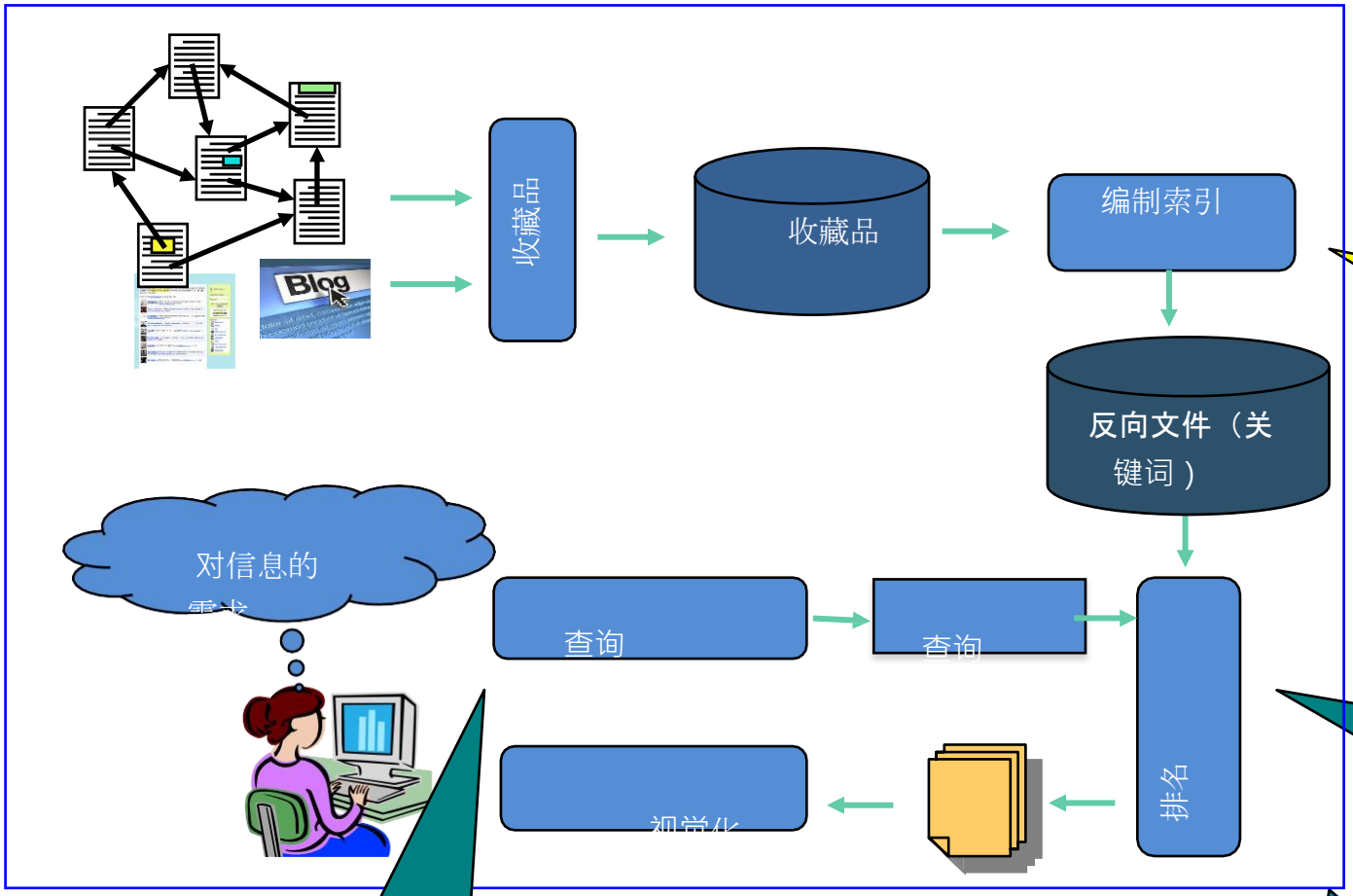
第二章：信息表示/索引

RI @MB课程

课程纲要

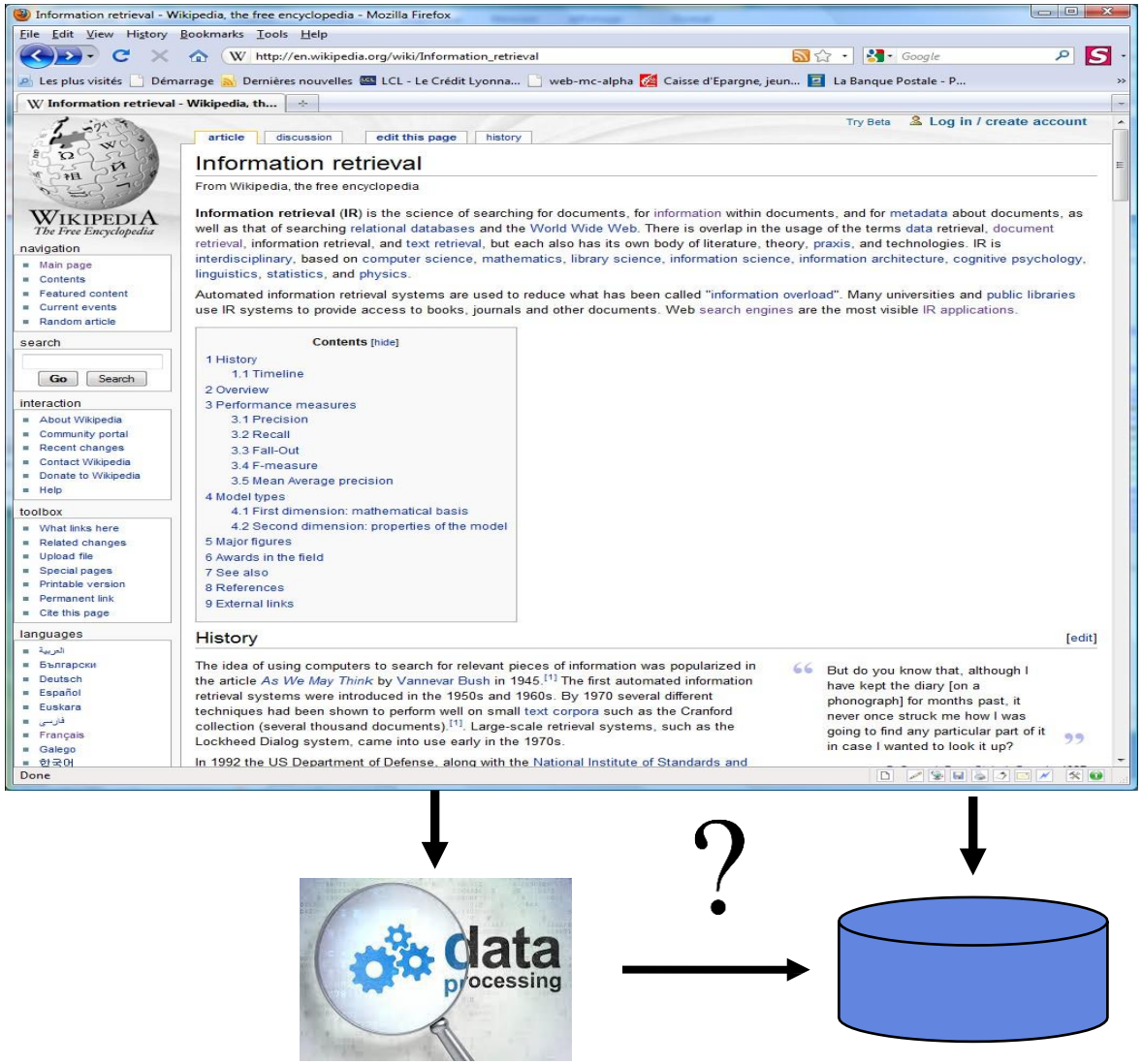


第二章。
- 编制索引。
- 反向文件
- 加权

第43章：IR模型：矢量。
概率论，ML

应用框架：网络搜索引擎

信息的表述？


Information retrieval - Wikipedia, the free encyclopedia - Mozilla Firefox
http://en.wikipedia.org/wiki/Information_retrieval
Information retrieval
From Wikipedia, the free encyclopedia
Information retrieval (IR) is the science of searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web. There is overlap in the usage of the terms data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its own body of literature, theory, praxis, and technologies. IR is interdisciplinary, based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, statistics, and physics.
Automated information retrieval systems are used to reduce what has been called "information overload". Many universities and public libraries use IR systems to provide access to books, journals and other documents. Web search engines are the most visible IR applications.
Contents [hide]
1 History
1.1 Timeline
2 Overview
3 Performance measures
3.1 Precision
3.2 Recall
3.3 Fall-Out
3.4 F-measure
3.5 Mean Average precision
4 Model types
4.1 First dimension: mathematical basis
4.2 Second dimension: properties of the model
5 Major figures
6 Awards in the field
7 See also
8 References
9 External links
History
The idea of using computers to search for relevant pieces of information was popularized in the article As We May Think by Vannevar Bush in 1945. The first automated information retrieval systems were introduced in the 1950s and 1960s. By 1970 several different techniques had been shown to perform well on small text corpora such as the Cranford collection (several thousand documents). Large-scale retrieval systems, such as the Lockheed Dialog system, came into use early in the 1970s.
