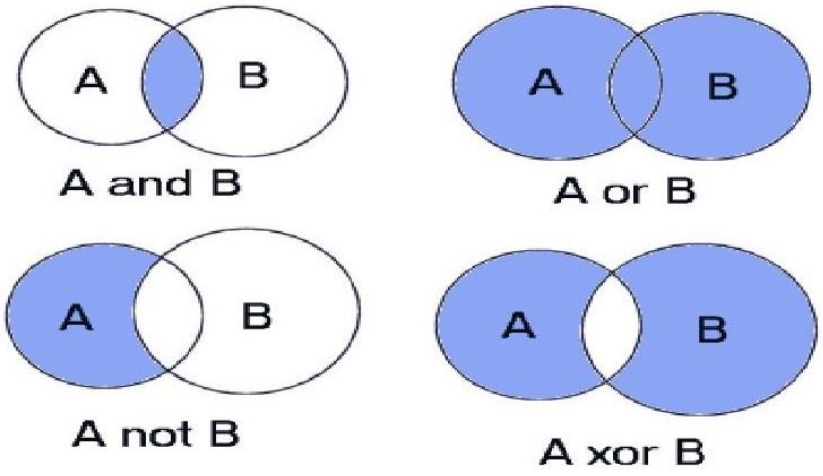
布尔检索

目录



如何度量相关性？

确定文档和查询之间的相关度是IR的核心问题

Query:

Bill Clinton

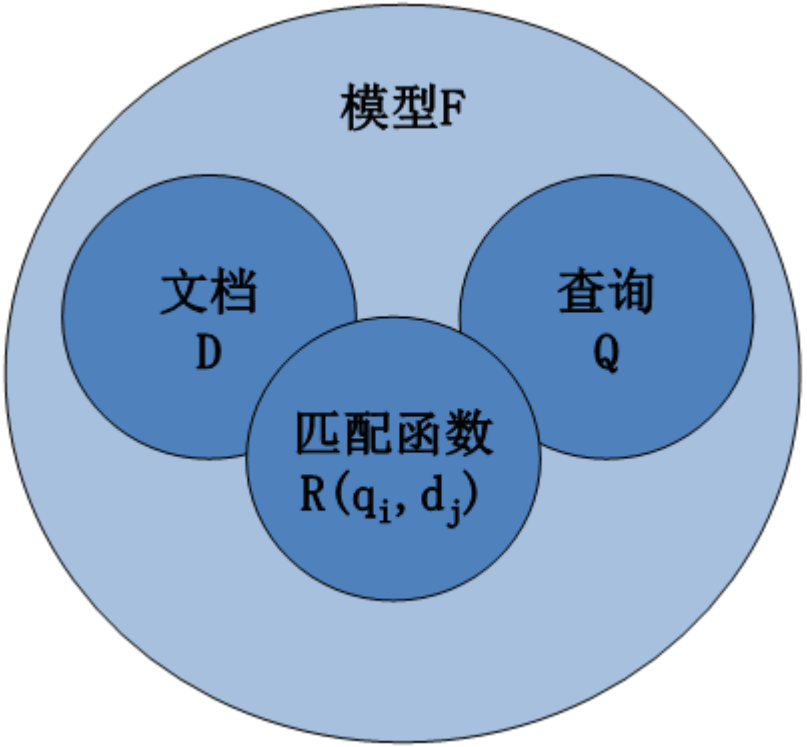
Document:

Relevant?

How relevant?

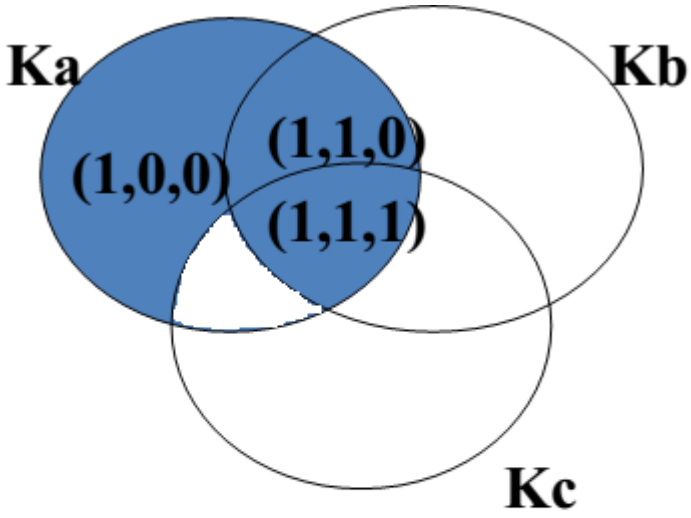
检索模型的定义

信息检索模型是描述信息检索中的文档、查询和它们之间关系(匹配函数)的数学模型。



布尔模型：定义

- **文档表示**
  - 一个文档被表示为关键词(bag of words)的集合
- **查询表示**
  - 查询式(Queries)被表示为关键词的布尔组合，用“与、或、非”连接起来
- **相关度计算**
  - 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
  - 检索策略是二值匹配



$$q = k_a \wedge (k_b \vee \neg k_c)$$

布尔模型：优缺点

优点

- 由于查询简单，因此容易理解
- 通过使用复杂的布尔表达式，可方便地控制查询结果
- 相当有效的实现方法
- 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

缺点

- 弱。不支持部分匹配，完全匹配会导致结果太多或太少
- 非常刚性：“与”意味着全部；“或”意味着任何一个
- 原则上讲，所有被匹配的文档都将被返回
- 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？

