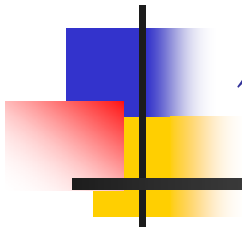
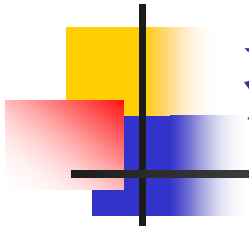


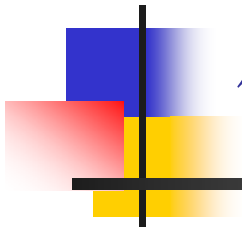
信息检索模型

刘挺
哈工大信息检索研究室
2004年秋

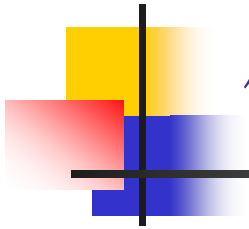


提纲

- 信息检索模型的概述
- 布尔模型
- 向量空间模型(VSM)
- 扩展的布尔模型
- 潜在语义索引模型(LSI)
- 概率模型
- 基于统计语言模型的信息检索模型
- 基于本体论的信息检索模型

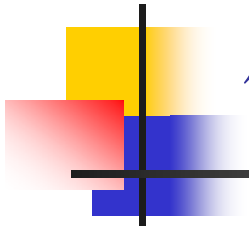


信息检索模型的概述



什么是模型？

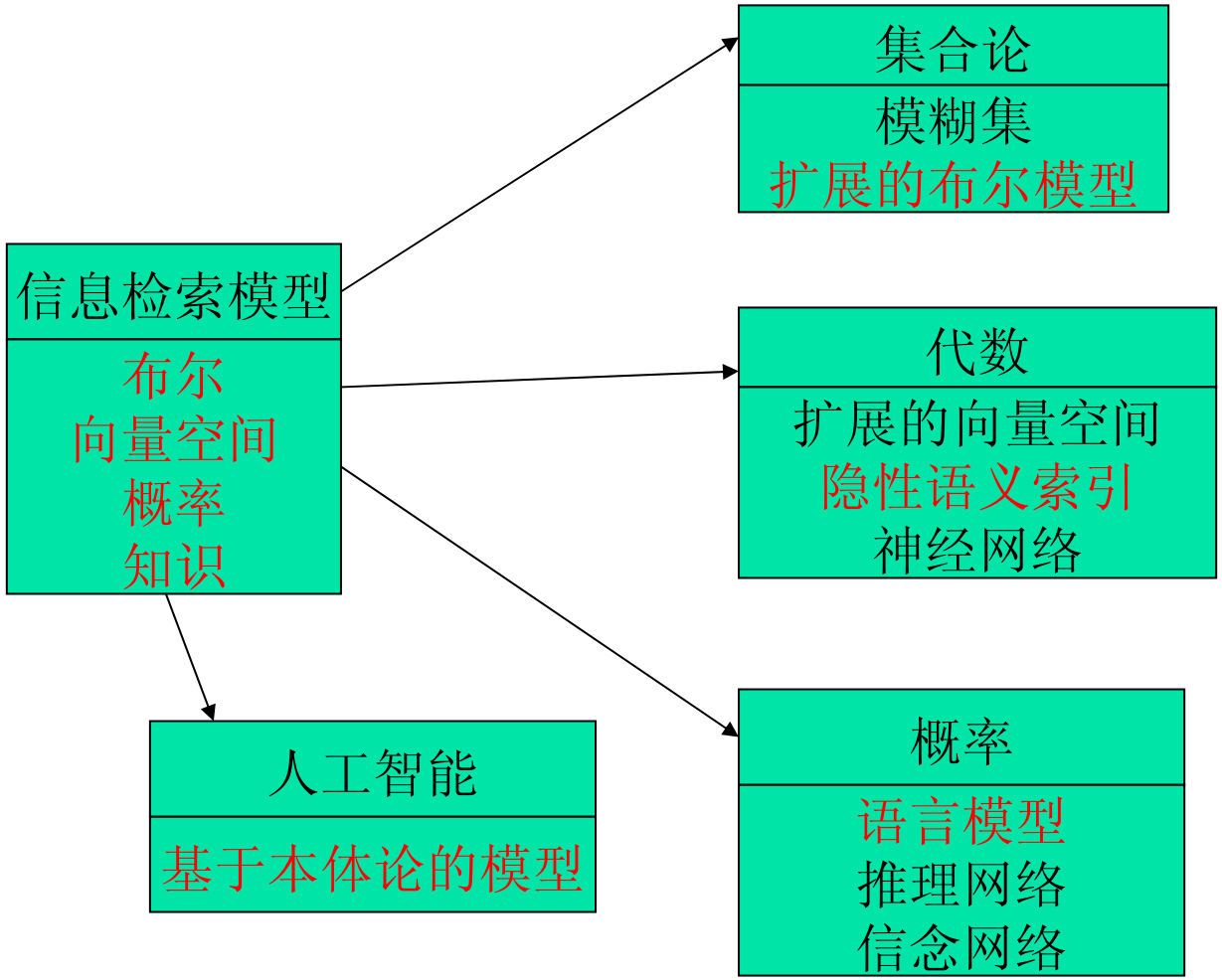
- 模型是采用数学工具，对现实世界某种事物或某种运动的抽象描述
- 面对相同的输入，模型的输出应能够无限地逼近现实世界的输出
  - 举例：天气的预测模型
- 模型和实现的区别
  - 一个模型可以用多种方法实现
  - 例如：布尔模型可以用倒排文档(inverted file)实现，也可以用B-tree实现

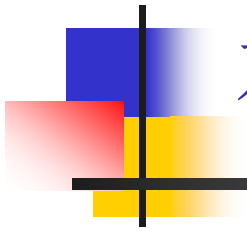


信息检索模型

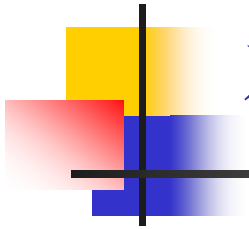
- 四元组[D, Q, F, $R(q_i, d_j)$]
  - D: 文档集的机内表示
  - Q: 用户需求的机内表示
  - F: 文档表示、查询表示和它们之间的关系的模型框架(Frame)
  - $R(q_i, d_j)$: 给query q_i 和document d_j评分
- 信息检索模型决定于：
  - 从什么样的视角去看待查询式和文档
  - 基于什么样的理论去看待查询式和文档的关系
  - 如何计算查询式和文档之间的相似度

模型分类


集合论
模糊集
扩展的布尔模型
信息检索模型
布尔
向量空间
概率
知识
代数
扩展的向量空间
隐性语义索引
神经网络
人工智能
基于本体论的模型
概率
语言模型
推理网络
信念网络

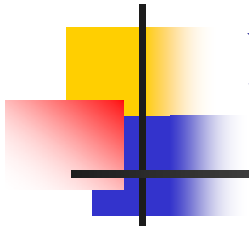


布尔模型(Boolean Model)



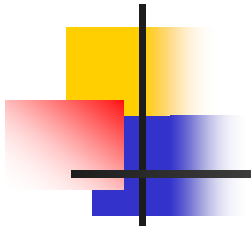
布尔模型

- 文档表示
  - 一个文档被表示为关键词的集合
- 查询式表示
  - 查询式(Queries)被表示为关键词的布尔组合，用“与或非”连接起来，并用括弧指示优先次序
- 匹配
  - 一个文档当且仅当它能够满足布尔查询式时，才将其检索出来
- 不同的系统可以使用:
  - 不同的去除停用词(*stopword removal)*策略和*stemming*策略
  - 索引中不同类型的辅助信息
  - 不同的实现方法



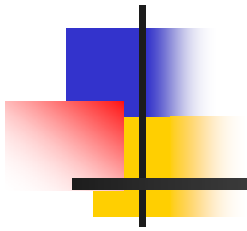
强调

- 到目前为止，布尔模型是最常用的检索模型，因为：
  - 由于查询简单，因此容易理解
  - 通过使用复杂的布尔表达式，可以很方便地控制查询结果
- 相当有效的实现方法
  - 相当于识别包含了一个某个特定term的文档
- 经过某种训练的用户可以容易地写出布尔查询式
- 布尔模型可以通过扩展来包含排序的功能，即“扩展的布尔模型”

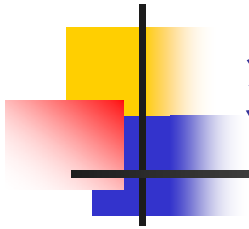


问题

- 布尔模型被认为是功能最弱的方式，其主要问题在于不支持部分匹配，而完全匹配会导致太多或者太少的结果文档被返回
- 非常刚性: “与”意味着全部; “或”意味着任何一个
  - 如果“我想要n个词中m个词同时出现的文档”，怎么表示？
  - 不可能企望用户自己规定m值
  - 系统可以从m=n开始，然后逐渐减少m，但很麻烦
- 很难表示用户复杂的需求
- 很难控制被检索的文档数量
  - 原则上讲，所有被匹配的文档都将被返回
- 很难对输出进行排序
  - 不考虑索引词的权重，所有文档都以相同的方式和查询相匹配
- 很难进行自动的相关反馈
  - 如果一篇文档被用户确认为相关或者不相关，怎样相应地修改查询式呢？

