

信息组织与检索

- 词汇表和倒排记录表

主讲人：张蓉

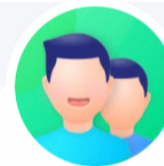
华东师范大学 数据科学与工程学院

自我介绍

- 姓名：张蓉
- 电子邮件：rzhang@dase.ecnu.edu.cn
- 隶属单位：数据科学与工程学院
- 研究工作：数据库系统

助教

- 王清帅：博士二年级
 - 52194501004@stu.ecnu.edu.cn
 - 叶田地：博士一年级
 - 18365225454@163.com
 - 每周四晚实验课
- 扫描入群，请改名：学号+姓名
- 示例：20200108飞飞



2020信息检索

群号：343384720



扫一扫二维码，加入群聊。



提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

上一讲回顾

➤ 倒排索引的基本知识

— 组成: 词典和倒排记录表

| | | | | | | | | | |
|--------|---|---|---|---|----|----|----|-----|-----|
| BRUTUS | → | 1 | 2 | 4 | 11 | 31 | 45 | 173 | 174 |
|--------|---|---|---|---|----|----|----|-----|-----|

| | | | | | | | | | | |
|--------|---|---|---|---|---|---|----|----|-----|-----|
| CAESAR | → | 1 | 2 | 4 | 5 | 6 | 16 | 57 | 132 | ... |
|--------|---|---|---|---|---|---|----|----|-----|-----|

| | | | | | |
|-----------|---|---|----|----|-----|
| CALPURNIA | → | 2 | 31 | 54 | 101 |
|-----------|---|---|----|----|-----|