目录

- 我想知道《莎士比亚全集》这本大部头的书中：哪些剧本包含Brutus和Caesar但不包含Calpurnia？
- 一种方式是采用Unix下的grep程序，先找出所有包含Brutus和Caesar的剧本，然后再将包含Calpurnia的剧本排除
- 但是很多情况下，采用上述线性扫描的方式是远远不够的。

为什么？

上述方式不合适的原因：

1. 对于大规模文档的搜索太慢。

2. 有时需要更灵活的匹配方式。比如，要求查找countrymen附近的Romans。

3. 需要对结果进行排序。用户希望能在多个能满足要求的文档中得到最佳答案。

词项文档索引

解决方法是采用**非线性的扫描方式**，一种方法就是事先给文档建立**索引(index)**

文档

	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

词项

如果文档（这里就是剧本）包含某个词，则对应的项为1，否则为0

Brutus AND Caesar BUT NOT Calpurnia

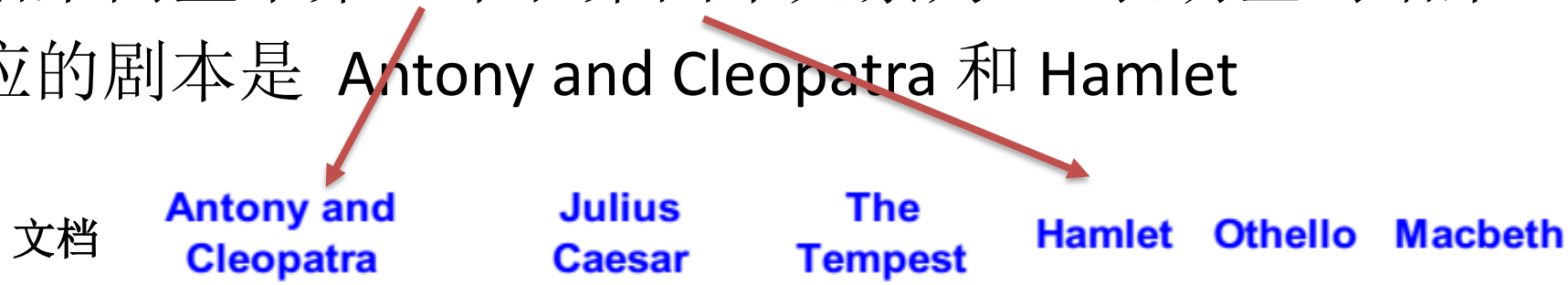
处理查询的过程

Brutus AND Caesar BUT NOT Calpurnia

- 分别取出Brutus、Caesar以及Calpurnia 3个词项对应的行向量110100，110111，010000，并对Calpurnia对应的向量求反101111。
- 进行按位与操作

110100 AND 110111 AND 101111 = 100100.

- 结果向量中第一个和第四个元素为1，表明查询结果对应的剧本是 Antony and Cleopatra 和 Hamlet



查询的返回结果

- Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius *Caesar* dead,
He cried almost to roaring; and he wept
When at Philippi he found *Brutus* slain.

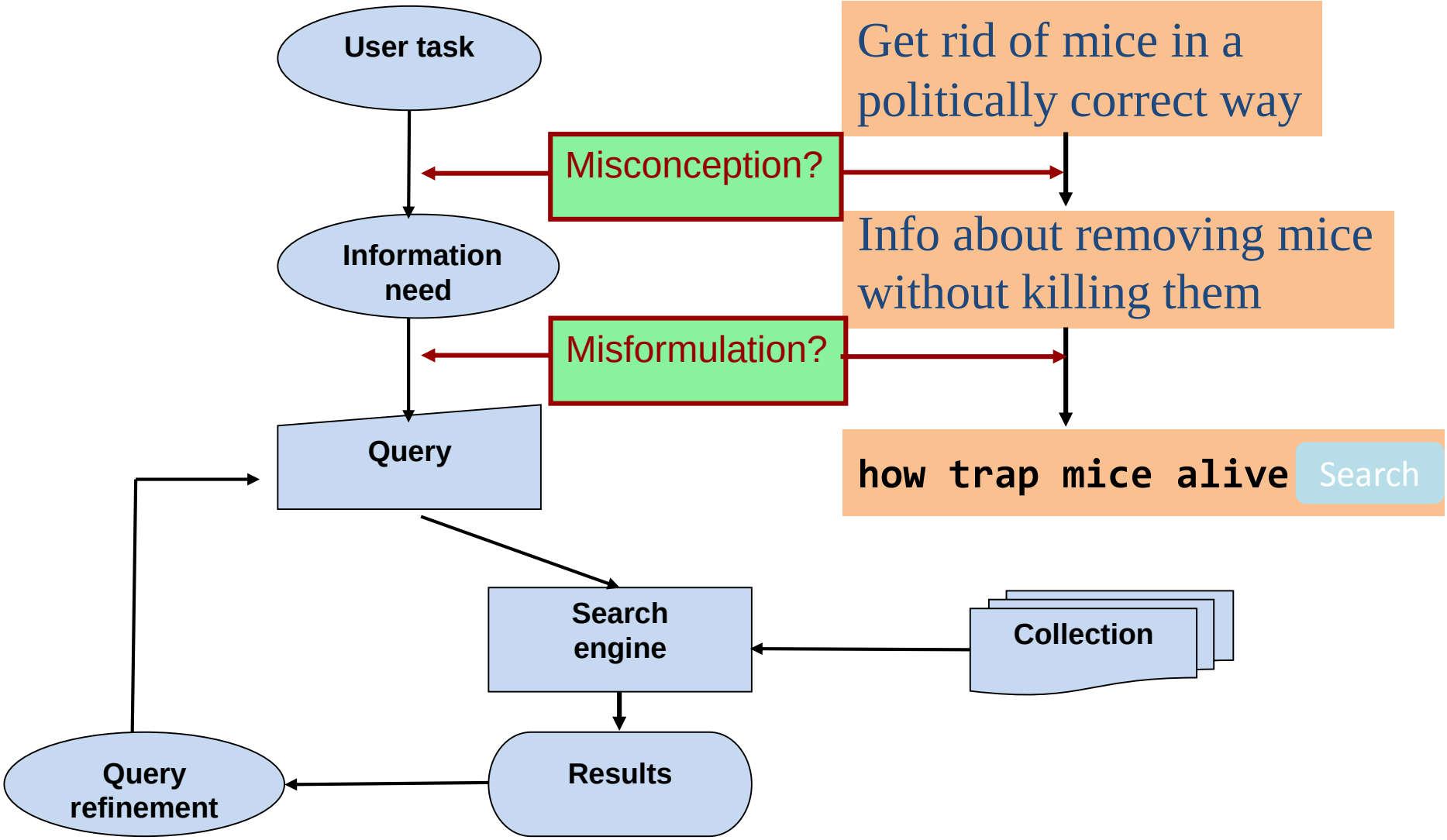
- Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius *Caesar* I was killed i' the Capitol; *Brutus* killed me.

