信息检索导论

An Introduction to Information Retrieval

第6讲 文档评分、词项权重计算及向量空间模型

Scoring, Term Weighting & Vector Space Model

授课人：李波

中国科学院信息工程研究所/国科大网络空间安全学院

*改编自"An introduction to Information retrieval"网上公开的课件，地址 http://nlp.stanford.edu/IR-book/

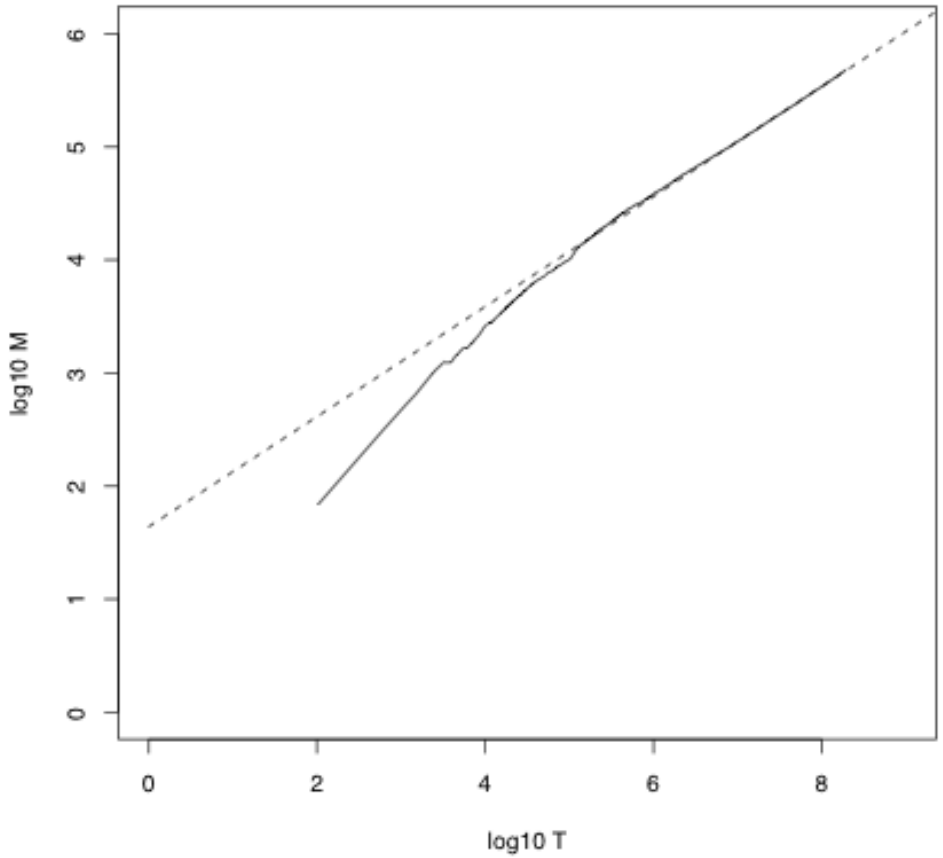
提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

