

自然语言处理
Information Retrieval

人工智能学院

大纲

- 数据库系统的局限（信息检索系统的动机）
- 信息检索（Information Retrieval, IR）
 - 索引
 - 检索模型：相似度计算
 - 评价
 - 其它的IR应用
- Web 搜索
 - 链接分析
 - PageRank

结构化数据

- 结构化数据一般指可以用二维表结构来逻辑表达实现的数据

| Employee | Manager | Salary |
|----------|---------|--------|
| Smith | Jones | 50000 |
| Chang | Smith | 60000 |
| Ivy | Smith | 50000 |

一般支持精确匹配的检索： e.g.,
Salary < 60000 AND Manager = Smith.

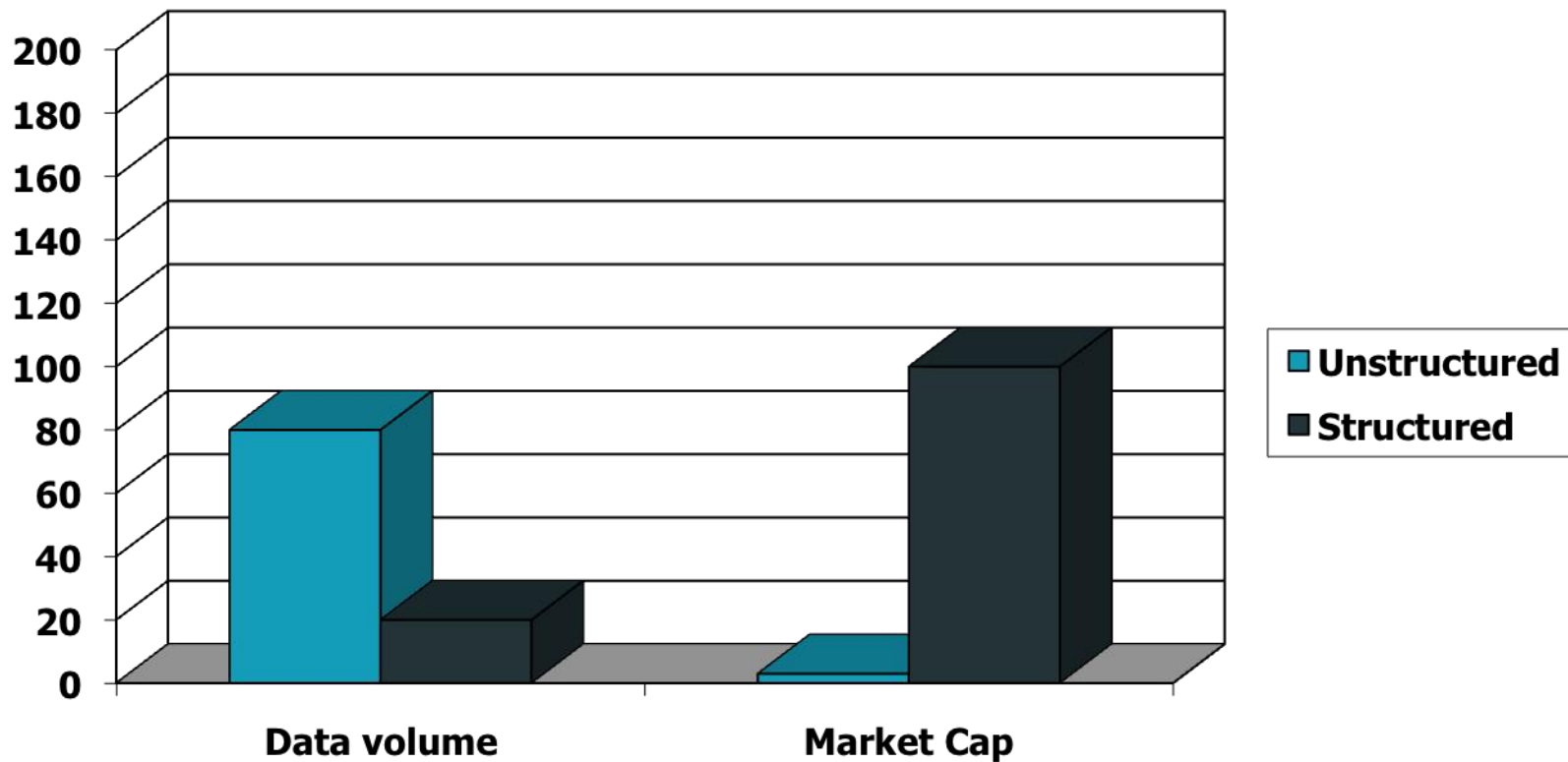
非结构化数据

- 非结构化数据：泛指文本、图片、图像和音频/视频信息等等
- 支持
 - 关键词检索（可以包含操作符）
 - e.g., 查找包含“Java” **AND** “线程”的文档
 - 更复杂的概念检索
 - e.g., 查找所有关于“澳门回归15周年”的网页

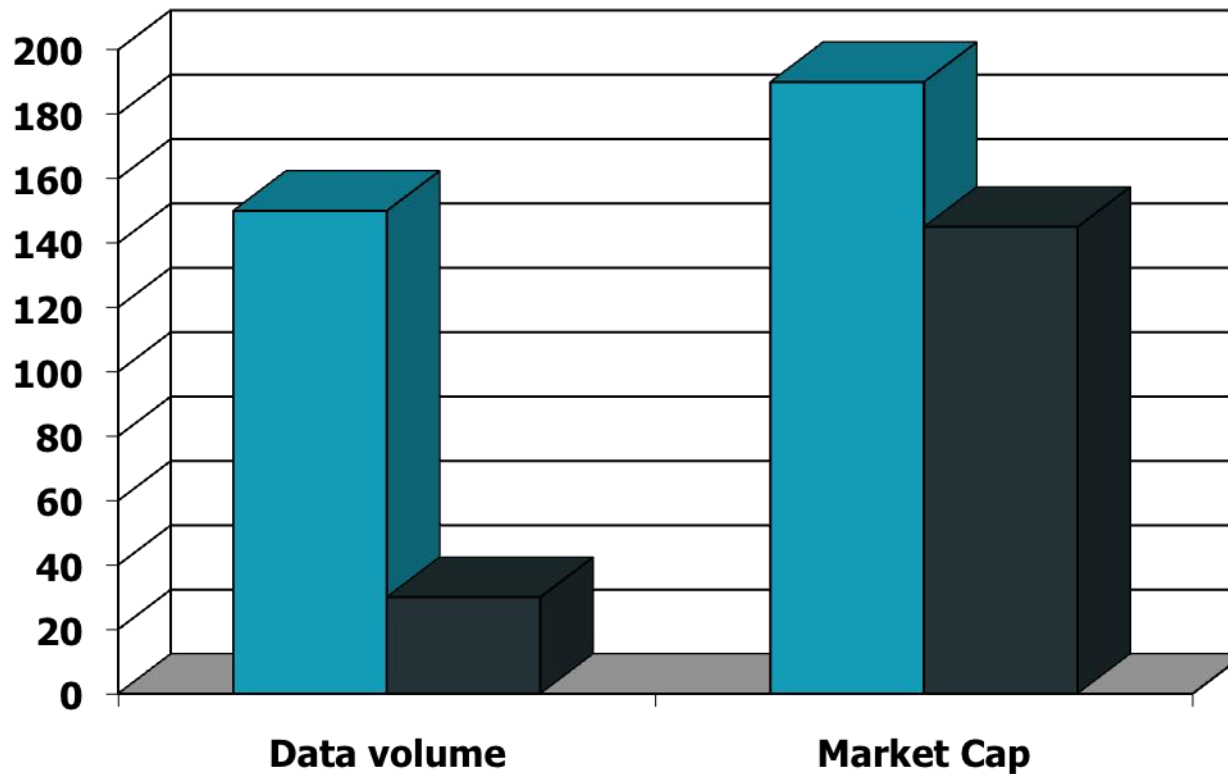
半结构化数据

- 严格意义上的非结构化数据并不存在
 - 比如：这张幻灯片由不同的区域构成，标题块、项目块
 - 再比如：一般文本中隐含可以结构化的语言结构信息（如文本的标题、段落、脚注等）
- 半结构数据的检索
 - E.g., 标题包含“数据” AND 项目块包含“检索”
- 甚或
 - 标题是关于“面向对象编程” AND 作者是“stro*rup”
 - 其中，* 是通配符