But do you know that, although I have kept the diary [on a phonograph] for months past, it never once struck me how I was going to find any particular part of it in case I wanted to look it up?
data processing
?

文件的表述

- 文件表述（索引）。
  - 构建一套元素 "关键词 "的过程，以描述文件内容的特征
- 主要内容
  - 单词：苹果
  - 词组：马铃薯
  - 来自本体论的概念
  - 单词向量
- 文本索引的结果
  - 每个文件由一个索引（关键词）列表表示
  - 该索引提供了对（选定）文件的访问

索引选项

- 编制索引的类型
  - 手动（由专家完成）对自动
  - 引导（控制）与自由
    - 指导：从受控词汇（词库、本体论、字典、词典等）中选择索引（关键词）。
    - 免费（索引（关键词）取自文件正文）
- 办法
  - 统计学（单词分布）和/或NLP（文本理解）。
  - 常见的方法是以简单的假设进行相当的统计
    - 一个词的冗余度标志着它的重要性
    - 词语的共同出现标志着一个文件的主题

编制索引：一般方法

- **分解文本（解析）。**
- **将字符序列分割成**单词（标记化）。
- 标准化
  - **文本**：标点符号、日期、方框
  - 语言学：词根/lematisatio n
- 去除停止词
  - **根据** "简短的清单"，"the"、"and"、"or "**或**频繁的词语
- 分组词

<标题>：信息检索

(Body text>：信息检索（IR）是一门搜索文件的科学。N.I.S.T推出TREC

信息检索信息检索是搜索文件的科学N.I.S.T推出TREC

信息检索信息检索IR是一门科学，是搜索、文件NIST推出的TREC

信息, 检索, 信息, 检索, IR, 科学, 搜索, 文件 NIST推出TREC

信息2，检索2，IR 1，科学1，搜索1。

文件1，NIST 1，发射1，TREC 1

一袋文字 (BOW)

分割(标记化)

-在中文和日文中没有空格

- **不保**证以独特的方式提取一个术语 **中国式的符号化**

1. Original text

旱灾在中国造成的影响

(the impact of droughts in China)

2. Word segmentation

旱灾 在 中国 造成 的 影响

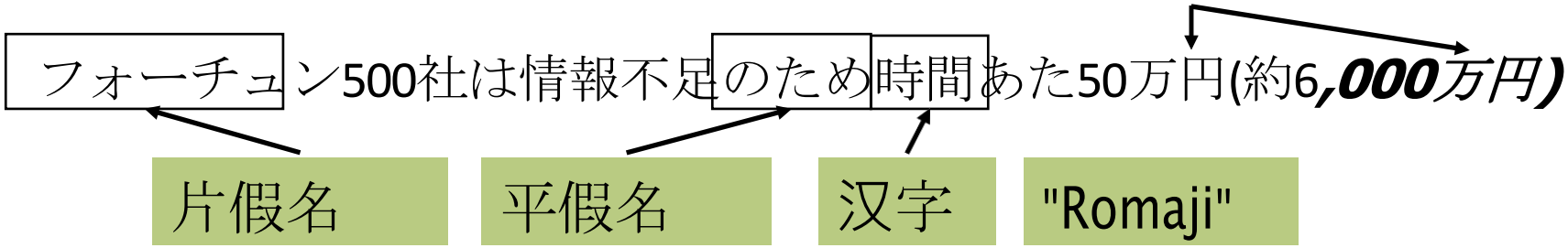
drought at china make impact

3. Bigrams

旱灾 灾在 在中 中国 国造
造成 成的 的影 影响

分割（标记化）

- 日语更复杂，有不同的字母



用户可以完全用平假名来表达他的请求

标准化

- 对一个词的变体进行分组的形态学过程
- 语言标准化
  - 词根化：去除后缀和转折词
    - 例：经济，经济上，经济学家，¢经济
    - 用于英语： retrieve, retrieving, retrieval, retrieved, retrieves ¢ retriev
  - 词组化：通过语言学分析，将单词还原为词组（不定式动词，名词的单数形式，.....）。

扎根

- 使用转换规则
  - 规则类型：条件行动
    - 例如：如果单词以s结尾，则删除结尾。
    - 几个最著名的算法：波特、洛文斯
    - Stemmer Snowball (http://snowball.tartarus.org/)可供下载
- X字符的截断

波特的算法

- 基于对元音-辅音序列的测量
  - 茎的度量m是[C](VC)^m[V]，其中C是一序列的辅音和V是元音序列 [] = 选项
  - m=0（树，由），m=1（麻烦，燕麦，树，常春藤），m=2（麻烦，私人）。
- 基于一组行动条件的算法
  - 旧后缀 新后缀
  - 这些规则分为几个阶段，并依次进行审查
  - 例如，步骤1a。
    - sses ss *(caresses caress)*
    - ies i *(ponies poni)*
    - s NULL *(cats 猫)*
  - 例如：步骤1b。
    - 如果m>0 eed ee *(同意同意)*
    - 如果*v*ed NULL *(抹灰的石膏，但流血的石膏)*
- 有几个地点可供选择
  - http://www.tartarus.org/~martin/PorterStemmer/