Heaps定律—词典大小的估计



- 词汇表大小M 是文档集规模T的一个函数

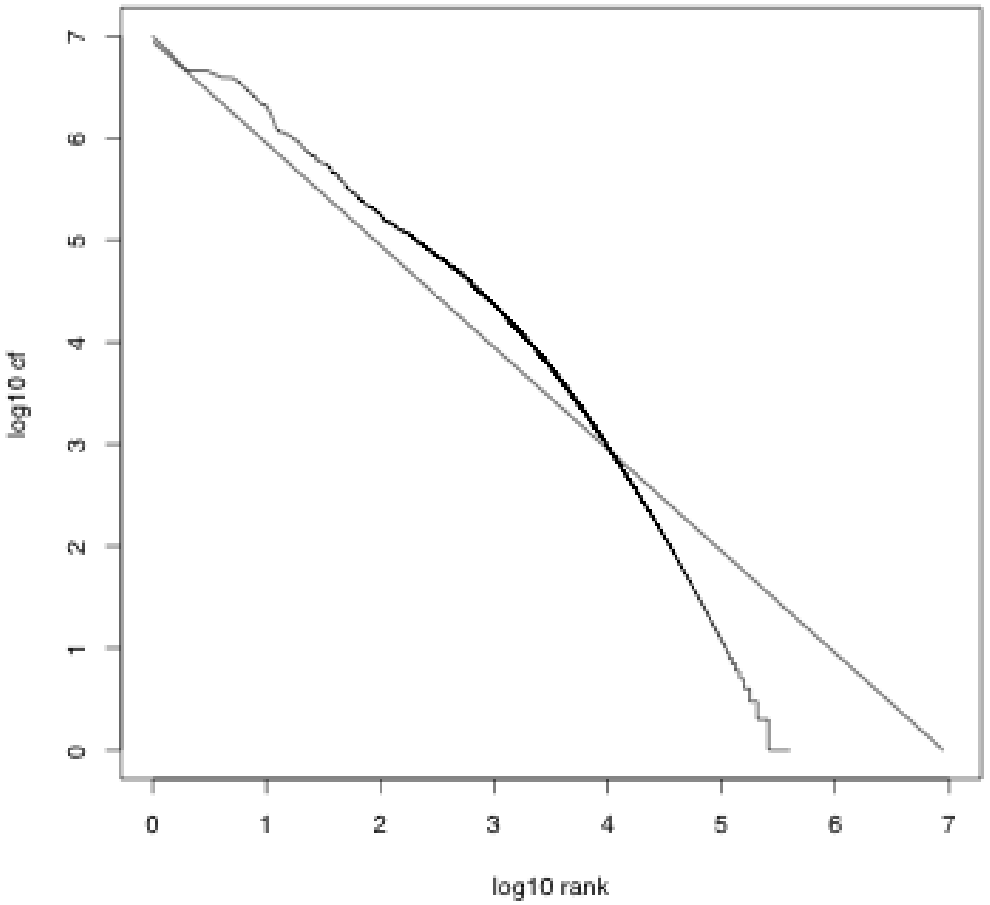
$$M = kT^b$$

- 图中通过最小二乘法拟合出的直线方程为：

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

- 于是有：
- $M = 10^{1.64} T^{0.49}$
- $k = 10^{1.64} \approx 44$
- $b = 0.49$

Zipf定律—倒排记录表大小的估计



反映词项的分布情况

$cf_i * i \approx c$

拟合度不是太高，但是基本能反映词项的分布规律：高频词少，低频词多。

Zipf定律

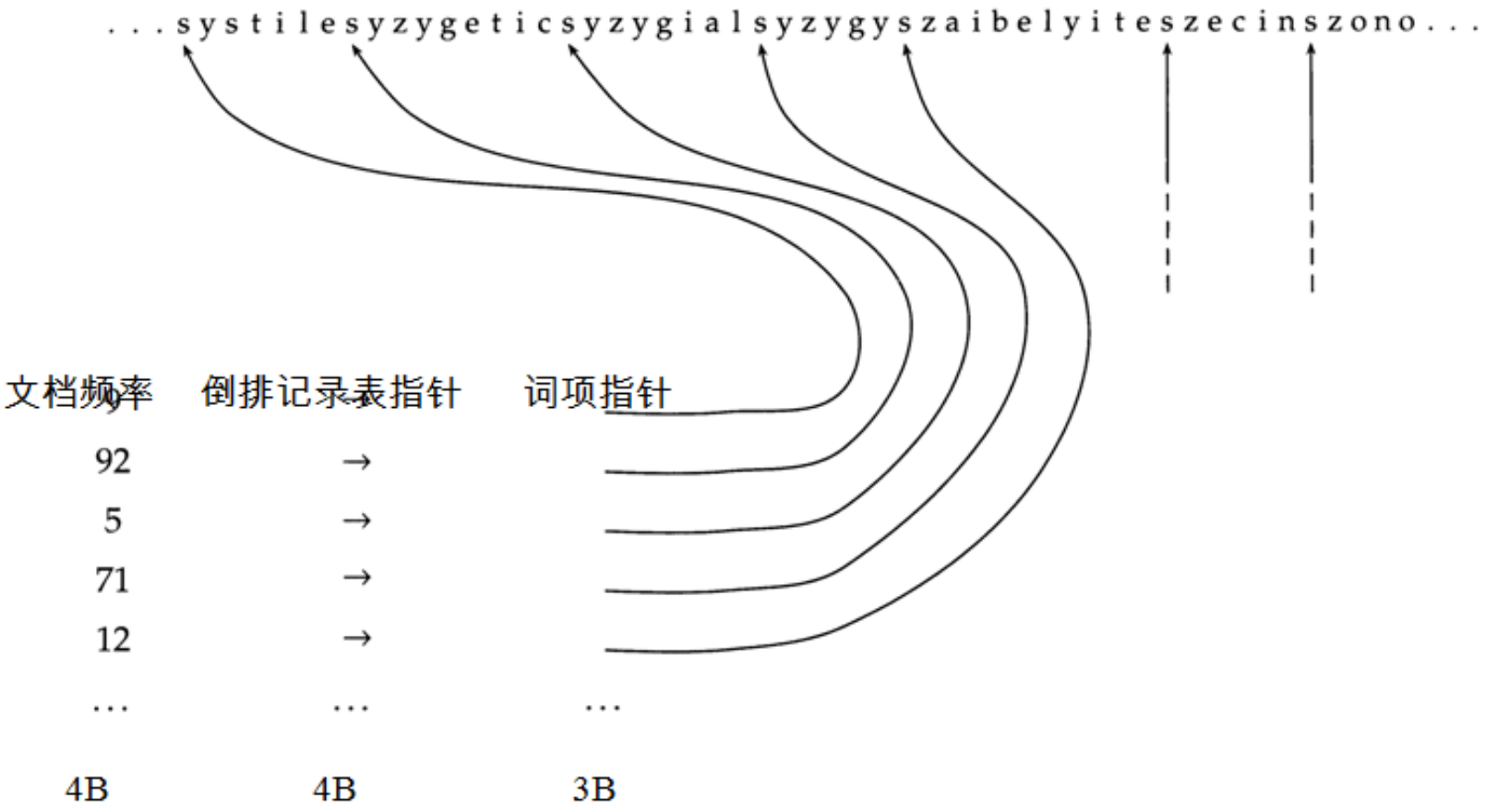
Zipf's law: Rank × Frequency ∼ Constant

English:	Rank R	Word	Frequency f	$R \times f$
	10	he	877	8770
	20	but	410	8200
	30	be	294	8820
	800	friends	10	8000
	1000	family	8	8000

German:	Rank R	Word	Frequency f	$R \times f$
	10	sich	1,680,106	16,801,060
	100	immer	197,502	19,750,200
	500	Mio	36,116	18,059,500
	1,000	Medien	19,041	19,041,000
	5,000	Miete	3,755	19,041,000
	10,000	vorläufige	1.664	16,640,000

*本页借用了Ronan Cummins的课件

词典压缩--将整部词典看成单一字符串



词典压缩---单一字符串方式下按块存储

词典压缩--前端编码(Front coding)

- 每个块当中 (下例k = 4)，会有公共前缀

8automata8automate9automatic10automation

⇓

- 可以采用前端编码方式继续压缩

8automat*a1⋄e2⋄ic3⋄ion

倒排记录表压缩--对间隔编码

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

倒排记录表压缩---可变字节(VB)码

- 设定一个专用位 (高位) *c*作为延续位(continuation bit)
  - 如果间隔表示少于7比特，那么*c* 置 1，将间隔编入一个字节的后7位中
  - 否则：将低7位放入当前字节中，并将*c* 置 0，剩下的位数采用同样的方法进行处理，最后一个字节的*c*置1（表示结束）
  - 比如，257=1 00000001 = 0000010 0000001 ➔ 00000010 10000001
- 被很多商用/研究系统所采用
- 变长编码及对齐敏感性(指匹配时按字节对齐还是按照位对齐)的简单且不错的混合产物

倒排记录表压缩--γ编码

- 将整数G 表示成长度(length)和偏移(offset)两部分
  - 偏移对应G的二进制编码，只不过将首部的1去掉
  - 例如 13 → 1101 → 101 = 偏移
  - 长度部分给出的是偏移的位数
  - 比如G=13 (偏移为 101), 长度部分为 3
  - 长度部分采用一元编码: 1110
  - 13的γ编码为1110 101
- G的γ编码就是将长度部分和偏移部分两者联接起来得到的结果。

Reuters RCV1索引压缩总表

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

索引压缩实例分析——Google

- 查询响应时间由3天降低到50.5秒！
- 继续降低到7.58秒！
- 降低到2毫秒！
- :-)

- *“This engine is incredibly, amazingly, ridiculously fast!”*
  - (from “Top Gear”)

本讲内容

- 对搜索结果排序(Ranking) : 为什么排序相当重要?
- 词项频率(Term Frequency, TF): 排序中的重要因子
- 逆文档频率(Inverse Document Frequency, IDF): 另一个重要因子
- TFIDF 权重计算方法: 最出名的经典排序方法
- 向量空间模型(Vector space model): 信息检索中最重要的形式化模型之一 (其他模型还包括布尔模型、概率模型、统计语言建模模型等)

提纲

1. 上一讲回顾
2. **排序式检索**
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

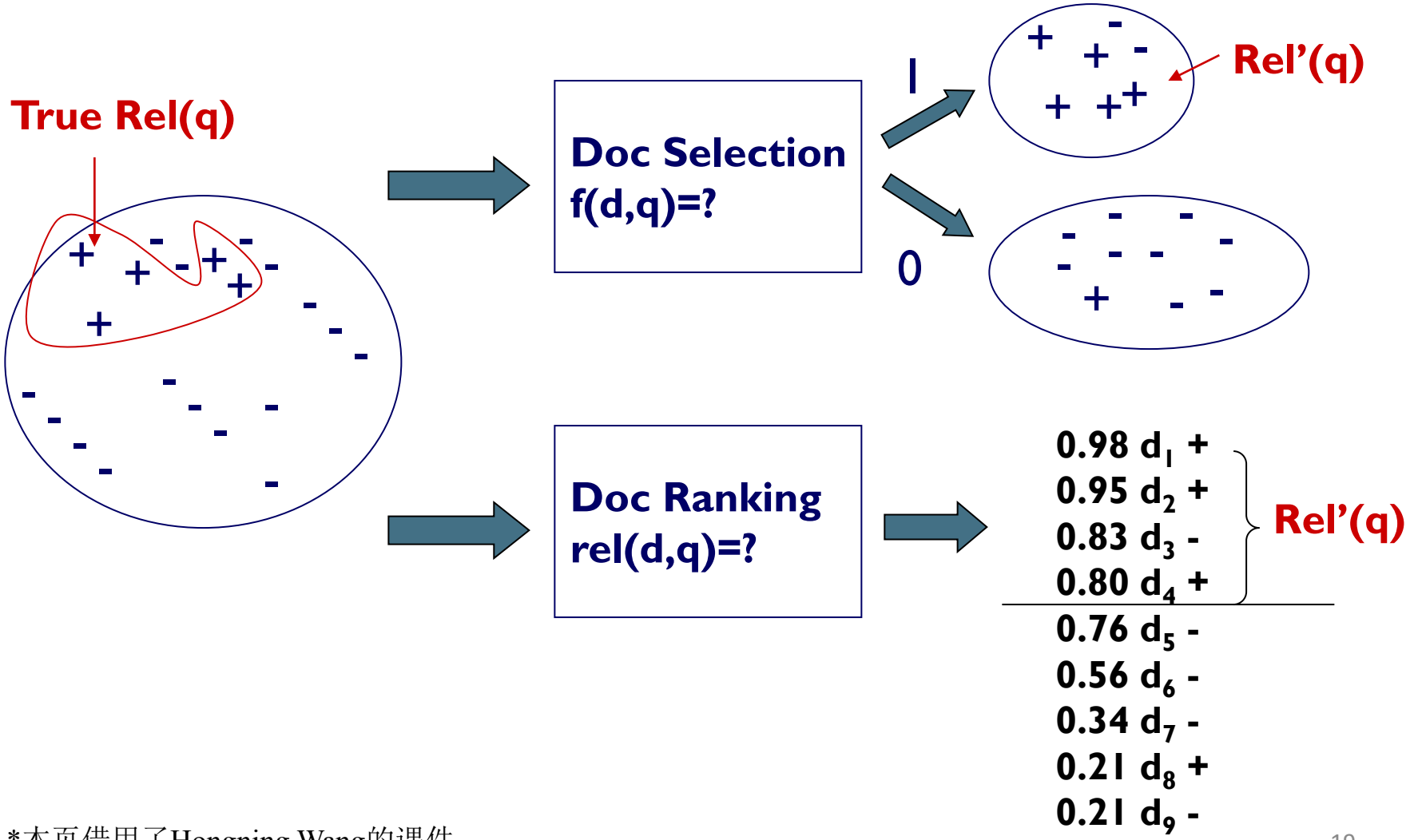
布尔检索

- 迄今为止，我们主要关注的是布尔检索，即文档(与查询)要么匹配要么不匹配
- 布尔检索的优点：
  - 对自身需求和文档集性质非常了解的专家而言，布尔查询是不错的选择
  - 对应用开发来说也非常简单，很容易就可以返回1000多条结果
- 布尔检索的不足：
  - 对大多数用户来说不方便
    - 大部分用户不能撰写布尔查询或者他们认为需要大量训练才能撰写出合适的布尔查询
    - 大部分用户不愿意逐条浏览1000多条结果，特别是对Web搜索更是如此

布尔检索的其他不足: 结果过少或者过多

- 布尔查询常常会导致过少(=0)或者过多(>1000)的结果
- 例子：
  - 查询 1 (布尔与操作): [standard user dlink 650]
    - → 200,000 个结果 – 太多
  - 查询2 (布尔与操作): [standard user dlink 650 no card found]
    - → 0 个结果 – 太少
- 结论：
  - 在布尔检索中，需要大量技巧来生成一个可以获得合适规模结果的查询

文档选择与评分的差异



*本页借用了Hongning Wang的课件

排序式检索(Ranked retrieval)

