolly capacity petroleum</b>	MEXICO: Hurricane Dolly heads for Mexico coast
9	1017	police security <b>russian</b> people military peace killed told <b>grozny court</b>	police killed military security peace told <b>troops</b> <b>forces rebels</b> people	RUSSIA: Russia's Lebed meets rebel chief in Chechnya
10	1259	00 000 tonnes traders futures wheat prices <b>cents september</b> tonne	<b>delivery</b> traders futures tonne tonnes <b>desk</b> wheat prices 000 00	USA: Export Business - Grain/oilseeds complex

- 三种方法：选择质心向量中的突出词项，使用MI的差别式标签，使用离质心最近的文档的标题
- 三种方法的结果都不错

# 四种HAC算法簇相似度数目

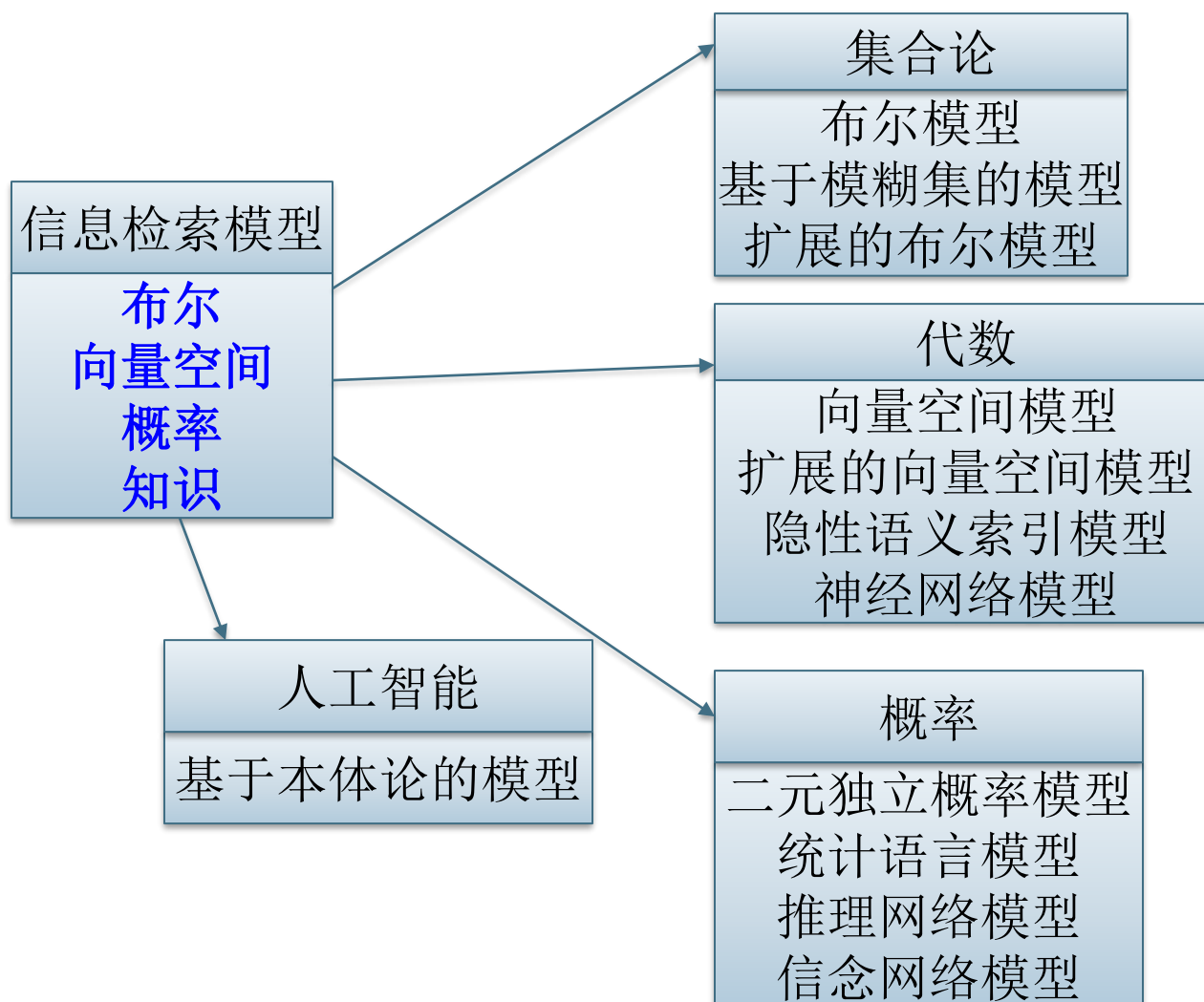
- 对于固定的N篇文档，单连接和全连接聚类中，簇之间相似度的计算均退化为两篇文档之间的相似度计算，因此，簇相似度最多有 $N(N-1)/2$ 个。那么对于GAAC和质心聚类算法，不同的簇相似度数目最多是多少？