90年代的结构化数据与非结构化数据对比



当今结构化数据与非结构化数据对比



Google™

YAHOO!®

■ Unstructured
■ Structured

bing™

Ask™
.com

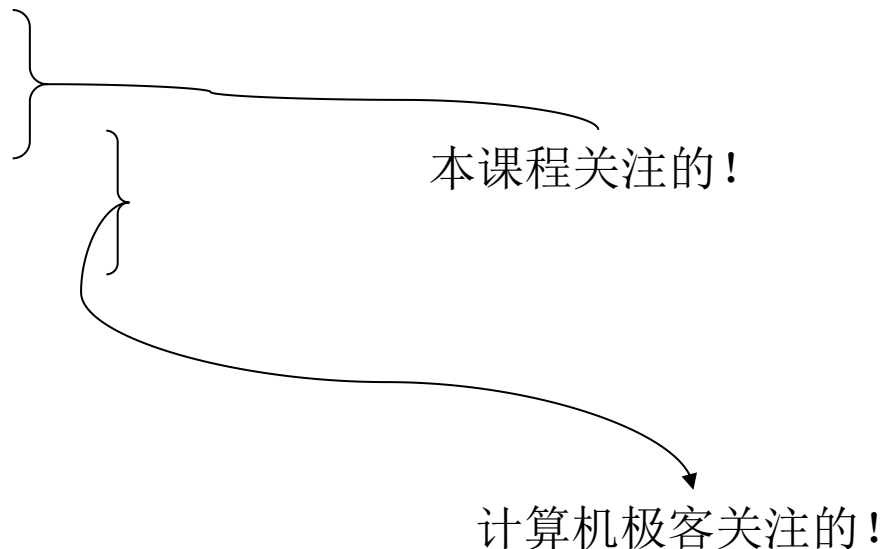
动机

- 数据搜索
 - 包含一组关键词的文档
 - 要求语义定义清晰
 - 细微差异可能导致整体搜索失败
- 信息检索 (IR, Information Retrieval)
 - 关于主题或主体的信息
 - 语义一般很模糊
 - 可以容忍小的错误
- IR系统
 - 深度解释信息内容
 - 根据相关度生成一个排序结果
 - 相关度的概念十分重要