信息检索的基本假设

- **集合**：固定数量的文档
- **目标**：找到与用户信息需求相关的含有信息量的文档，帮助用户完成一个任务

典型的搜索模型


User task
Get rid of mice in a politically correct way
Misconception?
Information need
Info about removing mice without killing them
Misformulation?
Query
how trap mice alive
Search
Search engine
Collection
Query refinement
Results

返回文档的好坏

- **查准率**：返回的能满足用户信息需求的文档占总的返回文档的百分比
- **召回率**：返回的能满足用户信息需求的文档占总的能满足用户信息需求的文档的百分比
- **信息检索的评价将在后续课程中详细介绍**

更大的数据集

- 假设有一个更大的数据集
  - 共有*N*=100万个文档，
  - 每个文档包含大概1000个词汇
- 假设一个词汇需要6个字节(将空格和标点计算在内)，则这些文档共有6G (100万*1000*6)
- 假设其中共有500k个唯一的不重复的词汇

在这种情况下，前面所使用的词项文档矩阵会如何？

无法构建矩阵

- 词项-文档矩阵中有 500*K*(不重复词汇数*t*) * 100万(文档数*n*) = 0.5T 个 0和1
- 但是其中只有不到10亿个1 (100万*1000个词汇/文档)，其他都是0。这个矩阵十分稀疏Sparse
- 更好的索引表示方法是？
  - 只记录1的位置

	T_1	T_2	….	T_t
D_1	0	0	…	1
D_2	0	0	…	0
:	:	:		:
:	:	:		:
D_n	0	1	…	0

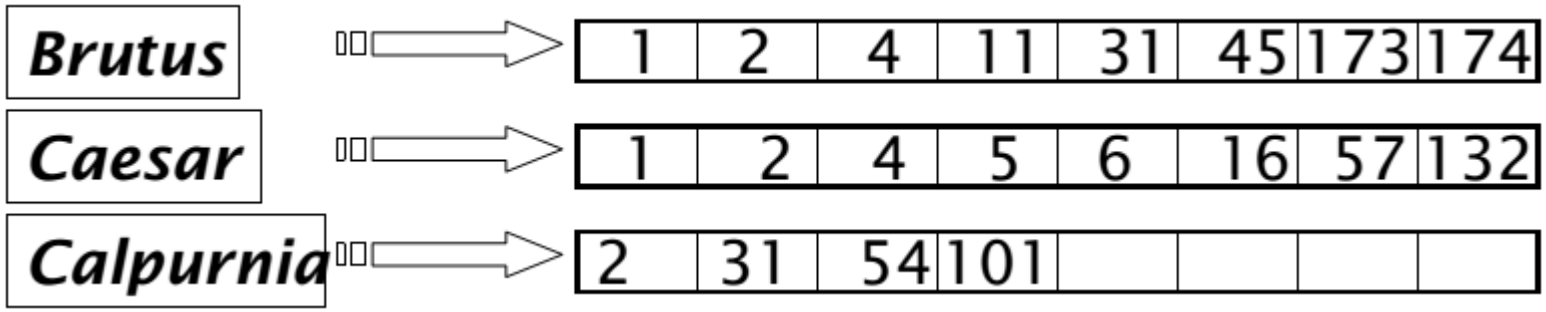
小结：一个简单的搜索示例

- 线性扫描的搜索：grep
- 非线性的扫描方式：index
- 处理查询
  - 构造矩阵→ 110100 AND 110111 AND 101111 = 100100
- 典型的搜索模型
  - 任务→信息需求→文字形式→查询→查询优化→结果
  - 返回文档的好坏：查准率、召回率
- 简单模型存在的问题：大的数据集无法构建矩阵