# 提纲

- ① 上一讲回顾
- ② 隐性语义索引
- ③ 空间降维处理
- ④ LSI 在IR中的应用

# 信息检索模型分类



# 信息检索中的隐性语义例子

查询	文档
电脑	计算机
加密	密码、解密、信息安全、隐私保护
达康书记	人民的名义
国科大	中国科学院大学
中国	China
隐性语义索引	Latent Semantic Idexing、潜在语义索引、LSI、LSA

同义词

同领域词

同主题

缩略词

中英文

简写

。 。 。

- 思考如何返回与查询语义相关的文档？


# 信息检索中的隐性语义例子

- LSI的基本思想是文本中的词与词之间不是孤立的，存在着某种潜在的语义关系，通过对样本数据的统计分析，让机器自动挖掘出这些潜在的语义关系，并把这些关系表示成计算机可以“理解”的模型。
- 使得检索结果的实际效果更接近于人的自然语言，在一定程度上提高检索结果的相关性

# Susan Dumais

- 来自微软美国研究院
  - 隐性语义索引LSI的提出者
  - 2009年Salton奖得主
  - <https://www.microsoft.com/en-us/research/people/sdumais/>





Susan Dumais

Technical Fellow & Managing Director, Microsoft Research New England, New York City and Montreal

## About

I am a technical fellow and managing director of [Microsoft Research New England](#), [Microsoft Research New York City](#) and [Microsoft Research Montreal](#). My research is at the intersection of information retrieval and human-computer interaction. I am interested in algorithms and interfaces for improved information retrieval, as well as general issues in human-computer interaction.

I have been at Microsoft Research since July 1997. My current research focuses on gaze-enhanced interaction, the temporal dynamics of information systems, user modeling and personalization, novel interfaces for interactive retrieval, and search evaluation. Previous research studied a variety of information access and management challenges, including personal information management, desktop search, question answering, text categorization, collaborative filtering, interfaces for improving search and navigation, and user/task modeling. I have worked closely with several Microsoft groups (Bing, Windows Desktop Search, SharePoint Portal Server and Office Online Help) on search-related innovations.

Prior to coming to Microsoft, I co-developed a...

[Read more](#) ▾

## Recent news

### SIGCHI Lifetime Research Award

Susan Dumais, Technical Fellow and Director of the Microsoft Research Labs in New England, New York City and Montréal, and adjunct professor at the University of Washington, received the SIGCHI Lifetime Research Award. This is the highest honor awarded in the Human Computer Interaction community and is a tribute to a lifetime of deep and broad contributions to the intellectual core of the field.

# 回顾一下词项-文档矩阵

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
...						

该矩阵是计算文档和查询相似度的基础，接下来我们要介绍，能否通过对该矩阵进行转换来获得文档和查询之间的一个更好的相似度计算方法？

# 隐性语义索引LSI简介

- 我们将词项-文档矩阵转换成多个矩阵的乘积
- 这里我们使用的是一个特定的分解方法：奇异值分解 (singular value decomposition，简称SVD)
- SVD:  $C = U\Sigma V^T$  (其中  $C$  = 词项-文档矩阵)。SVD分解具有“唯一性” (不考虑正负符号的变向)
- 利用SVD分解的结果我们来构造一个新的、改进的词项-文档矩阵  $C'$
- 通过  $C'$  我们可以得到一个更好的相似度计算方法(相对于  $C$ 而言)
- 为了这种目的使用SVD被称为隐性语义索引/标引( latent semantic indexing)或者隐性语义分析( latent semantic analysis)或者简称 LSI、LSA。