— 构建中的关键步骤: 按词项排序

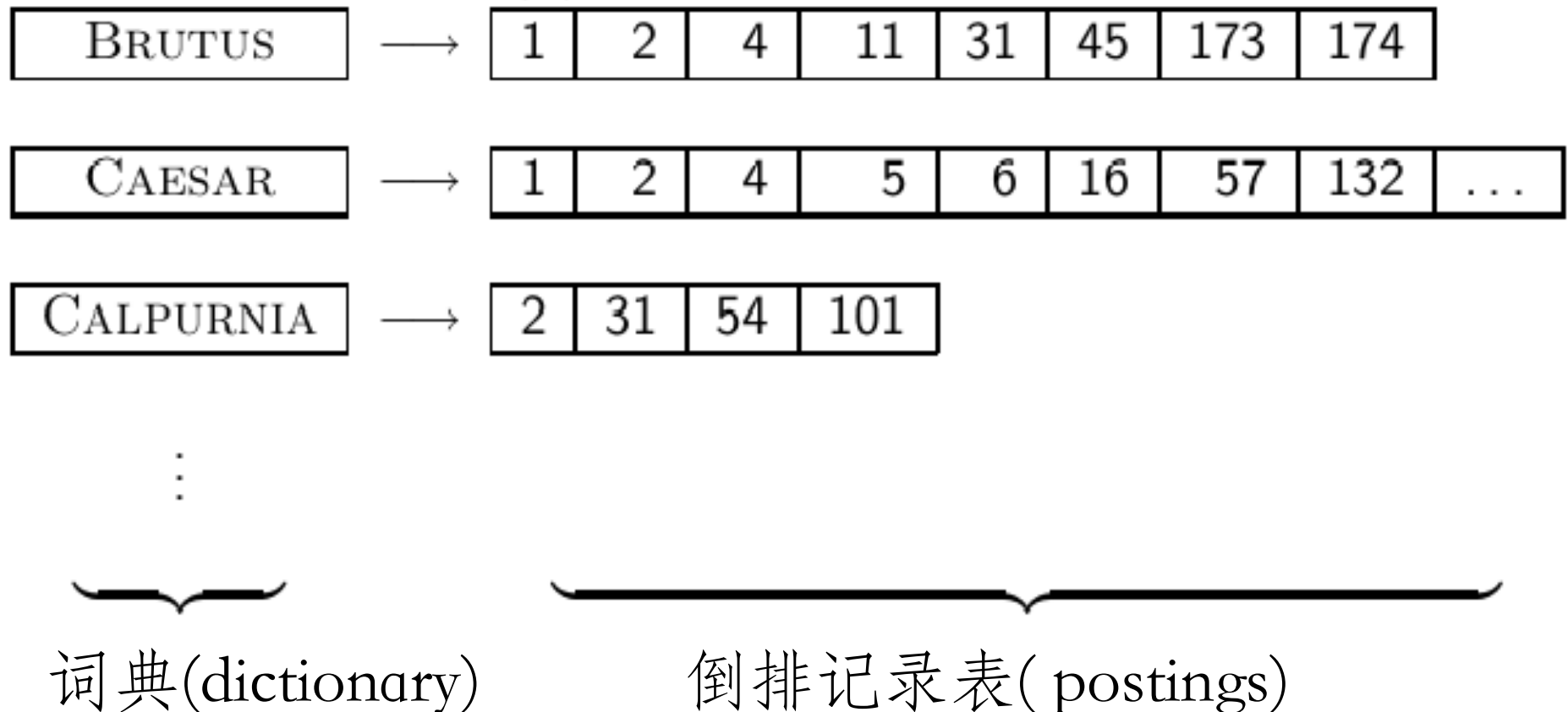
➤ 布尔查询的处理

— 线性时间内求交集

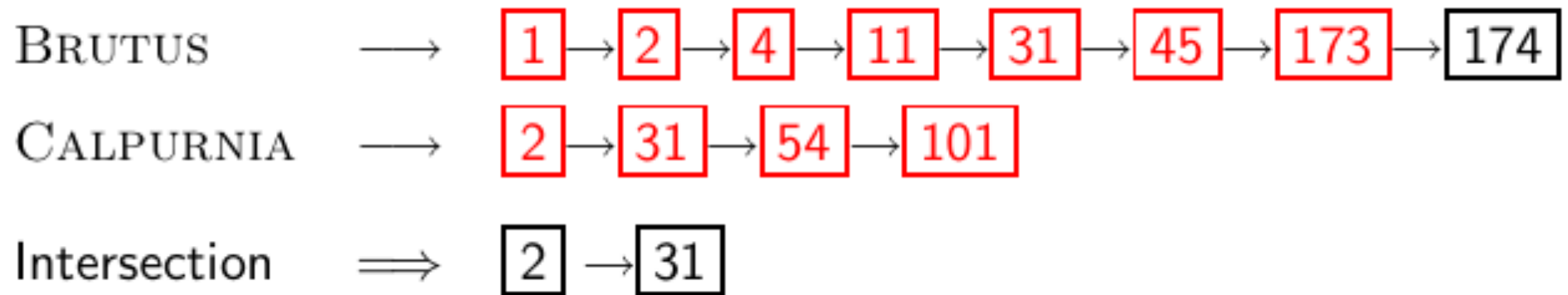
— 查询优化

倒排索引

对每个词项 t , 保存所有包含 t 的 文档列表



倒排记录表的合并



查询优化

- 按照表从小到大(即df从小到大)的顺序进行处理:
 - 每次从最小的开始合并

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

相当于处理查询 (*Calpurnia AND Brutus*) *AND Caesar*.

更通用的优化策略

- e.g., (madding OR crowd) AND (ignoble OR strife)
 - 每个布尔表达式都能转换成上述形式(合取范式)
- 获得每个词项的df
- (保守)通过将词项的df相加，估计每个OR表达式对应的倒排记录表的大小
- 按照上述估计从小到大依次处理每个OR表达式.

本讲的内容

- 索引构建过程(预处理)
- 如何对索引文档进行处理来得到词典
 - 理解文档(document)的概念
 - 词条化(tokenization), 理解词条(token)的概念
 - 词项生成, 理解词项(term)的概念
- 倒排记录表
 - 更快的合并算法: 跳表法(skip list)
 - 短语查询的处理及带位置信息的倒排索引

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

回顾倒排索引构建

待索引文档



Friends, Romans, countrymen.

Tokenizer

词条化工具

词条流

Friends

Romans

Countrymen

Linguistic modules

语言分析工具

修改后的词条

friend

roman

countryman

Indexer

倒排索引

friend

roman

countryman

2

4

1

2

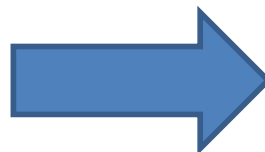
13

16

文档分析

➤ 文档格式处理

— pdf / word / excel / html?



➤ 文档语言识别

➤ 文档编码识别 GBK/UTF-8/Unicode

文档语言识别和编码识别理论上都可以看成分类问题，基于后面章节的分类方法可以处理。但是实际中，常常采用启发式方法.....

多格式/语言并存

- 待索引文档集可能同时包含多种语言的文档
 - 在同一索引中词汇表中包含来自多个语言的词项
- 有时文档或者其部件中包含多种语言/格式
 - 法语邮件中带一个德语的pdf格式附件
- 如何确定索引的单位？
 - 文件为单位？
 - 邮件为单位？
 - 如果邮件带有5个附件，怎么办？
 - 一组文件？（比如采用html格式写的某个PPT文档）

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

词条和词项

TOKENS AND TERMS

词条化(Tokenization)

- 输入: “Friends, Romans and Countrymen”
- 输出: 词条(Token)
 - Friends
 - Romans
 - Countrymen
- 词条 就是一个字符串实例
- 词条在经过进一步处理之后将放入倒排索引中的词典中
 - 词项
- 词条化中的问题-词条如何界定?