肢端化

- 将单词的转折变体还原为其基本形式前。
  - *是、是、是*
  - *汽车、汽车、汽车的、汽车的汽车*
  - *唱吧，唱吧，唱吧➔唱吧*
- 使用词库（词典）。
- 精细的语法分析

  - Tree-tagger (http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/)

索引结果的例子（与波特的正常化）。

- 原文。

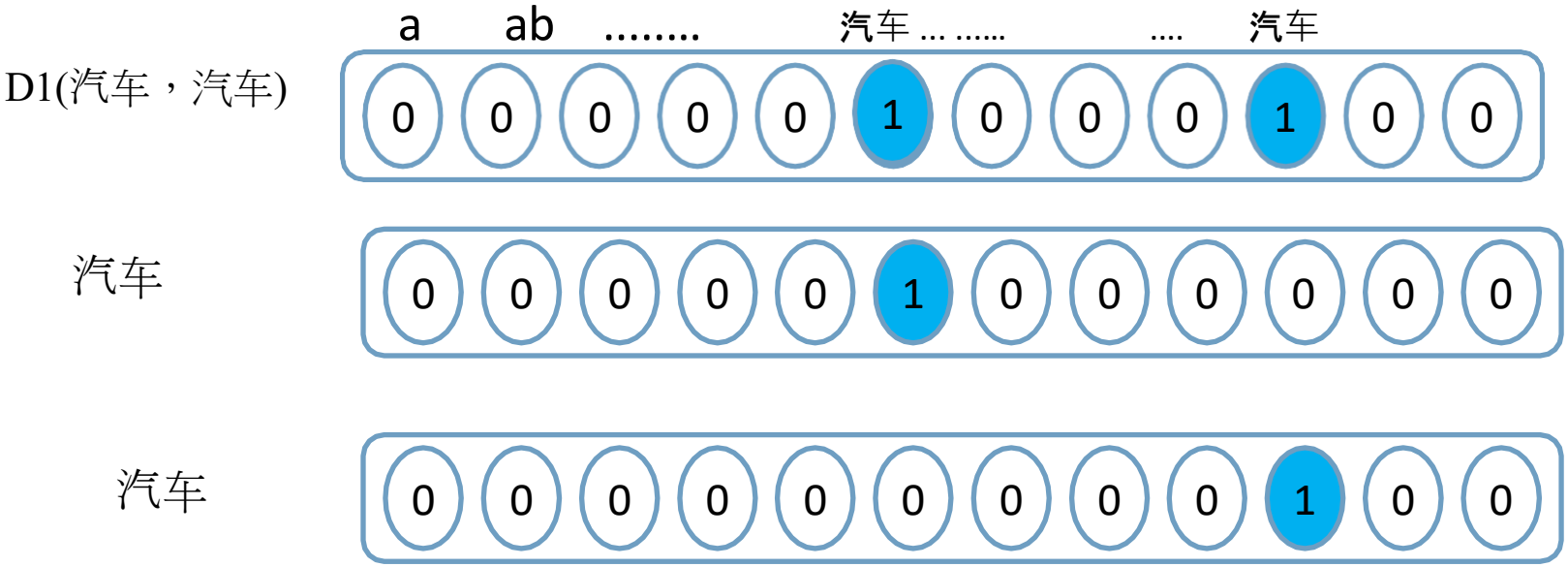
  美国公司对其农业化学品采取的营销策略，报告此类化学品的市场份额预测，或报告农业化学品、杀虫剂、除草剂、杀菌剂、杀虫剂、化肥的市场统计数据，预测销售量、市场份额、刺激需求、降价、销售量

- 携带后的文本+删除空字

  市场4，战略1，卡尔1，公司1，美国1，农业1，化学2。
  报告2，预测2，分享1，统计学家1，农业化学1，农药1。
  草药1，杀菌剂1，杀虫剂1，肥料1，销售2，刺激剂1，需求1，价格1，切割1，体积1

正式代表

- **正式代表**
  - 组成者
    - D1(车，车，...)
  - **向量表示**（"一热表示（本地）"）。）
    - D1(**汽**车，汽车)



相似性（车，车）=0

实物代表

- 反面文件
  - 然后，这些术语被存储在一个叫做倒置文件的结构中

反向文件

d1:
凯撒也是如此。高贵的布鲁特斯告诉你，凯撒是
有雄心的

d2:
我确实制定了凯撒大帝，我被杀了我在都城；布鲁特斯杀

处理方法 = 编制索引

术语	N个文档	总频率	ÄÄÄ
有雄心的	1	1	1
是	1	1	2
残暴	2	2	3
国会	1	1	5
凯撒	2	3	6
做了	1	1	
颁布	1	1	
阴阳师	1	1	
I	1	2	
i'	1	1	
它	1	1	
朱利叶斯	1	1	
被杀	1	2	
让	1	1	
我	1	1	
崇高的	1	1	
那么	1	1	
的	2	2	
告诉你	1	1	
你	1	1	
是	2	2	
与	1	1	

文件 #	频率
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

d1:
凯撒也是如此。高贵的布鲁特斯已经告诉你，凯撒是有野心的

d2:
我确实制定了凯撒大帝，我在国会大厦被杀；布鲁图斯

d3:
我确实制定了朱利叶斯

d4:

d5:
我确实制定了凯撒大帝，我被杀了

d6:
我确实制定了凯撒大帝，我被杀了

d7:
我确实制定了凯撒大帝，我被杀了

d8:
我确实制定了凯撒大帝，我被杀了

d9:

反向文件

词典

词语	Nb Doc	累计	ĀĀĀ
雄心勃勃	2	5	1
布鲁特斯	2	8	3
国会	5	15	6

简单张贴

文档	频率
doc1	3
doc2	2
doc1	1
doc3	7

- **排序的列表**
- **B型**轴
- **哈希代**码表
- …

该词在文件中的位置（对**短**语搜索很重要）

丰富的张贴

文档	频率	位置	信标
doc1	3	1, 4, 12	1, 5
doc2	2	1	
doc3	2	3	

标签（标题、正文、锚。..）

我确实制定了凯撒大帝，我被杀了
12 3 4 56 7 8
我在都城；布鲁特斯杀了我。

对请求作出回应

凯撒，布鲁

术语	N个文档	总频率	ÄÄÄ
有雄心的	1	1	1
是	1	1	2
残暴	2	2	3
国会	1	1	5
凯撒	2	3	6
做了	1	1	
颁布	1	1	
阴阳师	1	1	
I	1	2	
i'	1	1	
它	1	1	
朱利叶斯	1	1	
被杀	1	2	
让	1	1	
我	1	1	
崇高的	1	1	
那么	1	1	
的	2	2	
告诉你	1	1	
你	1	1	
是	2	2	
与	1	1	