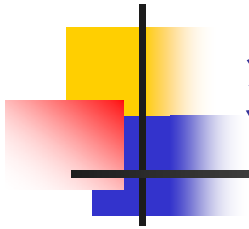


向量空间模型



统计模型

- 基于关键词(一个文本由一个关键词列表组成)
- 根据关键词的出现频率计算相似度
  - 例如：文档的统计特性
- 用户规定一个词项(term)集合，可以给每个词项附加权重
  - 未加权的词项: Q = 〈 database; text; information 〉
  - 加权的词项: Q = 〈 database 0.5; text 0.8; information 0.2 〉
  - 查询式中没有布尔条件
- 根据相似度对输出结果进行排序
- 支持自动的相关反馈
  - 有用的词项被添加到原始的查询式中
  - 例如：Q ⇒〈 database; text; information; *document* 〉

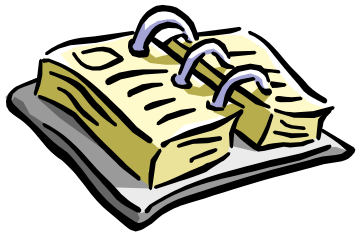


统计模型中的问题

- 怎样确定文档中哪些词是重要的词？
- 怎样确定一个词在某个文档中或在整个文档集中的重要程度？
- 怎样确定一个文档和一个查询式之间的相似度？
- 在WWW中，什么是文档集(collection)，链接、文档结构以及其它形式特征（如字体、颜色等）对统计模型有何影响？

向量空间模型

- 若干独立的词项被选作索引项(*index terms*) or 词表*vocabulary*
- 索引项代表了一个应用中的重要词项
  - 计算机科学图书馆中的索引项应该是哪些呢?



计算机科学
文档集

体系结构
总线
计算机
数据库
….
XML

文档集中的索引项

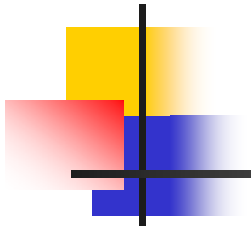
向量空间模型

- 这些索引项是不相关的*un-correlated (或者说是正交的orthogonal)*，形成一个向量空间*vector space*

计算机科学文档集

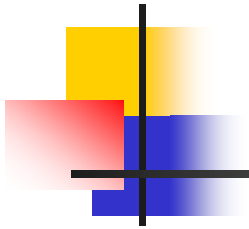
“计算机” “科学” “商务”

该文档集中的全部重要词项



向量空间模型

- 实际上，这些词项是相互关联的
  - 当你在一个文档中看到 “计算机”, 非常有可能同时看到“科学”
  - 当你在一个文档中看到 “计算机”, 有中等的可能性同时看到 “商务”
  - 当你在一个文档中看到“商务”，只有很少的机会同时看到“科学”



向量空间模型

- 2个索引项构成一个二维空间，一个文档可能包含0, 1 或2个索引项
  - d_i = 〈 0, 0 〉　(一个索引项也不包含)
  - d_j = 〈 0, 0.7 〉 (包含其中一个索引项)
  - d_k = 〈 1, 2 〉　(包含两个索引项)
- 类似的，3个索引项构成一个三维空间，n个索引项构成n维空间
- 一个文档或查询式可以表示为n个元素的线性组合

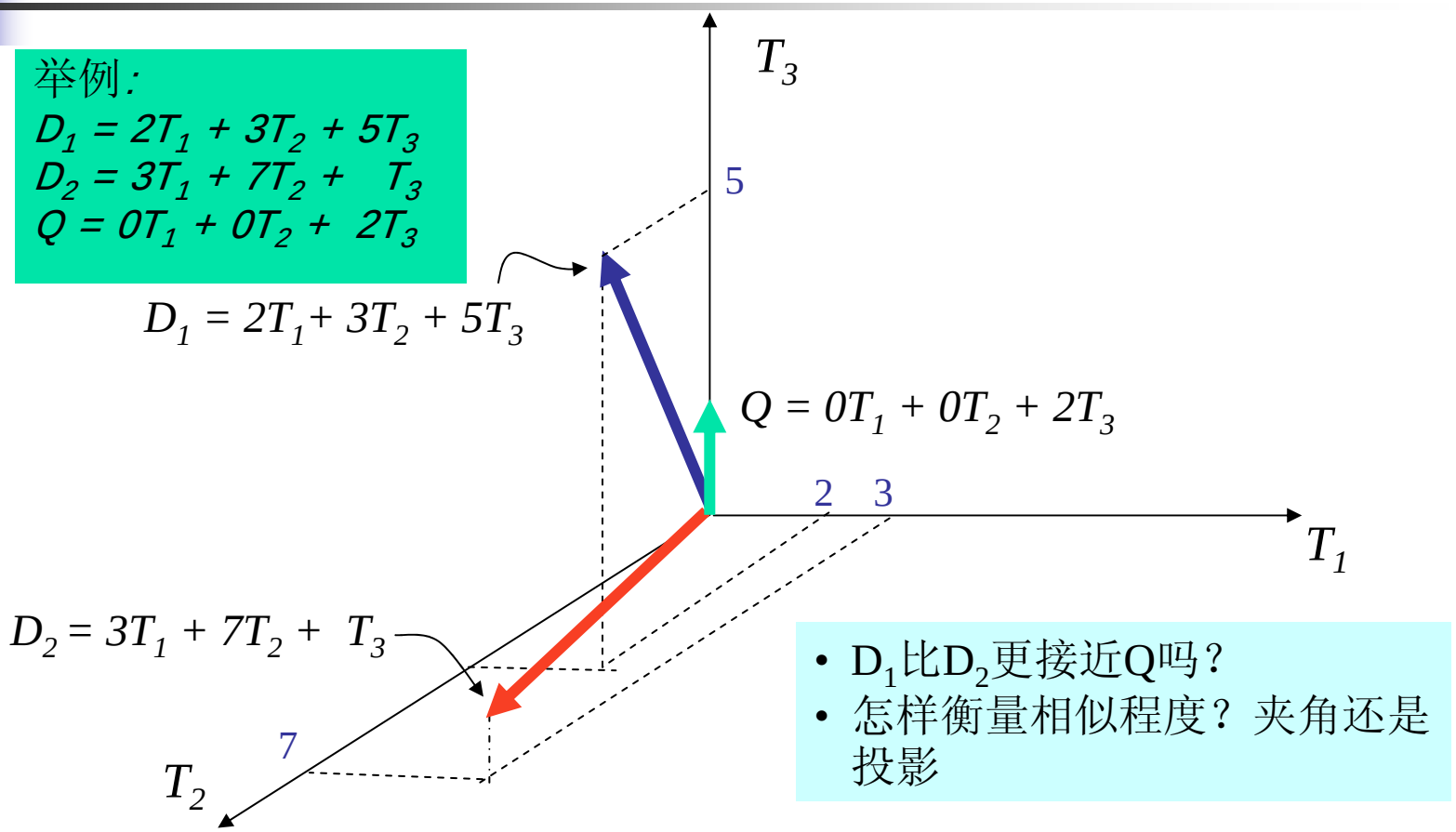
图示

举例:

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

$D_2 = 3T_1 + 7T_2 + T_3$

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

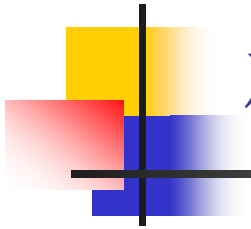


- D_1比D_2更接近Q吗？
- 怎样衡量相似程度？夹角还是投影

文档集 – 一般表示

- 向量空间中的N个文档可以用一个矩阵表示
- 矩阵中的一个元素对应于文档中一个词项的权重。“0”意味着该词项在文档中没有意义，或该词项不在文档中出现。

$$\begin{array}{c|cccc} & T_1 & T_2 & \ldots & T_t \\ D_1 & d_{11} & d_{12} & \ldots & d_{1t} \\ D_2 & d_{21} & d_{22} & \ldots & d_{2t} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & d_{n1} & d_{n2} & \ldots & d_{nt} \end{array}$$



相似度计算

- 相似度是一个函数，它给出两个向量之间的相似程度
  - 查询式和文档都是向量，各类相似度存在于：
    - 两个文档之间
    - 两个查询式之间
    - 一个查询式和一个文档之间
  - 人们曾提出大量的相似度计算方法，因为最佳的相似度计算方法并不存在。
- 通过计算查询式和文档之间的相似度，可以：
  - 可以根据预定的重要程度对检索出来的文档进行排序
  - 通过强制设定某个阈值，控制被检索出来的文档的数量
  - 检索结果可以被用于相关反馈中，以便对原始的查询式进行修正。(例如：将文档向量和查询式向量进行结合)