语言好难



生活不止眼前的苟且
还有读不懂的诗和到不了的远方

► 难度渐变：

- 三分钟韩语，
- 三小时英语，
- 三天的法语，
- 三个月的日语，
- 三年的德语，
- 三百年阿拉伯语

汉语

阿拉伯

优美的语言？

日语、意大利语、西班牙语

“さくら”，发音 [s]a[ku]a



汉语



“你的牙真好看。”

意思?

“哦，那是假的。”

(✓)

“真的?”

。

问：牙是真的还是假的?

意思相同。

小明：小意思，小意思。

领导：你这人真有意思。

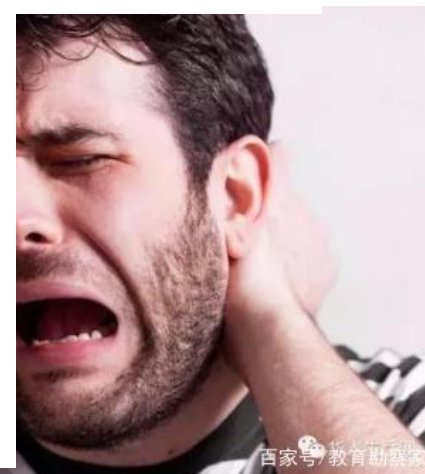
小明：其实也没有别的意思。领导：那我就不好意思了。

角

小明：是我不好意思。
所以选 D。

冬天能穿多少穿多少，
夏天能穿多少穿多少

：服是比较肥了。



百家号/教育观察家

词条化

➤ 一系列问题:

- Finland' s capital →
- Finland? Finlands? Finland' s?
- Hewlett-Packard → 看成Hewlett 和 Packard 两个词条?
 - ❖ state-of-the-art:
 - ❖ co-education
 - ❖ lowercase, lower-case, lower case ?
- San Francisco: 到底是一个还是两个词条?
 - ❖ 如何判断是一个词条? New York

词条化中数字的处理

- 3/20/91 Mar. 12, 1991 20/3/91
- 55 B.C.
- B-52
- PGP 密钥: 324a3df234cb23e
- (800) 234-2333
 - 通常中间有空格
 - 早期的IR系统可能不索引数字
 - ❖ 但是数字却常常很有用: 比如在Web上查找错误代码
 - ❖ (一种处理方法是采用n-gram: 见第三讲)
 - 元数据是分开还是一起索引
 - ❖ 创建日期、格式等等

专有名词(Named Entity)

► 文本中实体

- 例如人名、地名、组织机构名
- 人名：刘德华、张三
- 地名：北京、苏州
- 组织机构名：苏州大学
- 歌名：冬天里的一把火
- 其他等等



实体识别（例子）

➤ 识别：人名、时间、地名、机构名、职务、。。。

➤ 马云1964年9月10日生于浙江省杭州市，祖籍浙江省嵊州市谷来镇，阿里巴巴集团主要创始人，现担任阿里巴巴集团董事局主席、日本软银董事、大自然保护协会中国理事会主席兼全球董事会成员、华谊兄弟董事、生命科学突破奖基金会董事、联合国数字合作高级别小组联合主席。

语言问题：法语和德语

➤ 法语

- L ‘ensemble → 到底是一个还是两个词条?
 - ❖ L ? L’ ? Le ?
 - ❖ 但是常常希望 l’ ensemble 能和un ensemble匹配
 - 至少在2003年以前，Google没有这样处理
 - » 国际化问题!

➤ 德语中复合名词连写

- Lebensversicherungsgesellschaftsangestellter
- ‘life insurance company employee’
- 德语检索系统往往要使用一个复合词拆分的模块，而且该模块对检索结果的提高有很大帮助(可以提高15%)

语言问题：韩语

- 역에 가고 싶습니다(我想去车站)
- 中间有空格和拉丁语系一样
- 语法体系和日语接近
- 输入法

어



语言问题：中文和日文

- 中文和日文词之间没有间隔：
 - 莎拉波娃现在居住在美国东南部的佛罗里达。
 - 分词结果无法保证百分百正确，“和尚”
- 日文中可以同时使用多种类型的字母表
 - 日期/数字可以采用不同的格式



而终端用户可能完全用平假名方式输入查询！
情報不足 = じょうほうぶそく

中文分词(Chinese Word Segmentation)

