现代信息检索
Modern 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：王斌

http://ir.ict.ac.cn/~wangbin

*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. tf-idf权重计算
5. 向量空间模型

排序式检索(Ranked retrieval)

- 迄今为止，我们主要关注的是布尔查询
  - 文档要么匹配要么不匹配
- 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
- 然而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

布尔搜索的不足: 结果过少或者过多

- 布尔查询常常会倒是过少(=0)或者过多(>1000)的结果
- 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
- 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

排序式检索

- 排序式检索可以避免产生过多或者过少的结果
- 大规模的返回结果可以通过排序技术来避免
- 只需要显示前10条结果
- 不会让用户感觉到信息太多
- 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

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
- 后面我们将给出多种评分的方法

第一种方法: 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”
- 文档 “Caesar died in March”

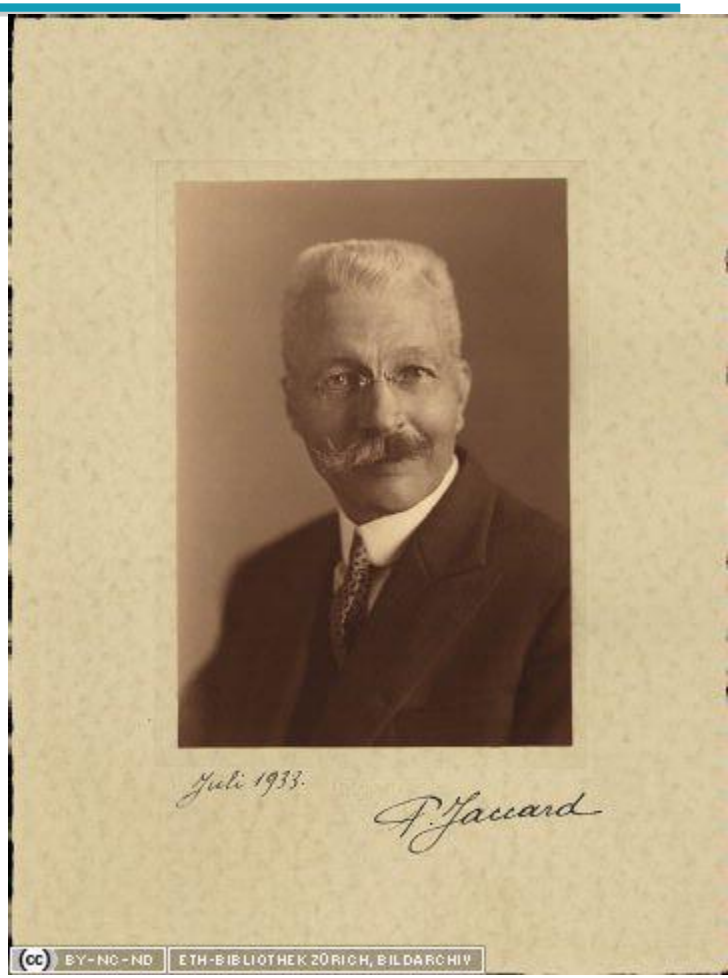
JACCARD(q, d) = 1/6

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
- 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，我们将使用 $|A \cap B|/\sqrt{|A \cup B|}$ (即余弦计算) 来代替 $|A \cap B|/|A \cup B|$，前者进行的长度归一化

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概念



提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. tf-idf权重计算
5. 向量空间模型

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(词频)

	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* are 的表示结果一样
- 这称为一个词袋模型(bag of words model)
- 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 $tf_{t,d}$ 是指*t* 在*d*中出现的次数
- 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
- 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 $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- 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”

提纲

1. 上一讲回顾
2. 排序式检索
3. 词项频率
4. tf-idf权重计算
5. 向量空间模型

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

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
- 文档频率指但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:

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

(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
- 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:

$$\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$$

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

- 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
- 对于单词项查询,idf对文档排序基本没有任何影响

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) : 文档集中出现的t词条的个数
- 词项*t*的文档频率: 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

tf-idf权重计算

- 词项的tf-idf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

tf-idf小结

- 词项t在文档d中的权重可以采用下次计算

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

- tf-idf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. tf-idf权重计算
5. 向量空间模型