# 特征分解

特征分解：将矩阵分解成一组特征向量和特征值。

$$Ax = \lambda x$$

A为n阶矩阵， $\lambda$ 为n的特征值，x为 $\lambda$ 对应的特征向量。

假设矩阵A有n个线性无关的特征向量 $\{v^{(1)}, \dots, v^{(n)}\}$ ，对应特征值 $\{\lambda_1, \dots, \lambda_n\}$ ，将特征向量连接成一个矩阵，使得每一列是一个特征向量： $V = [v^{(1)}, \dots, v^{(n)}]$ ，将特征值连接成向量 $\lambda = [\lambda_1, \dots, \lambda_n]$ ，则A的特征分解记作：

$$A = V \text{diag}(\lambda) V^{-1}$$

当A为实对称矩阵，就可以分解成实特征向量和实特征值：

$$A = Q \Lambda Q^T$$



# 特征分解

- 特征值分解可以得到特征值与特征向量，特征值表示的是这个特征到底有多重要，而特征向量表示这个特征是什么。
- 我们通过特征值分解得到的前N个特征向量，那么就对应了这个矩阵最主要的N个变化方向。我们利用这前N个变化方向，就可以近似这个矩阵（变换）。也就是提取了这个矩阵最重要的N个特征。

缺点：变换的矩阵必须是方阵。

如何表述普通矩阵的重要特征？——奇异值分解

# 奇异值分解

- 设矩阵  $A$  的维度为  $m \times n$ ，虽然  $A$  不是方阵，但是下面的矩阵却是方阵，且是对称方阵，维度分别为  $m \times m$ 、 $n \times n$ 。

$$AA^T \quad A^T A$$

- 分别对上面的方阵进行分解：

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

- 则  $A$  的奇异值分解为：

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

$$\begin{array}{c}
 \text{Blue box } A \\
 m \times n
 \end{array}
 =
 \begin{array}{c}
 \text{Green box } U \\
 m \times m
 \end{array}
 \times
 \begin{array}{c}
 \text{Blue box } \Sigma \\
 m \times n
 \end{array}
 \times
 \begin{array}{c}
 \text{Orange box } V^T \\
 n \times n
 \end{array}$$

# 更一般的形式

- 矩阵  $A$  的维度为  $m \times n$

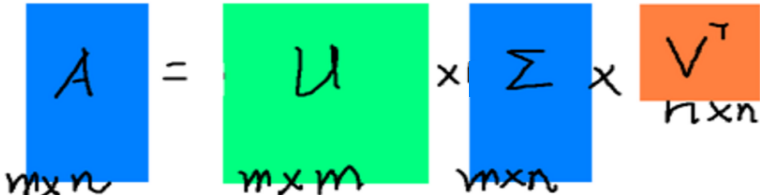
$$A = U \Sigma V^T$$


Diagram illustrating the SVD decomposition  $A = U \Sigma V^T$ . Matrix  $A$  is blue ( $m \times n$ ),  $U$  is green ( $m \times m$ ),  $\Sigma$  is blue ( $m \times n$ ), and  $V^T$  is orange ( $n \times n$ ). Dimensions are written below each matrix.

- 思考  $m > n$  和  $m < n$  时的奇异值分解情况

$U$  为  $M \times \min(M, N)$  的矩阵

$\Sigma$  为  $\min(M, N) \times \min(M, N)$  的对角方阵

$V$  为  $\min(M, N) \times N$  的矩阵

# 奇异值分解实现

- 时间复杂度较高
- 利用Matlab或Python的numpy  
如 $[u, s, v] = \text{svd}(a)$

```
1 import numpy as np
2
3 A = np.array([[1, 0, 1, 0, 1, 0], [1, 0, 1, 0, 0, 1], [0, 0, 1, 0, 1, 0], [1, 0, 1, 0, 1, 0], [0, 1, 0, 0, 0, 1], [0, 1, 0, 1, 0, 1], [0, 1, 0, 1, 0, 1]])
4
5 U, Sigma, VT = np.linalg.svd(A)
6
7 print(U)
8 print(Sigma)
9 print(VT)
```

- 重点关注奇异值分解的应用

# 例子 $C = U\Sigma V^T$ : 矩阵 $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

上面给出了一个标准的词项-文档矩阵( $M \times N$ ,  $M=5$ 是词项个数,  $N=6$ 是文档个数), 为简单起见, 这里使用了布尔矩阵。

# 例子 $C = U\Sigma V^T$ : 矩阵 $U$

$U$	1	2	3	4	5
ship	-0.44	-0.30	0.57	0.58	0.25
boat	-0.13	-0.33	-0.59	0.00	0.73
ocean	-0.48	-0.51	-0.37	0.00	-0.61
wood	-0.70	0.35	0.15	-0.58	0.16
tree	-0.26	0.65	-0.41	0.58	-0.09