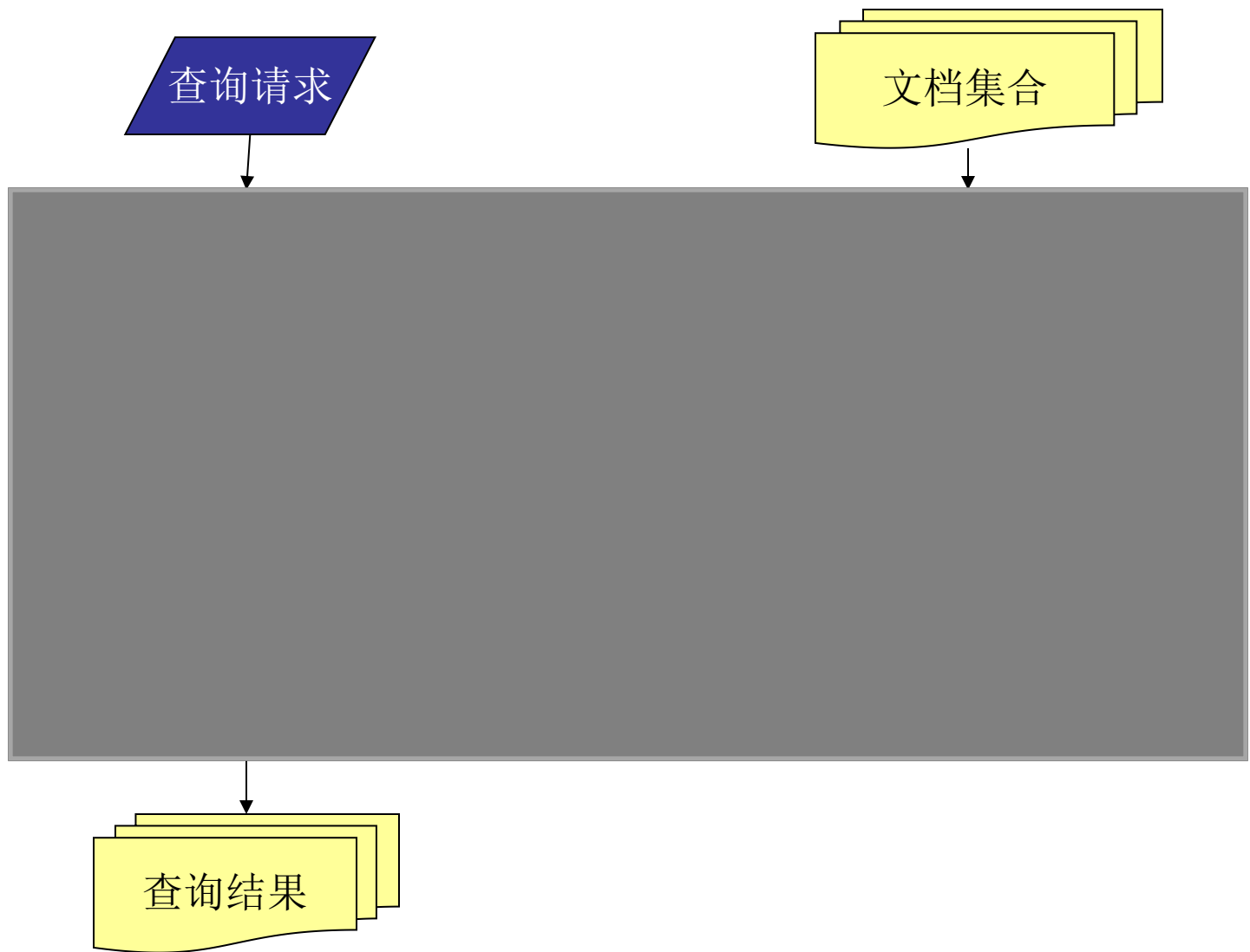
The Big Picture

- IR环境的三个基本构成：

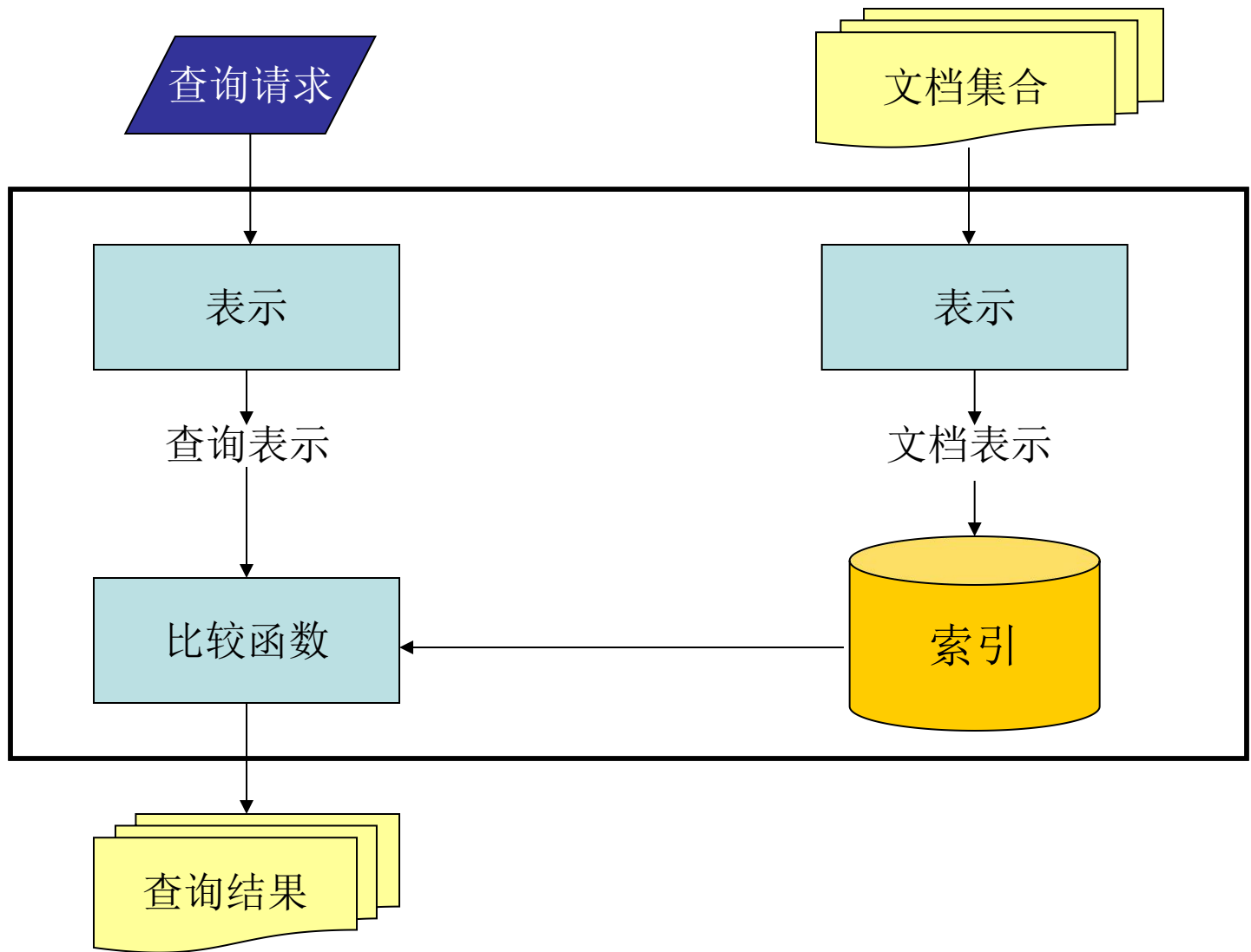
- 用户
- 检索
- 数据集



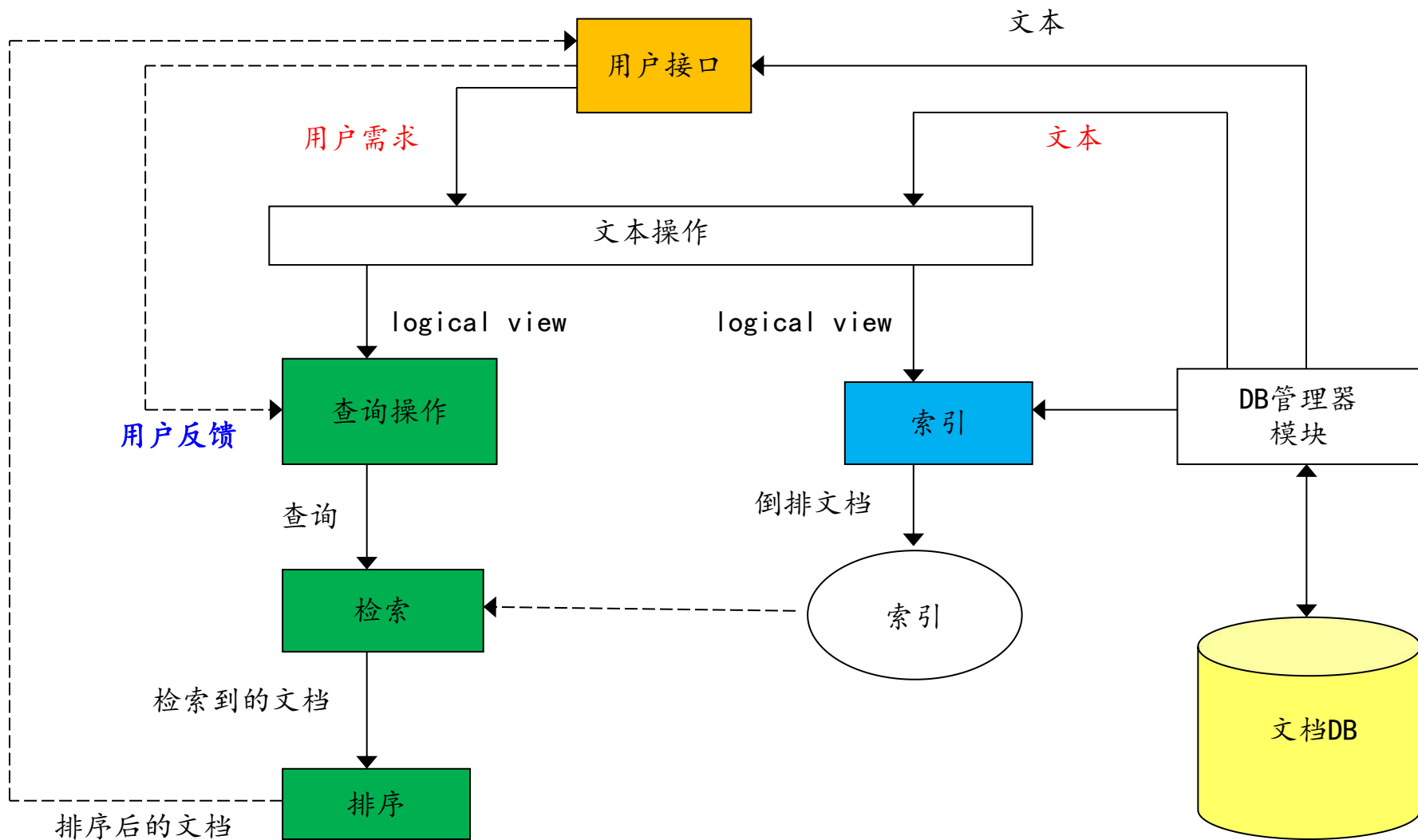
IR: 黑盒



IR: 内部结构



IR: 逻辑视图



IR: 核心问题

信息需求方（用户）



概念



查询项

信息生成方（作者）



概念



文档



查询项与文档是否表示同一个概念？

IR：任务分解

- 文档索引：为文档集合中的文档构建索引
- 查询操作：处理用户的查询请求
- 检索：相似度比较及排序
 - 找到和查询请求最匹配的文档集（相关文档集）
- 显示及优化：结果展示
 - E.g., 用户可能需要修改查询（反馈）、系统可能对查询结果进行组织（聚类等）

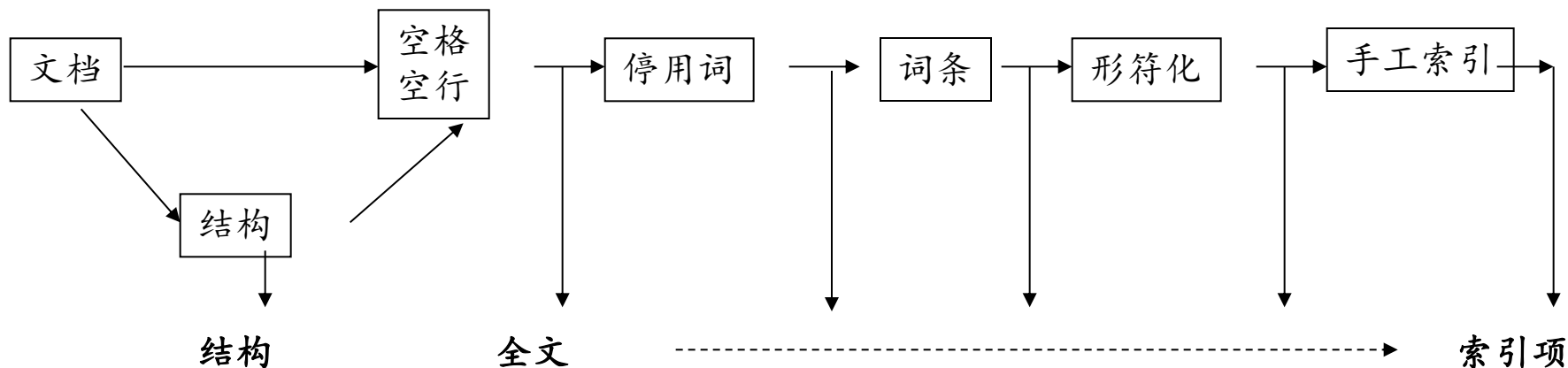
- What does each part do?

文档索引

- 动机：
- 一个例子：哪部莎士比亚的作品包含词条“**Brutus**”和“**Caesar**”，但不包括“**Calpurnia**”
(***Brutus AND Caesar NOT Calpurnia***)
 - 一种方式：扫描 (**grep**) 莎翁的作品，得到包含**Brutus** and **Caesar**的行，然后去除包含**Calpurnia**的行
 - 问题？
 - 慢（尤其是对于大规模语料）
 - ***NOT Calpurnia*** 是一个复杂操作
 - 扩展性差（例如，不能找到“**Romans** near **countrymen**”）
 - 不能对结果排序（返回最好的文档）

文档索引 (BoW方法)

• 文档的逻辑视图



• 文档索引

文档

This is a document in
information retrieval

索引

Document
Information
Retrieval
Is
This

索引：去停用词

- 依据预先定义的停用词词表，直接从词典里去除停用词
 - 停用词包含的语音信息较少： *the, a, and, to, be*
 - 停用词比例很大：最高频的前30个词可能占据索引表的大约30%
 - 高频词对不同概念或语义的区分能力往往较差
- 停用词词表：
 - *a, about, above, according, across, after, afterwards, again, against, albeit, all, almost, alone, already, also, although, always, among, as, at*
 - 啊吧呀罢了比方不仅不止不然的地等而且固然哈哼嘿哼哧或者即便即使既然假如进而咳啦哩嘛呢哦呸

索引：去停用词

- 但整体趋势是不去除停用词：
 - 一方面：索引表压缩技术可以使停用词占用的系统空间很小
 - 另一方面：往往需要停用词
 - 短语查询："King of Denmark"
 - 各种音乐、图书等实体的标题："Let it be", "To be or not to be"
 - “关系”查询："flights to London"
 - 一些看起来没用的停用词可能用于表达语义：动词“*can*”和名词“*can*”