文件 #	频率
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

d1:
凯撒也是如此。高贵的布鲁特斯已经告诉你，凯撒是有野心的

d2:
我确实制定了凯撒大帝，我在国会大厦被杀；布鲁图斯

d3:
我确实制定了朱利叶斯

d4:

d5:
我确实制定了凯撒大帝，我被杀了

d6:
我确实制定了凯撒大帝，我被杀了

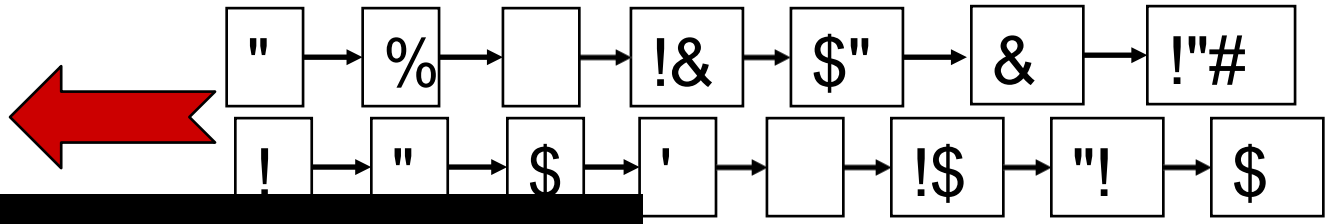
d7:
我确实制定了凯撒大帝，我被杀了

d8:
我确实制定了凯撒大帝，我被杀了

d9:

对请求作出回应

- 让我们的查询是:
  - *布鲁特斯和凯撒*
  - 在字典里查一下*布鲁特斯*。
    - 选择你的张贴物清单。
  - 在字典中查找*凯撒*。
    - 选择你的张贴物清单。
  - **合并**这两个帖子。

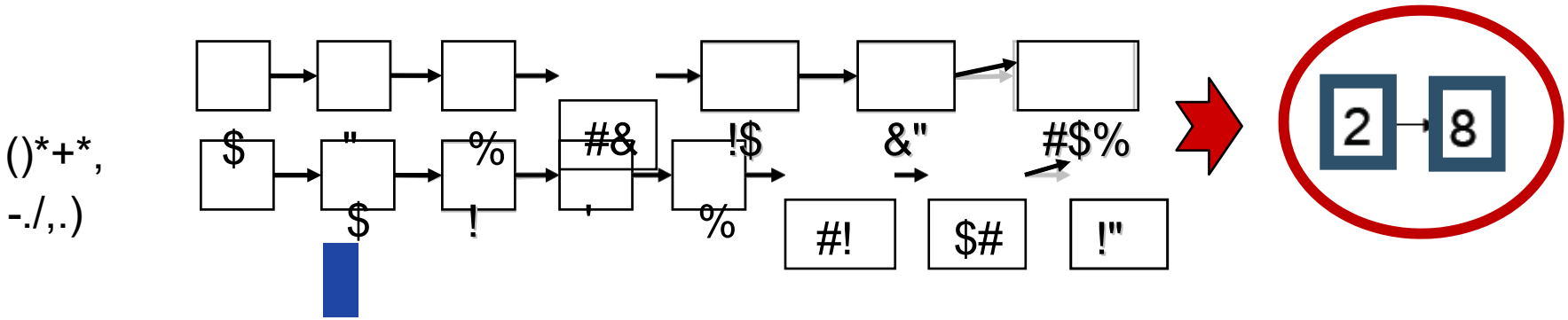


回应查询：合并

- 为了加快搜索速度，将*发布*的信息按文件号排序

合并 ➔同时运行两个列表

如果列表的长度为x和y，则该算法为O（x+y）。



构建反向文件

- 建立一个反向文件是一个 "重要的步骤"。
- 记忆中不可能做到的方法
  - 为每个文件分配一个编号
  - 提取单词（单词，DOC，频率）。
  - 按术语对文件进行排序
  - ➔ 矩阵转置

从每个文件中提取单词

- 指定编号：Doc1➔1, Doc2➔2

  将每份文件的术语提取到一个文件中（每份文件1个文件）或一份文件对应多个文件）。

1

我确实制定了凯撒大帝，我在国会大厦被杀；布鲁图斯

2

凯撒也是如此。高贵的布鲁特斯已经告诉你，凯撒是有野心的

术语	文件 #
I	1
做了	1
颁布	1
朱利叶斯	1
凯撒	1
I	1
是	1
被杀	1
i'	1
的	1
国会	1
残暴	1
被杀	1
我	1
所以	2
让	2
它	2
是	2
与	2
凯撒	2
的	2
崇高的	2
残暴	2
阴阳师	2
告诉你	2
你	2
凯撒	2
是	2
有雄心的	2

对术语-文件文件进行排序

- 按术语和文件的字母顺序排列文件

术语	文件 #
I	1
做了	1
颁布	1
朱利叶斯	1
凯撒	1
I	1
是	1
被杀	1
i'	1
的	1
国会	1
残暴	1
被杀	1
我	1
那么	2
让	2
它	2
是	2
与	2
凯撒	2
的	2
崇高的	2
残暴	2
阴阳师	2
告诉你	2
你	2
凯撒	2
是	2
有雄心的	2

术语	文件 #
有雄心的	2
是	2
残暴	1
残暴	2
国会	1
凯撒	1
凯撒	2
凯撒	2
做了	1
颁布	1
阴阳师	1
I	1
I	1
i'	1
它	2
朱利叶斯	1
被杀	1
被杀	1
让	2
我	1
崇高的	2
那么	2
的	1
的	2
告诉你	2
你	2
是	1
是	2
与	2

对术语-文件文件进行排序

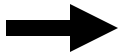
- 对于每一个条款，我们有
  - 从它出现的文件列表中
  - 含有该词的文件数量的百分比

术语	文件 #
有雄心的	2
是	2
残暴	1
残暴	2
国会	1
凯撒	1
凯撒	2
凯撒	2
做了	1
颁布	1
阴阳师	1
I	1
I	1
i'	1
它	2
朱利叶斯	1
被杀	1
被杀	1
让	2
我	1
崇高的	2
所以	2
的	1
的	2
告诉你	2
你	2
是	1
是	2
与	2

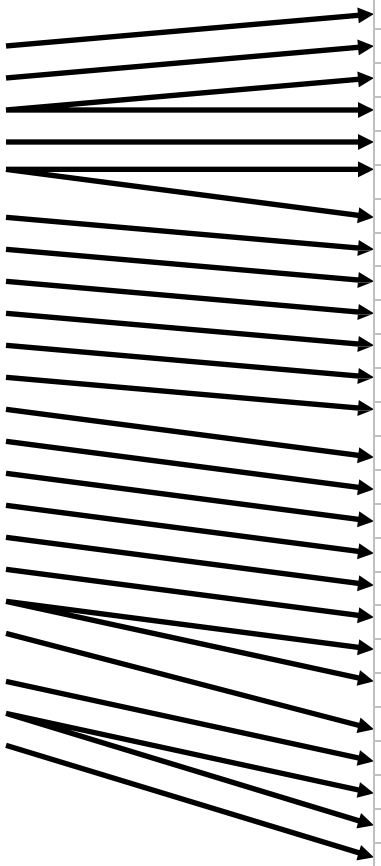
术语	文件 #	频率
有雄心的	2	1
是	2	1
残暴	1	1
残暴	2	1
国会	1	1
凯撒	1	1
凯撒	2	2
做了	1	1
颁布	1	1
阴阳师	2	1
I	1	2
i'	1	1
它	2	1
朱利叶斯	1	1
被杀	1	2
让	2	1
我	1	1
崇高的	2	1
那么	2	1
的	1	1
的	2	1
告诉你	2	1
你	2	1
是	1	1
是	2	1
与	2	1

