信息检索与数据挖掘

第3章 词项词典和倒排记录表

课程内容

- 第1章　绪论
- 第2章　布尔检索及倒排索引
- 第3章　词典查找及扩展的倒排索引
- 第4章　索引构建和索引压缩
- 第5章　向量模型及检索系统
- 第6章　检索的评价
- 第7章　相关反馈和查询扩展
- 第8章　概率模型
- 第9章　基于语言建模的检索模型
- 第10章　文本分类
- 第11章　文本聚类
- 第12章　Web搜索
- 第13章　多媒体信息检索
- 第14章　其他应用简介

第3章 词典查找及扩展的倒排索引

1. 如何建立**词项词典**（ term vocabulary ）？
   ① 文档集
   ② 文本**词条化**（ Tokenization ）
   ③ 语言学预处理
   ④ 建立索引

2. 如何实现倒排记录表？
   ① 快速合并算法：带跳表的倒排记录表(skip lists)
   ② 包含位置信息的倒排记录表以及短语查询

建立词项(Term)词典过程

待建索引文档集合

词条化模块
Tokenizer

词条化

Friends Romans Countrymen

语言学预处理模块
Linguistic modules

friend roman countryman

语言学预处理

索引器
Indexer

倒排记录表

friend → 2 → 4

roman → 1 → 2

countryman → 13 → 16

第3章 词典查找及扩展的倒排索引

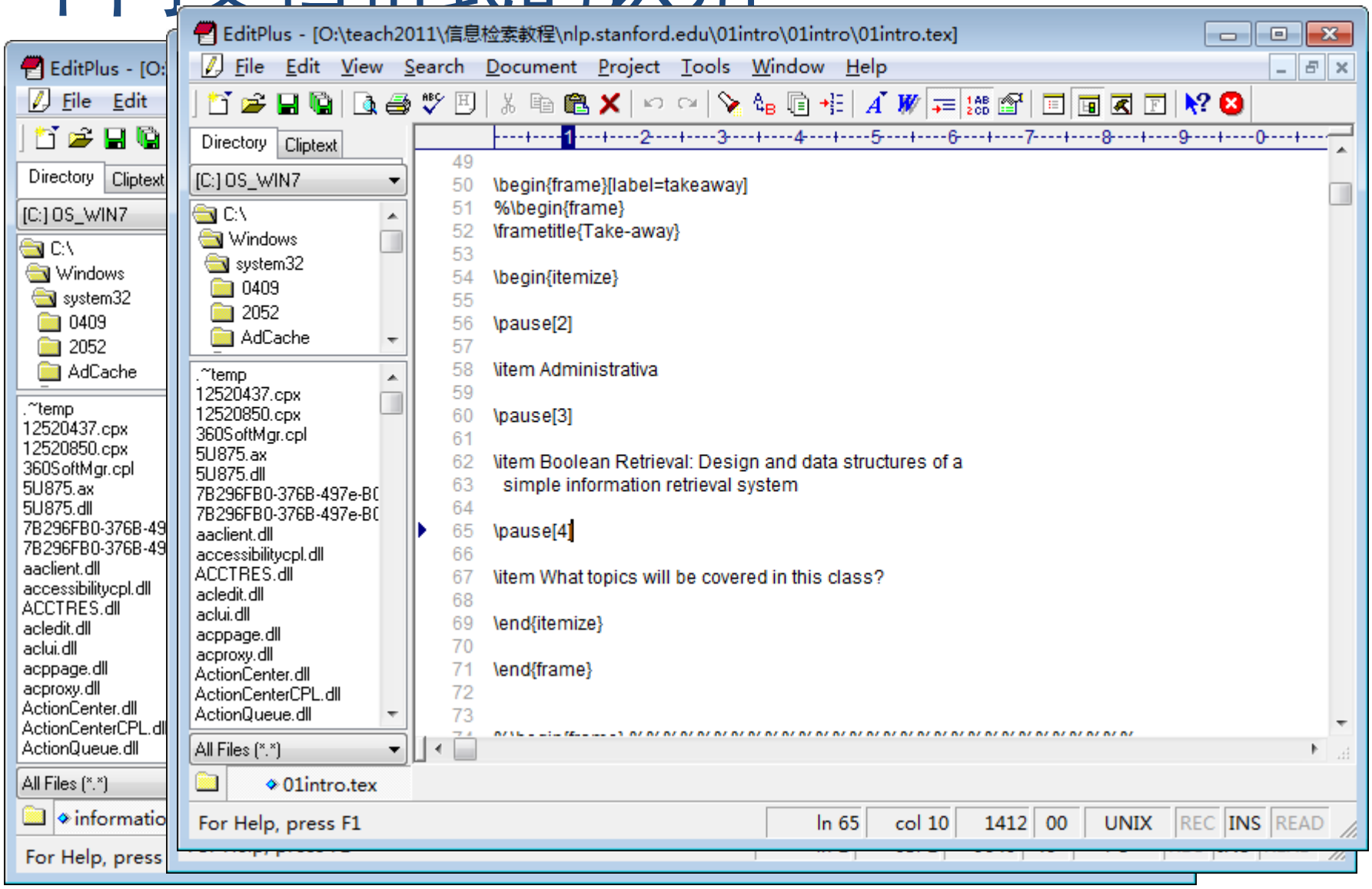
- 第一部分：如何建立词项词典？
  - 文档解析（Parsing a document）
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询