索引：形符化

- 最简单的情形：去除后缀
 - *develop* → *develop*
 - *developing* → *develop*
 - *development* → *develop*
 - *developments* → *develop*
- 形符化的影响：
 - 对于英文：提升召回率（但损失了精确率）
 - 形符化实际效果往往比预想中差

索引：词条文档矩阵

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|-----------|----------------------|---------------|-------------|--------|---------|---------|
| Antony | 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 |

Brutus AND Caesar but NOT Calpurnia

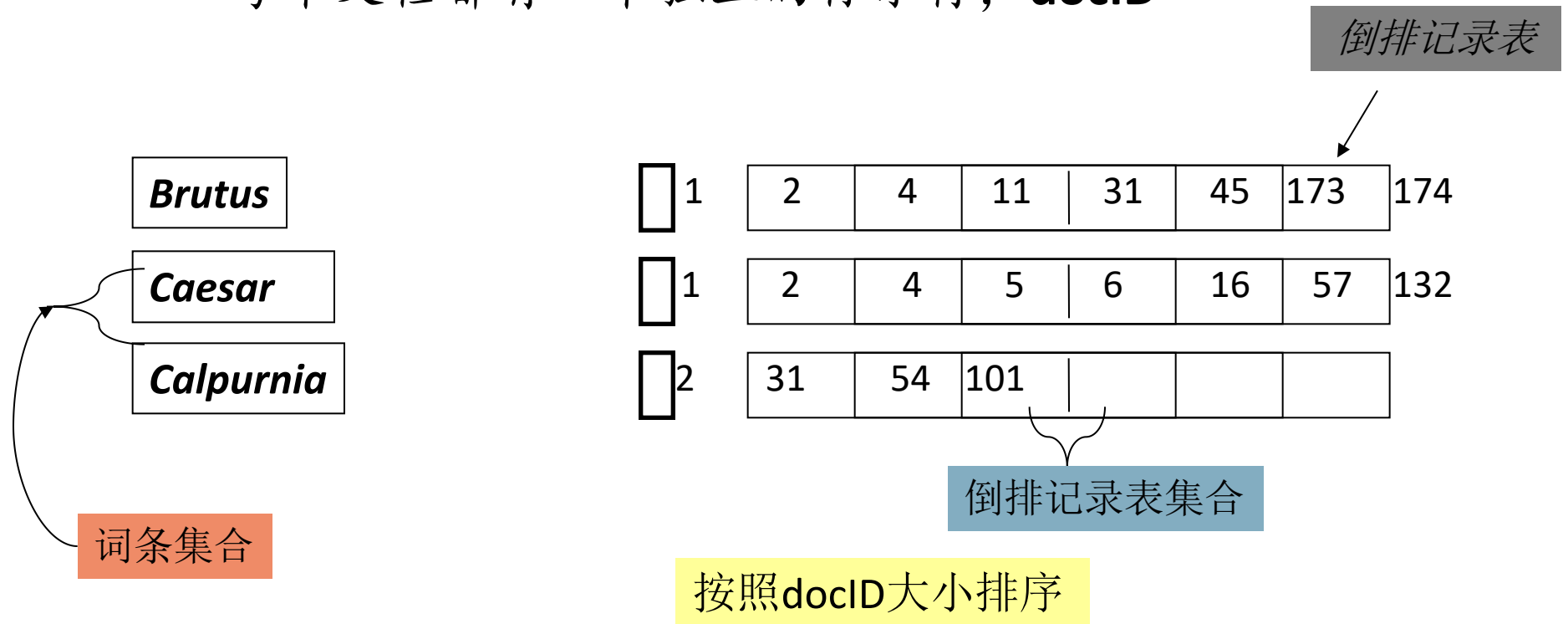
1 if **play** contains **word**, 0 otherwise

索引：词条文档矩阵

- 问题：文档词条矩阵非常稀疏，可能只包含很少的非零项
 - 50万个词条，100万篇文档，词条-文档矩阵有5000亿个元素（远远大于一台计算机内存的容量）
 - 设每个文档的平均长度为1000个词，则词条文档矩阵中对应大约10亿个1，剩余为0（大约占99.8%）
 - 极度稀疏！
- 如何更好的表示词条-文档的共现关系？
 - 只记录值为1的项？

倒排索引：索引表

- 对于一个词条 t ，保存一个记录出现该词条的所有文档的列表
 - 每个文档都有一个独立的标示符，**docID**



倒排索引：索引表

- 倒排索引的数据结构

- 定长数组

- 浪费，一些词在许多文档中出现，另一些词只在少数文档中出现
 - 如果在某个文档中插入一个词时，如何修改索引表也是一个问题

- 可变长度的倒排记录表

- 单链表便于文档的插入和更新
 - 可变长度的数组可以节省指针消耗的空间，同时由于采用连续的内存存储可以充分利用缓存技术提高访问速度

倒排索引：词条列表

(词条, 文档ID) 对构成的序列

Doc 1

I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.

Doc 2

So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious



| Term | docID |
|-----------|-------|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |
| | |
| | |
| | |

倒排索引：排序

依据字母顺序排序（然后依据文档顺序排序）



| Term | docID | Term | docID |
|-----------|-------|-----------|-------|
| I | 1 | ambitious | 2 |
| did | 1 | be | 2 |
| enact | 1 | brutus | 1 |
| julius | 1 | brutus | 2 |
| caesar | 1 | capitol | 1 |
| I | 1 | caesar | 1 |
| was | 1 | caesar | 2 |
| killed | 1 | caesar | 2 |
| i' | 1 | did | 1 |
| the | 1 | enact | 1 |
| capitol | 1 | hath | 1 |
| brutus | 1 | I | 1 |
| killed | 1 | I | 1 |
| me | 1 | i' | 1 |
| so | 2 | it | 2 |
| let | 2 | julius | 1 |
| it | 2 | killed | 1 |
| be | 2 | killed | 1 |
| with | 2 | let | 2 |
| caesar | 2 | me | 1 |
| the | 2 | noble | 2 |
| noble | 2 | so | 2 |
| brutus | 2 | the | 1 |
| hath | 2 | the | 2 |
| told | 2 | told | 2 |
| you | 2 | you | 2 |
| caesar | 2 | was | 1 |
| was | 2 | was | 2 |
| ambitious | 2 | with | 2 |
| | | | |
| | | | |
| | | | |

倒排索引：词条和倒排记录表

同一个文档中某个词条的多次出现合并在一起；

将词条和文档ID分开，
词条存储在词典
(dictionary) 中，每个
词项都对应一个倒排索引表 (posting)；

加入文档频次信息

Why doc frequency?

| Term | docID |
|-----------|-------|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |



| term | doc. freq. | → | postings lists |
|-----------|------------|---|----------------|
| ambitious | 1 | → | [2] |
| be | 1 | → | [2] |
| brutus | 2 | → | [1] → [2] |
| capitol | 1 | → | [1] |
| caesar | 2 | → | [1] → [2] |
| did | 1 | → | [1] |
| enact | 1 | → | [1] |
| hath | 1 | → | [2] |
| i | 1 | → | [1] |
| i' | 1 | → | [1] |
| it | 1 | → | [2] |
| julius | 1 | → | [1] |
| killed | 1 | → | [1] |
| let | 1 | → | [2] |
| me | 1 | → | [1] |
| noble | 1 | → | [2] |
| so | 1 | → | [2] |
| the | 2 | → | [1] → [2] |
| told | 1 | → | [2] |
| you | 1 | → | [2] |
| was | 2 | → | [1] → [2] |
| with | 1 | → | [2] |

创建倒排索引

文档集合



Friends, Romans, countrymen.