构建词典并发布

术语	文件 #	频率
有雄心的	2	1
是	2	1
残暴	1	1
残暴	2	1
国会	1	1
凯撒	1	1
凯撒	2	2
做了	1	1
颁布	1	1
阴阳师	2	1
I	1	2
i'	1	1
它	2	1
朱利叶斯	1	1
被杀	1	2
让	2	1
我	1	1
崇高的	2	1
那么	2	1
的	1	1
的	2	1
告诉你	2	1
你	2	1
是	1	1
是	2	1
与	2	1



术语	N个文档	总频率
有雄心的	1	1
是	1	1
残暴	2	2
国会	1	1
凯撒	2	3
做了	1	1
颁布	1	1
阴阳师	1	1
I	1	2
i'	1	1
它	1	1
朱利叶斯	1	1
被杀	1	2
让	1	1
我	1	1
崇高的	1	1
所以	1	1
的	2	2
告诉你	1	1
你	1	1
是	2	2
与	1	1

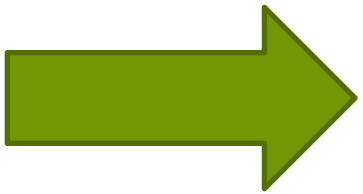


文件 #	频率
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

处理大型收藏品

术语	docID
I	同上。1
医师	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

rme	同上。医师
	2
	2
	2
	2
th	2
esar	2
e	2
ble	2
utus	2
th	2
d	2
u	2
caesar	2
was	2
ambitious	2



按术语排序

然后通过文件)

术语	同上。
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

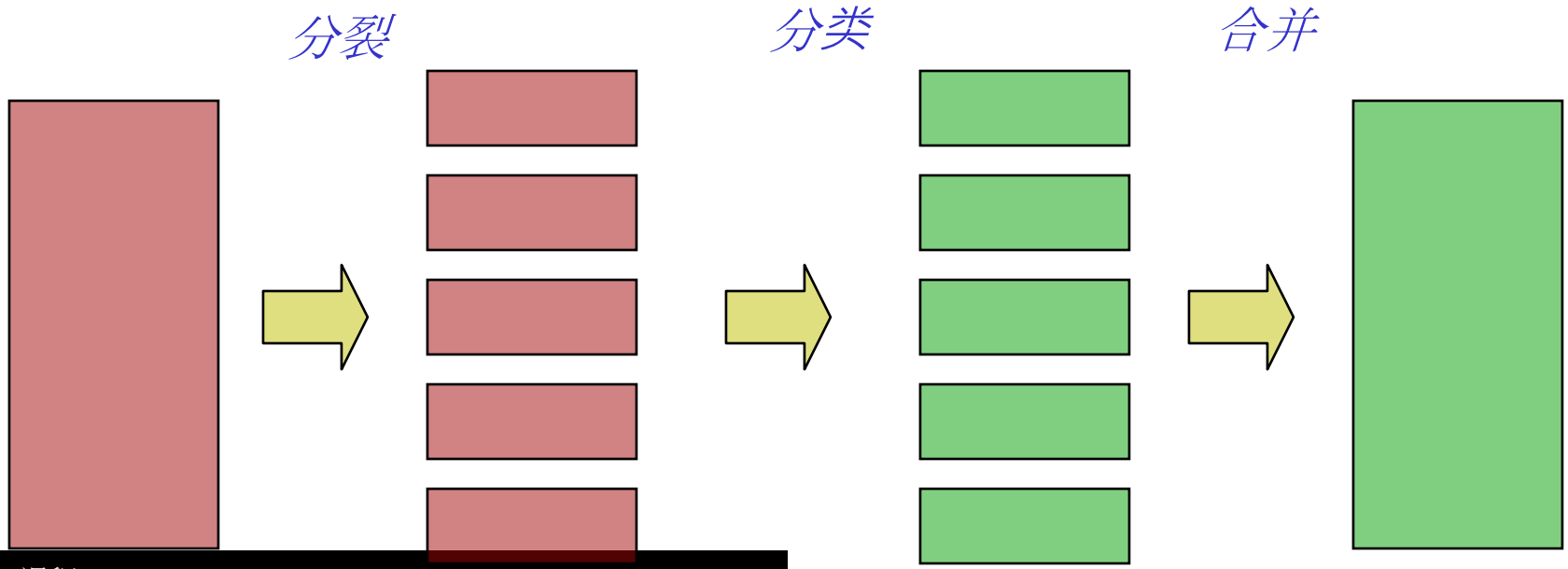
....

分拣阶段的问题

- 分拣，通常在**RAM中完成**
- 但是，……对于大型收藏品来说是不可能的
  - 每次分析一份文件
  - 索引列表只有在过程结束时才是完整的
- **可以在磁盘上进行（在磁盘上进行排序），但速度非常慢**
- 解决方案
  - ➔ 基于分块排序的索引
  - ➔ 分布式索引（➔）。

按区块排序

- 在内存中把集合分成*n*个可管理的部分
- 对每个部分分别排序，并将结果改写在磁盘
- 合并结果➔几种合并方法



分布式索引

- ➔ 分布式索引编制
  - 对于大型集合（网络）来说
  - 一个主服务器运行整个事情（必须是非常安全的）。
  - 它将索引任务划分为一组并行任务
  - 它将每个任务分配给网络中一个空闲的工作机器