目录

倒排索引(Inverted index)

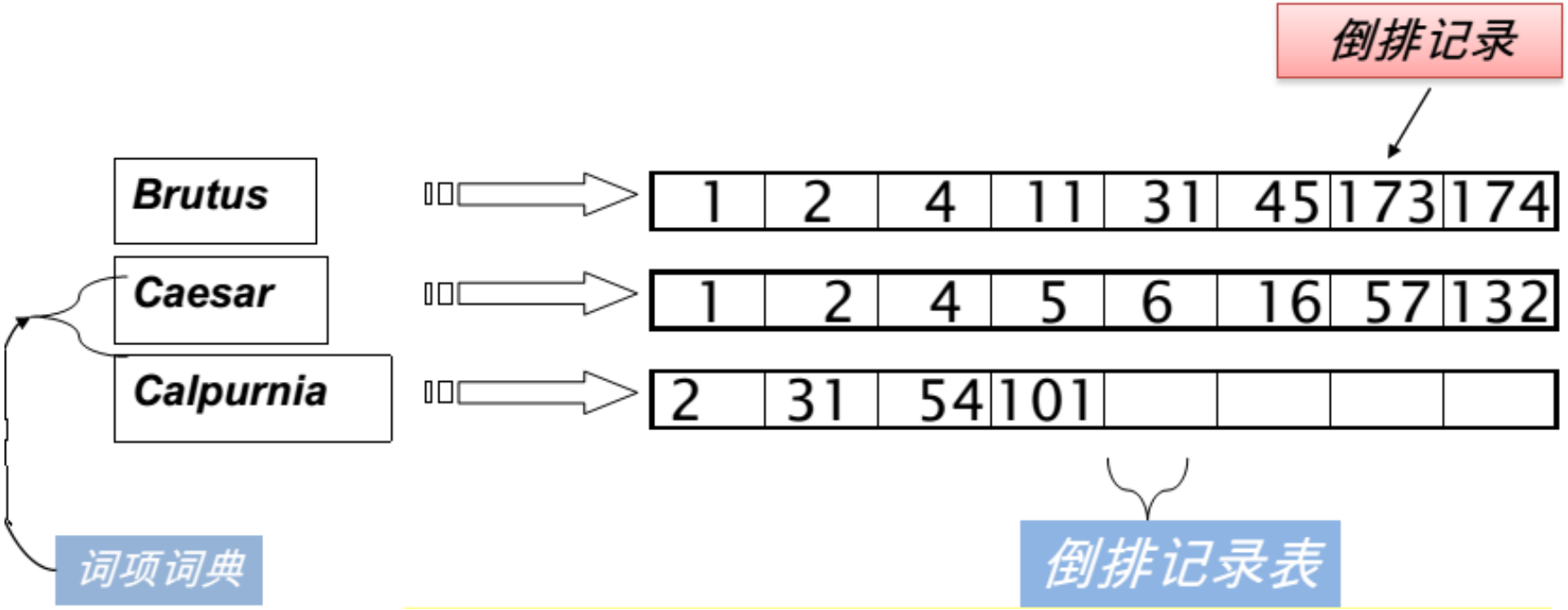
- 对于每一个词项，存储所有包含这个词项的文档的一个列表。一个文档用一个序列号docID来表示。
- 能用一个固定长度的数组来存储它吗？



- 如果词项Caesar被添加到14号文档中，索引如何变化?