文档解析

- 文档包含哪些格式？
  - pdf/word/excel/html？
- 文档中包含的语言？
- 文档使用何种编码方式？

上述问题都可以看成是机器学习中的分类问题，但在实际中往往采用启发式方法来实现。（后面章节讨论）

不同文档格式的识别

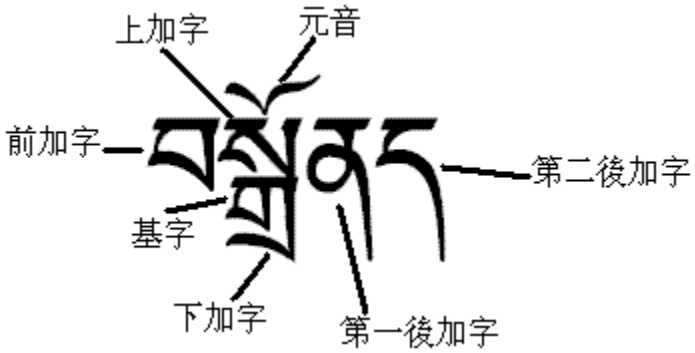

EditPlus - [O:\teach2011\信息检索教程\nlp.stanford.edu\01intro\01intro\01intro.tex]
File Edit View Search Document Project Tools Window Help
Directory Cliptext
[C:] OS_WIN7
C:\
Windows
system32
0409
2052
AdCache
.~temp
12520437.cpx
12520850.cpx
360SoftMgr.cpl
5U875.ax
5U875.dll
7B296FB0-376B-497e-B0
7B296FB0-376B-497e-B0
aaclient.dll
accessibilitycpl.dll
ACCTRES.dll
acledit.dll
aclui.dll
acppage.dll
acproxy.dll
ActionCenter.dll
ActionCenterCPL.dll
ActionQueue.dll
All Files (*.*)
01intro.tex
49
50 \begin{frame}[label=takeaway]
51 %\begin{frame}
52 \frametitle{Take-away}
53
54 \begin{itemize}
55
56 \pause[2]
57
58 \item Administrativa
59
60 \pause[3]
61
62 \item Boolean Retrieval: Design and data structures of a
63 simple information retrieval system
64
65 \pause[4]
66
67 \item What topics will be covered in this class?
68
69 \end{itemize}
70
71 \end{frame}
72
73
For Help, press F1
ln 65 col 10 1412 00 UNIX REC INS READ