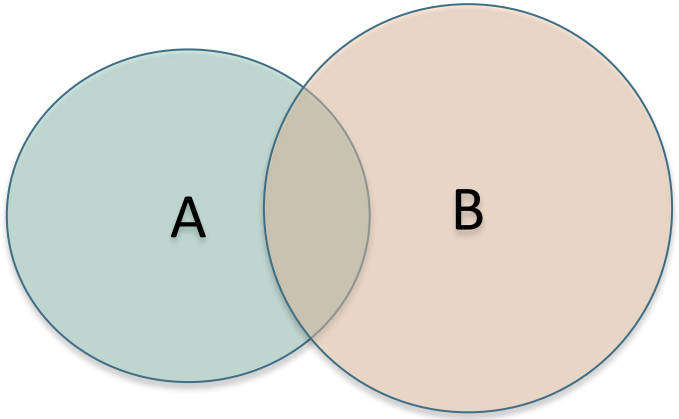
- 排序式检索会对查询和文档的匹配程度进行排序，即给出一个查询和文档匹配评分
- 排序式检索可以避免产生过多或者过少的结果
- 可以通过排序技术来避免大规模返回结果，比如只需要显示前10条结果，这样不会让用户感觉到信息太多
- 用户满意的前提：排序算法真的有效，即相关度大的文档结果会排在相关度小的文档结果之前

排序式检索中的评分技术

- 我们希望，在同一查询下，文档集中相关度高的文档排名高于相关度低的文档
- 如何实现？
  - 通常做法是对每个查询-文档对赋一个[0, 1]之间的分值
  - 该分值度量了文档和查询的匹配程度

方法一: Jaccard系数

- 计算两个集合重合度的常用方法
- 令 A 和 B 为两个集合
- Jaccard系数的计算方法:



$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$(A \neq \emptyset \text{ or } B \neq \emptyset)$

- JACCARD $(A, A) = 1$
- JACCARD $(A, B) = 0$ 如果 $A \cap B = 0$
- A 和 B 不一定要同样大小
- Jaccard 系数会给出一个0到1之间的值

Jaccard系数的计算样例

- 查询 “ides of March” ➔ 3月15日(恺撒的殉难日)
- 文档 “Caesar died in March”

JACCARD(q, d) = 1/6

Jaccard系数的不足

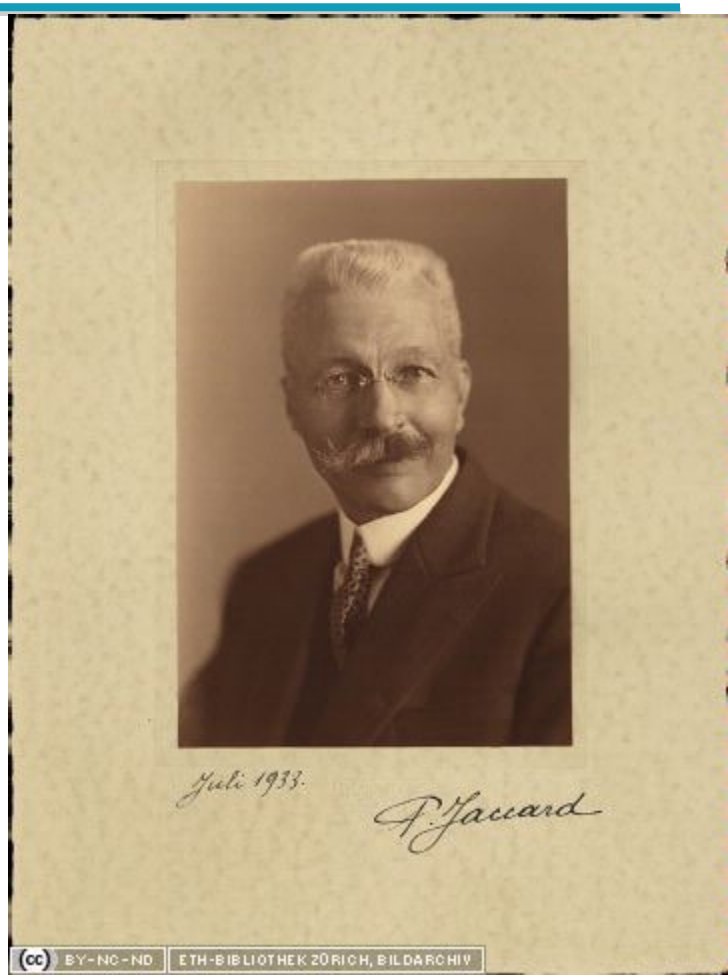
- 不考虑词项频率，即词项在文档中的出现次数(后面会定义)
- 一般而言，罕见词比高频词的信息量更大，Jaccard系数没有考虑这个信息
- 没有仔细考虑文档的长度因素
- 本讲义后面，我们将不考虑使用Jaccard系数来进行查询和文档的评分计算
- 提醒：在第19讲Web网页查重，我们还会介绍大规模网页下的Jaccard系数计算问题（网页和网页之间的相似度）

课堂练习

- 计算下列查询-文档之间的Jaccard系数
  - q: [information on cars] d: “all you’ve ever wanted to know about cars”
  - q: [information on cars] d: “information on trucks, information on planes, information on trains”
  - q: [red cars and red trucks] d: “cops stop red cars more often”

Paul Jaccard(1868-1944)

- 瑞士植物学家，ETH教授
- 1894年毕业于苏黎世联邦理工学院ETH(出过包括爱因斯坦在内的21位诺贝尔奖得主)
- 1901年提出Jaccard Index即Jaccard Coefficient(Jaccard系数)概念



提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

查询-文档匹配评分计算

- 如何计算查询-文档的匹配得分？
- 先从单词项查询(查询只包含一个词项)开始
  - 若该词项不出现在文档当中，该文档得分应该为0
  - 该词项在文档中出现越多，则得分越高
  - 这就是所谓**词项频率** (Term Frequency，简称TF)评分
- 后面我们将给出多种评分的方法

二值关联矩阵

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

每篇文档可以看成是一个二值的向量 $\in \{0, 1\}^{|V|}$

非二值关联矩阵(词项频率)

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

每篇文档可以表示成一个词频向量 $\in \mathbb{N}^{|V|}$

词袋(Bag of words)模型

- 不考虑词在文档中出现的顺序
- *John is quicker than Mary* 及 *Mary is quicker than John* 的表示结果一样
- 这称为一个词袋模型(bag of words model, BOW模型)
- 在某种意思上说，这种表示方法是一种“倒退”，因为位置索引中能够区分上述两篇文档
- 本课程后部将介绍如何“恢复”这些位置信息
- 这里仅考虑词袋模型

词项频率 TF

- 词项t的词项频率(以下简称词频) $tf_{t,d}$ 是指t 在d中出现的次数，是与文档相关的一个量，可以认为是**文档内代表度**的一个量，也可以认为是一种**局部信息**。

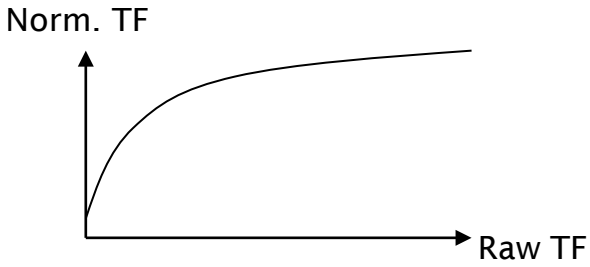
- 下面将介绍利用tf来计算文档评分的方法

- 第一种方法是采用原始的tf值(raw tf)
- 但是原始tf不太合适：
  - 某个词项在A文档中出现十次，即tf = 10，在B文档中tf = 1，那么A比B更相关
  - 但是相关度不会相差10倍，即相关度不会正比于词项频率tf

一种替代原始tf的方法: 对数词频

- t 在 d 中的对数词频权重定义如下：

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$



- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$:
  0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, 等等

- 文档-词项的词频匹配得分是所有同时出现在*q*和文档*d*中的词项的对数词频之和

$$\text{tf-matching-score}(q, d) = \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$$

- 如果两者没有公共词项，则得分为0

课堂练习

- 计算下列查询-文档之间的词频匹配得分
  - q: [information on cars] d: “all you’ve ever wanted to know about cars”
  - q: [information on cars] d: “information on trucks, information on planes, information on trains”
  - q: [red cars and red trucks] d: “cops stop red cars more often”

提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

文档中的词频 vs. 文档集中的词频

- 除词项频率tf之外，我们还想利用词项在整个文档集中的频率进行权重和评分计算

罕见词项所期望的权重

- 罕见词项比常见词所蕴含的信息更多
- 考虑查询中某个词项，它在整个文档集中非常罕见 (例如 ARACHNOCENTRIC).
- 某篇包含该词项的文档很可能相关
- 于是，我们希望像ARACHNOCENTRIC一样的罕见词项将有较高权重

- 物以稀为贵！

常见词项所期望的权重

- 常见词项的信息量不如罕见词
- 考虑一个查询词项，它频繁出现在文档集中 (如 GOOD, INCREASE, LINE等等)
- 一篇包含该词项的文档当然比不包含该词项的文档的相关度要高
- 但是，这些词对于相关度而言并不是非常强的指示词