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 $\in \{0, 1\}^{|V|}$

词频矩阵

	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 → 词频 → 权重矩阵

	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

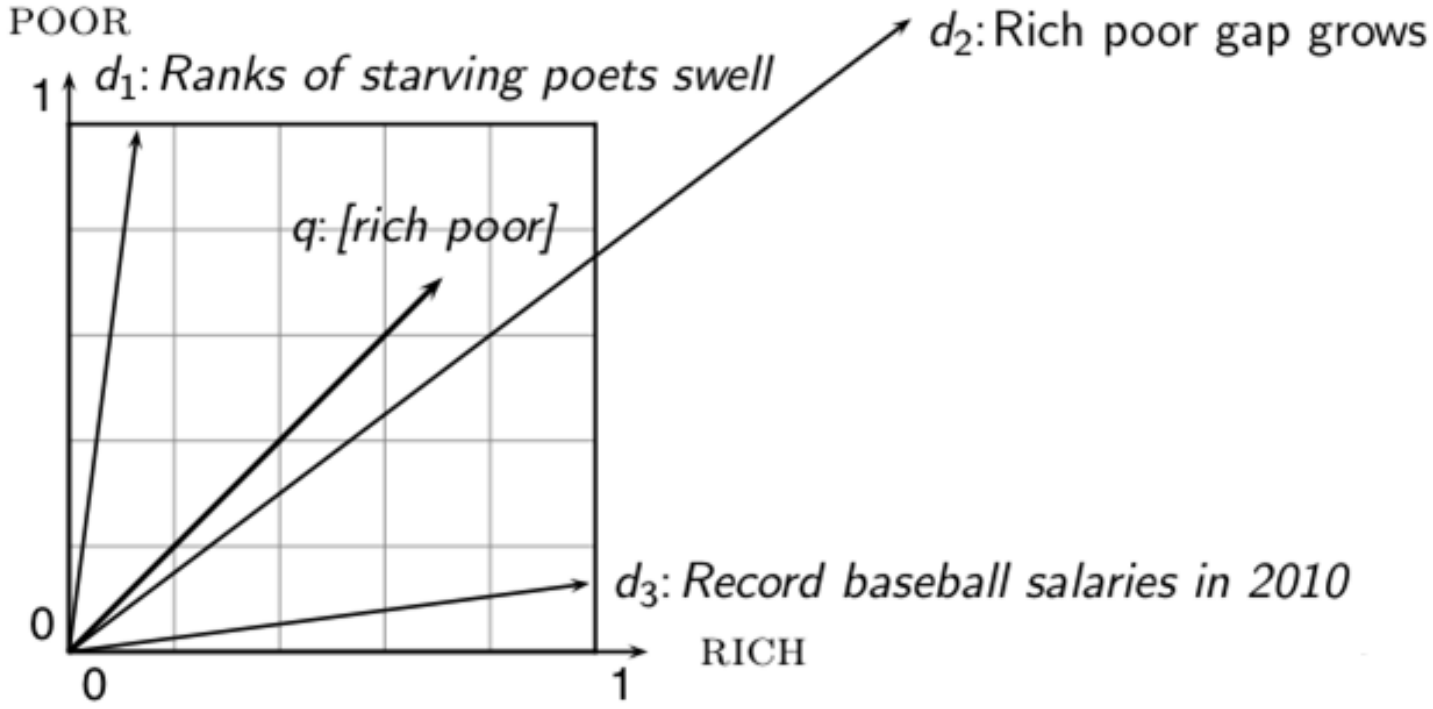
查询看成向量

- 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≈ 距离的反面
- 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。.

Questions about basic vector space setup?