相似度度量 – 内积(Inner Product)

- 文档D和查询式Q可以通过内积进行计算:

$$\text{sim}(D, Q) = \sum_{k=1}^{t} (d_{ik} \bullet q_k)$$

  - d_{ik}是文档d_i中的词项k的权重，q_k是查询式Q中词项k的权重
- 对于二值向量, 内积是查询式中的词项和文档中的词项相互匹配的数量
- 对于加权向量, 内积是查询式和文档中相互匹配的词项的权重乘积之和

内积 – 举例

retrieval database architecture computer text management information

- 二值（Binary）：
  - D = 1, 1, 1, 0, 1, 1, 0
  - Q = 1, 0 , 1, 0, 0, 1, 1
  - sim(D, Q) = 3

- 向量的大小 = 词表的大小 = 7
- 0 意味着某个词项没有在文档中出现，或者没有在查询式中出现

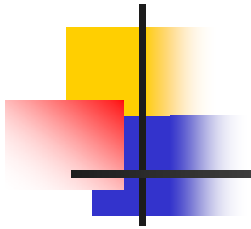
加权

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

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

$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$

$\text{sim}(D_2, Q) = 3*0 + 7*0 + 1*2 = 2$



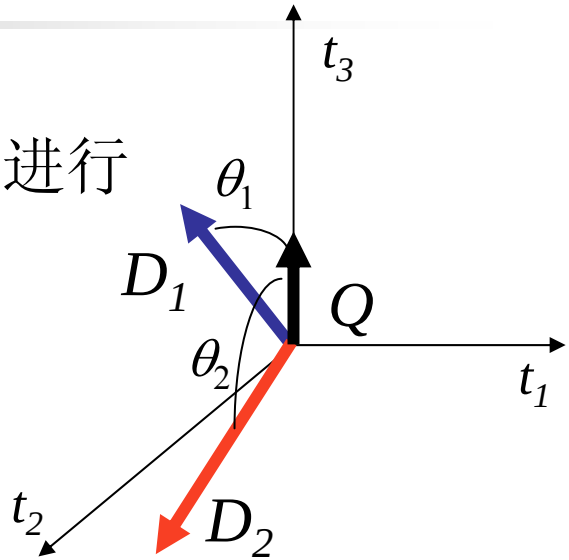
内积的属性

- 内积值没有界限
  - 不象概率值，要在(0,1)之间
- 对长文档有利
  - 内积用于衡量有多少词项匹配成功，而不计算有多少词项匹配失败
  - 长文档包含大量独立词项，每个词项均多次出现，因此一般而言，和查询式中的词项匹配成功的可能性就会比短文档大。

余弦(Cosine)相似度度量

- 余弦相似度计算两个向量的夹角
- 余弦相似度是利用向量长度对内积进行归一化的结果

$$\mathrm{CosSim}(D_i, Q) = \frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^{2} \bullet \sum_{k=1}^{t} q_k^{2}}}$$



$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad \mathrm{CosSim}(D_1, Q) = 5 / \sqrt{38} = 0.81$

$D_2 = 3T_1 + 7T_2 + T_3$ $\quad \mathrm{CosSim}(D_2, Q) = 1 / \sqrt{59} = 0.13$

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

用余弦计算，D_1 比 D_2 高6倍；
用内积计算，D_1 比 D_2 高5倍

其它相似度度量方法

- 存在大量的其它相似度度量方法

Jaccard Coefficient:

$$\frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sum_{k=1}^{t} d_{ik}^{2} + \sum_{k=1}^{t} q_k^{2} - \sum_{k=1}^{t}(d_{ik} \bullet q_k)}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ Sim(D_1 , Q) = 10 / (38+4-10) = 10/32 = 0.312
$D_2 = 3T_1 + 7T_2 + T_3$ Sim(D_2 , Q) = 2 / (59+4-2) = 2/61 = 0.033
$Q = 0T_1 + 0T_2 + 2T_3$

- D_1 比 D_2 高9.5倍

二值化的相似度度量

Inner Product: $\sum_{k=1}^{t}(d_{ik} \bullet q_k)$ $\quad |d_i \cap q_k|$

Cosine: $\dfrac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^2 \bullet \sum_{k=1}^{t} q_k^2}}$ $\quad \dfrac{|d_i \cap q_k|}{\sqrt{|d_i|} \times \sqrt{|q_k|}}$

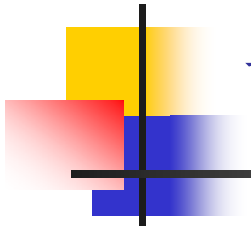
Jaccard : $\dfrac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sum_{k=1}^{t} d_{ik}^2 + \sum_{k=1}^{t} q_k^2 - \sum_{k=1}^{t}(d_{ik} \bullet q_k)}$ $\quad \dfrac{|d_i \cap q_k|}{|d_i| + |q_k| - |d_i \cap q_k|}$

d_i 和 q_k here are vector

d_i and q_k here are sets of keywords

文档和词项的权重

- 根据词项在文档(*tf*)和文档集(*idf*)中的频率(frequency)计算词项的权重
  - tf_{ij} = 词项j在文档i中的频率
  - df_j = 词项j的文档频率= 包含词项j的文档数量
  - idf_j = 词项j的反文档频率= $\log_2 (N/ df_j)$
    - *N*: 文档集中文档总数
    - 反文档频率用词项区别文档



词项权重

- 一种典型的词项重要性指示器

$$w_{ij} = tf_{ij} \bullet idf_j = tf_{ij} \bullet \log_2 (N/df_j)$$

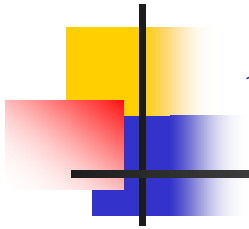
  - 一个在当前文档中频繁出现，但是在剩余的其它文档中很少出现的词项获得较高的权重
  - $tf_{ij} \bullet idf_j$ 直接正比于一个词项在文档中出现的次数

- 还有使用归一化词项频率的例子:
  - $w_{ij} = (tf_{ij}/max\{tf_{ij}\}) \cdot idf_j = (tf_{ij}/max_l\{tf_{lj}\}) \cdot \log_2 (N/df_j)$
  - $max\{tf_{ij}\}$ 是文档 j 中最高频率的词项的频率
  - 还有很多其它衡量词项权重的方法

TFIDF举例

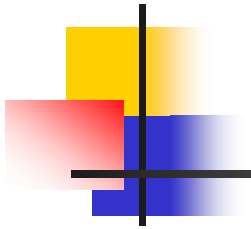
- 文本：“俄罗斯频繁发生恐怖事件，俄罗斯的安全部门加大打击恐怖主义的力度。”

	TF	IDF	TFIDF		TF	IDF	TFIDF
俄罗斯	2	较高	高	安全	1	中等	高
恐怖	2	较高	高	部门	1	较低	低
的	2	非常低	很低	加大	1	较低	低
频繁	1	较低	低	打击	1	中等	高
发生	1	较低	低	主义	1	较低	低
事件	1	较低	低	力度	1	中等	高



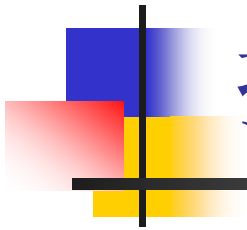
查询式词项权重

- 如果词项出现在查询式中，则该词项在查询式中的权重为1，否则为0
- 也可以用用户指定查询式中词项的权重
- 一个自然语言查询式可以被看成一个文档
  - 查询式：“有没有周杰伦的歌？”会被转换为:
    <周杰伦, 歌>
  - 查询式：“请帮我找关于俄罗斯和车臣之间的战争以及车臣恐怖主义首脑的资料”会被转换为:
    <俄罗斯 2, 车臣 2, 战争1, 恐怖主义1, 首脑 1>
  - 过滤掉了：“请帮我找”，“和”，“之间的”，“以及”，“的资料”
- 两个文档之间的相似度可以同理计算



向量空间

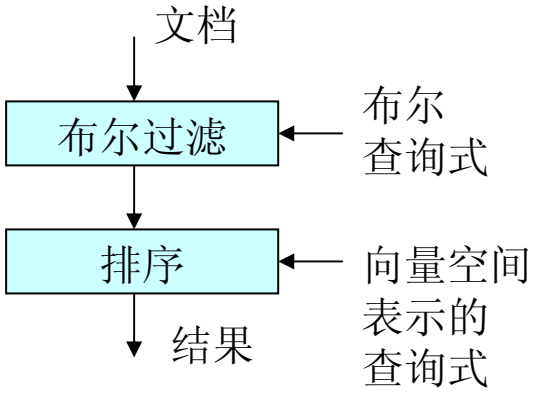
- 向量模型的优点在于：
  - 术语权重的算法提高了检索的性能
  - 部分匹配的策略使得检索的结果文档集更接近用户的检索需求
  - 可以根据结果文档对于查询串的相关度通过Cosine Ranking等公式对结果文档进行排序