- 于是，对于诸如GOOD、INCREASE和LINE的频繁词，会给一个正的权重，但是这个权重小于罕见词权重

文档频率(Document frequency, df)

- 对于罕见词项我们希望赋予高权重
- 对于常见词我们希望赋予正的低权重
- 接下来我们使用文档频率df这个因子来计算查询-文档的匹配得分
- 文档频率(document frequency, df)指的是出现词项的文档数目

idf 权重

- df_t 是出现词项t的文档数目
- df_t 是和词项t的信息量成反比的一个值
- 于是可以定义词项t的idf权重(逆文档频率):

$$idf_t = \log_{10} \frac{N}{df_t}$$

(其中N 是文档集中文档的数目)

- idf_t 是反映词项t的信息量的一个指标，是一种全局性指标，反应的是词项在全局的区别性。
- 实际中往往计算[log N/df_t]而不是 [N/df_t]，这可以对idf的影响有所抑制
- 值得注意的是，对于tf 和idf我们都采用了对数计算方式

idf的计算样例

- 利用右式计算idf_t:

$$idf_t = \log_{10} \frac{1,000,000}{df_t}$$

词项	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

idf对排序的影响

- 对于单词项查询,idf对文档排序没有任何影响
- idf 会影响至少包含2个词项的查询的文档排序结果
- 例如，在查询 “arachnocentric line”中, idf权重计算方法会增加arachnocentric的相对权重，同时降低 line的相对权重

文档集频率 vs. 文档频率

单词	文档集频率	文档频率
INSURANCE	10440	3997
TRY	10422	8760

- 词项*t*的文档集频率(Collection frequency，cf) : 文档集中出现的t词条的个数
- 词项*t*的文档频率df: 包含*t*的文档篇数
- 为什么会出现上述表格的情况？即文档集频率相差不大，但是文档频率相差很大
- 哪个词是更好的搜索词项？即应该赋予更高的权重
- 上例表明 df (和idf) 比cf (和“icf”)更适合权重计算

TFIDF权重计算

- 词项的TFIDF权重是tf权重和idf权重的乘积

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- 信息检索中最出名的权重计算方法
- 注意：上面的“-”是连接符，不是减号
- 其他叫法：tf.idf、tf x idf

TFIDF小结

- 词项t在文档d中的权重可以采用下式计算

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- TFIDF权重
  - 随着词项频率的增大而增大
  - 随着词项罕见度的增加而增大

课堂练习: 词项、文档集及文档频率

统计量	符号	定义
词项频率	$tf_{t,d}$	*t*在文档*d*中出现的次数
文档频率	df_t	出现 t的文档数目
文档集频率	cf_t	t在文档集中出现的总次数

- df和cf有什么关系?
- tf和cf有什么关系?
- tf和df有什么关系?

提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

信息检索模型

- 信息检索模型是描述文档和用户查询的表示形式以及它们之间关系的框架
- 信息检索模型是一个四元组[D, Q, F, $R(q_i, d_j)$]
  - D: 文档集的表示
  - Q: 查询的表示
  - F: 文档表示、查询表示和它们之间的关系的模型框架(Frame)
  - $R(q_i, d_j)$: 排序函数，给□ □ □ q_i和文档d_j相关度大小的评价
- 在信息检索的经典模型中：
  - D用一组索引项(也称词项)来描述
  - 查询也看成是一组词项的集合
  - 用(q_i, d_j)来描述查询词项q_i和文档d_j的相关程度，用符号$w_{i,j}$表示相关程度的具体权值

二值关联矩阵

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

每篇文档表示成一个二值向量 $\in \{0, 1\}^{|V|}$

tf矩阵

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

每篇文档表示成一个词频向量$\in \mathbb{N}^{|V|}$

二值 → → tfidf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
. . .						

每篇文档表示成一个基于tfidf权重的实值向量 $\in R^{|V|}$

文档表示成向量

- 每篇文档表示成一个基于tfidf权重的实值向量 $\in R^{|V|}$.
- 于是，我们有一个 $|V|$维实值空间
- 空间的每一维都对应词项
- 文档都是该空间下的一个点或者向量
- 极高维向量：对于Web搜索引擎，空间会上千万维
- 对每个向量来说又非常稀疏，大部分都是0

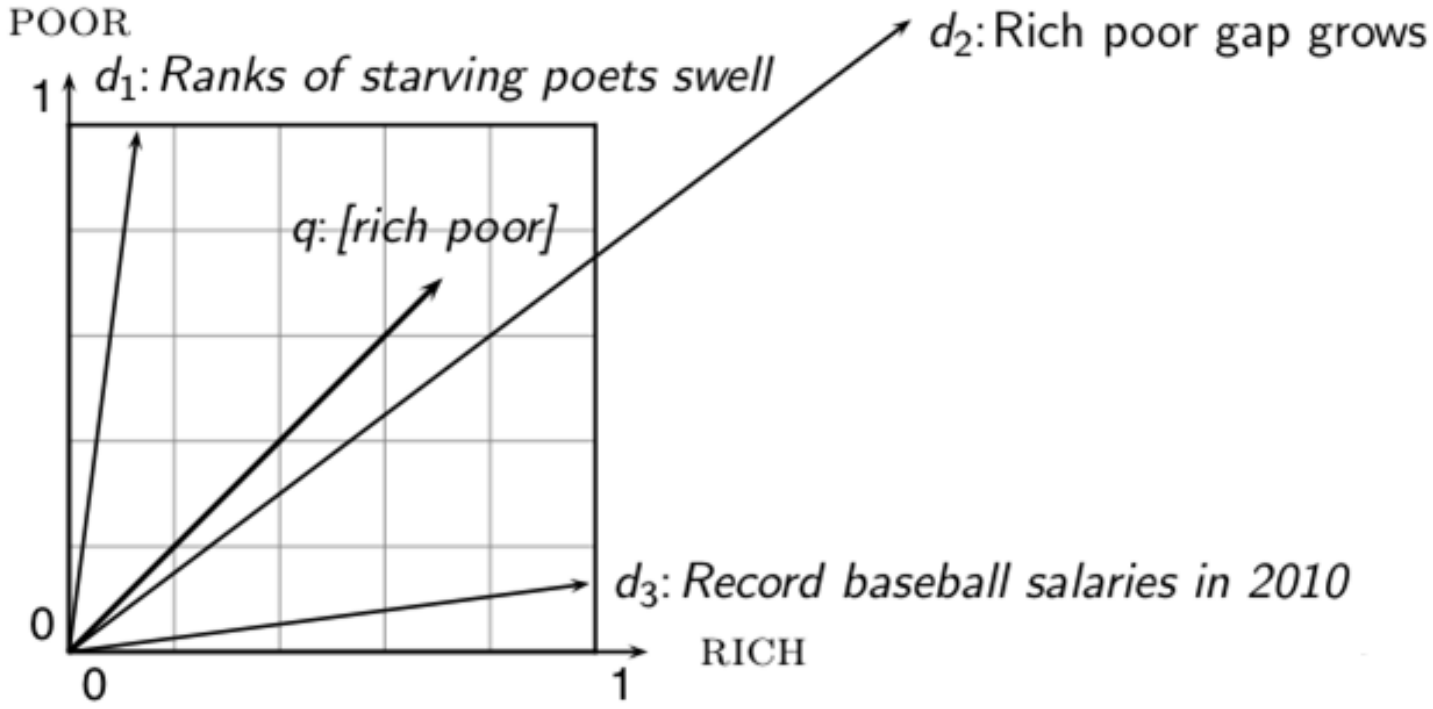
查询看成向量

- 关键思路1: 对于查询做同样的处理，即将查询表示成同一高维空间的向量
- 关键思路2: 按照文档对查询的邻近程度排序
  - 邻近度 = 相似度
  - 邻近度≈ 距离的倒数
- 回想一下，我们是希望和布尔模型不同，能够得到非二值的、既不是过多或也不是过少的检索结果
- 这里，我们通过计算出相关文档的相关度高于不相关文档相关度的方法来实现