词条序列

语言分析

friend

roman

countryman

索引

friend | 10

roman | 7

countryman | 6

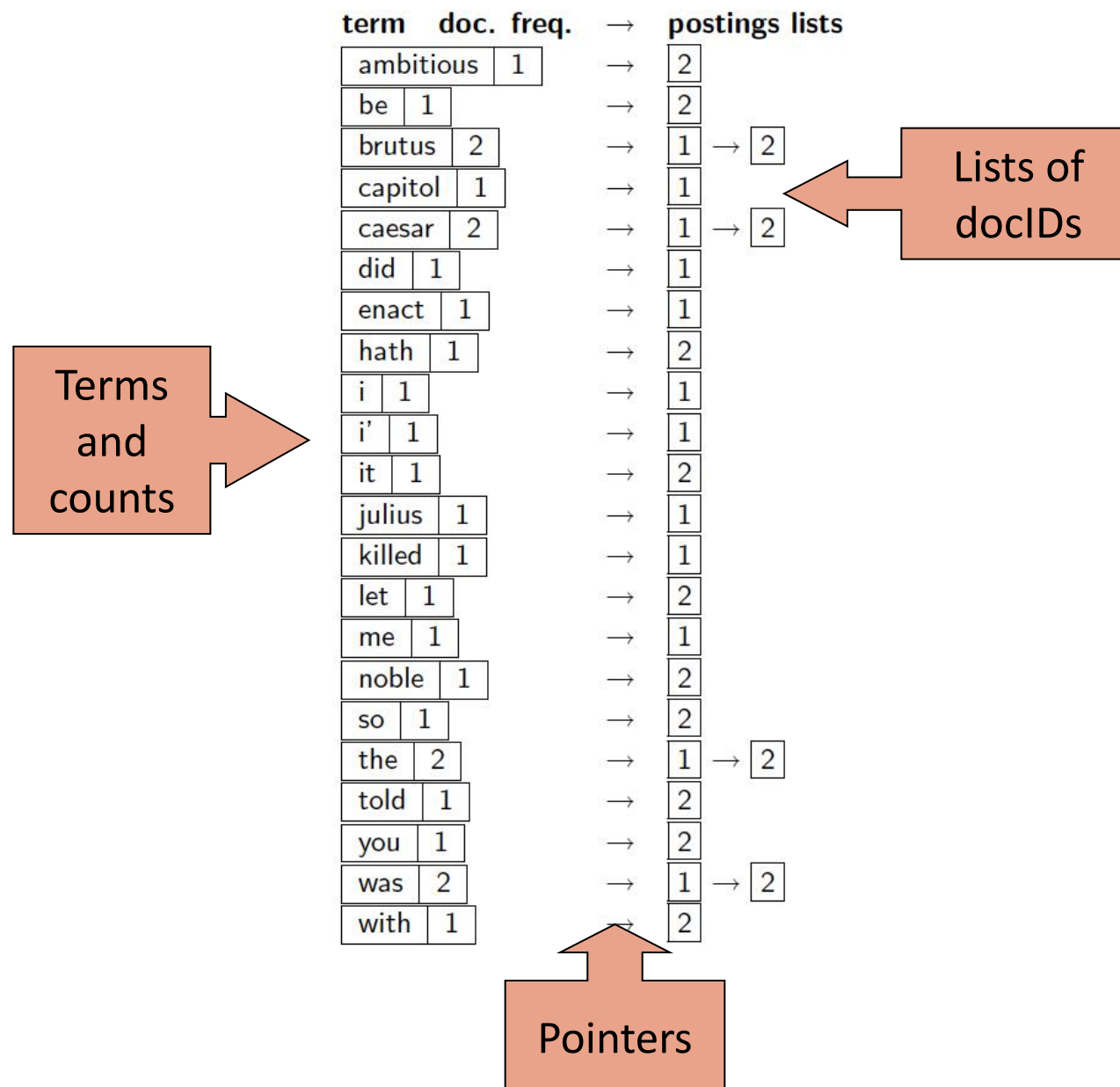
2 → 4 →

1 → 2 →

13 → 16

倒排索引

倒排索引的存储



相似度模型

- 布尔模型 (Boolean model)
- 向量空间模型 (Vector space model)
- 概率模型 (Probabilistic model)
- 语言模型 (Language model)
-

布尔模型

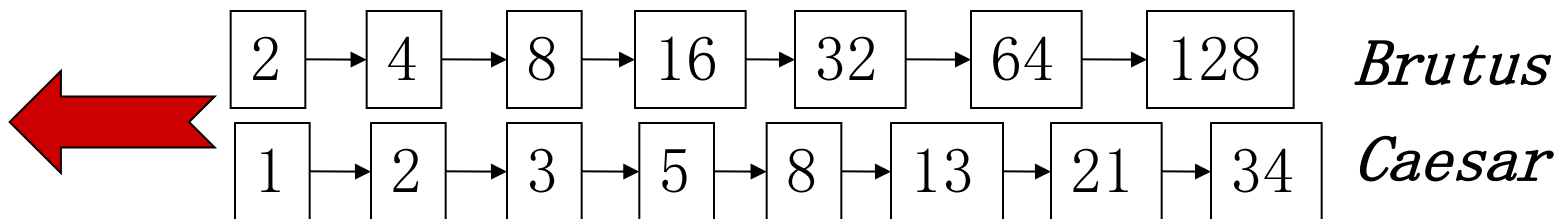
- 查询：“Which plays of Shakespeare contain the words **Brutus AND Caesar** but **NOT Calpurnia**?”
- 首先为每个词条创建一个0/1 词条-文档向量
- 为完成检索，取出**Brutus, Caesar** 和 **Calpurnia** (位取反)对应的向量 → 进行位与 (bitwise AND)
- $110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$

布尔模型

- AND操作

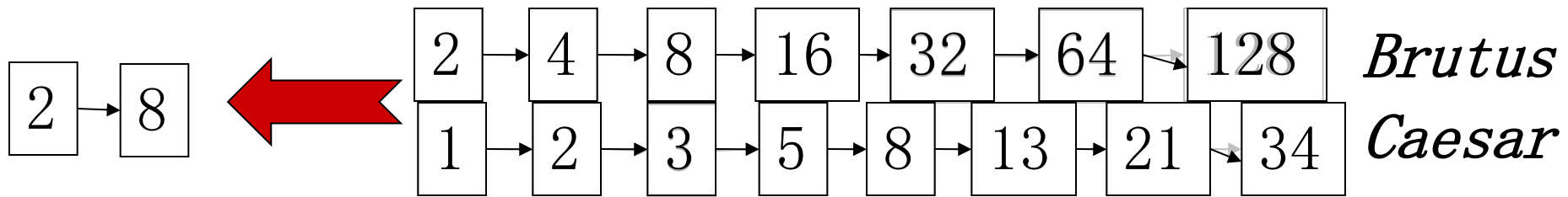
- E.g., ***Brutus AND Caesar***

- 在词典中查找 ***Brutus***
 - 返回其对应的倒排记录表
 - 在词典中查找 ***Caesar***
 - 返回其对应的倒排记录表
 - “合并”两个记录表（取交集）



布尔模型

- **交集：**将两个位置指针同时在两个列表中后移，比较两个指针所指向的DocID
 - 如果一样，则输出DocID到结果列表，然后指针同时后移一位
 - 如果不一样，则较小DocID对应的指针后移



如果两个列表的长度分别为 x 和 y ，则该操作的时间复杂度为 $O(x+y)$

关键：倒排记录表按照DocID排序

两个倒排记录表求交集 (“merge” 算法)

```
INTERSECT( $p_1, p_2$ )  
  1   $answer \leftarrow \langle \rangle$   
  2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$   
  3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$   
  4      then  $\text{ADD}(answer, \text{docID}(p_1))$   
  5           $p_1 \leftarrow \text{next}(p_1)$   
  6           $p_2 \leftarrow \text{next}(p_2)$   
  7      else if  $\text{docID}(p_1) < \text{docID}(p_2)$   
  8          then  $p_1 \leftarrow \text{next}(p_1)$   
  9          else  $p_2 \leftarrow \text{next}(p_2)$   
 10 return  $answer$ 
```

布尔模型

- 查询优化：
 - 哪种查询顺序最优？
 - 考虑 t 个词条的 AND 构成的查询
 - 对于任何一个词条，返回其倒排索引记录，然后执行 AND 操作

| | | | | | | | | |
|------------------|----|----|---|----|----|----|-----|----|
| <i>Brutus</i> | 2 | 4 | 8 | 16 | 32 | 64 | 128 | |
| <i>Calpurnia</i> | 1 | 2 | 3 | 5 | 8 | 16 | 21 | 34 |
| <i>Caesar</i> | 13 | 16 | | | | | | |

Query: *Brutus AND Calpurnia AND Caesar*

布尔模型

- 按照文档频率升序进行查询：
 - 从最短的倒排记录表开始，则所有中间结果的大小都不会超过最短的倒排记录表

在词典里保存文档频率的原因
!

Brutus

| | | | | | | | |
|---|---|---|----|----|----|-----|--|
| 2 | 4 | 8 | 16 | 32 | 64 | 128 | |
|---|---|---|----|----|----|-----|--|

Calpurnia

| | | | | | | | |
|---|---|---|---|---|----|----|----|
| 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |
|---|---|---|---|---|----|----|----|

Caesar

| | | | | | | | |
|----|----|--|--|--|--|--|--|
| 13 | 16 | | | | | | |
|----|----|--|--|--|--|--|--|

执行查询： (*Caesar AND Brutus*) *AND Calpurnia*.