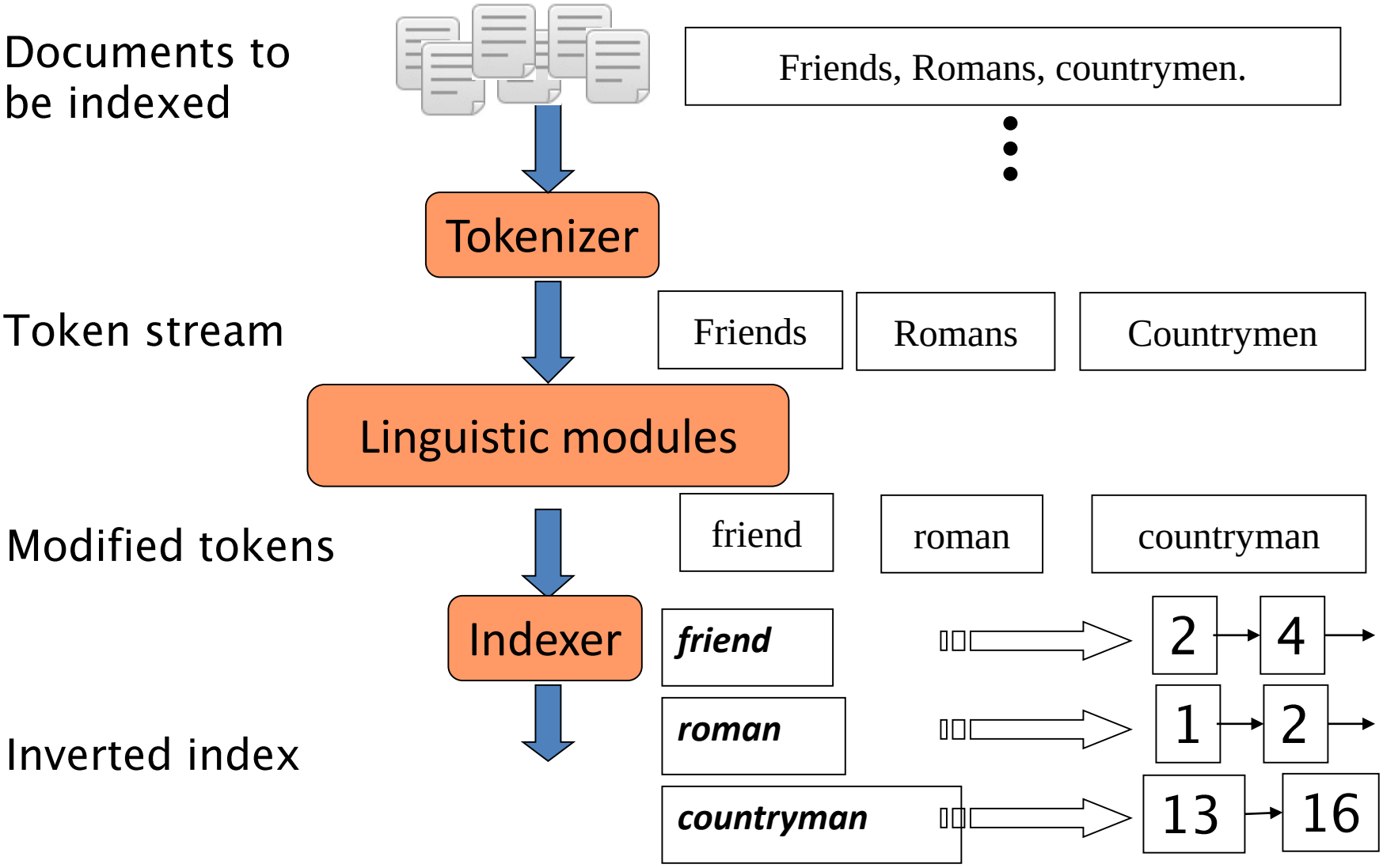
倒排索引

- 应当使用可变长度的记录列表
  - 在硬盘上，一串连续的记录是正常的，也是最好的
  - 在内存里，可以使用链表，或者可变长度的数组



- 按照docID排列(稍后会说为什么这么做？)