$M \times \min(M,N)$  矩阵，每个词项对应一行，共  $\min(M,N)$  列。

这是个正交矩阵：

- (i) 列向量都是单位向量；
- (ii) 任意两个列向量之间都是互相正交的。可以想象这些列向量分别代表不同的“语义”维度，比如政治、体育、经济等主题。矩阵元素  $u_{ij}$  给出的是词项  $i$  和第  $j$  个“语义”维度之间的关系强弱程度。

## 例子 $C = U\Sigma V^T$ : 矩阵 $\Sigma$

$\Sigma$	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

这是个  $\min(M,N) \times \min(M,N)$  的对角方阵。对角线上是矩阵  $C$  的奇异值。奇异值的大小度量的是相应“语义”维度的重要性。我们可以通过忽略较小的值来忽略对应的“语义”维度

## 例子 $C = U\Sigma V^T$ : 矩阵 $V^T$

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

每篇文档对应一列，每  $\min(M,N)$  对应一行。同样，这也是一个正交矩阵：

- (i) 每个行向量都是单位向量；
- (ii) 任意两个行向量之间都是正交的；

同样每个行向量代表的是一个语义维度，矩阵元素  $v_{ij}$  代表的是文档  $i$  和语义维度  $j$  的关系强弱程度



# 例子 $C = U\Sigma V^T$ : 所有的四个矩阵

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$	
ship	1	0	1	0	0	0	
boat	0	1	0	0	0	0	
ocean	1	1	0	0	0	0	=
wood	1	0	0	1	1	0	
tree	0	0	0	1	0	1	
$U$	1	2	3	4	5		
ship	-0.44	-0.30	0.57	0.58	0.25		
boat	-0.13	-0.33	-0.59	0.00	0.73		
ocean	-0.48	-0.51	-0.37	0.00	-0.61		×
wood	-0.70	0.35	0.15	-0.58	0.16		
tree	-0.26	0.65	-0.41	0.58	-0.09		
$\Sigma$	1	2	3	4	5		
1	2.16	0.00	0.00	0.00	0.00		
2	0.00	1.59	0.00	0.00	0.00		
3	0.00	0.00	1.28	0.00	0.00		×
4	0.00	0.00	0.00	1.00	0.00		
5	0.00	0.00	0.00	0.00	0.39		
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$	
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12	
2	-0.29	-0.53	-0.19	0.63	0.22	0.41	
3	0.28	-0.75	0.45	-0.20	0.12	-0.33	
4	0.00	0.00	0.58	0.00	-0.58	0.58	
5	-0.53	0.29	0.63	0.19	0.41	-0.22	

# LSI: 小结

- 词项-文档矩阵可以分解成3个矩阵的乘积
- 词项矩阵  $U$  – 每个词项对应其中的一个行向量
- 文档矩阵  $V^T$  – 每篇文档对应其中的一个列向量
- 奇异值矩阵  $\Sigma$  – 对角方阵，对角线上的奇异值代表的是每个“语义”维度的重要性
- 接下来我们要介绍这样做的原因。