向量空间下相似度的形式化定义

- 先考虑一下两个点之间的距离倒数
- 一种方法是采用欧氏距离
- 但是，欧氏距离不是一种好的选择，这是因为欧氏距离对向量长度很敏感

欧氏距离不好的例子



尽管查询q和文档d_2的词项分布$\vec{d}_2$常相似，但是采用欧氏距离计算它们对应向量之间的距离非常大。

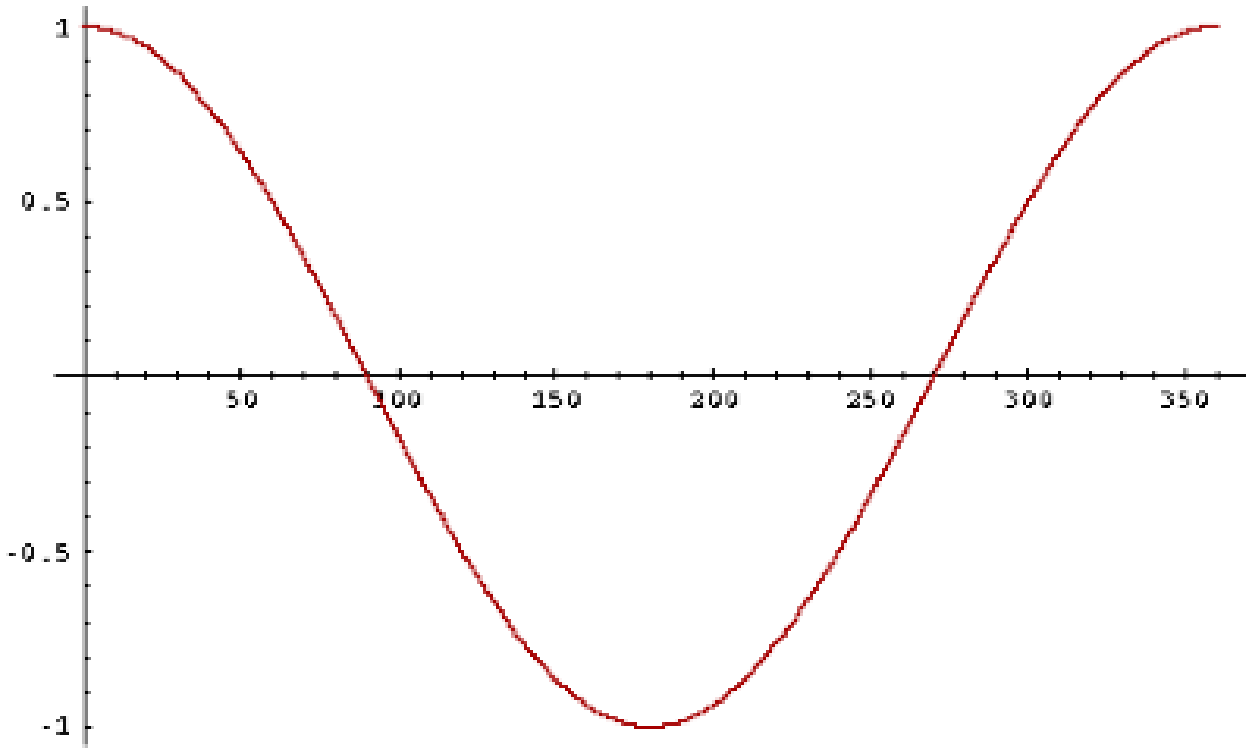
采用夹角而不是距离来计算

- 将文档按照其向量和查询向量的夹角大小来排序
- 假想实验：将文档 d 复制一份加在自身末尾得到文档d'. d' 是d的两倍
- 很显然，从语义上看， d 和 d' 具有相同的内容
- 两者之间的夹角为0，代表它们之间具有最大的相似度
- 但是，它们的欧氏距离可能会很大

从夹角到余弦

- 下面两个说法是等价的：
  - 按照夹角从小到大排列文档
  - 按照余弦从大到小排列文档
- 这是因为在区间[0°, 180°]上，余弦函数cosine是一个单调递减函数

Cosine函数



文档长度归一化

- 如何计算余弦相似度？
- 一个向量可以通过除以它的长度进行归一化处理，以下使用L_2（2范数）：

$$||x||_2 = \sqrt{\sum_i x_i^2}$$

- 这相当于将向量映射到单位球面上
- 这是因为归一化之后：$||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- 因此，长文档和短文档的向量中的权重都处于同一数量级
- 前面提到的文档 d 和 d' (两个d 的叠加) 经过上述归一化之后的向量相同

查询和文档之间的余弦相似度计算

$$\cos(\vec{q},\vec{d}) = \text{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i 是第i 个词项在查询q中的TFIDF权重
- d_i是第i 个词项在文档d中的TFIDF权重
- $|\vec{q}|$ 和 $|\vec{d}|$ 分别是 $\vec{q}$ 和 $\vec{d}$.的长度
- 上述公式就是 $\vec{q}$ 和 $\vec{d}$. 的余弦相似度，或者说向量 $\vec{q}$ 和 $\vec{d}$. 的夹角的余弦

归一化向量的余弦相似度

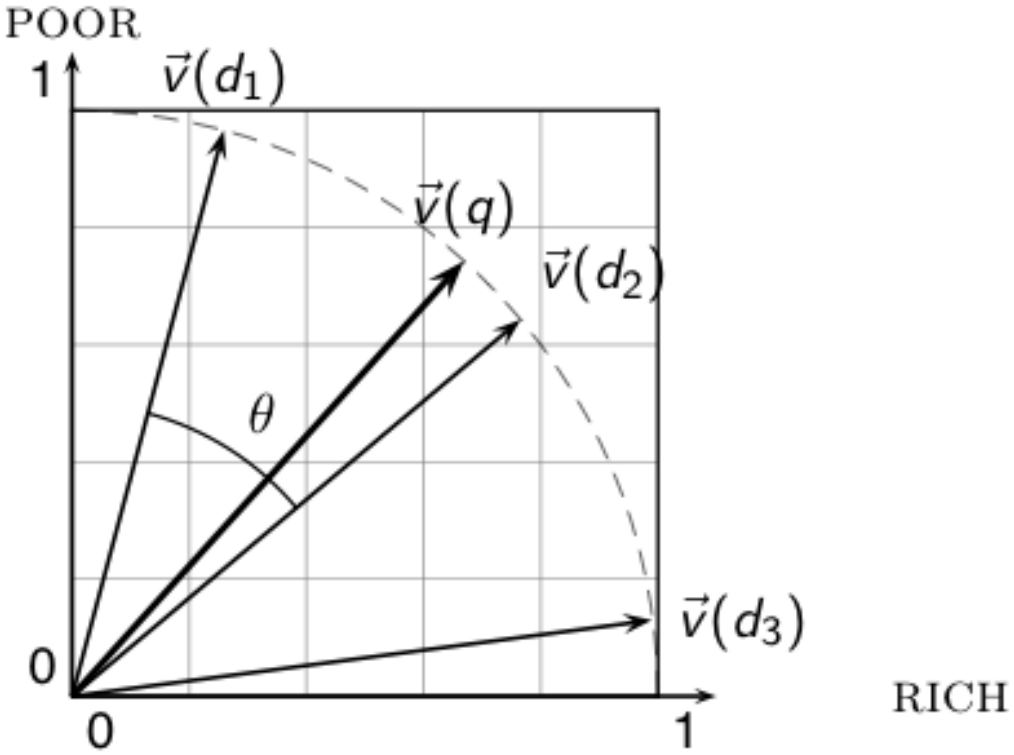
- 归一化向量的余弦相似度等价于它们的点积(或内积)

$$\cos(\vec{q},\vec{d}) = \vec{q}\cdot\vec{d} = \sum_i q_i \cdot d_i$$

如果 $\vec{q}$ 和 $\vec{d}$ 都是长度归一化后的向量

问题：文档的余弦相似度会不会为负值？

余弦相似度的图示



余弦相似度的计算样例

3本小说之间的相似度

(1) SaS(理智与情感): Sense and Sensibility
(2) PaP(傲慢与偏见): Pride and Prejudice
(3) WH(呼啸山庄): Wuthering Heights

词项频率tf

词项	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

余弦相似度计算

词项频率 tf

词项	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

对数词频（$1+\log_{10}tf$）

词项	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

为了简化计算，上述计算过程中没有引入idf

余弦相似度计算

对数词频$(1+\log_{10}tf)$

词项	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

对数词频的余弦归一化结果

词项	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

$\cos(SaS, PaP) \approx 0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$

$\cos(SaS, WH) \approx 0.79$

$\cos(PaP, WH) \approx 0.69$

$\cos(SaS, PaP) > \cos(SAS, WH) > \cos(PaP, WH)$

向量相似度计算算法

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d]+ = w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

夹角余弦相似度之外的其他相似度计算

- 考虑三个因素tf、idf、文档长度

- 为什么长度因素很重要？
  - 长度长的文档词项也多
  - 长度长的文档TF高

为什么长度因素很重要？

- 查询: *iphone 6s*
  - D1: iPhone 6s receives pre-orders on September 12.
  - D2: iPhone 6 has three color options.
  - D3: iPhone 6 has three color options. iPhone 6 has three color options. iPhone 6 has three color options.