分布式索引

- 搜索引擎使用类似的架构
  - 一个分布式文件系统
  - 一个工作调度器：哪个程序在哪个时间在哪个机器上运行
- 谷歌提出的初始架构

  (谷歌文件系统和Map Reduce)
- 在Hadoop项目中开发的开放源码实施方案

谷歌数据中心

- 谷歌的数据中心主要包含商品机，分布在世界各地。

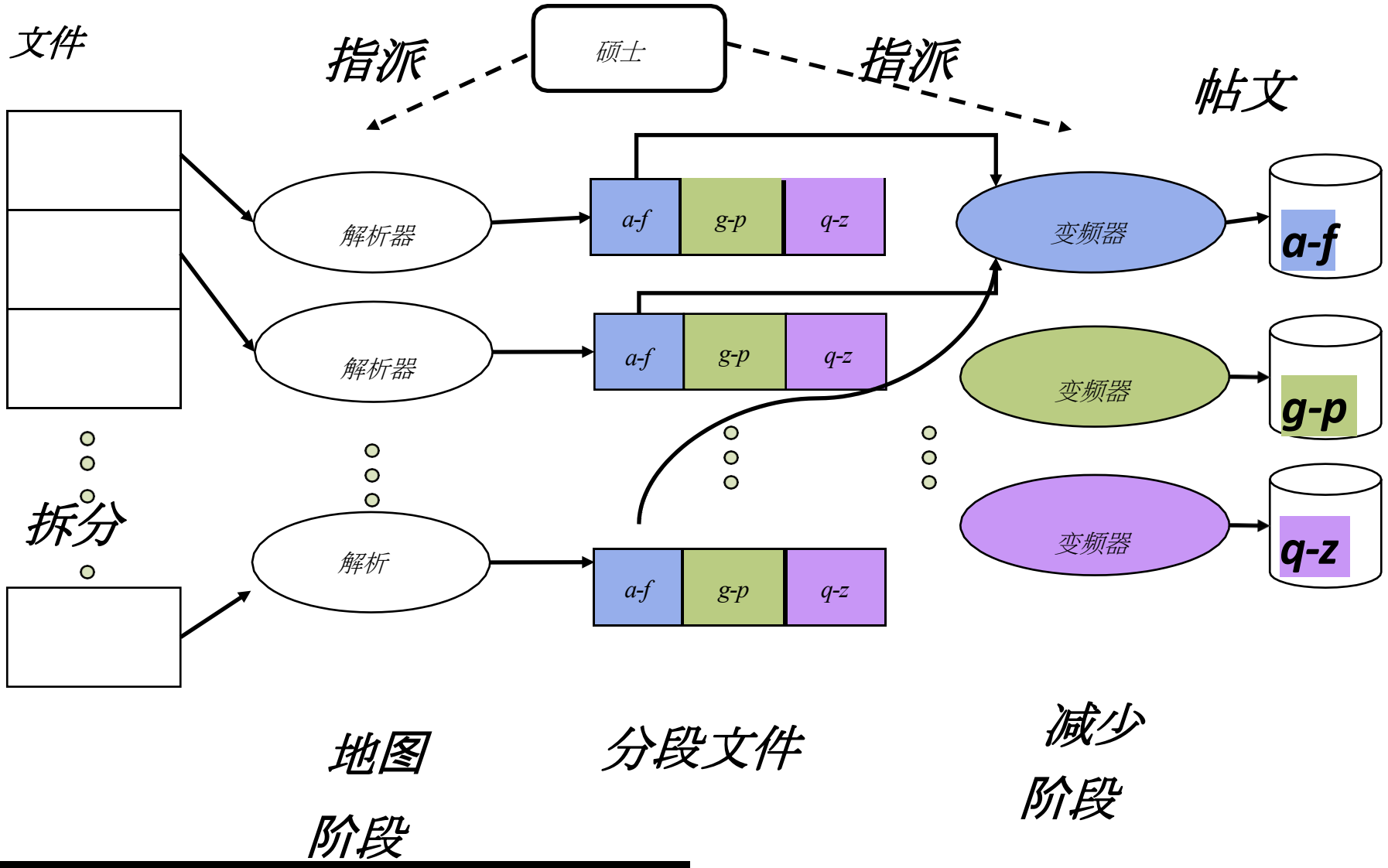
- 估计。
  - 共有100万台服务器，300万个处理器/核。
  - 谷歌每季度安装10万台服务器。
  - 每年2-2.5亿美元的支出

绘图还原（MapReduce

- 原则：
  - 所有的算法都写成两个函数。
    1. 一个执行数据处理的*地图*函数
    2. 一个*减少*函数，将*map*产生的中间结果合并。

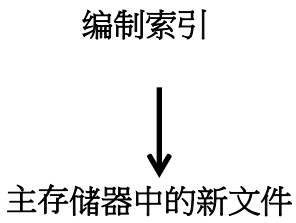
- 兴趣:
  - *地图*任务是在存储数据的机器上执行的
  - 它们是平行运行的

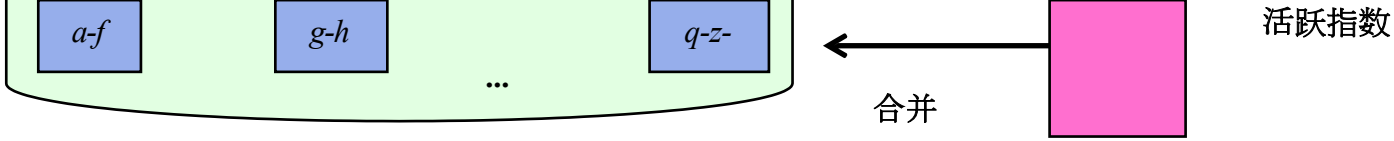
用MapReduce建立索引



对数据流进行索引（如推文）。

- 馆藏往往是动态的（如网络、Twitter）。
  - → **添加**、删除和修改文件
  - → 词典和*张贴的***更新**
- 解决方案（简单但无效）：
  - **从**头开始重建索引
- 一个更好的解决方案。
  - **在内存中保持一个索引，跟踪所有的**变化
  - **一旦索引"满"了，就与磁**盘上的索引合并。
  - 维护已删除文件的矢量

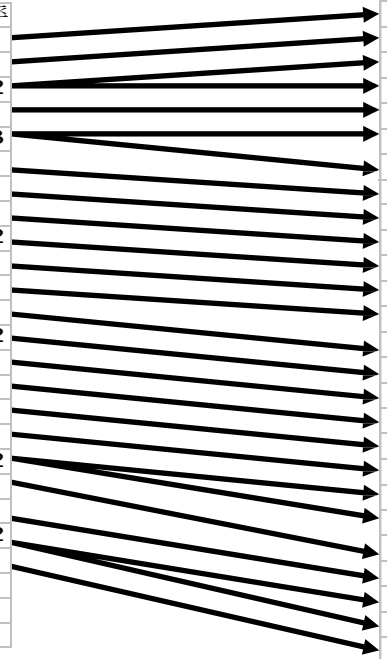



a-f
g-h
q-z-
...
合并
活跃指数

储存的成本

词典

术语	N个文档	总频率
有雄心的	1	1
是	1	1
残暴	2	2
国会	1	1
凯撒	2	3
做了	1	1
颁布	1	1
阴阳师	1	1
I	1	2
i'	1	1
它	1	1
朱利叶斯	1	1
被杀	1	2
让	1	1
我	1	1
崇高的	1	1
所以	1	1
的	2	2
告诉你	1	1
你	1	1
是	2	2
与	1	1



文件 #	频率
2	1
2	1
1	1
2	1
1	1
1	1
2	
1	
1	
2	
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