扩展的布尔模型

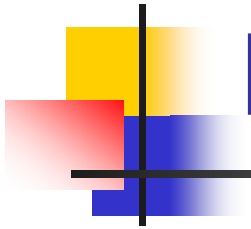
布尔模型

- 统计模型更像“或”操作
- 布尔模型可以和向量空间模型相结合，先做布尔过滤，然后进行排序：
  - 首先进行布尔查询
  - 将全部满足布尔查询的文档汇集成一个文档
  - 用向量空间法对布尔检索结果进行排序



如果忽略布尔关系的话，向量空间查询式和布尔查询式是相同的

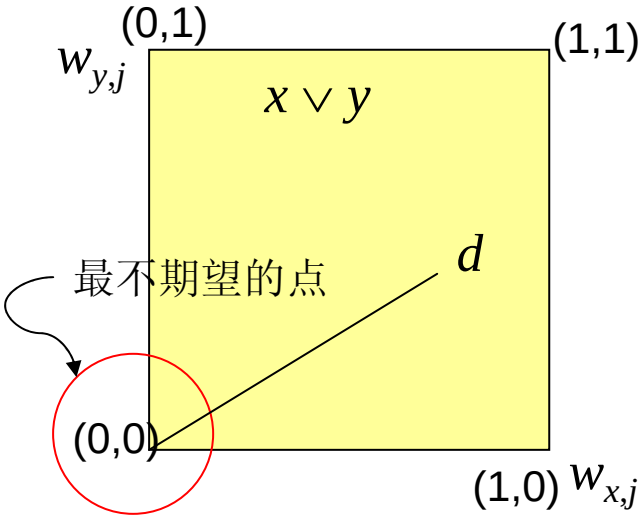
扩展的布尔模型
Extended Boolean Model



- 先“布尔”，后“排序”存在的问题：
  - 如果 “与” 应用于布尔查询式, 结果集可能太窄,因而影响了后面的排序过程
  - 如果 “或” 应用于布尔查询式, 就和纯向量空间模型没有区别了
  - 在第一步，如果最佳地应用布尔模型呢？

扩展布尔模型中的“或”关系

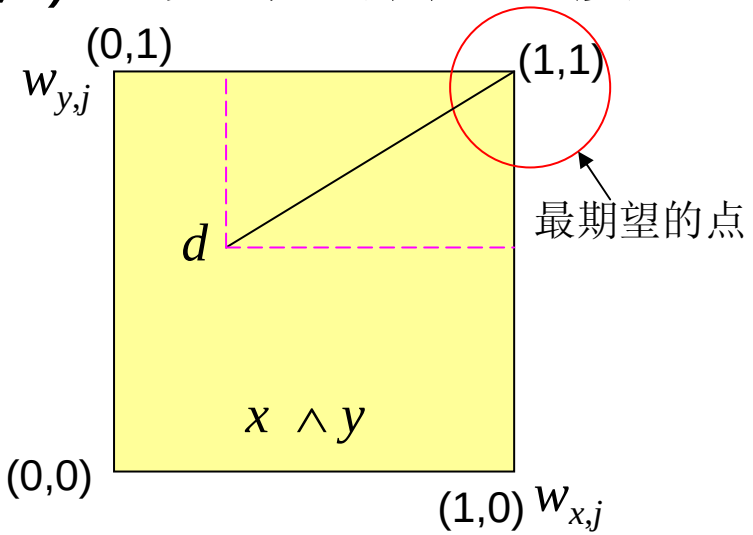
- 给定一个或关系的查询式：$x \vee y$
- 假设文档d_j中x和y的权重被归一化在(0,1)区间内:
  - $w_{x,j} = tf_{x,j} / max_l\ tf_{l,j} \times idf_x / max_i\ idf_i$
  - $\text{sim}(q_{or}, d_j) = [\ (x^2 + y^2)/2\ ]^{0.5}$ where $x = w_{x,j}$ and $y = w_{y,j}$



- 一个文档在(1,1)处获得最高的权重，此时意味着文档包含了全部两个查询词，并且查询词在文档中的权重也是最高的
- 函数sim()度量了从原点出发的向量长度

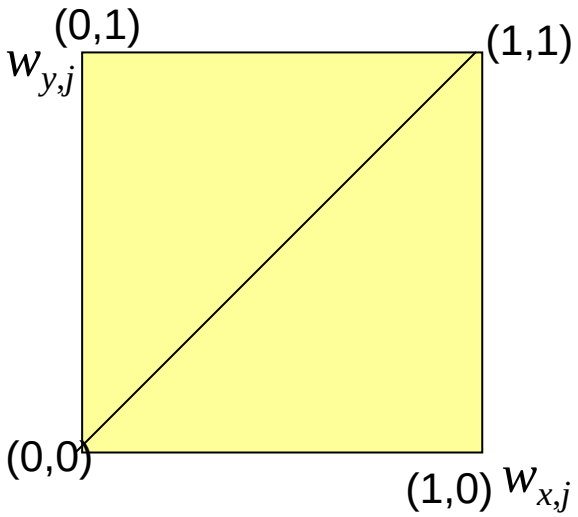
扩展布尔模型中的“与”关系

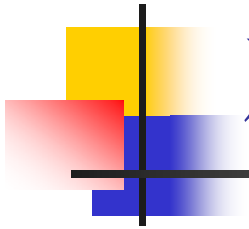
- 给定一个联合的查询式 $x \wedge y$
- $\text{sim}(q_{\text{and}}, d_j) = 1 - \{ [(1-x)^2 + (1-y)^2]/2 \}^{0.5}$
- 函数sim() 表示从(1,1) 出发到d的向量长度



观察

- 如果x出现在文档d_j中，则x在文档d_j中具有权重1，否则为0
- 当d_j包含x和y时
  $\text{sim}(q_{\text{and}}, d_j) = \text{sim}(q_{\text{or}}, d_j) = 1$
- 当d_j既不包含x也不包含y时
  $\text{sim}(q_{\text{and}}, d_j) = \text{sim}(q_{\text{or}}, d_j) = 0$
- 当d_j包含x和y二者之一时
  $\text{sim}(q_{\text{and}}, d_j) = 1 - 2^{0.5} = 0.293$
  $\text{sim}(q_{\text{or}}, d_j) = 2^{0.5} = 0.707$



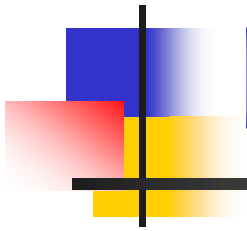


观察

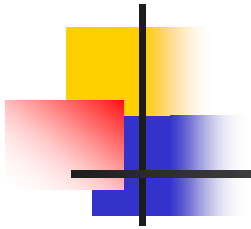
- 一个词项的存在将对“或”关系查询式提供0.707的增益值，但对“与”关系查询式仅提供0.293的增益值
  - 一个词项不存在，将给“与”关系的查询式提供0.707的罚分
- 当x和y有权值0.5, $\text{sim}(q_{\text{and}}, d) = \text{sim}(q_{\text{or}}, d) = 0.5$
  - 在一个“与”关系查询中，两个词项的权重均为0.5，则相似度为0.5。其中一个权重为1，另一个为0，相似度为0.293。
  - 在“或关系”查询中，情况恰好相反
- 在“与关系”查询中，如果一个词项的权重低于0.5，将给相似度贡献一个较大的罚分

p-norm 模型

- 扩展布尔模型可以被泛化为*m* 个查询项:
  $sim(q_{or}, d) = [(x_1^2 + x_2^2 + ... + x_m^2) / m]^{0.5}$
  $sim(q_{and}, d) = 1 - \{ [(1- x_1)^2 + (1- x_2)^2 + ... + (1- x_m)^2] / m \}^{0.5}$
- 它可以被进一步地 泛化为*p*-norm model:
  $sim(q_{or}, d) = [(x_1^p + x_2^p + ... + x_m^p) / m]^{1/p}$
  $sim(q_{and}, d) = 1 - \{ [(1- x_1)^p + (1- x_2)^p + ... + (1- x_m)^p] / m \}^{1/p}$
- 当$p = 1$时, $sim(q_{or}, d) = sim(q_{and}, d) = (x_1 + x_2 + ... + x_m)/ m$
  - 和向量空间中的内积相似
- 当$p = \infty$, $sim(q_{or}, d) = max(x_i)$; $sim(q_{and}, d) = min(x_i)$
  - 模糊逻辑模型(Fuzzy logic model)



隐性语义索引(LSI)



问题引出

- 自然语言文本中的词汇(术语)具有一词多义(polysemy)和一义多词(synonymy)的特点.
- 由于一词多义, 基于精确匹配的检索算法会报告许多用户不要的东西;
  - 处理
    - 什么地方处理旧家具?
    - 你去把那个叛徒处理了
    - 处理自然语言很难
- 由于一义多词, 基于精确匹配的检索算法又会遗漏许多用户想要的东西.
  - “互联网”，“万维网”，“因特网”，“国际互联网”等

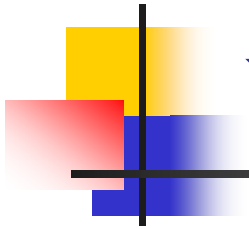
词汇-文档矩阵

- 设Doc1, Doc2, Doc3是三个文件. 一些术语在这三个文件中的出现情况如下表:

	Doc1	Doc2	Doc3
access	X		
document	X		
retrieval	X		X
information		X*	X*
theory		X	
database	X		
indexing	X		
computer		X*	X*

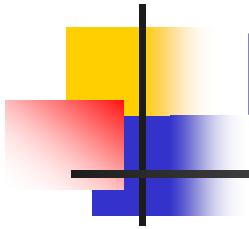
假定用"information" 和"computer"作为主题词进行检索, 那么Doc2和Doc3与之精确匹配, 因而中选.