# 提纲

- ① 上一讲回顾
- ② 隐性语义索引
- ③ 空间降维处理
- ④ LSI 在IR中的应用

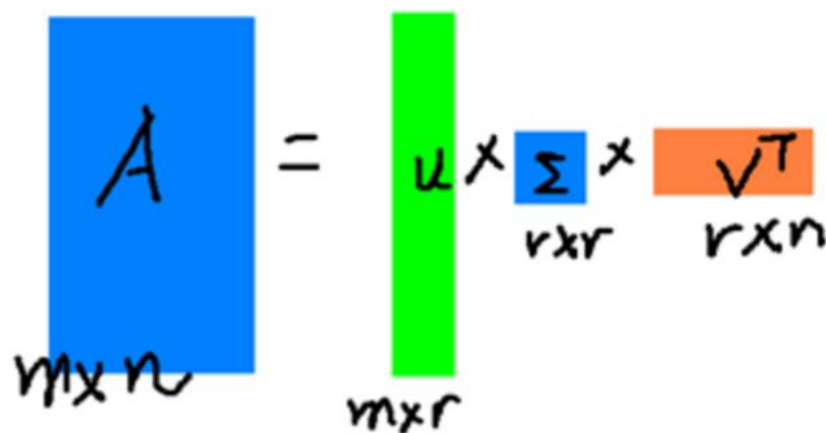
# 为什么在LSI中使用SVD分解

- 最关键的性质：每个奇异值对应的是每个“语义”维度的权重
- 将不太重要的维度的权重置为0，可以保留重要的信息，去掉一些信息“枝节”
- 这些“枝节”可能是：
  - 噪音 – 这种情况下，简化后的LSI 噪音更少，是一种更好的表示方法
  - 枝节信息可能会使本来应该相似的对象不相似，同样简化的LSI 由于其能更好地表达相似度，因而是一种更优的表示方式
- “细节越少越好”的一个类比
  - 鲜红色花朵的图像
  - 红黑花朵的图像
  - 如果忽略颜色，将更容易看到两者的相似性

# 低秩逼近

在很多情况下，前10%甚至1%的奇异值的和就占了全部的奇异值之和的99%以上了。也就是说，我们也可以用前r大的奇异值来近似描述矩阵，则**部分奇异值分解**为：

$$A_{m \times n} \approx U_{m \times r} \Sigma_{r \times r} V_{r \times n}^T \quad r\text{-秩逼近}$$



在存储观点中，矩阵面积越小，存储量就越小。  
存下这里的三个矩阵：U、 $\Sigma$ 、V，其面积和要远远小于原始的矩阵A。

# 将空间维度降为 2

$U$	1	2	3	4	5	
ship	-0.44	-0.30	0.00	0.00	0.00	
boat	-0.13	-0.33	0.00	0.00	0.00	
ocean	-0.48	-0.51	0.00	0.00	0.00	
wood	-0.70	0.35	0.00	0.00	0.00	
tree	-0.26	0.65	0.00	0.00	0.00	
$\Sigma_2$	1	2	3	4	5	
1	2.16	0.00	0.00	0.00	0.00	
2	0.00	1.59	0.00	0.00	0.00	
3	0.00	0.00	0.00	0.00	0.00	
4	0.00	0.00	0.00	0.00	0.00	
5	0.00	0.00	0.00	0.00	0.00	
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

实际上，我们只需将矩阵 $\Sigma$ 中相应的维度置为0即可。此时，相当于矩阵 $U$ 和 $V^T$ 的相应维度被忽略，然后计算  $C_2 = U\Sigma_2V^T$  .

# 维度降为 2

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49
$U$	1	2	3	4	5	
ship	-0.44	-0.30	0.57	0.58	0.25	
boat	-0.13	-0.33	-0.59	0.00	0.73	
ocean	-0.48	-0.51	-0.37	0.00	-0.61	$\times$
wood	-0.70	0.35	0.15	-0.58	0.16	
tree	-0.26	0.65	-0.41	0.58	-0.09	
$\Sigma_2$	1	2	3	4	5	
1	2.16	0.00	0.00	0.00	0.00	
2	0.00	1.59	0.00	0.00	0.00	
3	0.00	0.00	0.00	0.00	0.00	$\times$
4	0.00	0.00	0.00	0.00	0.00	
5	0.00	0.00	0.00	0.00	0.00	
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

# 回顾原始未分解的矩阵 $C=U\Sigma V^T$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$	
ship	1	0	1	0	0	0	
boat	0	1	0	0	0	0	
ocean	1	1	0	0	0	0	=
wood	1	0	0	1	1	0	
tree	0	0	0	1	0	1	
$U$	1	2	3	4	5		
ship	-0.44	-0.30	0.57	0.58	0.25		
boat	-0.13	-0.33	-0.59	0.00	0.73		
ocean	-0.48	-0.51	-0.37	0.00	-0.61		×
wood	-0.70	0.35	0.15	-0.58	0.16		
tree	-0.26	0.65	-0.41	0.58	-0.09		
$\Sigma$	1	2	3	4	5		
1	2.16	0.00	0.00	0.00	0.00		
2	0.00	1.59	0.00	0.00	0.00		
3	0.00	0.00	1.28	0.00	0.00		×
4	0.00	0.00	0.00	1.00	0.00		
5	0.00	0.00	0.00	0.00	0.39		
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$	
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12	
2	-0.29	-0.53	-0.19	0.63	0.22	0.41	
3	0.28	-0.75	0.45	-0.20	0.12	-0.33	
4	0.00	0.00	0.58	0.00	-0.58	0.58	
5	-0.53	0.29	0.63	0.19	0.41	-0.22	