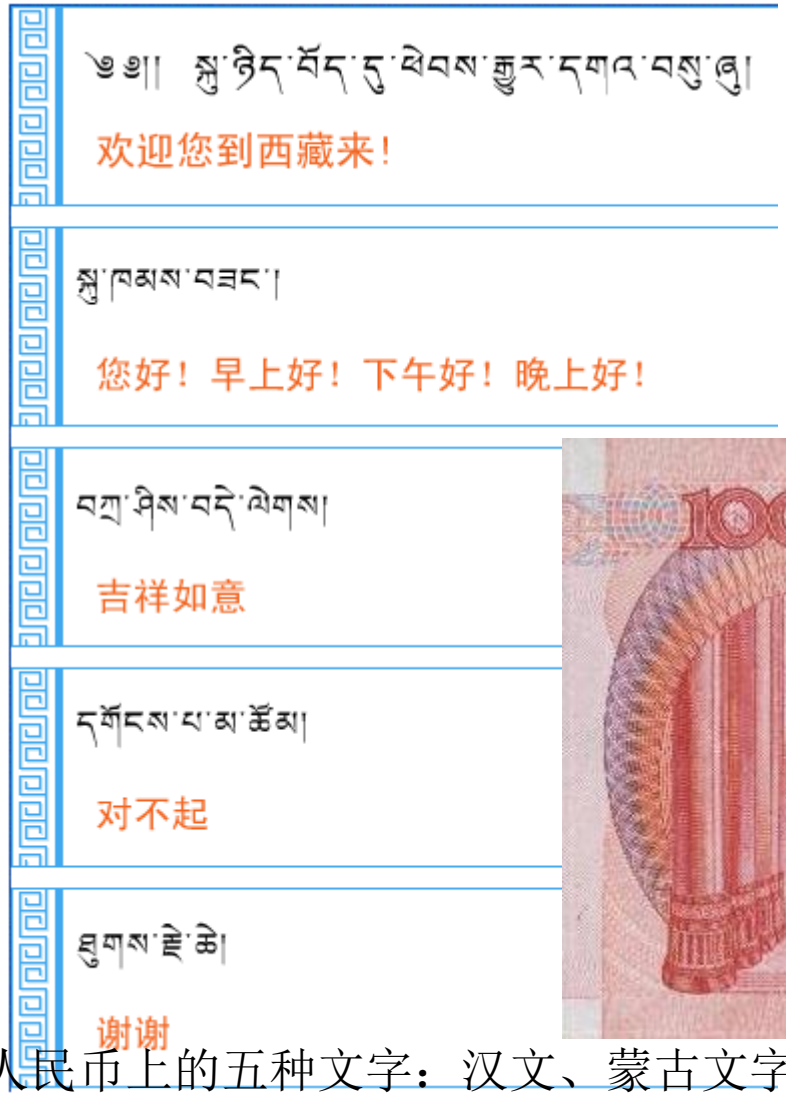
语言的自动检测

- 你好
- こんにちは
- **Hallo**
- 안녕하세요.
- **Алло**
- .أهلا و سهلا

- 我爱你
- 私はあなたを愛して
- **Ich liebe dich**
- 나는 당신을 사랑합니다
- **Я люблю тебя**
- .أحبك

文档中的语言





人民币上的五种文字：汉文、蒙古文字、藏族文字、维吾尔文字、壮文

文档中的编码方式

- 7bit ASCII?
- UNICODE?
  - UTF-8、UTF-16、UTF-32
- Email对二进制附件的编码
  - Content-Type: text/html;
  - charset="gb2312"
  - Content-Transfer-Encoding: base64

复杂因素：格式/语言

- **待索引文档集中包含不同语言的文档**
  - 单独的一个索引应该包含不同语言的文档
- 一个文档或者其附件中包含多种语言或格式
  - 例子：一封法语的邮件中包含德语的pdf
- 文档单位的选择？
  - 一个文件？
  - 一封email？
  - 一封带有5个附件的email？
  - 一组文件？

第3章 词典查找及扩展的倒排索引

- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 合并算法回顾
  - 基于跳表指针的快速合并算法
  - 短语查询

什么是词条化(Tokenization)

- 词条化：将给定的字符序列拆分成一系列子序列的过程，其中每一个子序列称之为一个“词条”。
- **输入：** “*Friends, Romans and Countrymen*”
- **输出：**
  - *Friends*
  - *Romans*
  - *Countrymen*

词条（Tokens）
词项（Terms）

- 每个词条都作为候选的索引。
- 但是什么是有效的索引？

"Friends, Romans, countrymen, lend me your ears" is the first line of a speech by Mark Antony in the play Julius Caesar, by William Shakespeare. Occurring in Act III, scene II, it is one of the most famous lines in all of Shakespeare's works.