然而, Doc2是用户并不想要的文件, Doc1才是想要的查不出来, 不想要的倒查了出来. 这说明精确匹配不能很好地反映用户的意图.



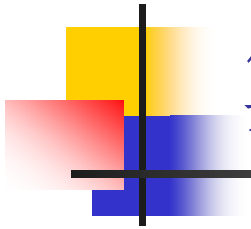
词汇-文档矩阵

- LSI将自然语言中的每个文档视为以词汇为维度的空间中的一个点，认为一个包含语义的文档出现在这种空间中，它的分布绝对不是随机的，而是服从某种语义结构。
- 同样地，也将每个词汇视为以文档为维度的空间中的一个点。文档是由词汇组成的，而词汇又要放到文档中去理解，体现了一种“词汇—文档”双重概率关系。



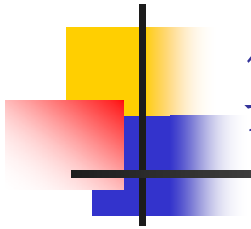
LSI地提出

- 当然, 如果能基于自然语言理解来做这件事, 那一切问题就都没有了。问题是:
  - 自然语言理解的目前水平还是有限度的;
  - 即使用自然语言理解, 效率也会很低
- 我们希望找到一种办法, 既能反映术语之间内在的相关性, 又具有较高的效率.
- 1990年，来自University of Chicago、Bell Communications Research和University of Western Ontario的Scott Deerwester、Thomas K. Landauer等五位学者共同提出了潜在语义分析（Latent Semantic Analysis，缩写为LSA）这一自然语言处理的方法



算法步骤

- 以词项(terms)为行, 文档(documents)为列做一个大矩阵(matrix). 设一共有t行d列, 矩阵名为X. 矩阵的元素为词项在文档中的出现频度.
- 数学上可以证明:
  - X可以分解为三个矩阵T0, S0, D0'(D0的转置)的积.
    - T0和D0的列向量都是正交归一化的
    - S0是对角矩阵
    - T0是t*m矩阵, S0是m*m矩阵,D0是d*m矩阵
    - m是X的秩
  - 这种分解叫做单值分解
    - singlar value decomposition,简称SVD
  - X=T0*S0*D0‘



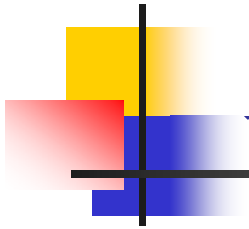
算法步骤

- 一般要求T0, S0, D0都是满秩的. 不难做到把S0的元素沿对角线从大到小排列.
- 现在, 把S0的m个对角元素的前k个保留, 后m-k个置0, 我们可以得到一个新的近似的分解:
  - Xhat=T*S*D'
  - T, S, D三个矩阵在文档检索中有重要的应用价值.
- 奇妙的是, Xhat在最小二乘意义下是X的最佳近似! 这样, 我们实际上有了一个"降维"的途径.
- K值的选择
  - k越大失真越小, 但开销越大
  - k的选择是按实际问题的要求进行平衡的结果

SVD示例

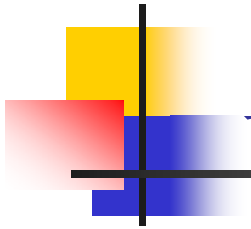
$$\mathbf{A}=\begin{bmatrix}0.96 & 1.72\\ 2.28 & 0.96\end{bmatrix}=\mathbf{U\Sigma V}^{T}=\begin{bmatrix}0.6 & -0.8\\ 0.8 & 0.6\end{bmatrix}\begin{bmatrix}3 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}0.8 & 0.6\\ 0.6 & -0.8\end{bmatrix}^{T}$$

$$\begin{bmatrix}0.6\\ 0.8\end{bmatrix}\begin{bmatrix}3\end{bmatrix}\begin{bmatrix}0.8 & 0.6\end{bmatrix}=\begin{bmatrix}1.44 & 1.08\\ 1.92 & 1.44\end{bmatrix}$$



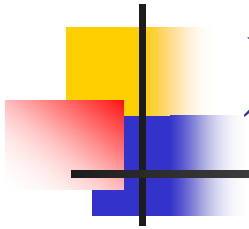
三个问题

- 给定矩阵X, 基于X可以问三类同文件检索密切有关的问题
  - 术语i和j有多相似?
    - 即术语的类比和聚类问题
  - 文件i和j有多相似?
    - 即文件的类比和聚类问题
  - 术语i和文件j有多相关?
    - 即术语和文件的关联问题



三个问题的答案

- 比较两个术语
  - 做"正向"乘法:
  - Xhat*Xhat'=T*S*D'*D*S*T'=T*S^2*T‘
  - D'*D=I, 因为D已经是正交归一的
  - 它的第i行第j列表明了术语i和j的相似程度
- 比较两个文件做"逆向"乘法:
  - Xhat'*Xhat=D*S*T'*T*S*D'=D*S^2*D‘
  - T'*T=I, 因为T已经是正交归一的
  - 它的第i行第j列表明了文件i和j的相似程度
- 比较一个文件和一个术语恰巧就是Xhat本身.
  - 它的第i行第j列表明了术语i和文件j的相关联程度.

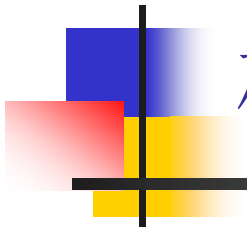


对LSI的理解

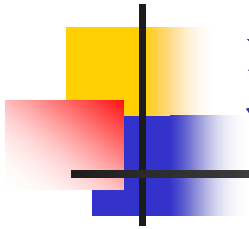
- 最佳近似矩阵
  - 从数据压缩的角度看，*B*是秩为*k*的前提下矩阵*A*的全局最佳近似矩阵。
- 降维
  - LSA不同于向量空间模型（VSM）中文档和词汇的高维表示，而是将文档和词汇的高维表示投影在低维的潜在语义空间（Latent Semantic Space）中，缩小了问题的规模，得到词汇和文档的低维表示。
- 语义关联的发现
  - 对应于小奇异值的奇异向量被忽略后，噪声被大量消减，而使语言单元之间的意义上的相关性显示出来。
  - 潜在语义空间中（不论是文档空间，还是词汇空间），每个维度代表了一个潜概念（Latent Concept）

利用LSI进行检索

- 对查询式的要求
  - 和传统的基于关键词的查询不同，潜语义检索允许用户提交类似于自然语言的查询条件，而不一定必须是几个分离的词汇。
  - 查询式越长，提供的信息需求越充分，越明确
- 对查询式q进行处理
  - 和对文本的处理一样，LSI也根据用户提问式中各词汇的出现频率生成提问式向量Xq，Xq中第i个元素的数值，表示第i个词汇在提问式q中出现的频次。
  - 对Xq进行截断奇异值分解，使其可以和文本矩阵D进行比较。得到：$Dq=X_q'*T*S^{-1}$
  - 得出的Dq即为提问式q在k维语义空间内的坐标向量。这样，词汇、文本、提问式三者的坐标向量，构成了我们所需的隐含语义空间。
- 检索过程
  - 检索过程就是把查询式的集合视为是一个虚拟的文件，检索的任务是把这个虚拟的文件和其他文件做相似性比较，挑选最相似的出来
  - 相似度计算方法可以采用线性代数理论中的各种方法，比如向量夹角等，根据实际情况而定