# 原始矩阵 $C$ vs. 简化的矩阵 $C_2 = U\Sigma_2V^T$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1
$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

$C_2$  可以看成矩阵  $C$  的一个二维表示。

我们将表示的维度缩减至2维。

# 为什么新的低维空间更好？

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1
$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

在原始空间中， $d_2$ 和 $d_3$ 的相似度为0；  
 但是在新空间下， $d_2$ 和 $d_3$ 的相似度为：  
 $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

# 为什么新的低维空间更好？

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1
$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

“boat” 和“ship” 语义上相似。低维空间能够反映出这一点。

SVD的什么性质会导致相似度计算有所提高？

# 矩阵的F范数

- 矩阵 $C$ 的F范数( Frobenius norm)定义为:

$$\|C\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$$

- 两个矩阵 $A$ 、 $B$ 的逼近程度可以用矩阵 $A-B$ 的F范数来度量

# 课堂练习

- 习题18-8 利用公式 (18-13) 所示的SVD分解, 计算矩阵  $C$  (某词项-文档矩阵) 的1-秩逼近  $C_1$ , 并给出该逼近下的F范数误差值

$$C = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (18-12)$$

公式 (18-12) 所示矩阵的 SVD 结果是：

$$U = \begin{pmatrix} -0.816 & 0.000 \\ -0.408 & -0.707 \\ -0.408 & 0.707 \end{pmatrix}, \Sigma = \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \end{pmatrix}, V^T = \begin{pmatrix} -0.707 & -0.707 \\ 0.707 & -0.707 \end{pmatrix}. \quad (18-13)$$

- 其中, 两个矩阵的差 ( $X=C-C_k$ ) 的F范数为:

$$\|X\|_F = \sqrt{\sum_{i=1}^M \sum_{j=1}^N X_{ij}^2}$$

# 课堂练习-解答

$$C_1 = \begin{pmatrix} -0.816 \\ -0.408 \\ -0.408 \end{pmatrix} \times 1.732 \times \begin{pmatrix} -0.707 & -0.707 \end{pmatrix}$$

$$= \begin{pmatrix} -1.413 \\ -0.707 \\ -0.707 \end{pmatrix} \times \begin{pmatrix} -0.707 & -0.707 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

$$F = \|C - C_1\| = 2 * 0.5 = 1$$

# 提纲

- ① 上一讲回顾
- ② 隐性语义索引
- ③ 空间降维处理
- ④ LSI 在IR中的应用

# LSI在IR中使用的原因

- LSI 能够发现文档的语义上的关联 ...
- ... 但是在原始向量空间中这些文档相似度不大 (因为它们使用不同的词语) ...
- ... 于是通过LSI可以将它们映射到新的低维向量空间中 .  
..
- ... 在新的空间下，两者相似度较高
- 因此， LSI能够解决一义多词([synonymy](#)) 和语义关联问题
- 在标准向量空间下，同义词对文档相似度计算没有任何贡献
- LSI所期望的效果：同义词对文档相似度贡献很大



# LSI是如何解决一义多词和语义关联问题的

- 降维迫使我们忽略大量“细节”
- 我们将原始空间下不同的词映射到低维空间的同一维中
- 将同义词映射到同一维的“开销”远小于无关词的聚集
- SVD选择开销最小的映射方法
- 因此，SVD会将同义词映射到同一维
- 但是，它同时能避免将无关词映射到同一维

# LSI与其它方法的比较

- 如果查询和文档没有公共词项时，前面我们介绍的相关反馈和查询扩展可以用于提高IR的召回率
- LSI会提高召回率但是损害正确率
- 因此，它和相关反馈查询扩展解决的是同一问题...
- ... 同样它们的缺点也一致

# LSI实现

- 对词项-文档矩阵进行SVD分解
- 计算在新的低维空间下的文档表示
- 将查询映射到低维空间中  $\vec{q}_2^T = \Sigma_2^{-1} U_2^T \vec{q}^T$ .
- 上述公式来自： $C_2 = U_2 \Sigma_2 V_2^T \Rightarrow V_2^T = \Sigma_2^{-1} U_2^T C_2$
- 计算  $q_2$  和  $V_2$  中的所有文档表示的相似度
- 像以往一样按照相似度高低输出文档结果
- LSI可以看成是向量空间模型的降维方法