Inverted index construction



建立索引的步骤：词条序列Token Sequence

(修改过的词条，文档ID)对序列

Doc1

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

Doc2

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

建立索引的步骤：排序

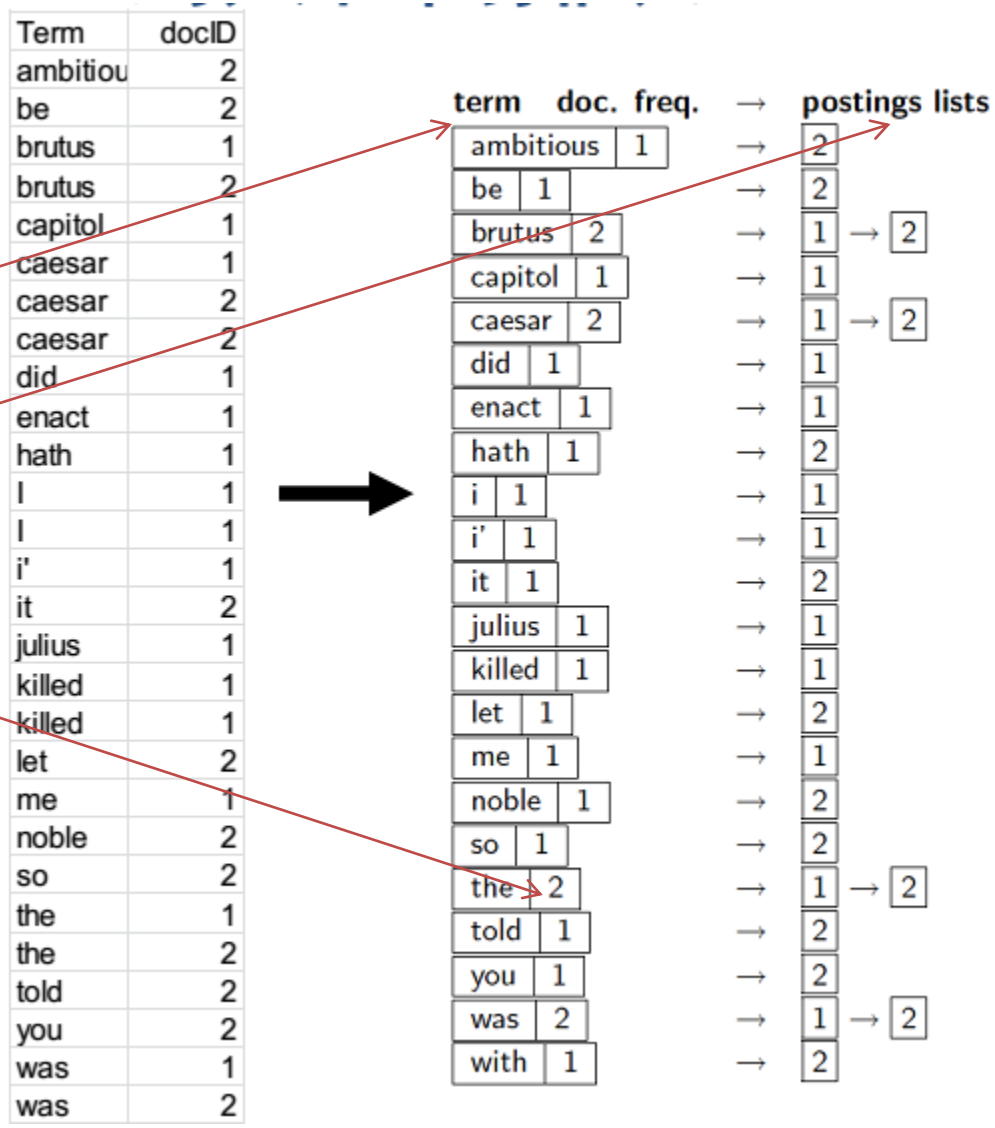
- 先按照词条排序，
- 再按照docID排序

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

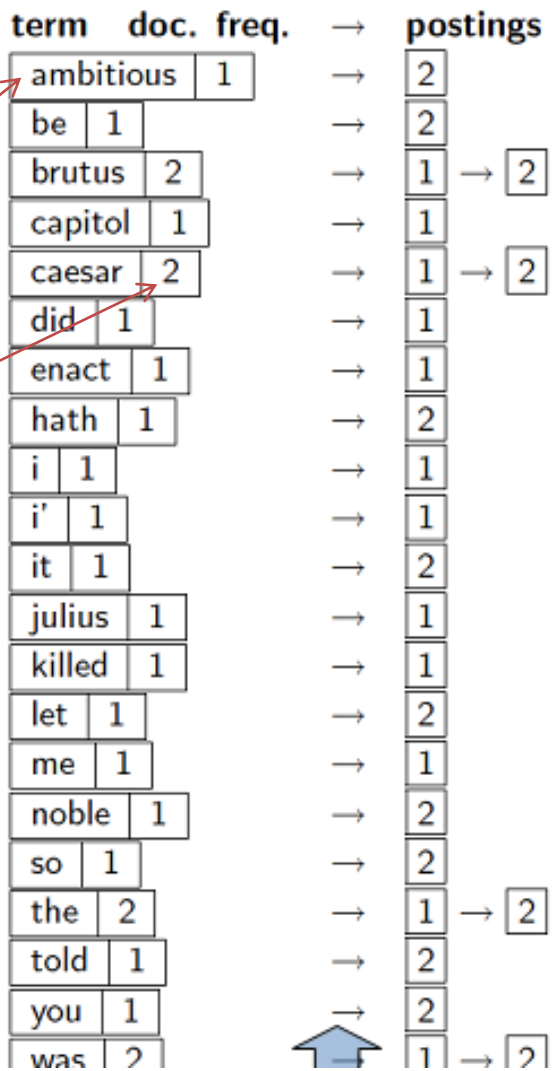
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

建立索引的步骤：词典和倒排表

- 同一篇文档中多次出现的词被合并
- 分割成词典和倒排表
- 词汇的**文档频率**也被记录



存储开销



词条和文档频率

docID的列表

指针

稍后的课程会讨论:

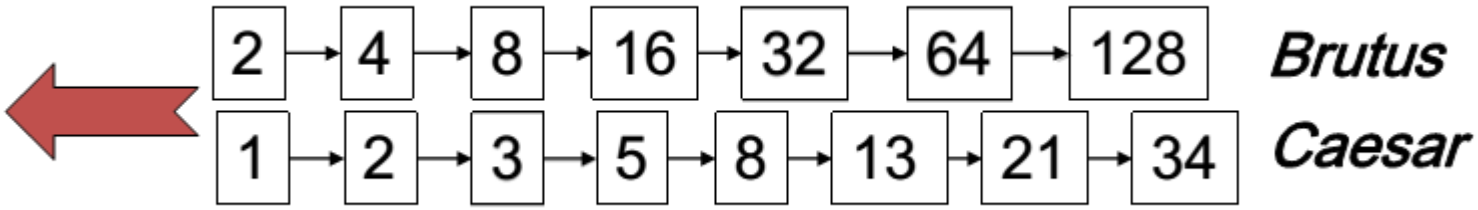
- 如何高效地建立索引?
- 需要多少存储空间?

查询的处理：AND

- 考虑处理这样的查询：

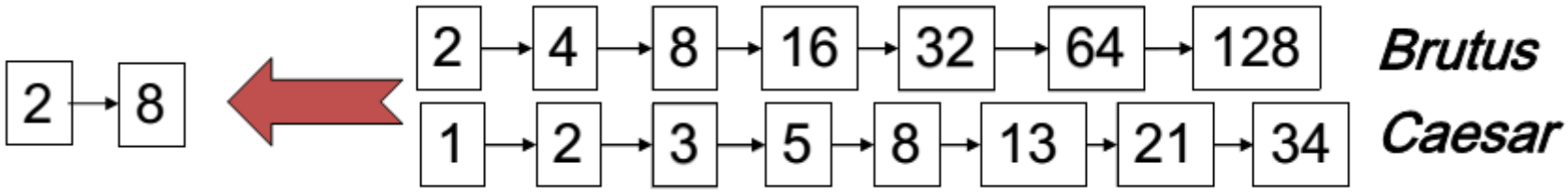
  Brutus AND Caesar

  – 在字典中找到Brutus，得到它的倒排记录表
  – 在字典中找到Caesar，得到它的倒排记录表
  – 合并两个倒排列表



倒排记录表的“合并”