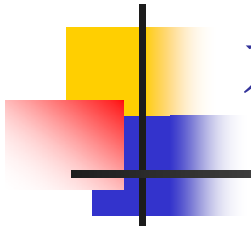


概率模型



理想结果

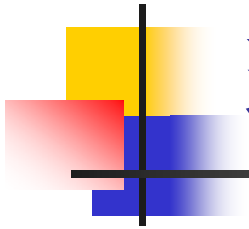
- 理想答案集(ideal answer set)
  - 给定一个用户的查询串，相对于该串存在一个包含所有相关文档的集合
  - 我们把这样的集合看作是一个理想的结果文档集
- 用索引项刻画理想答案集的属性
  - 把查询处理看作是对理想结果文档集属性的处理
  - 我们并不能确切地知道这些属性，我们所知道的是存在索引词来刻画这些属性



初始估计和相关反馈

- 初始估计
  - 由于在查询期间这些属性都是不可见的，这就需要在初始阶段来估计这些属性。
  - 这种初始阶段的估计允许我们对首次检索的文档集合返回理想的结果集，并产生一个初步的概率描述。

- 相关反馈(relevance feedback)
  - 为了提高理想结果集的描述概率，系统需要与用户进行交互式操作，具体处理过程如下：
    - 用户大致浏览一下结果文档，决定哪些是相关的，哪些是不相关的；
    - 然后系统利用该信息重新定义理想结果集的概率描述；
    - 重复以上操作，就会越来越接近真正的结果文档集。



理论

- 概率模型是基于以下理论：
  - 给定一个用户的查询串 q和集合中的文档d_j概率模型来估计用户查询串与文档d_j 相关的概率。
- 概率模型假设这种概率只决定于查询串和文档。
- 更进一步说，该模型假定存在一个所有文档的集合，即相对于查询串q的结果文档子集，这种理想的集合用R表示，集合中的文档是被预料与查询串相关的。
- 这种假设存在着缺点，因为他没有明确定义计算相关度的概率，下面将给出这种概率的定义。

相关度概率的定义

- 在概率模型中索引术语的权重都是二元的
  - $w_{i,j} \in \{0,1\}$, $w_{i,q} \in \{0,1\}$,
- 查询式q是索引词项集合的子集
- 设R是相关文档集合（初始的猜测集合），$\bar{R}$是R的补集（非相关文档的集合）
- $P(R|\vec{d}_j)$表示文档d_j和查询式q相关的概率；
- $P(\bar{R}|\vec{d}_j)$表示文档d_j和查询式q不相关的概率；

定义

- 文档d_j对于查询串q的相关度值定义为：
  - $Sim(d_j,q)=P(R|\vec{d_j})/P(\overline{R}|\vec{d_j})$
- 根据贝叶斯原理
  - $Sim(d_j,q)=(P(\vec{d_j}|R)P(R))/(P(\vec{d_j}|\overline{R})P(\overline{R}))$
  - $P(d_j|R)$代表从相关文档集合R中随机选取文档d_j的概率，$P(R)$表示从整个集合中随机选取一篇文档作为相关文档的概率，依此定义$P(d_j|\overline{R})$和$P(\overline{R})$

推导

- 因为对于集合中所有的文档$P(R)$和$P(\overline{R})$ 是相同的，所以
- $Sim(d_j,q)=P(d_j|R) / P(d_j|R)$
- 假设索引术语是相互独立的则：

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

$g_i(d_j)=1, g_i(d_j)=0$

最终的相似的计算公式

- $P(k_i|R)$表示集合R中随机选取的文档中出现索引术语k_i的概率，$P(\overline{k_i}|R)$表示集合R中随机选取的文档中不出现索引术语的概率，$P(k_i|R)+P(\overline{k_i}|R)=1$
- 类似定义$P(k_i|\overline{R})$和$P(\overline{k_i}|\overline{R})$，最终得到：

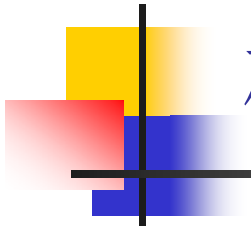
$$sim(d_j,q) \approx \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{p(k_i|R)}{1-p(k_i|R)} + \log \frac{p(k_i|\overline{R})}{1-p(k_i|\overline{R})}\right)$$

$i=1\ldots t$

初始化方法

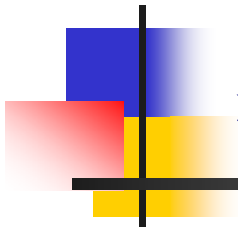
- 由于我们在开始时并不知道集合R，因此必须设计一个初始化计算$P(k_i|R)$和$P(k_i|R)$的算法。

- 在查询的开始间段只定义了查询串，还没有得到结果文档集。我们不得不作一些简单的假设，例如：
  - $P(k_i|R)=0.5$
    - 假定$P(k_i|R)$对所有的索引术语来说是常数（一般等于0.5）
  - $P(ki|R)=n_i/N$
    - n_i表示出现索引术语k_i的文档的数目，N是集合中总的文档的数目。
    - 假定索引术语在非相关文档中的分布可以由索引术语在集合中所有文档中的分布来近似表示。



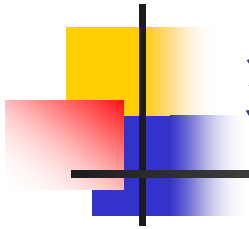
概率模型

- 优点
  - 文档可以按照他们相关概率递减的顺序来计算秩（rank）。
- 缺点
  - 开始时需要猜想把文档分为相关和不相关的两个集合，一般来说很难
  - 实际上这种模型没有考虑索引术语在文档中的频率（因为所有的权重都是二元的），而索引术语都是相互独立的
- 概率模型是否要比向量模型好还存在着争论，但现在向量模型使用的比较广泛。



基于统计语言模型的信息检索模型

统计语言模型



- 统计语言模型在语音识别中产生
- $argmax\ p(s|a)$，s是文字串，a是声学参数串
- $argmax\ p(s|a)=argmax\ p(a|s)p(s)/p(a)$
  - 忽略$p(a)$，$p(a|s)$是声学模型
  - $p(s)$是语言模型
- $p(s)=p(w_1,w_2,w_3,\ldots,w_n)=\Pi_{i=1\ldots n}p(w_i|h_i)$
  - n表示句子长度
  - $h_i=w_1,w_2,\ldots,w_{i-1}$，代表上下文

从文档中建立语言模型

- 原始文本
  - *<s0> <s> He can buy you the can of soda </s>*
- 一元模型(*Unigram*): *(8 words in vocabulary)*
  - *p1(He) = p1(buy) = p1 (you) = p1 (the) = p1(of) = p1(soda)= .125, p1(can) = .25*
- 二元模型(*Bigram*):
  - *p2(He|<s>) = 1, p2(can|He) = 1, p2(buy|can) = .5, p2(of|can) = .5, p2(you |buy) = 1,...*
- 三元模型(*Trigram*):
  - *p3(He|<s0>,<s>) = 1, p3(can|<s>,He) = 1, p3(buy|He,can) = 1, p3(of|the,can)= 1, ..., p3(</s>|of,soda) = 1.*

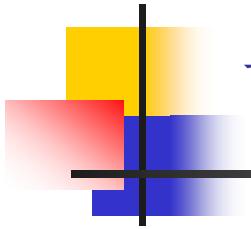
举例——智能拼音输入问题

- yi zhi xiao hua mao

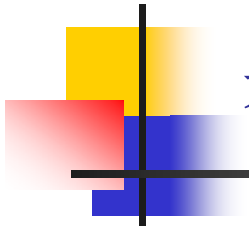
一	之	小	华	毛
以	只	校	话	贸
异	之	销	化	猫
已	枝	…	花	…
…	值		…	
	…			

- 基于大规模语料库建立的语言模型应该能够告诉我们：
  - p(“一只小花猫”)>p(“一枝小花猫”)>p(任何其它候选字串)

语言模型和搜索引擎的相似性



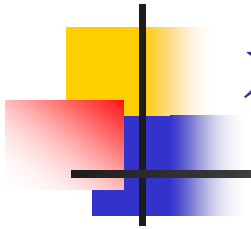
- 利用搜索引擎查找一个词串的过程很象在建立语言模型时统计N-gram出现频度的过程
- 相同的数据稀疏问题
  - 如果在Google中输入的查询式太长，则很难找到满意的结果
    - 原因：如果查询式包括8个词，索引表中有10万词，则$100000^8=10^{40}$，目前互联网的字节数在T级，也就是10^{12}，因此输入太长的查询式无法找到结果，因为数据稀疏
  - 在建立语言模型时同样存在严重的数据稀疏问题
- 有人在探讨利用互联网建立语言模型



基于语言模型的IR模型的概念

- 文档语言模型
  - 每个文档对应一个统计语言模型，称为文档的语言模型(Language Model)。
  - 它主要描述了该文档中各个单词的统计分布特征。
  - 因此每个文档看作是由其语言模型抽样产生的一个样本。

- 基于文档语言模型计算查询式的出现概率
  - 一个查询式也可以看作是由文档的语言模型抽样产生的一个样本。
  - 因此可以根据每个文档的语言模型抽样生成检索的概率来对其排序，其概率值越大，在该文档就越满足该检索要求。