# LSI应用举例

- 给定三篇文档

d1: *Shipment of gold damaged in a fire.*

d2: *Delivery of silver arrived in a silver truck.*

d3: *Shipment of gold arrived in a truck.*

- 查询: gold silver truck

# 构造词项-文档矩阵和查询矩阵

Terms ↓	d1 ↓	d2 ↓	d3 ↓	q ↓
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
shipment	1	0	1	0
silver	0	2	0	1
truck	0	1	1	1

$A =$

$q =$

# 矩阵分解

$$A = USV^T$$

$$U = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

## 2-秩逼近

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix}$$

$k = 2$

$$S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

降维空间中的文档向量

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix}$$

$$V^T \approx V_k^T = \begin{matrix} d_1 & d_2 & d_3 \\ \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix} \end{matrix}$$

# 降维空间中的查询向量

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix}$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$



# 查询与文档的相似度

$$\text{sim}(q, d) = \frac{q \bullet d}{|q| |d|}$$

$$\text{sim}(q, d_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

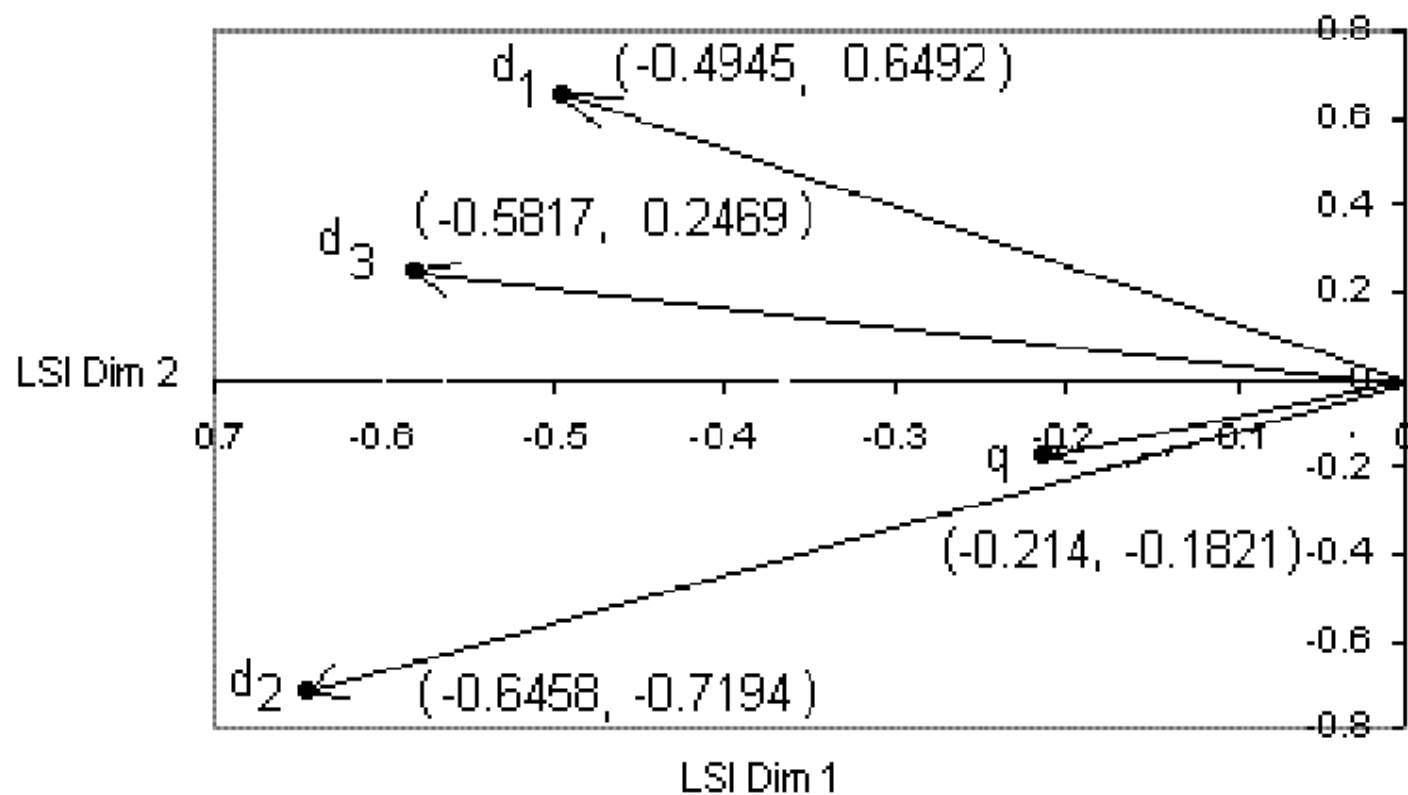
$$\text{sim}(q, d_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$\text{sim}(q, d_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2} \sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$d_2 > d_3 > d_1$$