- 需要对长度进行规一化(normalization)处理！

TFIDF权重计算的三要素

词项频率tf		文档频率df		归一化方法	
n(natural)	$tf_{t,d}$	n(no)	1	n(none)	1
l(logarithm)	$1+\log(tf_{t,d})$	t(idf)	$\log\frac{N}{df_t}$	c(cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\cdots+w_M^2}}$
a(augamented)	$0.5+\frac{0.5\times tf_{t,d}}{\max_t(tf_{t,d})}$	p(prob idf)	$\max\left\{0,\log\frac{N-df_t}{df_t}\right\}$	u(pivoted unique)	1/*u*(17.4.4节)
b(boolean)	$\begin{cases}1 \text{ if } tf_{t,d}>0\\0 \text{ otherwise}\end{cases}$			b(byte size)	$1/CharLength^a, a<1$
L(log ave)	$\frac{1+\log(tf_{t,d})}{1+\log(ave_{t\in d}(tf_{t,d}))}$				

TFIDF权重机制举例

- 对于查询和文档常常采用不同的权重计算机制
- 记法: ddd.qqq
- 例如: lnc.ltn
  - 文档: 对数tf，无idf因子，余弦长度归一化
  - 查询: 对数tf，idf，无归一化
- 文档当中不用idf结果会不会很差？
  - 查询: “best car insurance”
  - 文档: “car insurance auto insurance”

TFIDF 计算样例: Inc.ltn

查询: “best car insurance”. 文档: “car insurance auto insurance”.

word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

$$\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$$

1/1.92 ≈ 0.52

1.3/1.92 ≈ 0.68

最终结果 $\sum w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08$

向量空间模型小结

- 将查询表示成TFIDF权重向量
- 将每篇文档表示成同一空间下的 TFIDF权重向量
- 计算两个向量之间的某种相似度(如余弦相似度)
- 按照相似度大小将文档排序
- 将前K（如K =10）篇文档返回给用户

- 向量空间(检索)模型中包含一个向量表示模型，可以广泛用于其他领域，比如：判断论文抄袭、代码抄袭、图像相似度计算等等

Gerard Salton(1927-1995)

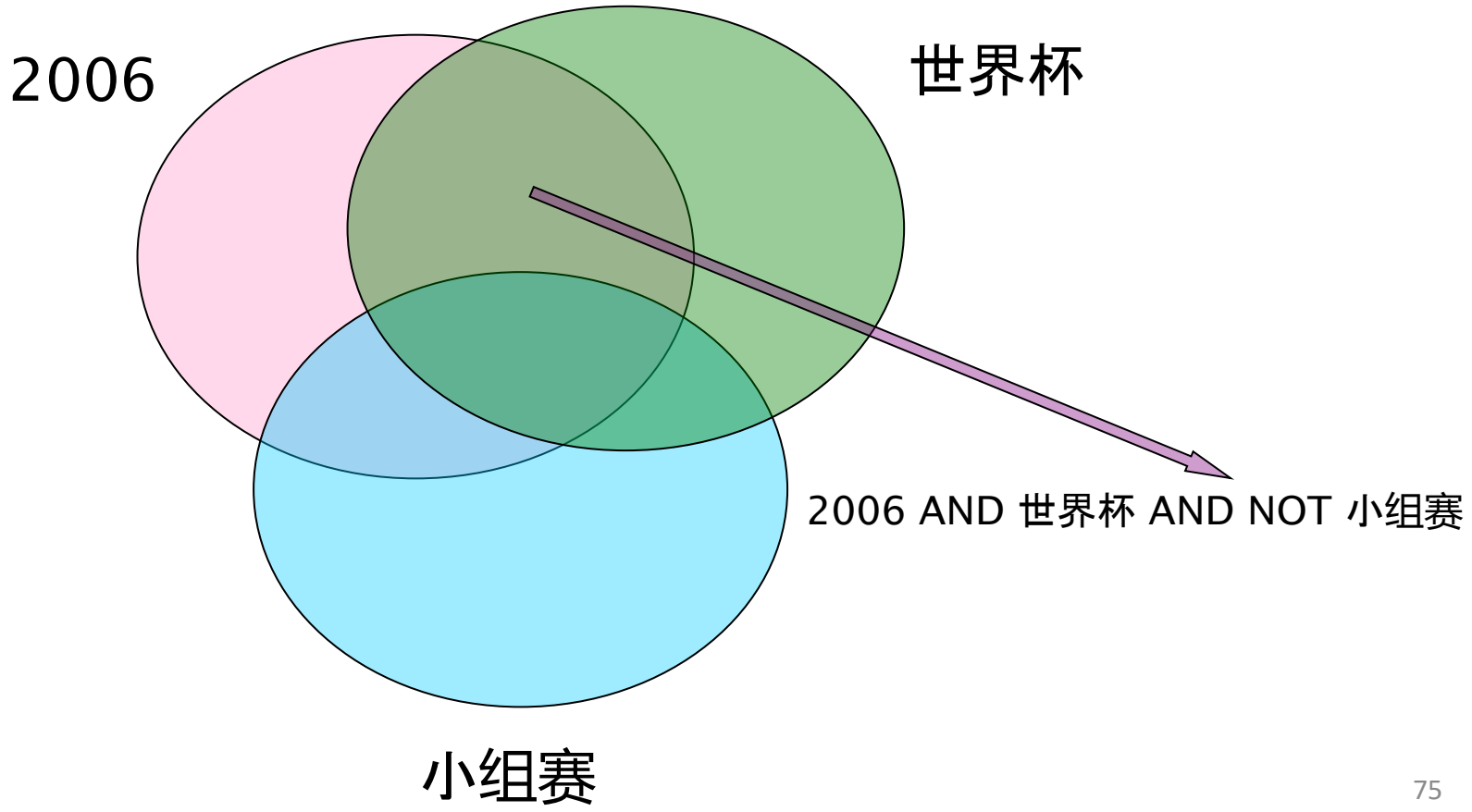
- 信息检索领域的奠基人之一，向量空间模型的完善者和倡导者，SMART系统的主要研制者，ACM Fellow
- 1958年毕业于哈佛大学应用数学专业，是Howard Aiken的关门博士生。 Howard Aiken是IBM第一台大型机ASCC的研制负责人。
- Salton是康奈尔大学计算机系的创建者之一。

提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. TFIDF权重计算
5. 向量空间模型
6. 布尔模型的扩展

方法二：基于模糊集的检索模型

布尔检索可以看成普通集合的交、并、非运算



普通集合

- 对于论域U上的一个子集A，可以定义函数：

$$\chi_A(x)=\begin{cases}1, \text{ if } x\in A\\ 0, \text{ if } x\notin A\end{cases}, \text{ 即}\chi_A\colon\ U\to\{0,1\}$$

  - 该函数刻画了论域U上的元素x到A的**隶属度**，当隶属度为1时，x属于A，当隶属度为0时，x不属于A，**该函数是二值函数**
  - 例子：“大于1的实数”用集合表示为 $A=\{x|x>1, x\in R\}$

模糊集合(Fuzzy Set)

- 模糊集合：简称模糊集，对于论域U上的一个“模糊子集”A，可以定义函数：

$$\mu_A: U \to [0,1]$$

  - 该函数刻画了论域U上的元素x到A的隶属度，函数值域为[0,1]。
  - 例子：“所有比1大得多的实数”用模糊集A可以定义为：

$$\mu_A(x) = \begin{cases} 0, & x \le 1 \\ \dfrac{1}{1+100/(x-1)^2}, & x > 1 \end{cases}$$

模糊集隶属函数的性质

- 模糊集合也存在非、并、交等运算，它们满足：

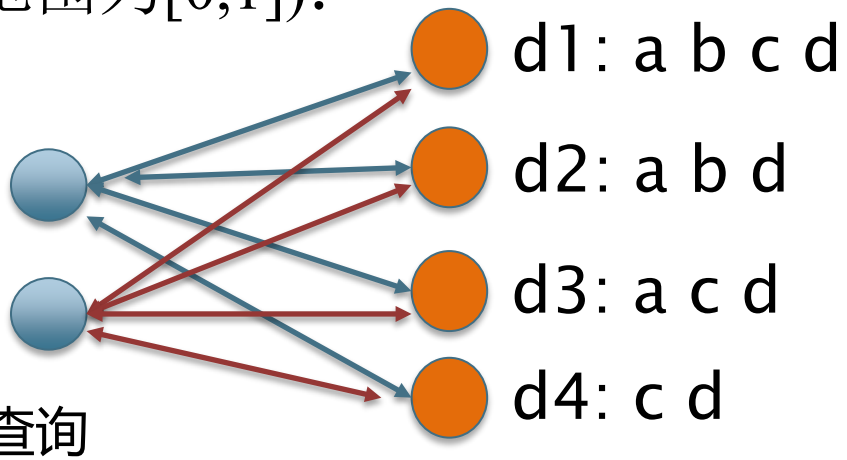
$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$
$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

基于模糊集的检索模型

- 对于文档集合$\{d_1,d_2,\ldots,d_n\}$，词项集合$\{t_1,t_2,\ldots,t_m\}$
- 第一步，定义词项之间的关系：建立一部相关性词典，定义所有词项和词项之间的相关性。t_i和t_l的相关性$c_{i,l}$可以通过下式计算(下式实际计算的是共现频率（如文档内共现频率），$c_{i,l}$取值范围为[0,1])：

$$c_{i,l}=\frac{n_{i,l}}{n_i+n_l-n_{i,l}}$$

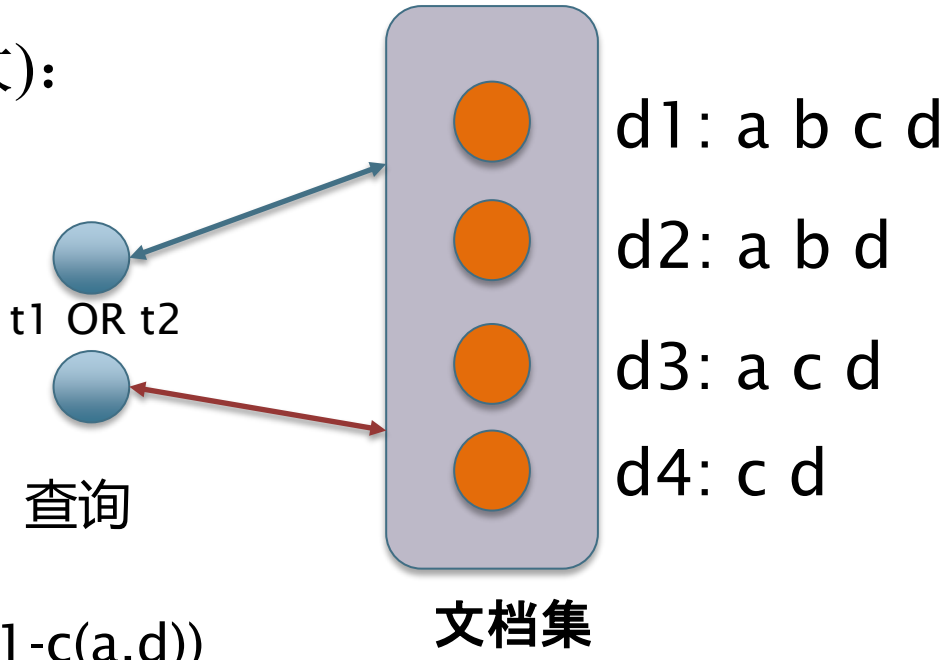
c(a,b)=？



基于模糊集的检索模型(续)