举例

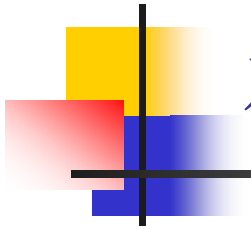
- 假设文档集合中只有1和2两个文本
- 文本1产生的语言模型1
  - $p_1(a)=0.25, p_1(b)=0.5, p_1(\alpha)=1/64, \alpha \in \{c..r\}$,剩下的s,t,u,v,w,x,y,z均为0
- 文本2产生的语言模型2
  - $p_2(a)=0.7, p_2(b)=0.05, p_2(\alpha)=1/64, \alpha \in \{c..r\}$,剩下的s,t,u,v,w,x,y,z均为0
- 查询式：q=abacaad
  - $p_1(q)=0.25*0.5*0.25*1/64*0.25*0.25*1/64 \approx 4.8*10^{-7}$
  - $p_2(q)=0.7*0.05*0.7*1/64*0.7*0.7*1/64 \approx 2.9*10^{-6}$

例子中的检索结果

- 从上例中可以看出
  - q在语言模型1下获得了较低的概率$4.8*10^{-7}$
  - q在语言模型2下获得了较高的概率$2.9*10^{-6}$
- 说明
  - 文本2比文本1更有可能生成q
  - 若输入q，应该检索出文本2，而不是文本1

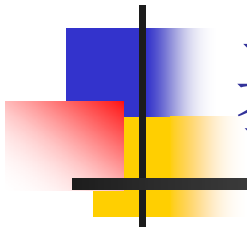
和传统概率模型的比较

- 基本思想完全不同
  - 传统的信息检索概率模型
    - 文档*d*与检索*q*的相关度排序函数定义为事件*R*(文档是否满足检索要求)的概率，即：f(q,d)=P(R|d) ；
    - 相关度排序函数定义虽然比较直观，但相关性是一个抽象的概念，该定义本身没有也无法具体给出*R*的定义，所以该模型在理论上存在很大的模糊性。
  - 基于语言模型的检索模型
    - 相关度排序函数则定义为由文档的语言模型生成检索的概率，即f(q,d)=p(q|d)。
    - 建立在统计语言模型理论基础上，定义明确，便于操作。

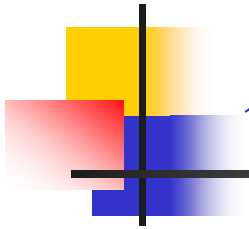


和传统概率模型的比较（续）

- 具体实施方法不同
  - 传统的概率模型
    - 由于没有也无法对相关性做出明确定义，因此一般需要在检索中，首先给定带有相关性标记的文档作为建立模型的基础。
    - 在实际中，要针对每个检索给定学习数据，几乎不可能。该问题是传统信息检索模型存在的一个主要问题。
  - 基于语言模型的信息检索模型
    - 可以基于每个文档直接计算出相关度排序函数，从而有效地避免这个问题
    - 还可以用该模型为传统概率模型形成初始检索。



基于本体论的信息检索模型



本体论

- 本体论（Ontology）最早是哲学的分支，研究客观事物存在的本质。
  - 本体（ontology）的含义是形成现象的根本实体(常与“现象”相对)。从哲学的范畴来说，本体是客观存在的一个系统的解释或说明，关心的是客观现实的抽象本质。
  - 它与认识论（Epistemology）相对，认识论研究人类知识的本质和来源。本体论研究客观存在，认识论研究主观认知。

各种关于本体的定义

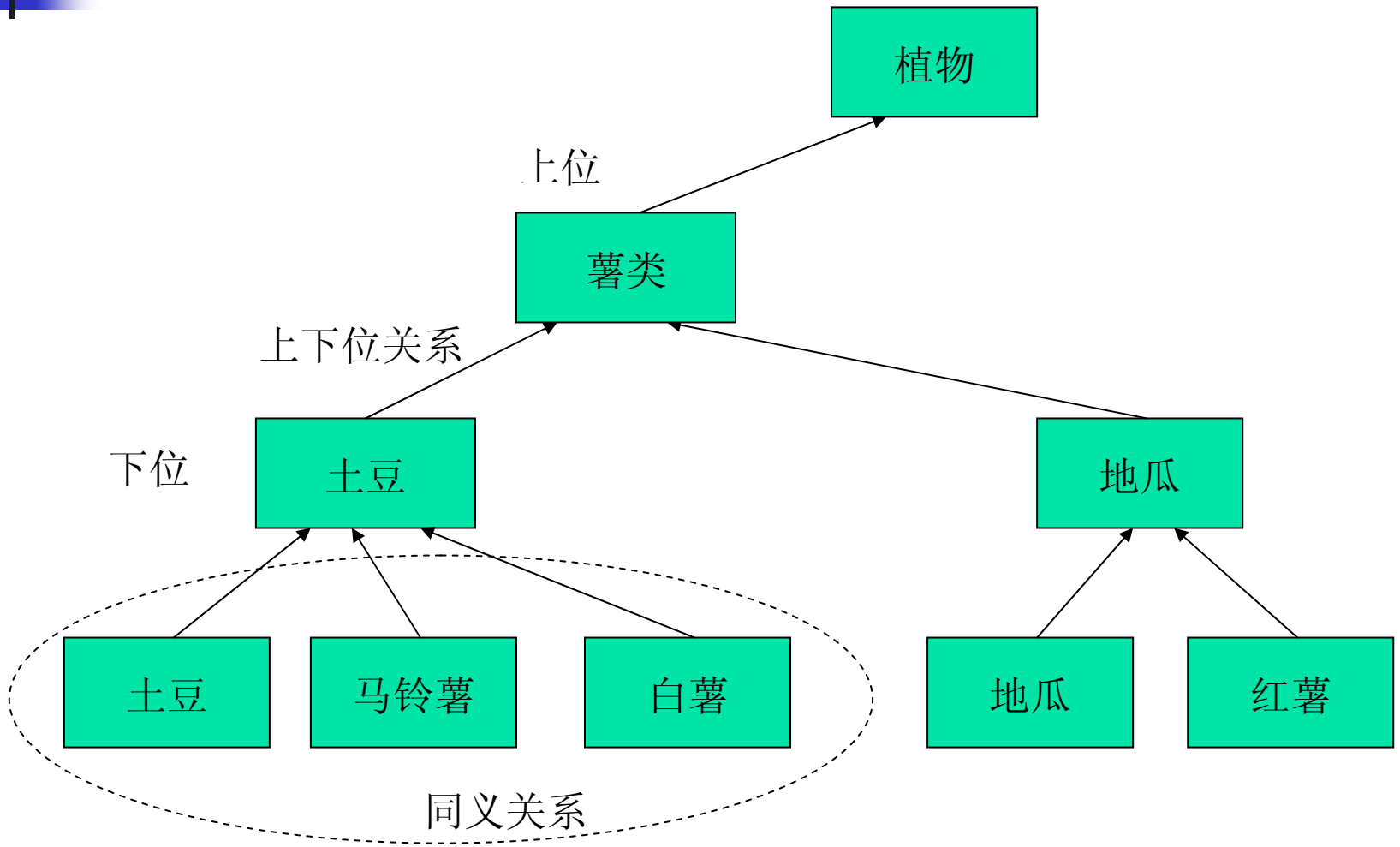
- 在人工智能界，最早给出本体定义的是Neches等人，将本体定义为“给出构成相关领域词汇的基本术语和关系，以及利用这些术语和关系构成的规定这些词汇外延的规则的定义”。
- 1993年，Gruber给出了本体的一个最为流行的定义，即“本体是概念模型的明确的规范说明”。
- 后来，Borst在此基础上,给出了本体的另外一种定义：“本体是共享概念模型的形式化规范说明”。
- Studer等对上述两个定义进行了深入的研究，认为“本体是共享概念模型的明确的形式化规范说明”。