- 对于中文，分词的作用实际上是要找出一个个的索引单位
- 例子：李明天天都准时上班
- 索引单位
 - 字：李明天天都准时上班
 - ❖ 索引量太大，查全率百分百，但是查准率低，比如查“明天”这句话也会出来
 - 词：李明天天都准时上班
 - ❖ 索引量大大降低，查准率较高，查全率不是百分百，而且还会受分词错误的影响，比如上面可能会切分成：李明天天都准时上班，还有：他和服务人员照相
 - 字词混合方式/k-gram/多k-gram混合
 - 一般原则，没把握的情况下细粒度优先

中文分词和检索

➤ 并非分词精度高一定检索精度高

- 评价标准不同
- 分词规范问题：鸡蛋、鸭蛋、鹌鹑蛋.....
- 目标不同

➤ 检索中的分词：

- 查询和文档切分采用一致的分词系统
- 速度快
- 倾向细粒度，保证召回率
- 多粒度并存

➤ 搜索引擎中的分词方法

- 猜想：大词典+统计+启发式规则，不明觉厉



语言问题：阿拉伯文

- 阿拉伯文(或希伯来文)通常从右到左书写，但是某些部分(如数字)是从左到右书写
- 词之间是分开的，但是单词中的字母形式会构成复杂的连接方式

استقلت الجزائر في سنة 1962 بعد 132 عام من الاحتلال الفرنسي.

← → ← →

← 开始

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’
- 在Unicode编码方式下，表面的表示方式很复杂，但是存储上倒是十分直接

多种语言



- 【中文】 谢谢
- 【英语】 Thank you
- 【菲律宾语】 Salamat Do (撒拉玛特朵)
- 【日语】 ありがとう (阿里嘎豆)
- 【韩语】 감사합니다. (勘三哈咪瘩)
- 【马来语】 terima Kasih (得力马卡系)
- 【越南语】 Cám o*n (嘉蒙)
- 【泰语】 kob-khun (寇布库恩)
- 【印度语】 dhanyavaad (达尼阿瓦德)
- 【希伯来语】 toda (透达)
- 【俄语】 Spasibo (思巴喜柏)
- 【德语】 Danke (但可)
- 【阿拉伯语】 shokran (休克朗)
- 【法语】 Merci (梅呵西)
- 【西班牙语】 Gracias (格拉喜亚思)
- 【意大利语】 Grazie (格啦姬)

停用词 (一)

- 根据停用词表(stop list), 将那些最常见的词从词典中去掉。
比如直观上可以去掉:
 - 一般不包含语义信息的词: the, a, and, to, be
 - 汉语中的 “的”、“得”、“地” 等等。
 - 这些词都是高频词: 前30个词就占了 ~30% 的倒排记录表空间
- 但是现代信息检索系统中倾向于不去掉停用词:
 - 在保留停用词的情况下, 采用良好的压缩技术(第五章)后, 停用词所占用的空间可以大大压缩, 最终它们在整个倒排记录表中所占的空间比例很小
 - 采用良好的查询优化技术(第七章)基本不会增加查询处理的开销
 - 所谓的停用词并不一定没用, 比如: 短语查询: “King of Denmark”、歌曲名或者台词等等: “Let it be”, “To be or not to be”、“关系型” 查询 “flights to London”

词条归一化(Normalization)成词项

- 将文档和查询中的词归一化成同一形式：
 - U.S.A. 和 USA
- 归一化的结果就是词项，而词项就是我们最终要索引的对象
- 可以采用隐式规则的方法来表示多个词条可以归一成同一词项，比如
 - 剔除句点
 - ❖ U.S.A., USA \ USA
 - 剔除连接符
 - ❖ anti-discriminatory, antidiscriminatory \ antidiscriminatory

归一化中的语言问题

- 重音符: 如法语中 *résumé* vs. *resume*.
- 日耳曼语系中的元音变化: 如德语中的 *Tuebingen* vs. *Tübingen*
 - 应该是一致的
- 最重要的准则:
 - 用户在输入查询时遇到这些词如何输入?
- 即使在有重音符号的语言中, 用户也往往不输入这些符号
 - 常常归一化成不带重音符号的形式
 - ❖ *Tuebingen*, *Tübingen*, *Tubingen* \ *Tubingen*

归一化中的语言问题

➤ 时间格式

- 7月30日 vs. 7/30
- 日语中用假名或者汉字表示日期

➤ 词条化和归一化都可能与语言相关，因此必须要做语言识别

Morgen will ich in MIT ...