- 第二步，定义词项和文档之间的关系：对于一个t_i建立一个模糊集合A_i，其元素是所有文档集合，其中每个文档d_j对该模糊集A_i的**隶属度函数**通过下式计算(d_j中和t_i越相关的词项越多，函数值越大)：

$$\mu_{i,j} = 1 - \prod_{t_l \in d_j} (1 - c_{i,l})$$

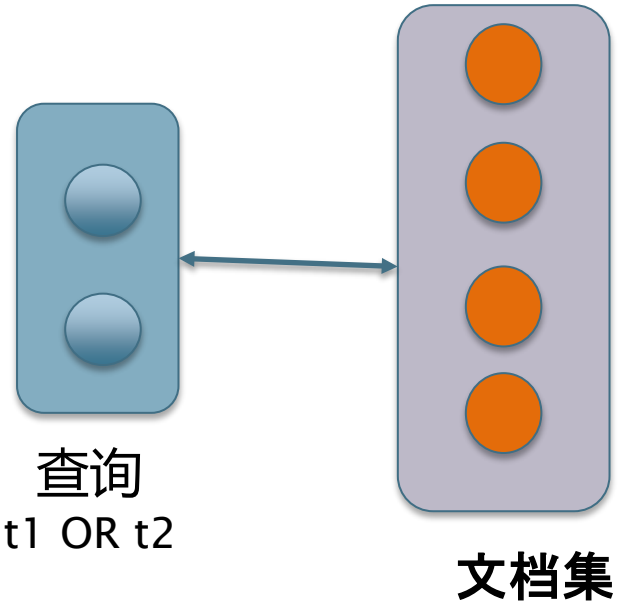


μ(a,d1)=1-(1-c(a,b)).(1-c(a,c)).(1-c(a,d))

基于模糊集的检索模型(续)

- 最后，定义查询q和文档d_j之间的关系，查询q对应一个模糊集合，求q与每个文档d_j的隶属度
  - 令词项t_i在文档d_j中的权值为$\mu_{ij}(0 \le \mu_{ij} \le 1)$，在查询中出现的词项$t_i$的权值为$\mu_{iq}$，认为$\mu_{ij} = \mu_{iq}$
  - 与第二步类似，定义q_l为q的DNF范式中的每一个合取表达式，则q与d_j的**隶属度函数**可以用下式计算

$$\mu_{q,j} = 1 - \prod_{q_l \in q} (1 - \mu_{l,j})$$



基于模糊集的检索模型(续)

- 可使用代数和与积代替min、max计算

布尔检索式	赋值公式
$sim(d_j, t_m \ and \ t_n)$	$min\{\mu_{m,j}, \mu_{n,j}\}$ $\mu_{m,j} \bullet \mu_{n,j}$
$sim(d_j, t_m \ or \ t_n)$	$max\{\mu_{m,j}, \mu_{n,j}\}$ $\mu_{m,j} + \mu_{n,j} - \mu_{m,j} \bullet \mu_{n,j}$
$sim(d_j, t_m \ and \ not \ t_n)$	$\mu_{m,j} \bullet (1 - \mu_{n,j})$

- 例子：对于布尔查询 $q=a \wedge (b \vee \neg c)$
- 可以求得其DNF范式
  - $q=abc \vee ab \neg c \vee a \neg b \neg c$
- 令cc1= abc,cc2= $ab \neg c$,$cc3= a \neg b \neg c$

$$\mu_{q,j} = \mu_{cc_1+cc_2+cc_3,j}$$
$$= 1 - \prod_{i=1}^{3}(1 - \mu_{cc_i,j})$$
$$= 1 - \left(1 - \mu_{a,j}\mu_{b,j}\mu_{c,j}\right)$$
$$\times (1 - \mu_{a,j}\mu_{b,j}(1 - \mu_{c,j}))$$
$$\times (1 - \mu_{a,j}(1 - \mu_{b,j})(1 - \mu_{c,j}))$$

举例（另一种计算方法）

- 对于标准布尔检索式，直接利用上表中公式计算
- 对于包含多个运算符的检索式，先处理深层的检索子式,再逐层递归处理

- 假设有一查询：$q = (t_1\ or\ t_2)\ and\ not\ t_3$
- 文档为d，其中$\mu_{1,d}$=0.7, $\mu_{2,d}$=0.2, $\mu_{3,d}$=0.1,计算文档d与q的相似度
- 内层检索子式：　　$sim(d, t_1\ or\ t_2) = 0.7$
- 外层检索式：

$$sim(d, (t_1\ or\ t_2)\ and\ not\ t_3 = sim(d, t_1\ or\ t_2) \times (1 - sim(d, t_3))$$
$$= 0.7 \times (1 - 0.1) = 0.63$$

- 最后得到文档d与查询q的相似度为0.63

基于模糊集的检索模型的优缺点

- 优点：
  - 克服原始布尔模型不能部分匹配的缺点
- 缺点：
  - 通常在模糊集研究领域涉及，在检索领域不流行
  - 缺乏大规模语料上的实验证实其有效性

方法三:扩展的布尔模型

- 基本想法：在布尔模型的基础上加入向量空间模型的一些思想，称Extended Boolean Model。
- 和向量空间模型一样，首先将每篇文档表示成N维空间上的一个点
- 比如：文档d_j中词项 k_x 对应的分量可以这样计算：

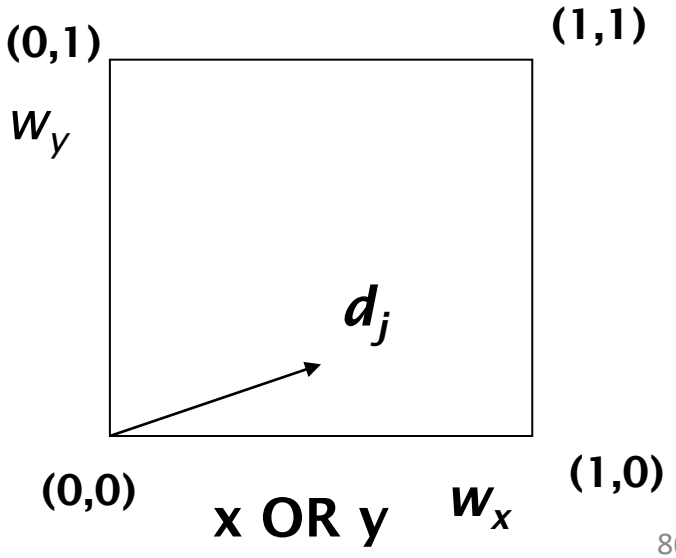
$$w_{x,j} = \frac{\mathrm{tf}_{x,j}}{\max_l f\,req_{l,j}} \times \frac{idf_x}{\max_i i\,df_i}$$

- $\mathrm{tf}_{x,j}$使用文档j中最大的tf进行归一化
- idf_x使用文档集中最高的idf进行归一化

扩展的布尔模型(续)

- OR情况处理：对于布尔查询$q=k_x$ OR k_y情况的处理，文档d_j和查询q的相关度：
  - 在(1,1)上的文档得分最高，(0,0)是尽量避免的点
  - 因此可以定义(0,0) 到$(w_{x,j}, w_{y,j})$的距离并归一化来计算文档和查询的相似度

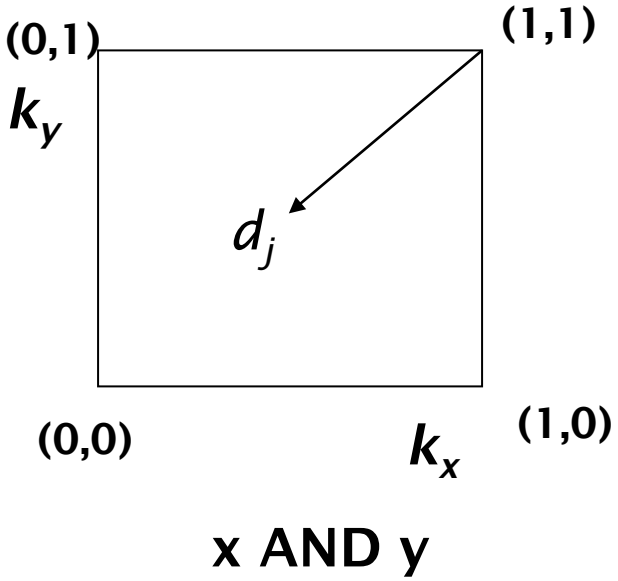
$$sim(q_{or}, d_j) = \sqrt{\frac{{w_{x,j}}^2 + {w_{y,j}}^2}{2}}$$