本体的分类和内容

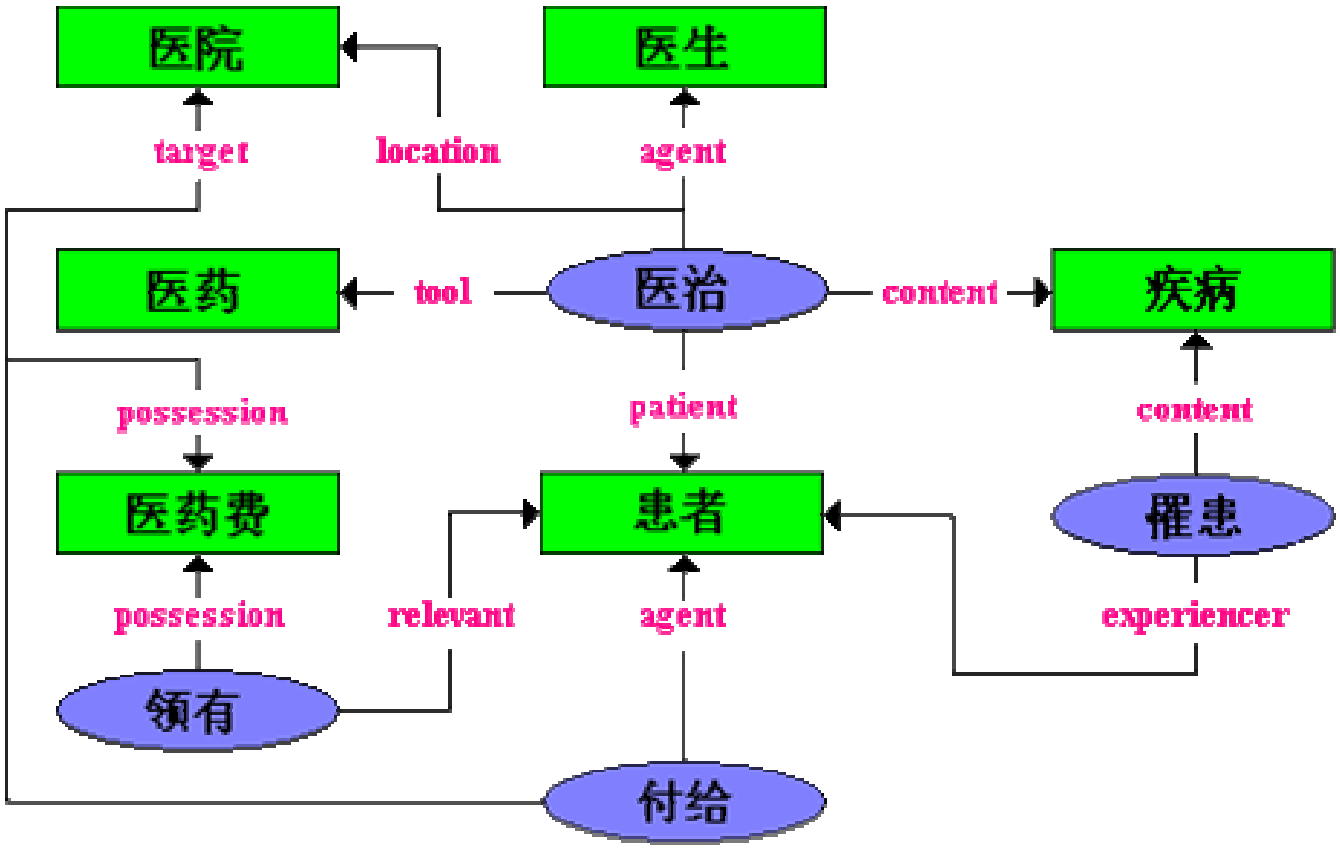
- 本体的分类
  - 本体是采用某种语言对概念化的描述，本体的分类按照表示和描述的形式化的程度不同，可以分为：完全非形式化的、半形式化的、严格形式化的，形式化程度越高，越有利于计算机进行自动处理。

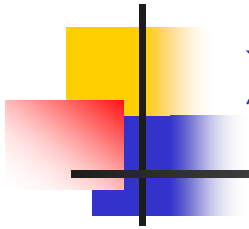
- 本体的内容
  - 从概念化对象的定义来看，一个领域的术语、术语的定义以及各个术语之间的语义网络，应是任一个领域本体论所必须包含的基本信息。
  - 概念之间的关系
    - 同义关系：表达了在相似数据源间的一种等价关系，是一种对称关系
    - 上下位关系：不对称的，是一种偏序关系，具有传递性
    - 其它各种语义关系
  - 各个概念间复杂的语义关系组成了语义网络图，概念在其中表现为节点，而节点间的弧则代表了上述的关系。

上下位关系和同义关系



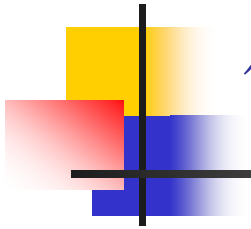
语义关系


医院
医生
target
location
agent
医药
tool
医治
content
疾病
possession
patient
content
医药费
患者
罹患
possession
relevant
agent
experiencer
领有
付给



构造本体的要点

- 出于对各自问题域和具体工程的考虑，构造本体的过程各不相同。目前没有一个标准的本体的构造方法。
- 最有影响的是Gruber在1995年提出的5条规则：
  - 清晰（Clarity）
    - 本体必须有效的说明所定义术语的意思。定义应该是客观的，形式化的
  - 一致（Coherence）
    - 它应该支持与其定义相一致的推理
  - 可扩展性（Extendibility）
    - 应该提供概念基础，支持在已有的概念基础上定义新的术语
  - 编码偏好程度最小（Minimal encoding bias）
    - 概念的描述不应该依赖于某一种特殊的符号层的表示方法
  - 本体约定最小（Minimal ontological commitment）
    - 本体约定应该最小，只要能够满足特定的知识共享需求即可。



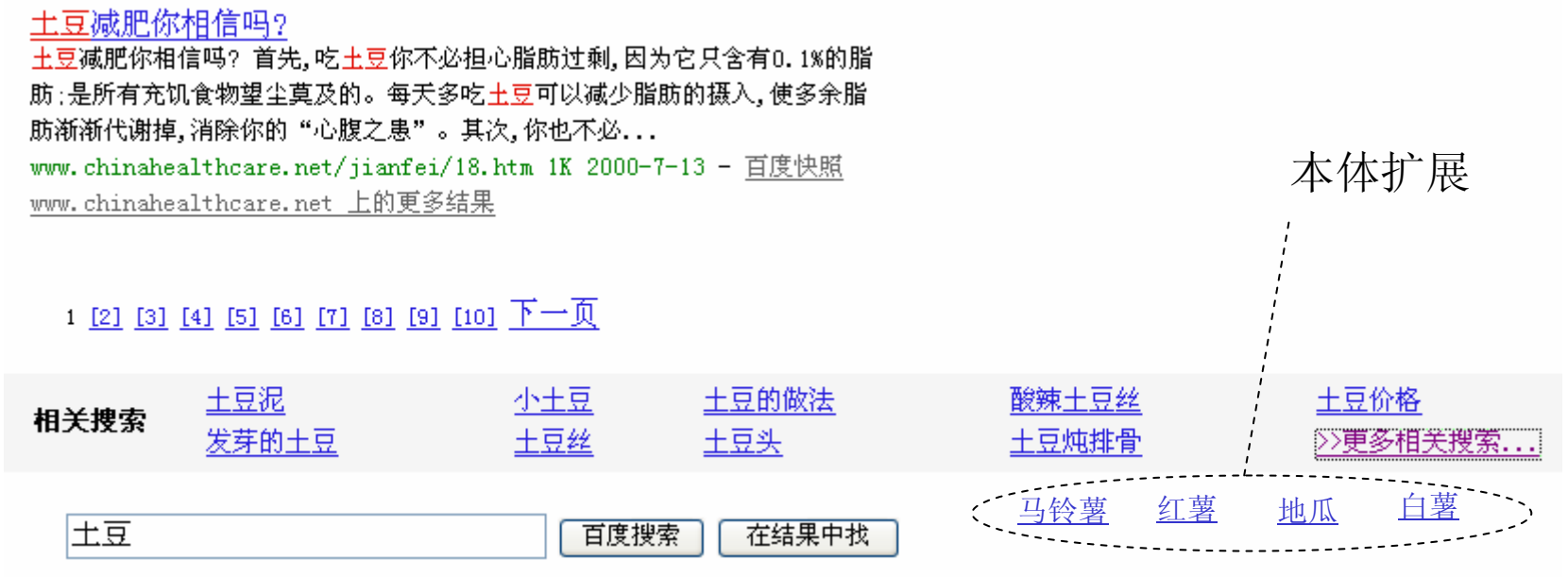
领域本体

- 领域本体(Domain ontology)的概念
  - 提供了某个专业学科领域中概念的词表以及概念间的关系
  - 在该领域里占主导地位的理论，是某一领域的知识表示
- 建立本体的方式
  - 借助某种本体描述语言，采用“恳谈法”从人类专家那里获得知识，经过抽象组织成领域本体
- 应用实例
  - IBM中国研究中心在信息集成项目中运用本体
  - 哈工大机器翻译研究室基于本体进行跨语言检索的研究

基于本体的检索过程

- 用户向信息检索系统提出检索申请。
- 信息检索系统产生一个界面与用户交互。界面接收用户提出的查询关键字后，系统查询本体库，从中找出出现该关键字的各个领域，然后将其领域以及在该领域下的关键字的含义罗列给用户。
- 用户此时可根据自己的意图，在界面上确定所需查找的领域及含义。
- 系统将经过本体规范后的请求交给全文搜索引擎进行检索。
- 全文搜索引擎检索后返回给用户检索信息。

利用本体进行检索的好处



- 解决从查询语言到检索语言之间转换过程中出现的语义损失和曲解等问题
- 保证在检索过程中能够有效地遵循用户的查询意图，获得预期的检索信息。

谢谢！