是德语的“mit”吗？

➤ 另外，谨记要将文档和查询中的同义词归一化成同一形式

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

英文词处理？

语言太多太乱，专心做英文吧！！

大小写问题

- 可以将所有字母转换成小写形式
 - 例外: 句中的大写单词?
 - ❖ e.g., General Motors (GM, 通用公司)
 - ❖ Fed (美联储) vs. fed (饲养)
 - ❖ SAIL (印度钢铁管理局) vs. sail (航行)
 - 通常情况下将所有字母转成小写是一种很合适的方式, 因为用户倾向于用小写方式输入

- Google 的例子:
 - 查询 C.A.T.
 - 排名第一的结果是 “cat” 而不是 Caterpillar Inc.

归一化成词项

- 除了前面互换方式(即能够归一化成同一词项的词条之间完全平等, 可以互换)之外, 另一种方式是非对称扩展 (asymmetric expansion)
- 一个非对称扩展更适合的例子
 - 输入: window 搜索: window, windows
 - 输入: windows 搜索: Windows, windows, window
 - 输入: Windows 搜索: Windows
 - 为什么反过来不行?
- 这种方法可能更强大, 但是效率低一些

同义词词典(Thesauri)及soundex方法

➤ 同义词和同音/同形异义词的处理

- E.g., 手动建立词典，记录这些词对

 - ❖ car = automobile color = colour 电脑/计算机

- 利用上述词典进行索引

 - ❖ 当文档包含 automobile 时，利用 car-automobile 进行索引

- 或者对查询进行扩展

 - ❖ 当查询包含 automobile 时，同时也查 car

➤ 拼写错误的处理(Clinton→Klinten)

- 一种解决方法是soundex方法，基于发音建立词之间的关系(Soundex方法将在后面介绍)

词形归并(Lemmatization)

➤ 将单词的屈折变体形式还原为原形

➤ 例子:

— am, are, is → be

— car, cars, car's, cars' → car

— the boy's cars are different colors → the boy car be
different color

➤ 词性归并意味中将单词的变形形式“适当”还原成一般词典中的单词形式

— found → find? found?

词干还原 (Stemming)

- 将词项归约(reduce)成其词干(stem), 然后再索引
- “词干还原” 意味着词缀的截除
 - 与语言相关
 - 比如, 将 automate(s), automatic, automation都还原成 automat

for example compressed and compression are both accepted as equivalent to compress.



for exampl compress and
compress ar both accept
as equival to compress

Porter算法

- 英语词干还原中最常用的算法
 - 结果表明该方法不差于其他的词干还原方法
- 一些规定 + 5 步骤的归约过程
 - 这些步骤有先后顺序
 - 每一步都包含一系列命令
- 一些规定，比如：选择可应用规则组中包含最长词缀的规则
 - SSES → SS caresses → caress
 - S → cats → cat

Porter中的典型规则

➤ $sses \rightarrow ss$

➤ $ies \rightarrow i$

➤ $ational \rightarrow ate$

➤ $tional \rightarrow tion$

➤ 规则适用条件的表达

— $(m > 1) \text{ EMENT} \rightarrow$

❖ $\text{replacement} \rightarrow \text{replac}$

❖ $\text{cement} \rightarrow \text{cement}$

Martin Porter



- 英国人，剑桥大学
- 2000年度 Tony Kent Strix award得主
 - 信息检索领域一个著名的奖项
- Porter's stemmer，有很多语言的版本
- Snowball 工具，支持多种语言的stemming(法语、德语、葡萄牙语、西班牙语挪威语等等)

https://en.wikipedia.org/wiki/Martin_Porter

其他词干还原工具(stemmer)

- Lovins: <http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm>
- 单遍扫描，最长词缀剔除(大概 250条规则)
- 全部基于词形分析 - 对于检索来说最多只能提供一定的帮助(at most modest benefits for retrieval)
- 词干还原及其它归一化工作对检索的帮助
 - 英语：结果要一分为二，对某些查询来说提高了召回率，但是对另外一些查询来说降低了正确率
 - 比如, operative (dentistry) ➔ oper
 - 对西班牙语、德语、芬兰语等语言非常有用
 - 其中对于芬兰语有30% 的性能提高!