词条化可能遇到的问题

- **FAST**
  - **Five hundred meters Aperture Spherical Radio Telescope**

- **SUN**
  - **Stanford University Network**
  - **Sun Microsystems**
    - **2009年被Oracle收购**

词条化可能遇到的问题(英文)

e.g.: Finland’s capital →
Finland? Finlands? Finland’s?

- 连字符问题?
  - Hewlett-Packard → Hewlett和Packard 是二个词条吗?
  - State-of-the-art
  - Co-education
- 空格问题?
  - San Francisco是一个词条还是二个词条?
- 连字符和空格相互影响
  - Lowercase，lower-case，lower case
- 英文句号的考虑
  - IEEE 802.3, 802.11ax, X.509

词条化可能遇到的问题(数字)

- 3/20/91 Mar. 12, 1991 20/3/91
- Tel:63601000 (800) 234-2333
- 查询2009至2011年间车祸死亡的人数
- B-52 AK-47
- PGP 密钥：324a3df234cb23e
- 双11

词条化可能遇到的问题(中文)

- Out of Vocabulary
  - 人名、地名、机构名
  - 一些新词、流行词
    - 重要的事情说三遍、世界那么大，我想去看看、城会玩、为国护盘、明明可以靠脸吃饭，却偏偏要靠才华、我想静静、吓死宝宝了、内心几乎是崩溃的、我妈是我妈、主要看气质……
- Ambiguity
  - 同一句子有多种可能的分词结果
    - 南京市_长江大桥 南京市长_江大桥
    - 我们小组_合成氢气　　我们小_组合成氢气
    - 发展中_国家　　发展_中国_家

补充：数学之美 系列二 -- 谈谈中文分词

- 最容易想到的，也是最简单的分词办法就是查字典。这种方法最早是由北京航天航空大学的梁南元教授提出的。用“查字典” 法，其实就是我们把一个句子从左向右扫描一遍，遇到字典里有的词就标识出来，遇到复合词（比如 “上海大学”）就找最长的词匹配，遇到不认识的字串就分割成单字词，于是简单的分词就完成了。这种简单的分词方法完全能处理上面例子中的句子。
- 八十年代，哈工大的王晓龙博士把它理论化，发展成最少词数的分词理论，即一句话应该分成数量最少的词串。这种方法一个明显的不足是当遇到有二义性 （有双重理解意思）的分割时就无能为力了。
- 90年前后，清华大学的郭进博士用统计语言模型成功解决分词二义性问题，将汉语分词的错误率降低了一个数量级。

基于字符串→基于统计

词条化的策略

- 针对不同的语言，采取不同策略的词条化
- 分词的基本方法：
  - 基于词典的最大匹配法
  - 机器学习

正向最大匹配(基于词典的方法)

0	1	2	3	4	5	6
他	说	的	确	实	在	理

指针位置	剩余词串	首字	最大匹配词条
0	他说的确实在理	他	他
1	说的确实在理	说	说
2	的确实在理	的	的确
4	实在理	实	实在
6	理	理	理

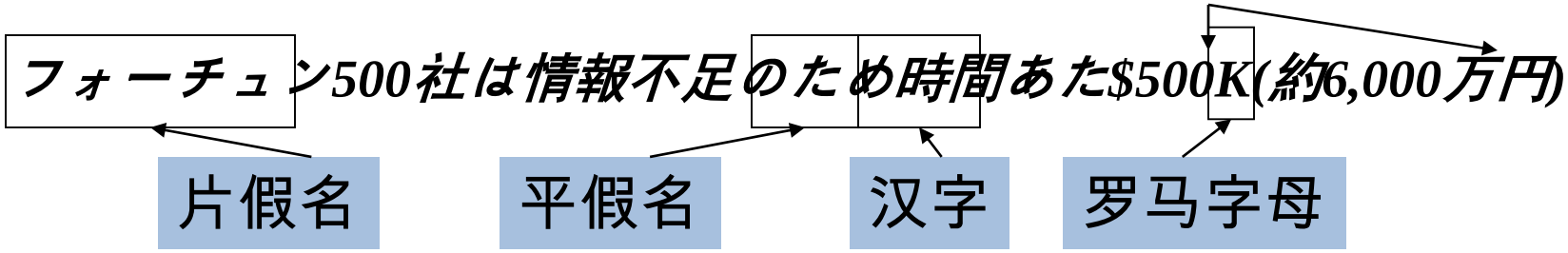
逆向最大匹配(基于词典的方法)