文

颠倒的文件的简单实现。

- 词典：每个术语20字节，4字节nbDoc，4字节指针
- *发布*：4个字节为文件标识，2个字节为频率。

降低存储成本 → 压缩

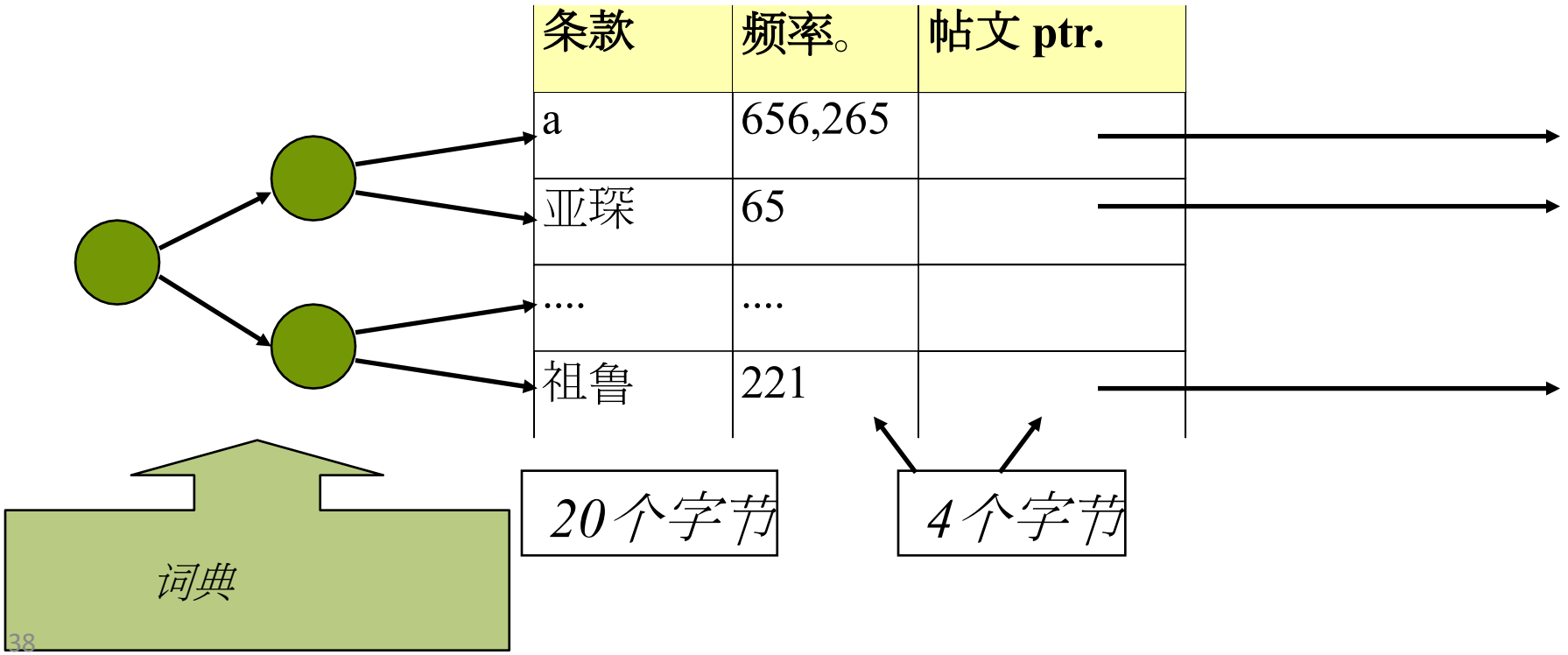
结束

- **关于字典和索引**压缩技术（张贴）的其余课程**可以在以下几**页中找到。

附加文件 词典的压缩(附件)

词典大小

- 固定尺寸的桌子
  - ~400,000个术语；28字节/术语=11.2MB。



大量浪费的空间

- 大量浪费的空间，单字母单词（a，à，…）所占空间与长字相同
  - 有一些话是不能通过的"违宪"。
    "supercalifragilisticexpialidocious" 或使用了 "*氢氯氟烃*"这一术语。