语言特性

- 上述很多转换处理具体实现时
 - 都与语言本身有关，并且
 - 常常和具体应用有关
- 上述过程可以插件方式植入索引过程
- 存在很多开源和商业插件可用

词典入口示意图

| |
|-----------------------------|
| <i>ensemble.french</i> |
| <i>時間.japanese</i> |
| <i>MIT.english</i> |
| <i>mit.german</i> |
| <i>guaranteed.english</i> |
| <i>entries.english</i> |
| <i>sometimes.english</i> |
| <i>tokenization.english</i> |



可以按(或不按)语言分组，后面还会讲到

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

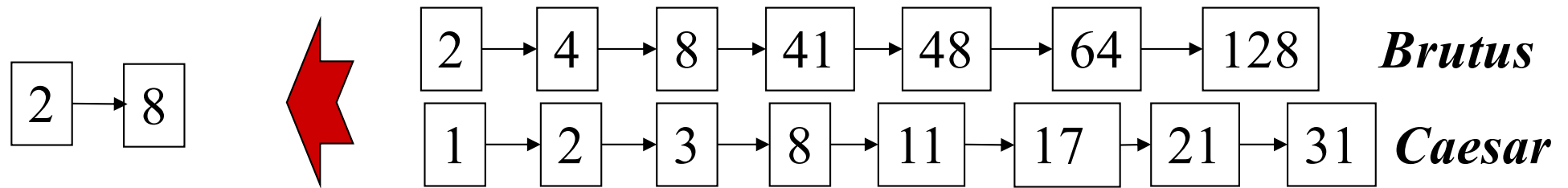
⑤ 短语查询

快速倒排表合并—跳表法

FASTER POSTINGS MERGES: SKIP POINTERS/SKIP LISTS

基本合并算法的回顾

➤ 两个指针，同步扫描，线性时间



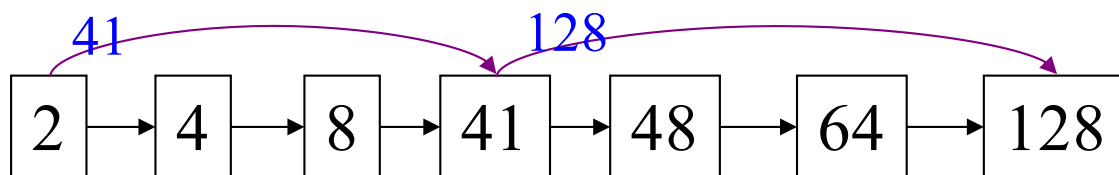
两个表长度为 m 和 n 的话，上述合并时间复杂度为 $O(m+n)$

能否做得更好？答案是可以(如果索引不常变化的话)

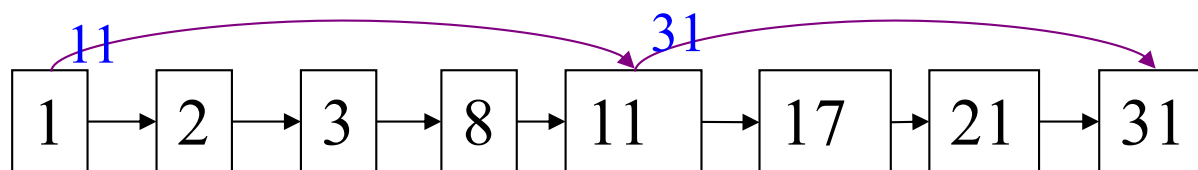
索引构建时为倒排记录表增加跳表指针

➤ 为什么可以加快速度?

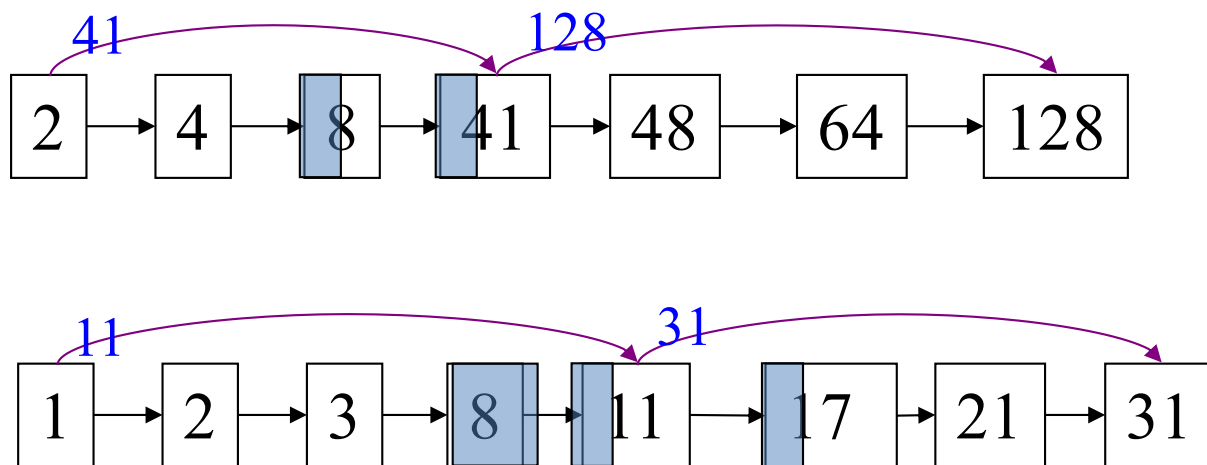
- 可以跳过那些不可能的检索结果



➤ 如何做?也就是在什么地方加跳表指针?