布尔模型

- 每个索引词条要么 **存在** 要么 **不存在**
- 文档要么 **相关** 要么 **不相关**（不排序）
 - 尽管可以扩展到短语查询，比如 "***stanford university***"
 - 但是对于句子 "*I went to university at Stanford*" 很难处理
- 优点
 - Simple
- 缺点
 - 没有引入排序概念（精确匹配）
 - 所有的索引词条具有相同的重要性

向量空间模型 (VSM)

- 查询及文档都被表示为索引词条的向量
(**vectors of index terms**)
- 通过比较两个向量的相似度计算相关程度
(e.g., **COSINE similarity**)
 - 引入基于相似度的排序

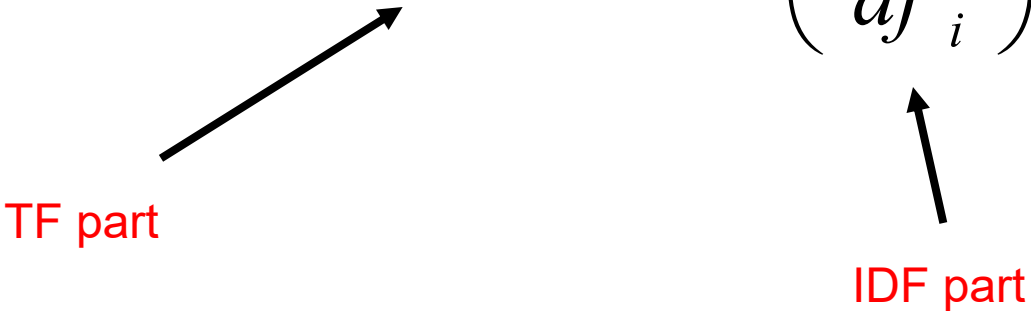
VSM: Binary \rightarrow count \rightarrow weight matrix

- 查询和文档被表示为实值向量，词条权重可采用 *tf-idf* 权重 ($tf-idf \in \mathbb{R}^{|V|}$)

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|-----------|----------------------|---------------|-------------|--------|---------|---------|
| Antony | 5.25 | 3.18 | 0 | 0 | 0 | 0.35 |
| Brutus | 1.21 | 6.1 | 0 | 1 | 0 | 0 |
| Caesar | 8.59 | 2.54 | 0 | 1.51 | 0.25 | 0 |
| Calpurnia | 0 | 1.54 | 0 | 0 | 0 | 0 |
| Cleopatra | 2.85 | 0 | 0 | 0 | 0 | 0 |
| mercy | 1.51 | 0 | 1.9 | 0.12 | 5.25 | 0.88 |
| worser | 1.37 | 0 | 0.11 | 4.15 | 0.25 | 1.95 |

向量空间模型 (VSM)

- TF-IDF

$$tfidf_{i,k} = f_{i,k} * \log \left(\frac{N}{df_i} \right)$$


TF part

IDF part

The diagram illustrates the TF-IDF formula. An arrow points from the text 'TF part' to the term $f_{i,k}$ in the formula. Another arrow points from the text 'IDF part' to the logarithmic term $\log \left(\frac{N}{df_i} \right)$.

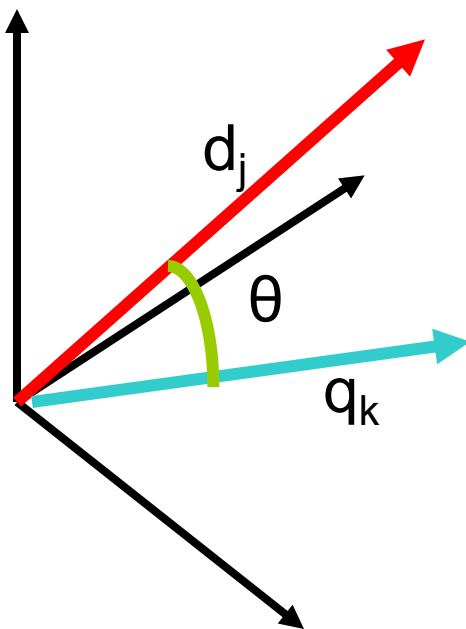
向量空间模型 (VSM)

- TF-IDF 权重的不同变种

| Term frequency | | Document frequency | | Normalization | |
|----------------|---|--------------------|---|--------------------|--|
| 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) | $\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$ |
| a (augmented) | $0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$ | p (prob idf) | $\max\{0, \log \frac{N - df_t}{df_t}\}$ | u (pivoted unique) | $1/u$ |
| b (boolean) | $\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$ | | | b (byte size) | $1/CharLength^\alpha$, $\alpha < 1$ |
| L (log ave) | $\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$ | | | | |

向量空间模型 (VSM)

- Cosine相似度



Cosine:

$$\text{sim}(\vec{Q}, \vec{D}_i) = \frac{\sum_{j=1}^N w_{qj} * w_{ij}}{\sqrt{\sum_{j=1}^N (w_{qj})^2 * \sum_{j=1}^N (w_{ij})^2}}$$

概率模型

- 在布尔模型或向量空间模型中，对于匹配虽有形式化定义，但在语义上并不精确：
 - 给定一个查询，IR系统只能得到信息需求的**非确定性**理解
 - 给定查询表示和文档表示，IR系统只能给出文档内容和查询是否相关的一个**非确定性**推测
- 概率论可以为这种非确定性推理提供一个基本的理论
 - 概率检索模型可以计算文档和查询相关的可能性

概率模型

- 通过概率的方法将查询和文档联系起来
- 定义3个随机变量：
 - 相关度 $R=\{0,1\}$
 - 查询 $Q=\{q_1, q_2, \dots\}$
 - 文档 $D=\{d_1, d_2, \dots\}$
 - 则可以通过计算条件概率 $P(R=1 | Q=q, D=d)$ 来度量文档和查询的相关度
 - 对于同一个 q , $P(R=1 | q, d)$ 可以简记为 $P(R=1 | d)$

概率模型

- 概率排序原理 (Probability Ranking Principle, PRP)
 - $p(R=1|d)$ – 文档 d 与给定查询相关的概率
 - $p(R=1), p(R=0)$ – 检索到相关/不相关文档的先验概率
 - $p(d|R=1), p(d|R=0)$ – 检索到的相关/不相关文档为 d 的概率

$$p(R=1|d) = \frac{p(d|R=1)p(R=1)}{p(d)}$$

$$p(R=0|d) = \frac{p(d|R=0)p(R=0)}{p(d)}$$

- 贝叶斯最优决策原理: d 是 q 的相关文档, 当且仅当 $P(R=1|d, q) > P(R=0|d, q)$
- 最终根据 $P(R=1|d, q)$ 的大小, 完成对文档的排序

概率模型

- 概率模型包括一系列模型
 - 回归模型(Logistic Regression)
 - 二值独立概率模型(Binary Independence Model)
 - Okapi BM25模型
 - 贝叶斯网络模型
 -
- 1998出现的基于统计语言建模的信息检索模型本质上也是概率模型的一种

概率模型

- 二值独立概率模型(Binary Independence Model, BIM)

- 二值：指文档和查询被表示为词条出现与否的0-1向量，用 \vec{q} 表示 q ， \vec{x} 表示 d
- 定义优势率（Odd，反应概率变化的放大器）：

$$\begin{aligned} O(R|\vec{x}, \vec{q}) &= \frac{P(R=1|\vec{x}, \vec{q})}{P(R=0|\vec{x}, \vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1, \vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0, \vec{q})}{P(\vec{x}|\vec{q})}} \\ &= \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})} \cdot \frac{P(\vec{x}|R=1, \vec{q})}{P(\vec{x}|R=0, \vec{q})} \end{aligned}$$

- 其中， $P(\vec{x}|R=1, q)$ 、 $P(\vec{x}|R=0, q)$ 分别表示在相关和不相关情况下生成文档 x 的概率
- 显然，排序函数与 $P(x|R=1, q)$ 成正比

概率模型

- 对于给定的 q , $P(R=1|q)/P(R=0|q)$ 是常量, 记为 $O(R|q)$
- 另, 根据朴素贝叶斯条件独立性假设:

$$\frac{P(\vec{x}|R=1, \vec{q})}{P(\vec{x}|R=0, \vec{q})} = \prod_{t=1}^M \frac{P(x_t|R=1, \vec{q})}{P(x_t|R=0, \vec{q})}$$

- 则:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^M \frac{P(x_t|R=1, \vec{q})}{P(x_t|R=0, \vec{q})}$$