扩展的布尔模型(续)

- AND情况处理：对于布尔查询q=k_x AND k_y情况的处理，文档d_j和查询q的相关度：
  - (1,1)是最希望达到的点，离(1,1)越远相关度越低
  - 因此可以定义1减去(1,1) 到(w_x, w_y)的距离并归一化来计算相似度

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1 - w_{x,j})^2 + (1 - w_{y,j})^2}{2}}$$



扩展的布尔模型(续)

- 当文档中同时包含两个词项x和y时，AND、OR的文档评分一致
- 当文档中只出现1个词项时，OR得分大于AND得分：0.707 vs. 0.293

d_j contains	Term weights	Document Score	
		$sim(x \vee y, d_j)$	$sim(x \wedge y, d_j)$
$xx..xx$	$wt_{x,j} = 0.5$	0.353	0.209
$xx..xx$	$wt_{x,j} = 1.0$	0.707	0.293
$xx..yy$	$wt_{x,j} = wt_{y,j} = 0.5$	0.5	0.5
$xx..yy$	$wt_{x,j} = wt_{y,j} = 1.0$	1.0	1.0

扩展布尔模型(续)

- 对于包含m个查询词项的情况，可以对二维平面的情况进行扩展（使用x_i表示$w_{x_i,j}$）
  - 在多维空间上计算归一化欧式距离

$$sim(q_{or}, d) = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \ldots + {x_m}^2}{m}}$$

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1-x_1)^2 + (1-x_2)^2 + \ldots + (1-x_m)^2}{m}}$$

扩展布尔模型(续)

- p范数（p-norm） 模型不使用欧氏距离，而是使用p-距离(p是引入的参数，p>=1,也可以是无穷大)
- 查询分别为：

$$q_{and} = k_1 \wedge^p k_2 \wedge^p \ldots \wedge^p k_m$$

$$q_{or} = k_1 \vee^p k_2 \vee^p \ldots \vee^p k_m$$

- 这样，相似度计算公式为：

$$sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + \ldots + x_m^p}{m}\right)^{\frac{1}{p}}$$

$$sim(q_{and}, d_j) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + \ldots + (1-x_m)^p}{m}\right)^{\frac{1}{p}}$$

扩展布尔模型(续)

- 对于p-norm模型：
  - p=1时，相似度计算公式类似于向量模型

$$sim(q_{or}, d_j) = sim(q_{and}, d_j) = \frac{x_1 + x_2 + \ldots + x_m}{m}$$

  - p=∞时，类似于基于模糊集的检索模型

$$sim(q_{or}, d_j) = \max(x_i)$$

$$sim(q_{and}, d_j) = \min(x_i)$$

扩展布尔模型的优缺点

- 优点：提供了一个统一的框架，将向量空间模型、基于模糊集的模型都包括在一个框架内。
- 缺点：不够自然简洁，目前在信息检索领域使用较少。

本讲内容

- 对搜索结果排序(Ranking) : 为什么排序相当重要？
- 词项频率(Term Frequency, TF): 排序中的重要因子
- TFIDF 权重计算方法: 最出名的经典排序方法
- 向量空间模型(Vector space model): 信息检索中最重要的形式化模型之一 (其他模型还包括布尔模型和概率模型)

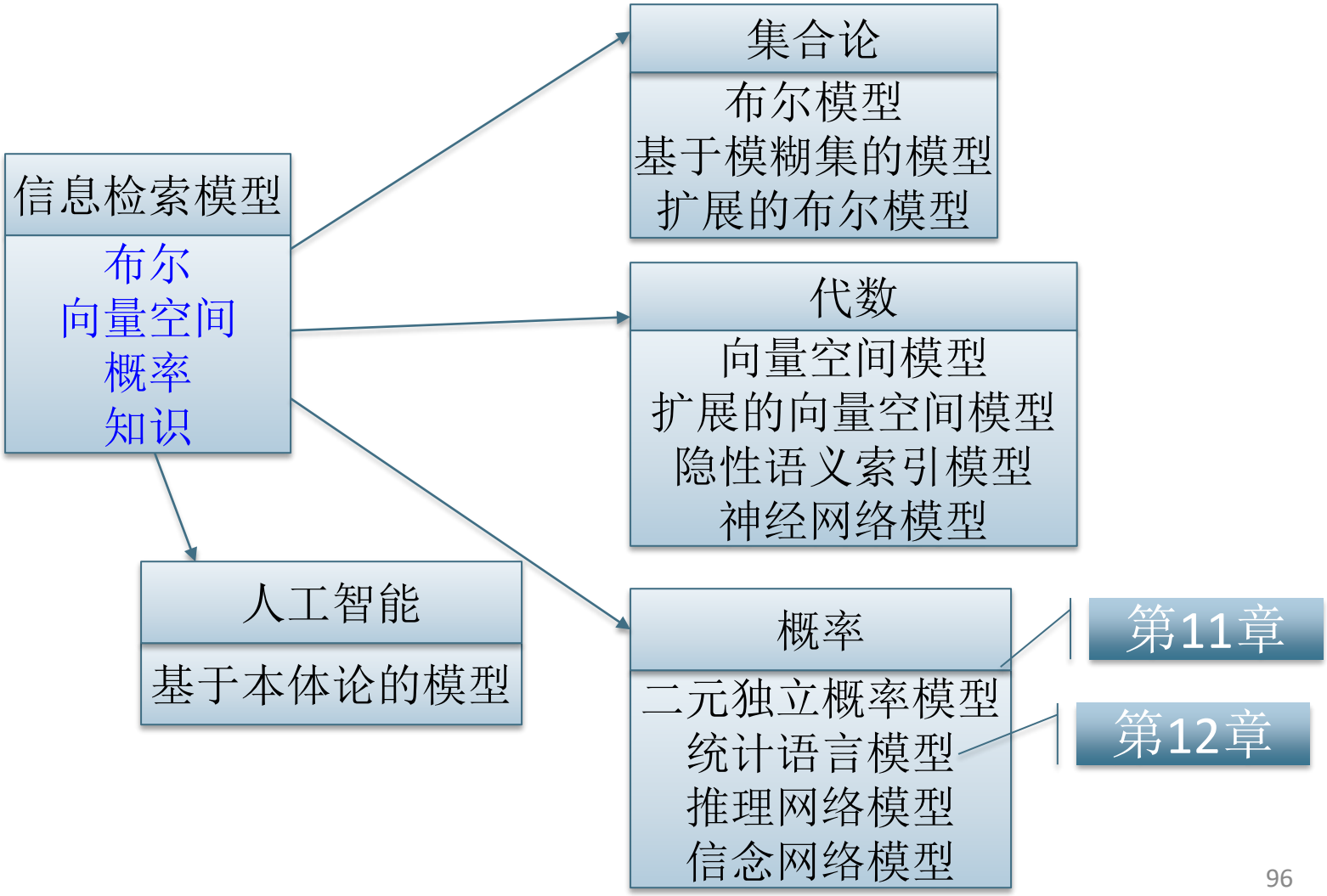
本讲小结

- 向量空间模型的三个步骤：词项选择、权重计算、相似度计算
- tf，反映词项在文档内部的权重，局部权重
- idf，反映词项在文档集(文档间)的权重，全局权重
- TFIDF，综合以后反映词在文档中的权重
- 长度因素，归一化

传统布尔模型和向量空间模型的优缺点

优缺点 / 模型	优　点	缺　点
传统布尔模型	检索式的结构化—用布尔算法明确的揭示了索引项之间的关系。	（1）不能对结果按相似度进行排序； （2）不能控制返回文档的数量； （3）不能进行相关性反馈。
向量空间模型	(1)检索结果的相关性排序； (2)可以控制输出结果的数量； (3)能够进行相关性反馈。	（1）理论上不够，基于直觉的经验性公式； （2）认为索引项相互独立，未能揭示词语之间的关系。 （例如 王励勤 乒乓球 的出现是不独立的）

信息检索模型分类


信息检索模型
布尔
向量空间
概率
知识
集合论
布尔模型
基于模糊集的模型
扩展的布尔模型
代数
向量空间模型
扩展的向量空间模型
隐性语义索引模型
神经网络模型
人工智能
基于本体论的模型
概率
二元独立概率模型
统计语言模型
推理网络模型
信念网络模型
第11章
第12章

参考资料

- 《信息检索导论》第6、7章
- Exploring the similarity space (Moffat and Zobel, 2005)
- Okapi BM25 (另外一种著名的权重计算方法，《信息检索导论》11.4.3节)
- TFIDF是一种熵？
  - 参考吴军《数学之美》

课后练习

- 有待补充