基于跳表指针的查询处理



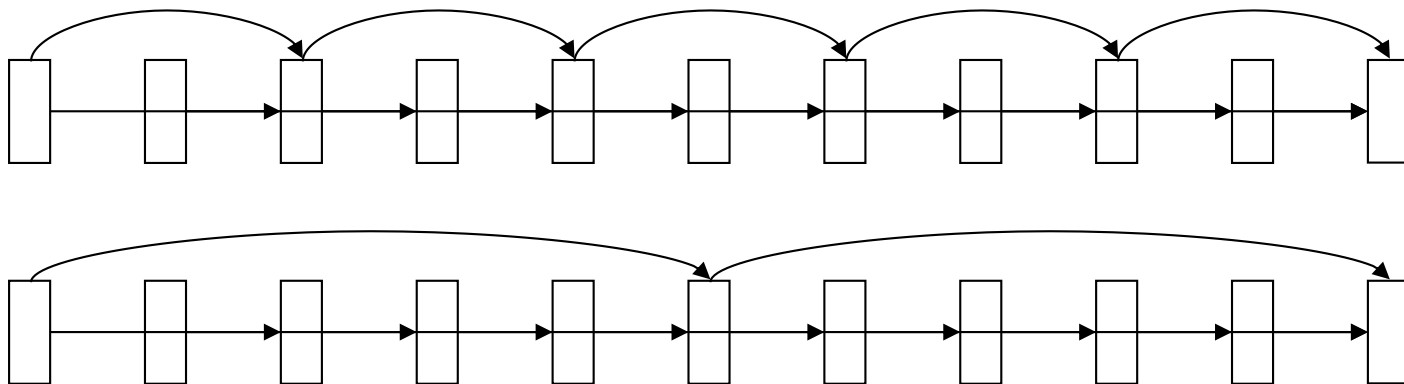
假定匹配到上下的指针都指向8，接下来两个指针都向下移动一位。

比较41和11，11小

此时看11上面的跳表指针，指向31，31仍然比41小，于是下指针可以直接跳过中间的11、17、21、31

跳表指针的位置

- 指针数目过多过少都不合适，要有一个均衡性：
- 指针越多 → 跳步越短 ⇒ 更容易跳转，但是需要更多的与跳表指针指向记录的比较
 - 指针越少 → 比较次数越少，但是跳步越长 ⇒ 成功跳转的次数少



跳表指针的位置

- **简单的启发式策略**：对于长度为 L 的倒排记录表，每 \sqrt{L} 处放一个跳表指针，即均匀放置。均匀放置方法忽略了查询词项的分布情况
- 如果索引相对静态，均匀方式方法是一种很简便的方法，但是如果索引经常更新造成 L 经常变化，均匀方式方法就很不方便
- 跳表方式在过去肯定是有用的，但是对于现代的硬件设备而言，如果合并的倒排记录表不能全部放入内存的话，上述方式不一定有用 (Bahle et al. 2002)
 - **更大的倒排记录表(含跳表)的 I/O 开销可能远远超过内存中合并带来的好处**

提纲

① 上一讲回顾

② 文档

③ 词项

- 通常做法+非英语处理
- 英语

④ 跳表指针

⑤ 短语查询

短语查询及位置索引

PHRASE QUERIES AND POSITIONAL INDEXES

短语查询

- 输入查询作为一个短语整体，
 - 比如 “stanford university” “中国 科学院”
- 因此，句子 “I went to university at Stanford” 就不应该是答案 （“我去了中国 农业 科学院”）
 - 有证据表明，用户很容易理解短语查询的概念，这也是很多搜索引擎“高级搜索”中比较成功的一个功能。
 - 但是很多查询是隐式短语查询， information retrieval textbook → [information retrieval] textbook
- 这种情况下，倒排索引仅仅采用如下方式是不够的
 - Term + docIDs
 - “information retrieval textbook”

第一种做法: 双词(Biword)索引

- 每两个连续的词组成词对(作为短语)来索引
- 比如文本片段 “Friends, Romans, Countrymen”
会产生两个词对
 - friends romans
 - romans countrymen
- 索引构建时, 将每个词对看成一个词项放到词典中
- 这样的话, 两个词组成的短语查询就能直接处理