- 平均字数（英文），大约是~8个字符
  - 我们怎样才能利用这个数字（每个术语~8个字符）？

词典压缩

- 将字典存储为一个（长）的字符串
  - 指向下一个术语的指针，给出当前术语的结束。

战斗战斗战斗战斗战斗战斗战斗战斗战斗战斗可燃

400K条款x19
7.6MB

1
5
11

42
11
235
63

4个字节

3个字节

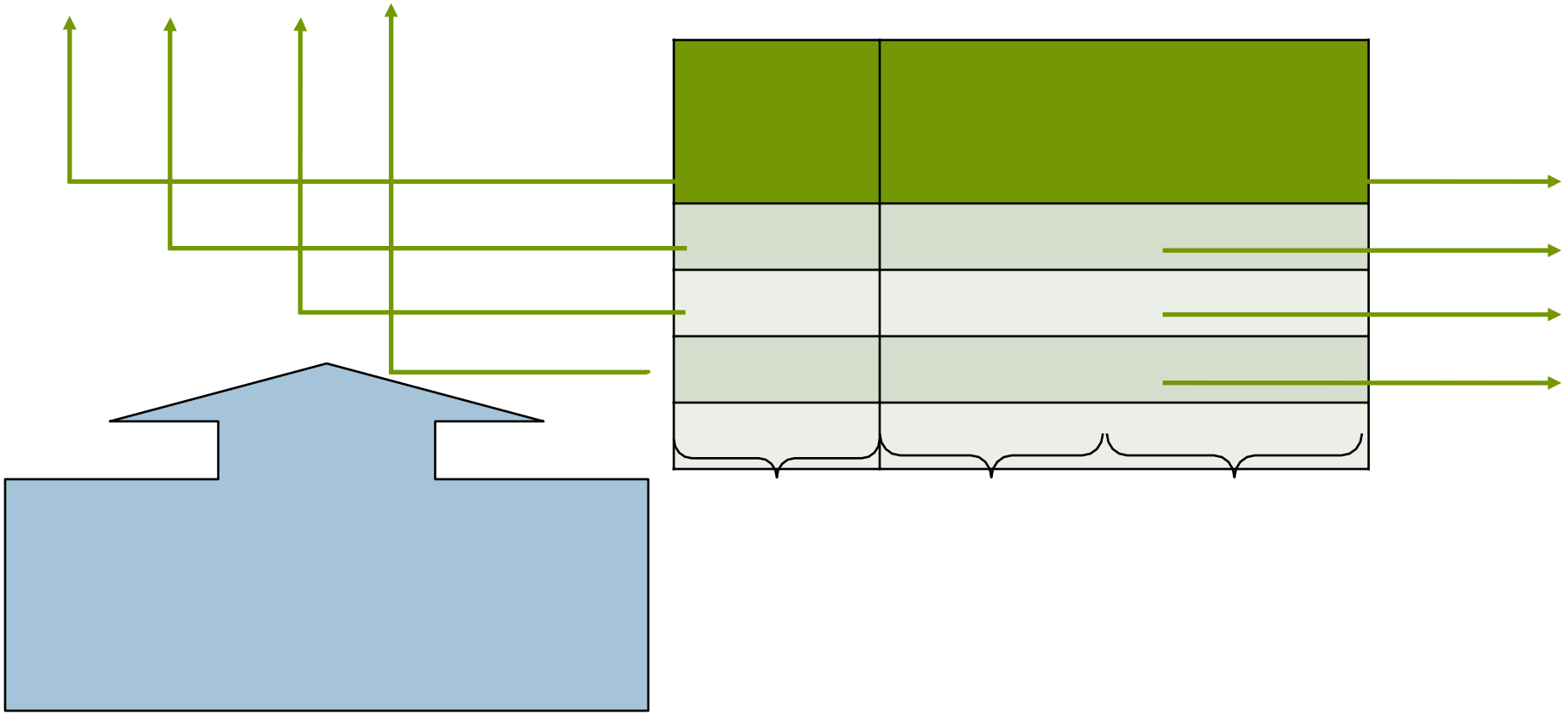
如果一个术语的长度为8字节，那么字典的长度为11字节

➔ *每个术语19个字节，而不是28个字节*

节

练习：确定400个千人条款的最佳指针大小

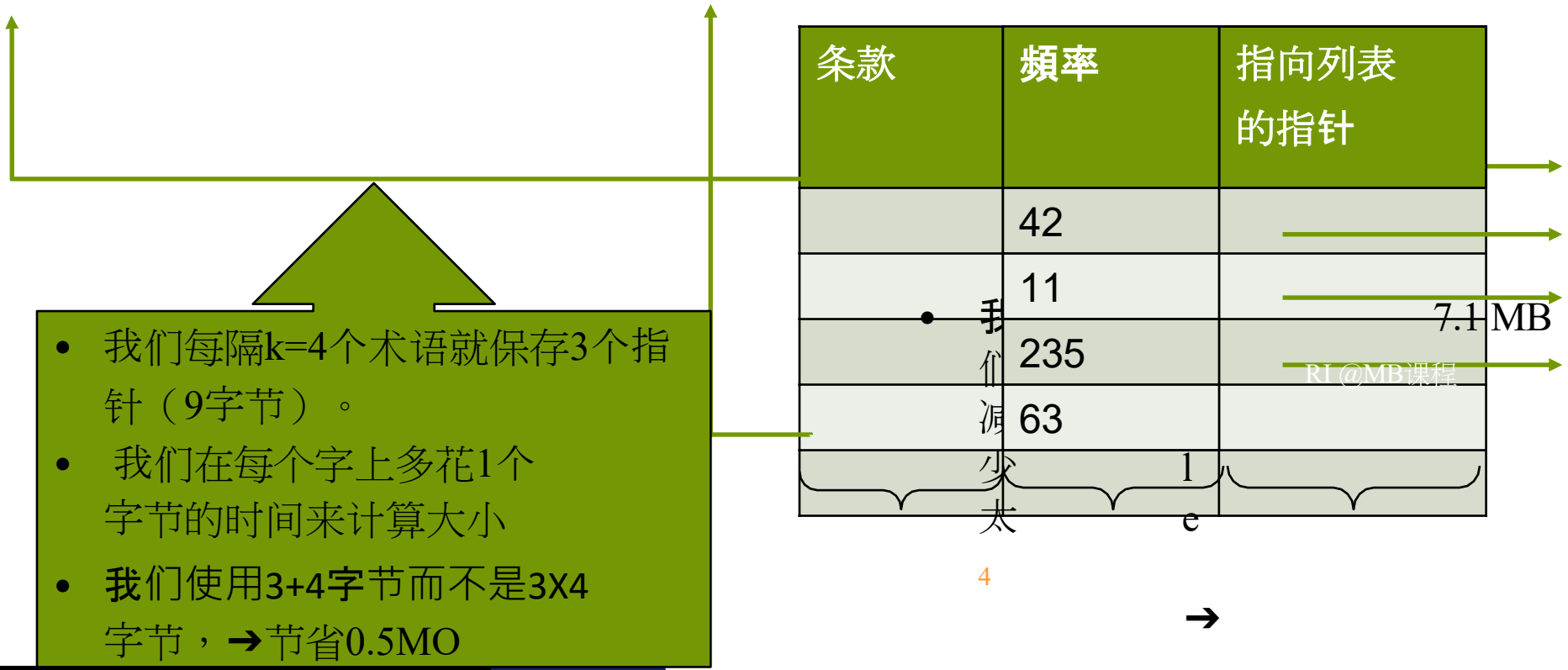
4个字



术语列表的压缩：块状指针

- 存储指向每k个术语的指针（例如：k=4）。
- 需要添加一个字节来存储术语的大小

4coma6combat5combe11combination7combler11fuel



3个字节

4个字节

4个字节

锻炼身体

- 为什么不增加K呢？
- 估计索引所需的空间（什么7.6 MO），*k=4，8*和*16.*

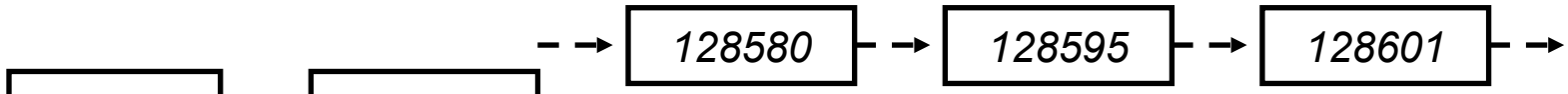
压缩*发帖（待读）*

压缩*发布*的内容

- *发帖*文件至少比字典大10倍。
  - *发布*是由DocId(s)（文件编号）➔一个4字节的整数组成的。
  - 在最好的情况下，对于1M的文件，在对数$_2$ 1,000,000≈20比特

频繁使用的术语不多，稀有术语很多

  - "***arachnocentric***
    "在整个文献库中可能出现一次➔，所以对于一个有一百万个文件的文献库来说，20比特应该是足够的。
  - **"the**
    "可能出现在所有的文件中，所以可能有20bitsX1M=20M比特来存储这个列表（这太多了！！）。



压缩*发布*的内容

- *发布的*文件编号按升序存储。
  - *计算机*：33,47,154,159,202 …
- 存储docid(s)之间的间隙，而不是docids。
  - 33,14,107,5,43 …
- 希望能够用不到20比特的时间来存储间隙（间隔）（比我们保留docIds的比特数少）。

压缩发布的内容

例如，三个张贴条目

	encoding	postings list									
THE	docIDs	...		283042		283043		283044		283045	...
	gaps				1		1		1		...
COMPUTER	docIDs	...		283047		283154		283159		283202	...
	gaps				107		5		43		...
ARACHNOCENTRIC	docIDs	252000		500100							
	gaps	252000	248100								

20比特对于"the"来说仍然是过度的

压缩发布的内容

- **目的。**
  - 对于***蛛丝马迹***，**使用的是~20比特/散布。**
  - 对于，你可以使用**~1比特/**间隙。

- **→**
  - 对于一个偏差值l，我们希望使用尽可能少的比特（高于log的整数$_2$ l）。

  在实践中，我们四舍五入到下一个字节

- **→** *可变*编码

压缩发布的内容

- 可变编码
  - 一个字节的七位用来表示数字（间隙），最后一位是延续位*c*。
  - 如果$1 \le 127$，7位就足够了➔ $c = 1$。
  - 否则，$c = 0$，我们继续进行下一个字节。
  - $c = 1$总是意味着数字在这个字节结束。

例子

docIDs	824	829	215406
差距		5	214577
VB代码	00000110 10111000	10000101	00001101 00001100 10110001

以字节连接的方式存储的帖子

00000110 10111000 10000101 00001101 00001100 10110001

延续位

对于小间隙（5），VB使用整个字节。

总结

- 索引是IR过程的核心
  - 允许选择描述文件内容的重要术语
  - 通常是一个简单的术语列表 ➔ 词汇袋模型
  - 理想情况下是语言+统计学的结合➔，现在由统计学模型主导。
- 多媒体索引（图像、视频等）。
  - 语境（以围绕对象的文本为例）➔我们又回到了文本的IR问题上
  - 内容(基于信号)