概率模型

- 设 $p_i = p(x_i | R=1, q)$, $u_i = p(x_i | R=0, q)$
- 将 d 看做由多元贝努利分布得到的二值词袋模型
- 则：

$$\begin{aligned} p(\vec{x} | R=1, \vec{q}) &= \prod_{x_i \in d} p(x_i | R=1, \vec{q}) \prod_{x_i \notin d} p(x_i | R=1, \vec{q}) \\ &= \prod_{x_i} p_i^{e_i} (1-p_i)^{1-e_i}, \text{ if } x_i \in d, \text{ then } e_i = 1, \text{ else } e_i = 0 \end{aligned}$$

$$\begin{aligned} p(\vec{x} | R=0, \vec{q}) &= \prod_{x_i \in D} p(x_i | R=0, \vec{q}) \prod_{x_i \notin D} p(x_i | R=0, \vec{q}) \\ &= \prod_{x_i} u_i^{e_i} (1-u_i)^{1-e_i}, \text{ if } x_i \in d, \text{ then } e_i = 1, \text{ else } e_i = 0 \end{aligned}$$

概率模型

- 例子：
 - 查询为：信息 检索 教程
 - 所有词项的在相关、不相关文档中出现的概率为 p_i 、 u_i

| 词项 | 信息 | 检索 | 教材 | 教程 | 课件 |
|---------------|-----|-----|------|------|------|
| R=1时的概率 p_i | 0.8 | 0.9 | 0.3 | 0.32 | 0.15 |
| R=0时的概率 u_i | 0.3 | 0.1 | 0.35 | 0.33 | 0.10 |

文档 x : 检索 课件

则: $P(x|R=1) = (1-0.8)*0.9*(1-0.3)*(1-0.32)*0.15$

$P(x|R=0) = (1-0.3)*0.1*(1-0.35)*(1-0.33)*0.10$

$P(x|R=1) / P(x|R=0) = 4.216$

概率模型

$$\begin{aligned}
 \log \frac{P(\vec{x} | R=1, \vec{q})}{P(\vec{x} | R=0, \vec{q})} &= \log \frac{\prod_{x_i \in d \cup \bar{d}} p_i^{e_i} (1-p_i)^{1-e_i}}{\prod_{x_i \in d \cup \bar{d}} u_i^{e_i} (1-u_i)^{1-e_i}} = \sum_{x_i \in d \cup \bar{d}} \log \left(\frac{p_i}{u_i} \right)^{e_i} \left(\frac{1-p_i}{1-u_i} \right)^{1-e_i} \\
 &= \sum_{x_i \in d \cup \bar{d}} \left(e_i \log \frac{p_i}{u_i} + (1-e_i) \log \frac{1-p_i}{1-u_i} \right) = \sum_{x_i \in d \cup \bar{d}} \left(e_i \log \frac{p_i}{u_i} - e_i \log \frac{1-p_i}{1-u_i} + \log \frac{1-p_i}{1-u_i} \right) \\
 &\propto \sum_{x_i \in d \cup \bar{d}} \left(e_i \log \frac{p_i / (1-p_i)}{u_i / (1-u_i)} \right) = \sum_{x_i \in d} \left(\log \frac{p_i / (1-p_i)}{u_i / (1-u_i)} \right) \quad \text{xi在d中权重0或1} \\
 &= \sum_{x_i \in q \cap d} \left(\log \frac{p_i / (1-p_i)}{u_i / (1-u_i)} \right) + \sum_{x_i \notin q \wedge x_i \in d} \left(\log \frac{p_i / (1-p_i)}{u_i / (1-u_i)} \right) \quad \text{求和类似于向量内积计算} \\
 &\approx \sum_{x_i \in q \cap d} \left(\log \frac{p_i / (1-p_i)}{u_i / (1-u_i)} \right) \quad \text{假设对不属于q的term, } p_i=u_i, \text{ 则此项为零} \\
 &= \sum_{x_i \in q \cap d} W_i^{BIM} \quad \text{ti在q中权重, 只与q相关}
 \end{aligned}$$

概率模型

- 给定 q , 对每个文档定义 排序(Ranking)函数 RSV_d (Retrieval Status Value)

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

- 如何估计 p_t 、 u_t ?

概率模型

- 初始情况：检索初始并没有相关和不相关文档集合
- 此时可以进行假设： p_i 是常数， u_i 近似等于term i 在所有文档集合中的分布(假定相关文档很少， $R_i=r_i=0$)

$$p_i = 0.5$$

$$q_i = \frac{n_i}{N}$$

$$\begin{aligned} \sum_{t_i \in D \cap Q} \log \frac{p_i / (1 - p_i)}{q_i / (1 - q_i)} &= \sum_{t_i \in D \cap Q} \log \frac{N - n_i}{n_i} \\ &\approx \sum_{t_i \in D \cap Q} \log \frac{N - n_i + 0.5}{n_i + 0.5} = \sum_{t_i \in D \cap Q} W_i^{IDF} \end{aligned}$$

因此，BIM在初始假设情况下，其检索公式实际上相当于对所有同时出现在 q 和 d 中的词项的IDF的求和

概率模型

- 基于前面的检索结果：假定检索出的结果集合 V (可以把 V 看成全部的相关文档结合), 其中集合 V_i 包含term i , 则可以进一步进行计算
- 避免较小的 V 和 V_i 集合, 加入常数或非常数平滑因子(用 V 和 V_i 表示同名集合的大小)

$$\begin{array}{ccc} p_i = \frac{V_i}{V} & \rightarrow & p_i = \frac{V_i + 0.5}{V + 1} & \rightarrow & p_i = \frac{V_i + \frac{n_i}{N}}{V + 1} \\ q_i = \frac{n_i - V_i}{N - V} & & q_i = \frac{n_i - V_i + 0.5}{N - V + 1} & & q_i = \frac{n_i - V_i + \frac{n_i}{N}}{N - V + 1} \end{array}$$

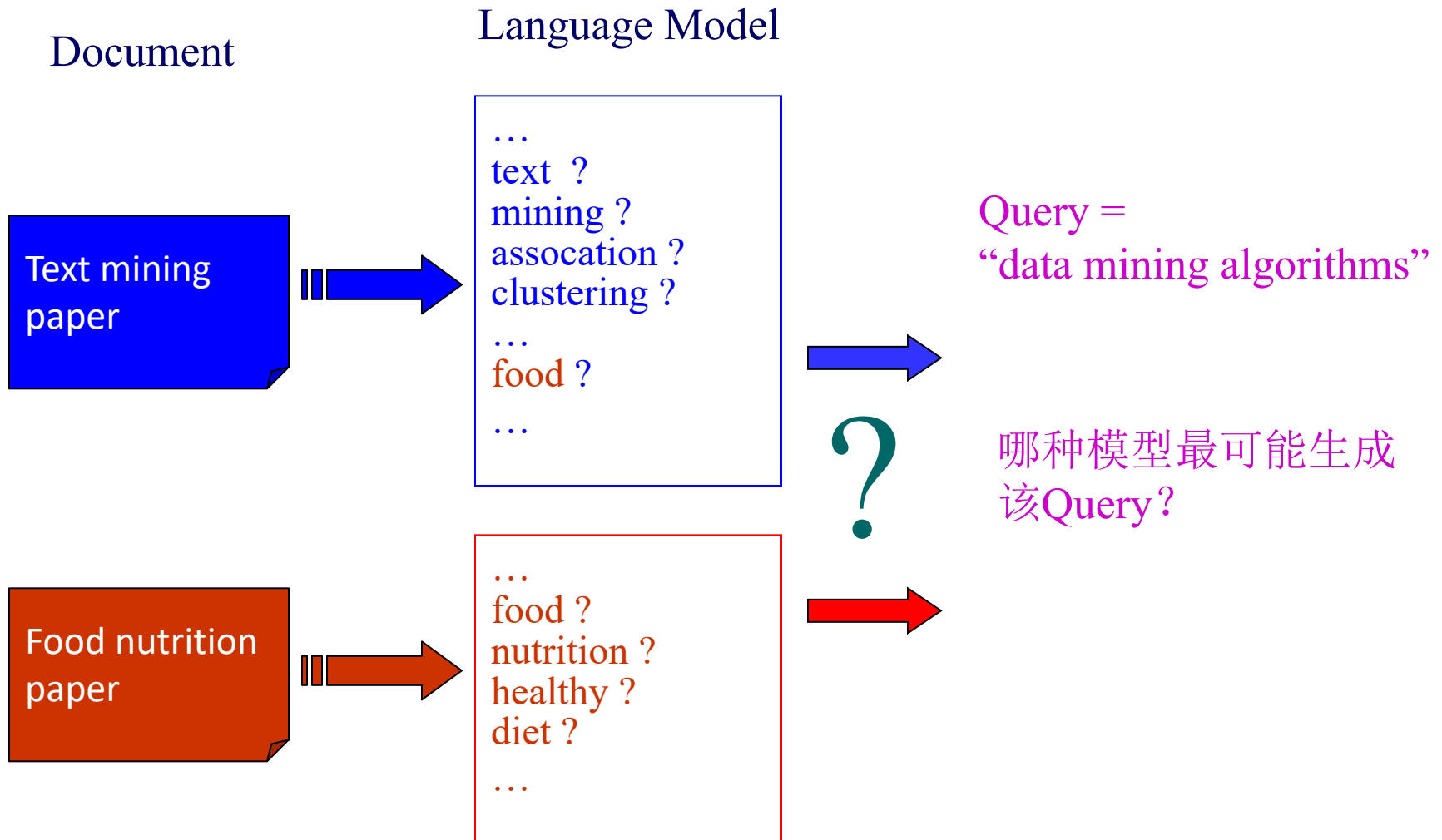
概率模型

- BIM建模过程：
 - 目标是求排序函数 $P(d|R=1,q)/P(d|R=0,q)$
 - 首先估计或计算每个term分别在相关文档和不相关文档中的出现概率 $p_i=P(x_i|R=1,q)$ 及 $u_i=P(x_i|R=0,q)$
 - 然后根据独立性假设，将 $P(d|R=1,q)/P(d|R=0,q)$ 转化为 p_i 和 u_i 的某种组合，将 p_i 和 u_i 代入即可求解。
- 优点：
 - BIM模型建立在数学基础上，理论性较强
- 缺点：
 - 需要估计参数
 - 原始的BIM没有考虑TF、文档长度因素
 - BIM中同样存在词项独立性假设