更长的短语查询处理

- 例子： stanford university palo alto，处理方法：
将其拆分成基于双词的布尔查询式：
- stanford university AND university palo AND palo alto
- $A B C = AB \ BC : .. \textcolor{red}{AB}... \textcolor{red}{BC}/.. \textcolor{red}{BC}.... \textcolor{red}{AB}../ \ ABC...$
- 如果不检查文档，无法确认满足上述表达式的文档是否真正满足上述短语查询。也就是说满足上述布尔表达式只是满足短语查询的充分条件。

很难避免伪正例的出现！**为什么？**

扩展的双词 (Extended Biword)

- 对待索引文档进行词性标注
- 将词项进行组块，每个组块包含名词 (N) 和冠词/介词 (X)
- 称具有NX*N形式的词项序列为扩展双词(extended biword)
 - 将这样扩展词对作为词项放入词典中
- 例子: catcher in the rye (书名: 麦田守望者)
 - N X X N
- 查询处理: 将查询也分析成 N和X序列
 - 将查询切分成扩展双词
 - 在索引中查找: catcher rye

关于双词索引

- 会出现伪正例子
- 由于词典中词项数目剧增，导致索引空间也激增
 - 如果3词索引，那么更是空间巨大，无法忍受
- 双词索引方法并不是一个标准的做法 (即倒排索引中一般不会全部采用双词索引方法)，但是可以和其他方法混合使用

第二种解决方法: 带位置信息索引 (Positional indexes)

- 在倒排记录表中, 对每个term在每篇文档中的每个位置(偏移或者单词序号)进行存储:
 - <term, 出现term的文档篇数;
 - doc1: 位置1, 位置2 ...;
 - doc2: 位置1, 位置2 ...;
 - 等等>

位置索引的例子

- 对于输入的短语查询，需要在文档的层次上进行迭代(不同位置上)合并
- 不仅仅简单合并，还要考虑位置匹配

<**be**: 993427;

1: 7, 18, 33, 72, 86, 231;

2: 3, 149;

4: 17, 191, 291, 430, 434;

5: 363, 367, ...>



1,2,4,5这几篇文章
中哪几篇可能包含 “*to be
or not to be*” ?

短语查询的处理

- 短语查询: “to be or not to be”
- 对每个词项, 抽出其对应的倒排记录表: to, be, or, not.
- 合并<docID:位置>表, 考虑 “to be or not to be” .
 - to:
 - ❖ 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...
 - be:
 - ❖ 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...
- 邻近搜索中的搜索策略与此类似, 不同的是此时考虑前后位置之间的距离不大于某个值

邻近式查询(Proximity query)

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
 - /k 表示 “在 k 个词之内”
- 很明显，位置索引可以处理邻近式查询，而双词索引却不能

位置索引的大小

- 位置索引增加了位置信息，因此空间较大，但是可以采用索引压缩技术进行处理(参见第五讲)
- 当然，相对于没有位置信息的索引，位置索引的存储空间明显大于无位置信息的索引
- 另外，位置索引目前是实际检索系统的标配，这是因为实际中需要处理短语(显式和隐式)和邻近式查询

位置索引的大小

- 词项在每篇文档中的每次出现都需要一个存储单元
- 因此索引的大小依赖于文档的平均长度
 - 平均Web页面的长度 < 1000 个词项
 - 美国证监会文件(SEC filings), 书籍, 甚至一些史诗 ... 和容易就超过 100,000 个词项

一些经验规律

- 位置索引的大小大概是无位置信息索引的2-4倍
- 位置索引大概是原始文本容量的35-50%
- 提醒：上述经验规律适用于英语及类英语的语言

混合索引

➤ 上述两种索引方式可以混合使用

- 对某些特定的短语 (如 “Michael Jackson”, “Britney Spears”), 如果采用位置索引的方式那么效率不高
 - ❖ 还有 “The Who” (英国一著名摇滚乐队), 采用位置索引, 效率更低

➤ Williams et al. (2004)对一种混合的索引机制进行了评估

- 采用混合机制, 那么对于典型的Web查询(比例)来说, 相对于只使用位置索引而言, 仅需要其 $\frac{1}{4}$ 的时间
- 相对于只使用位置索引, 空间开销只增加了26%

本讲小结

- 索引构建过程(特别是预处理)
- 如何对索引文档进行处理来得到词典
 - 理解文档(document)的概念
 - 词条化(Tokenization), 理解词条(token)的概念
 - 词项生成, 理解词项(term)的概念
- 倒排记录表
 - 更快的合并算法: 跳表法(skip list)
 - 短语查询的处理及带位置信息的倒排索引

课后作业3

- 在索引构建过程中，中文的处理和英文的处理有什么不同？
 - 至少写5点，并给出例子。
 - 5分钟