- 同时扫描两个倒排记录表求交集，所需时间和倒排记录的数量呈线性关系。



如果倒排记录表的长度分别为x和y, 则合并共需 O(x+y)次操作

注意：倒排记录已经按照docID排好序了

倒排记录表的“合并”算法(求交集)

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

小结：倒排索引

- **倒排索引(Inverted index)**
  - 对于每一个词项，存储所有包含这个词项的文档的一个列表
  - 词项 + 倒排记录
- **倒排索引的建立过程**
  - 词条化模块 → 语言学模 → 索引模块
- **查询的处理 ←→ 倒排记录表的“合并”**
  - AND

目录

布尔检索模型：定义

- 文档表示
  - 一个文档被表示为关键词的集合Bag of Words
- 查询表示
  - 查询式(Queries)被表示为关键词的布尔组合，用“与、或、非”连接起来
- 相关度计算
  - 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
  - 检索策略是二值匹配{0,1}

布尔检索模型：形式化表示

- 定义：用q_{dnf}表示查询q的析取范式，q_{cc}表示q_{dnf}的任意合取分量。
- 文献d_j与查询q的相似度为

$$sim(d_j,q)=\begin{cases}1 \ \text{if} \ \exists q_{cc} \mid (q_{cc} \in q_{dnf}) \wedge (\forall k_i, g_i(d_j)=g_i(q_{cc})) \\ 0 \ \text{otherwise}\end{cases}$$

- 如果 $sim(d_j,q)=1$，则表示文献d_j与q相关，否则为不相关。
  - $sim(d_j, q)$ 为该模型的匹配函数。

布尔检索模型：布尔代数

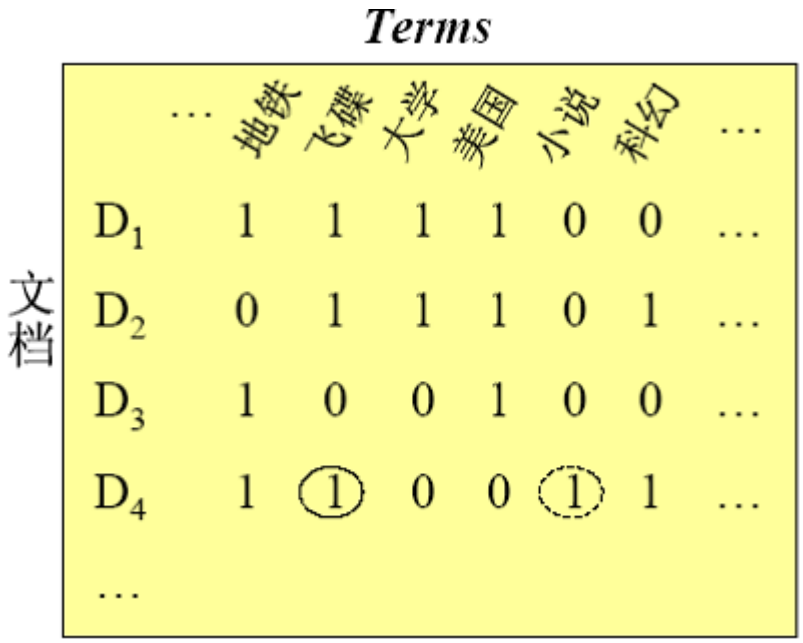
- 布尔变量
  - 只有“真”、“假”取值的变量
  - 计算机中常常用1表示“真”true，0表示“假”false
- 布尔操作(关系)
  - 与(AND)：(A AND B) = true iff A=true and B=true
  - 或(OR)：(A OR B) = true iff A=true OR B=true
  - 非(NOT)：(NOT A) = true iff A=false
- 布尔表达式
  - 多个布尔变量之间通过布尔操作组成的表达式
    - 如：A AND (B OR C) AND NOT D
  - 蕴含：两个布尔表达式P、Q，如果A为true，那么Q为true，则称P蕴含Q，记为P→Q

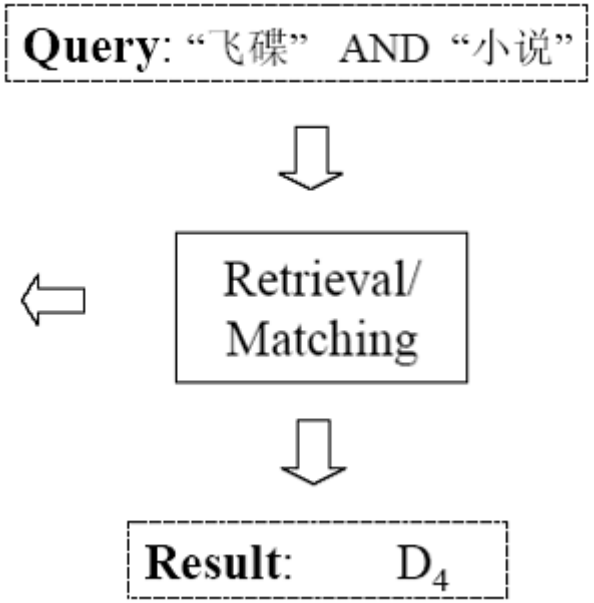
布尔检索：精确匹配

- 布尔模型可以用来处理布尔表达式形式的查询
  - 布尔查询使用AND，OR，NOT来连接查询词汇
    - 将文档看作词汇的集合
    - 精确：匹配 或 不匹配
  - 布尔模型也许是IR系统中最简单的模型
  - 是近30年来最主要的商业搜索工具
  - 当前使用的很多系统依然使用布尔模型
    - 电子邮件，图书馆分类系统，mac osx的spotlight