0	1	2	3	4	5	6
他	说	的	确	实	在	理

指针位置	剩余词串	尾字	最大匹配词条
6	他说的确实在理	理	在理
4	他说的确实	实	确实
2	他说的	的	的
1	他说	说	说
0	他	他	他

词条化可能遇到的问题(语言问题)

- 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
- 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！

词条化可能遇到的问题(语言问题)

- 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 عاما من الاحتلال الفرنسي.

- ← → ←

→ ← 开始

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’

词条化可能遇到的问题(语言问题)

- 屈折语（俄语、英语、法语等）
  - 有比较丰富的词形变化；
  - 一种词形变化的语素可以表示几种不同的语法意义；
  - 词尾和词干或词根结合十分紧密

		主格	宾格	所有格
	我	I	me	my (我的)
单数	你	you	you	your (你的)
单数	他	he	him	his (他的)
单数	她	she	her	her (她的)
单数	它	it	it	its (它的)
复数	我们	we	us	our (我们的)
复数	你们	you	you	your (你们的)
复数	他们	they	them	their (他们的)
复数	她们	they	them	their (她们的)
复数	它们	they	them	their (它们的)

词条化可能遇到的问题(语言问题)

- 孤立语（以汉藏语系为代表）：
  - 词序严格；
  - 虚词重要；
    - 18个古代虚词：而、何、乎、乃、其、且、若、所、为、焉、也、以、因、于、与、则、者、之
  - 复合词多、派生词少
- 黏着语（日语、朝鲜语、蒙古语、维吾尔语等）
  - 只是词的尾部发生变化；
  - 一种变化只表示一种语法意义；
  - 词根与变词语素结合不很紧密，两者有很大的独立性

词条化不可协调的矛盾(二义性)

- 用红墨水写一个“蓝”字，
- 请问，这个字是红字还是蓝字？

第3章 词典查找及扩展的倒排索引

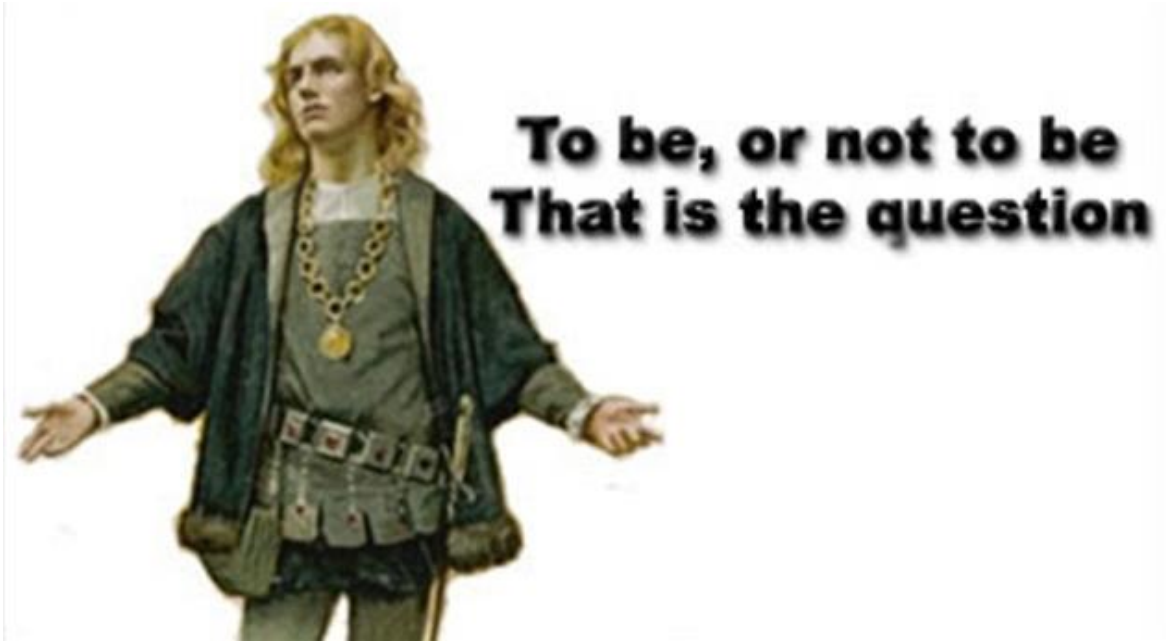
- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询