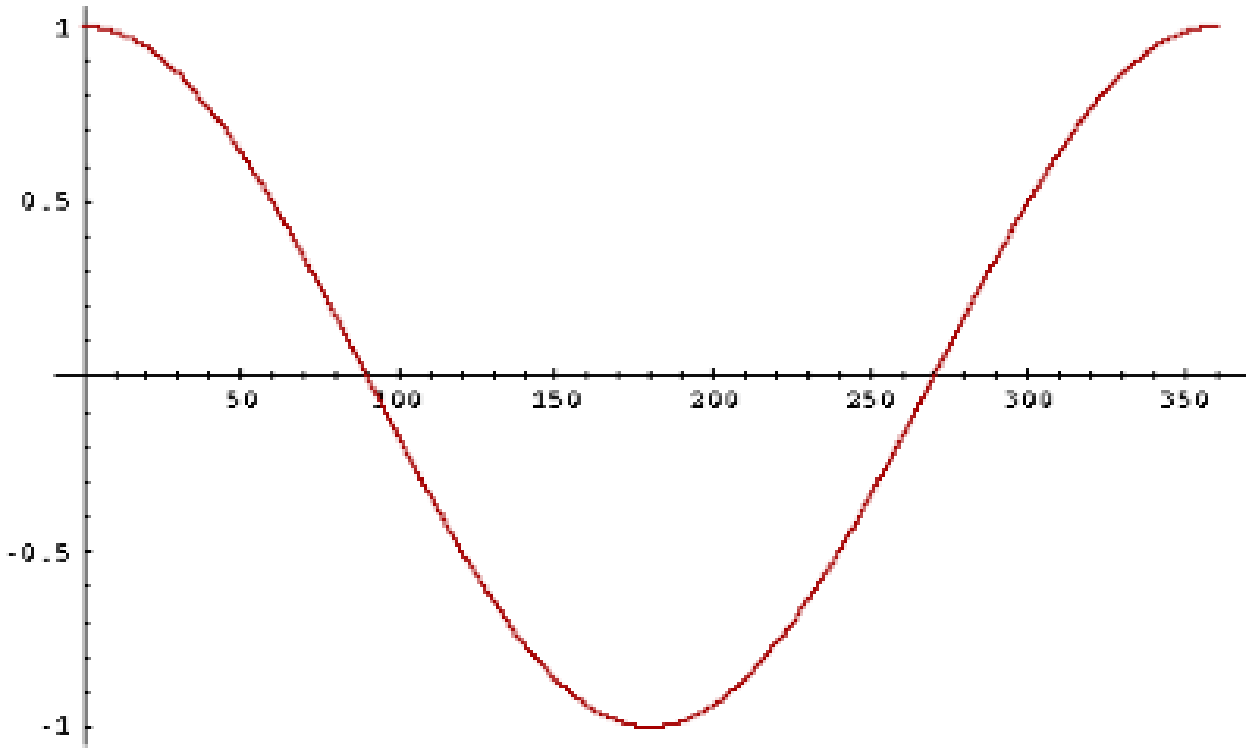
采用夹角而不是距离来计算

- 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tf-idf权重
- d_i是第i 个词项在文档d中的tf-idf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

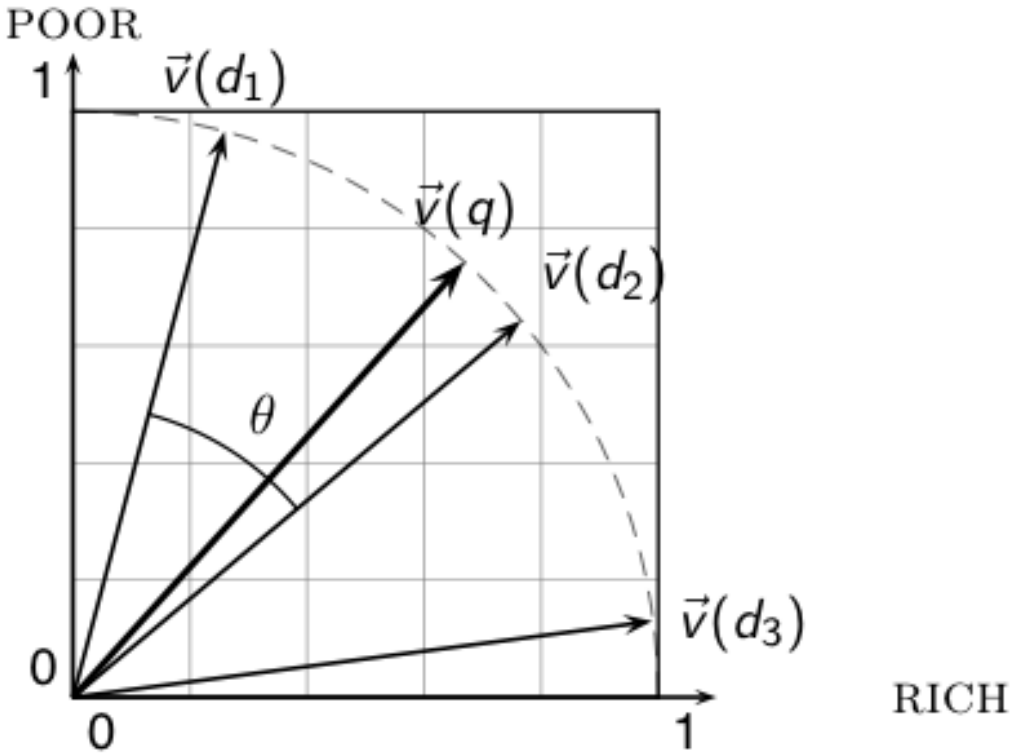
归一化向量的余弦相似度

- 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

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

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

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

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

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

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

余弦相似度计算算法

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** $Scores[d] += w_{t,d} \times w_{t,q}$
7 Read the array *Length*
8 **for each** d
9 **do** $Scores[d] = Scores[d]/Length[d]$
10 **return** Top K components of *Scores*[]

tf-idf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)	$\frac{1+\log(tf_{t,d})}{1+\log(ave_{t\in d}(tf_{t,d}))}$				

tf-idf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”

tf-idf 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tf-idf权重向量
- 将每篇文档表示成同一空间下的 tf-idf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

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。
- 是康奈尔大学计算机系的创建者之一。

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
- Tf-idf 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)

参考资料

- 《信息检索导论》第6、7章
- http://ifnlp.org/ir
  - 向量空间入门
  - 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)

课后练习

- 习题6-10
- 习题6-12
- 习题6-19
- 习题6-23