# 查询与文档的相似度



# 最优性

- LSI的最优性：
  - 对矩阵 $C$ 进行SVD分解，然后保留  $k$  个最大的奇异值并将其他奇异值置为0，这种做法得到是原始矩阵 $C$ 的最佳逼近(参考[Eckart-Young 定理](#))
  - 即不存在其它同秩的矩阵能够更加逼近 $C$

# LSI优缺点

- 优点：
  - 算法原理很简单
  - 对于隐私语义的检索，实验效果不错；
- 缺点：
  - 计算开销大； NMF
  - K-秩逼近中K的选择； HDP
  - 不是概率模型，缺乏统计基础，结果难以直观解释 pLSA  
LDA

# 推荐系统中的SVD分解

- 项/用户矩阵，稀疏矩阵
- 通过优化目标函数，将0值填上

item\user	Ben	Tom	John	Fred
item 1	5	5	0	5
item 2	5	0	3	4
item 3	3	4	0	3
item 4	0	0	5	3
item 5	5	4	4	5
item 6	5	4	5	5

# 课堂练习

? 习题 18-11 假定有一个文档集合，其中每篇文档可以是英文或者是西班牙文。整个文档集如图 18-4 所示。图 18-5 给出了与图 18-4 相关的英语和西班牙语的术语表。当然，该术语表只用于帮助理解，对检索系统来说是不可见的。

DocID	document text
1	hello
2	open house
3	mi casa
4	hola Profesor
5	hola y bienvenido
6	hello and welcome

图 18-4 习题 18-11 中的文档

Spanish	English
mi	my
casa	house
hola	hello
profesor	professor
y	and
bienvenido	welcome

图 18-5 习题 18-11 中的术语表

- (1) 构造该文档集的词项-文档矩阵  $C$ 。为简单起见，只使用原始词频而不是归一化的 tf-idf 权重。请务必清晰标出矩阵的每一维。
- (2) 写出矩阵  $U_2$ 、 $\Sigma'_2$  及  $V_2$ ，并推出 2-秩逼近矩阵  $C_2$ 。
- (3) 简述矩阵  $C^T C$  的元素  $(i, j)$  代表的意义。
- (4) 简述矩阵  $C_2^T C_2$  的元素  $(i, j)$  所代表的意义，解释它为什么与  $C^T C$  中的元素  $(i, j)$  不同。

# 课堂练习-解答

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- (3)  $C^TC$ 中的第 $i$ 行、 $j$ 列元素表示的是文档  $d_i$ 、 $d_j$ 中包含的公共词项的个数。
- (4)  $C_2^TC_2$ 中的 $i$ 行、 $j$ 列元素表示的是文档  $d_i$ 、 $d_j$ 在隐性空间下的相似度，两者是不同的。

# 参考资料

- 《信息检索导论》第 18 章
  - Furnas、Deerwester、Dumais等人写的第一篇LSI的文章
  - 正则化LSI (RLSI): 王泉老师博士论文
  - 非负矩阵分解(NMF)
  - Thomas Hofmann提出的概率LSI (PLSI)
- 
- George W. Furnas, Scott Deerwester, Susan T. Dumais, Thomas K. Landauer, Richard A. Harshman, Lynn A. Streeter, and Karen E. Lochbaum. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In Proceedings of SIGIR88. ACM, New York, NY, USA, 465-480. DOI: <https://doi.org/10.1145/62437.62487>
  - Latent Semantic Indexing (LSI) A Fast Track Tutorial. <https://apluswebservices.com/wp-content/uploads/2012/05/latent-semantic-indexing-fast-track-tutorial.pdf>
  - <https://blog.csdn.net/xpy870663266/article/details/101849044>