概率模型

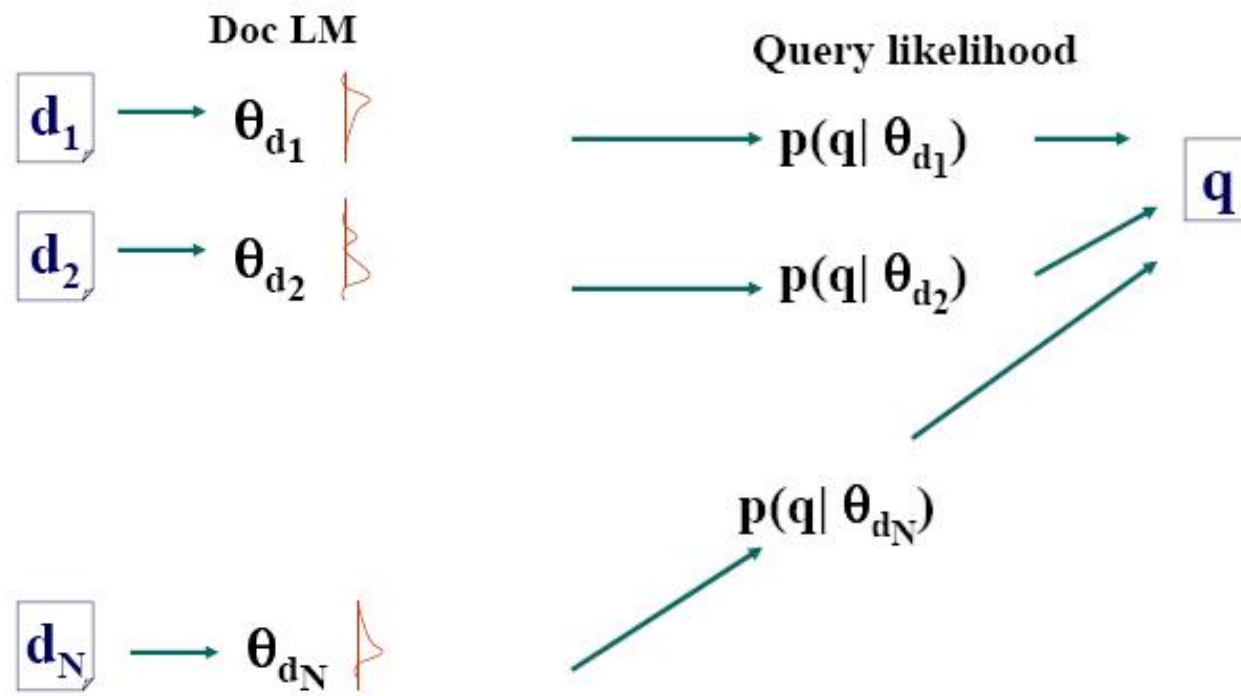
- Probability Ranking Principle (Robertson, 70ies; Maron, Kuhns, 1959)
- Information Retrieval as Probabilistic Inference (van Rijsbergen & co, since 1970ies)
- Probabilistic Indexing (Fuhr & Co., late 1980ies-1990ies)
- Probabilistic Logic Programming in IR (Fuhr & co, 1990ies)
- Bayesian Nets in IR (Turtle, Croft, 1990ies ; Blei and Lafferty, 2006ies)

语言模型



语言模型

- 根据query的似然对文档排序：



语言模型

- 基于query似然的文档排序：

$$\log p(q | d) = \sum_i \log p(w_i | d)$$

where, $q = w_1 w_2 \dots w_n$

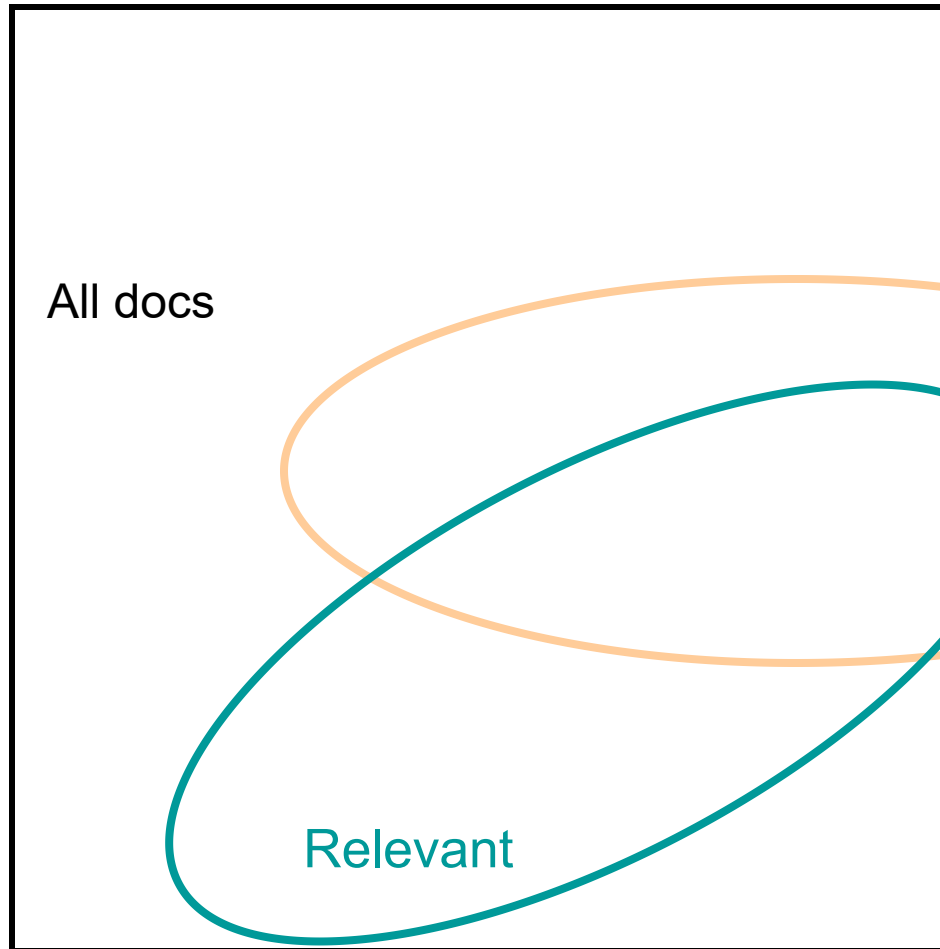
文档语言模型

- 检索问题 $\approx p(w_i | d)$ 估计问题

- How to evaluate retrieval results?

- 评价内容：
 - 信息的覆盖率
 - 信息的正确率
 - 结果的呈现形式
 - 时空效率
 - 用户体验
 -

IR 评价



$$\text{Recall} = \frac{|\text{RelRetrieved}|}{|\text{Rel in Collection}|}$$

$$\text{Precision} = \frac{|\text{RelRetrieved}|}{|\text{Retrieved}|}$$

$$F1 = \frac{2 * P * R}{P + R}$$

IR 评价

- 平均正确率均值: Mean average precision (MAP)

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$$

- R_{jk} : 前 k 个返回的排序结果
- 对于一个query, 返回结果中每篇相关文档位置上的正确率的平均值 (average precision)
- 对考察的所有query平均, 得到平均正确率均值

IR 评价

- 归一化折损累计增益：Normalized discounted cumulative gain (NDCG)

$$\text{NDCG}(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^k \frac{2^{R(j,m)} - 1}{\log_2(1 + m)}$$

- 在前k个返回结果上的评价指标
- $R(j, d)$ 为给定查询j，文档d的相关度分数
- Z_{jk} 为归一化因子

Next lecture

Web Search
Why Different?