示例：精确匹配的欠缺

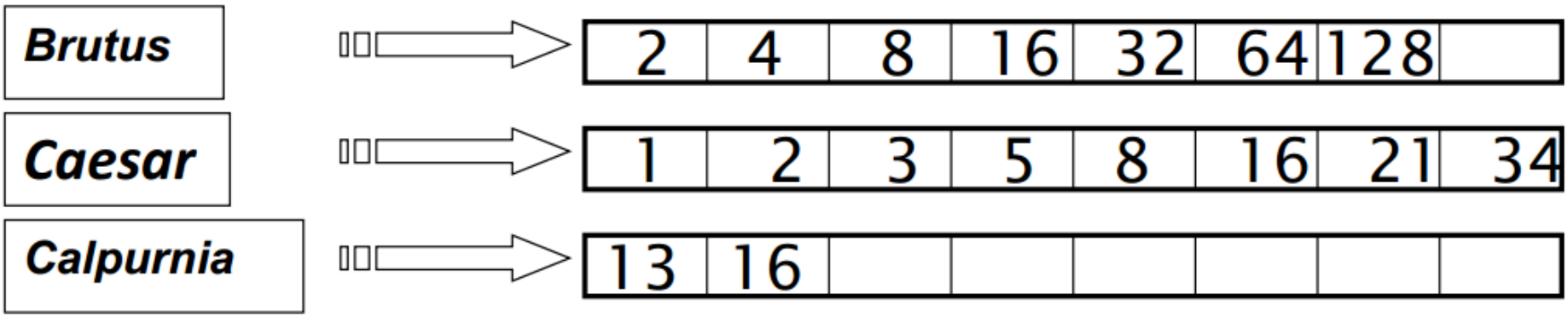
- “飞碟”AND“小说”：只能检索出D_4，无法显现D_1，D_2，D_3的差异
- “飞碟”OR“小说”：可以检出D_1，D_2，D_4，但无法显现它们的差异





查询优化

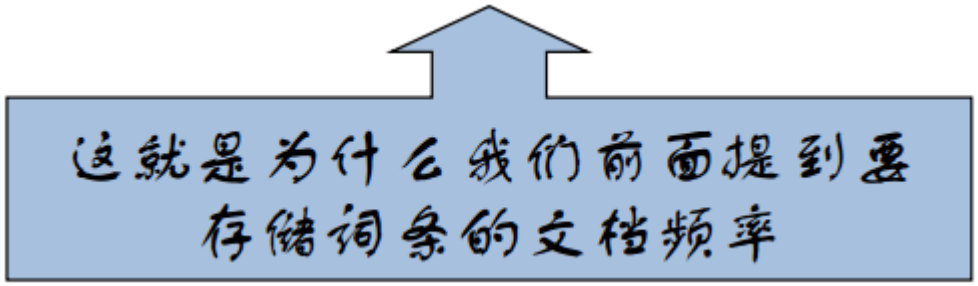
- 处理查询的最佳顺序是什么？
- 考虑一个使用**AND**连接*n*个词汇的查询
- 对于每一个词汇，都得到它的倒排记录表，然后进行“合并”操作

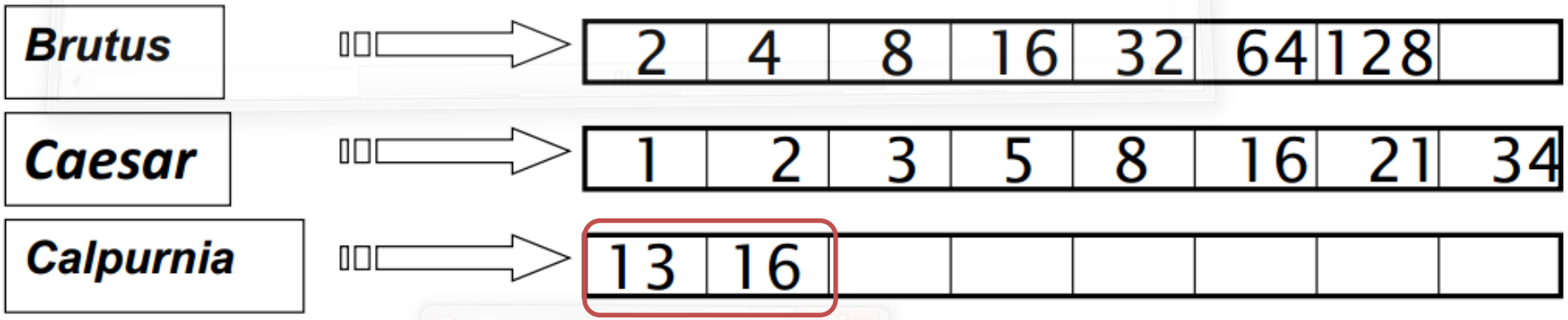


查询: ***Brutus AND Calpurnia AND Caesar***

查询优化的例子

- 按照文档频率的顺序进行处理：
  - 先处理文档频率小的，再处理大的





按照(*Calpurnia AND Brutus*) *AND Caesar*的顺利处理查询

更一般的优化

- e.g., (madding OR crowd) AND (ignoble OR strife) AND (killed OR slain)
- 获得所有词项的文档频率
- 保守地估计出每个OR操作后的结果大小
- 按照结果从小到大的顺序执行AND