停用词

- 停用词表
  - 将词项按照文档集频率（collection frequency），从高到底排列
  - 选取与文档意义不大，高频出现的词，比如，a, an, the, to, and, be…
  - 停用词使用的趋势
  - 现代搜索引擎发展的趋势使用少量的停用词表
  - 现代IR系统更加关注利用语言的统计特性来处理常见词问题
    - 第五章介绍采用压缩技术降低停用词的存储开销
    - 第六章介绍词项权重，将高频常用词来文档的排序影响降到最小
    - 第七章介绍一个索引按影响度大小排列的IR系统，权重很小时，停止扫描停用词表

消除停用词问题和可能的方法

- 优点：停用词消除可以减少term的个数
- 缺点：有时消除的停用词对检索是有意义的。
  - “的士”、“to be or not to be”
- 消除方法：
  - 查表法
  - 基于文档频率



第3章 词典查找及扩展的倒排索引

- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询

词项归一化

- 我们需要将文档和查询中的词条“归一化”成一致的形式
  - 希望USA和U.S.A.之间也能形成匹配
  - 归一化的结果：
  - 在IR系统的词项词典中，形成多个近似词项的一个等价类
  - 隐式的建立等价类
  - 例如将USA和U.S.A.映射为USA
  - 例如将anti-discrimination和antidiscrimination映射为antidiscrimination

词项归一化：不同语言之间的区别

- **重音符号：**
  - **e. g. ： 法语中*résumé* *vs.* *resume.***
  - 变音符号：
  - **e. g. ： 德语中*Tuebingen* *vs.* *Tübingen.***
    （其实他们应该是等价的）
  - 最重要的标准：
    - **最重要的问题不是规范或者语言学的问题，而是用户将会如何根据这些词来构造查询？**
  - 即使在一些语言中，有的词有了标准的读音，但是用户有自己的读音/拼写方式
    - **e. g. ：** *Tuebingen, Tübingen, Tubingen* →*Tubingen*

词项归一化：不同语言之间的区别

- 其他
  - 中文中日期的表示7月30日 vs. 英文中7/30
  - 日语中使用的假名汉字 vs. 中文中的汉字
  - 词条化和归一化
  - 二者都依赖于不同的语言种类，因此，在整个索引建立过程中要综合考虑
  - e. g. :

Morgen will ich in MIT …

Is this German “mit”?

德语Morgen will ich in MIT 的意思是“我明天在MIT”，而德语中的“MIT”其实是“与”的意思

词项归一化：大小写转换

- 一般策略
  - 将所有字母转换为小写
  - 绝大多数情况下，用户在构造查询时都忽略首字母的大写
  - 一些专有名词除外
    - e.g.： *General Motors*
    - *Fed vs. fed*
    - *SAIL vs. sail*
- Google的例子
  - 输入查询词C.A.T.
  - 首页是关于猫的网站，而不是卡特彼勒公司（Caterpillar Inc.）（2005年的时候）

词项归一化

- 词项归一化的策略：建立同义词扩展表。

- 例子：

查询：	检索：
window	*window, windows*
windows	*Windows, windows, window*
Windows	*Windows*

扩展词表和soundex算法

- 如何处理同义词和同音词？
  - e.g.：手工建立同义词词表
    - *car = automobile* *color = colour*

①为每个查询维护一张包含多个词的查询扩展词表
  - 例如：查询*automobile*的同时，也查询*car*

②在建立索引建构时就对词进行扩展
  - 例如：对于包含*automobile*的文档，我们同时也使用*car*来索引

- 如何处理拼写错误？
  - 其中的一种处理方法，就是根据发音相同来进行词项扩展
  - 后续章节中有讨论

第3章 词典查找及扩展的倒排索引

- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并（Lemmatization）
  - 词干还原（Stemming）
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询

词形归并（Lemmatization）

- 减少屈折变化的形式，将其转变为基本形式。
- e. g.
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
  - *the boy's cars are different colors* → *the boy car be different color*
  - 词形归并可以减少词项词典中的词项数量

词干还原（Stemming）

- 通常指的就粗略的去除单词两端词缀的启发式过程。
  - e.g., *automate(s), automatic, automation* → *automat.*

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

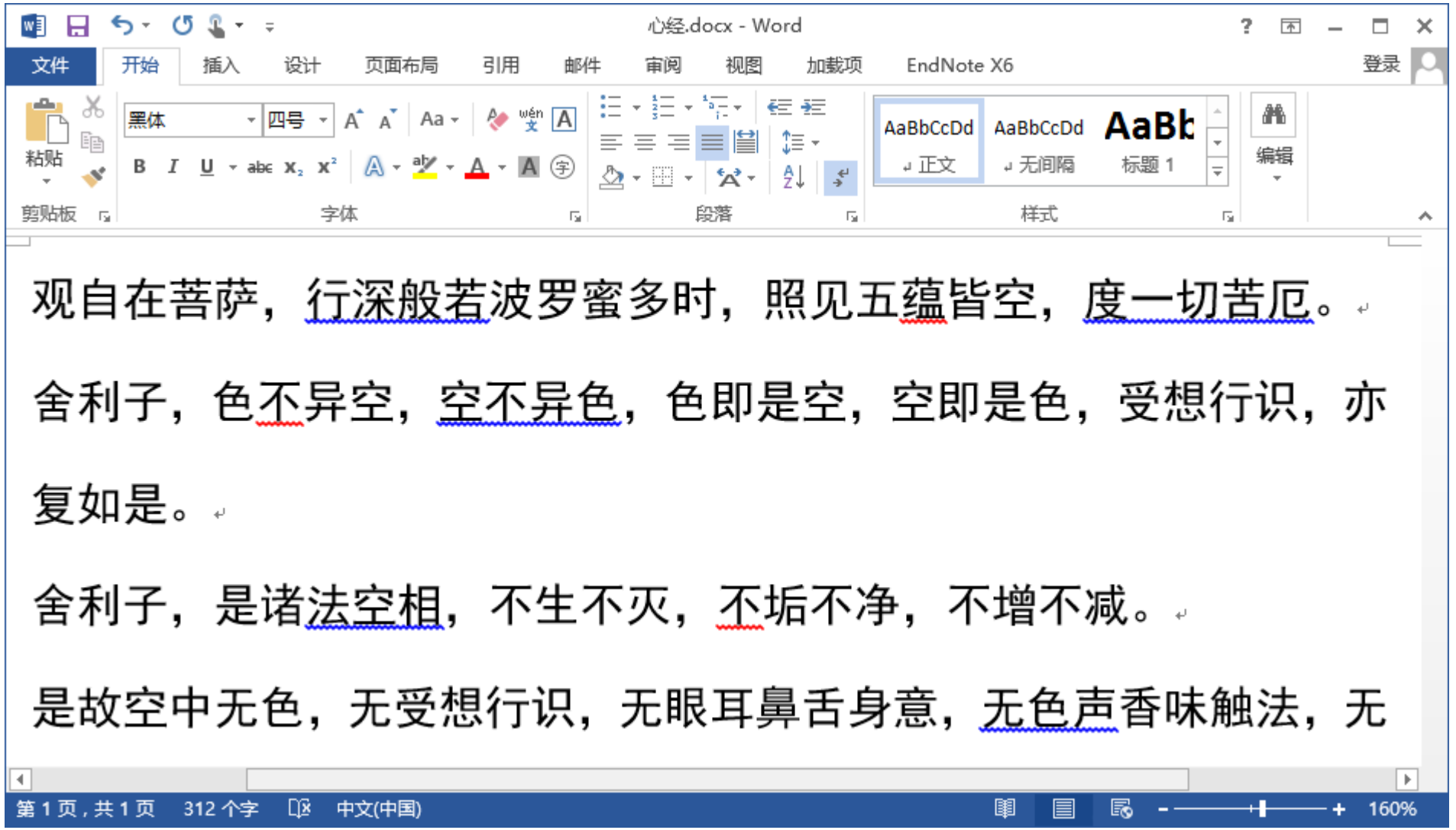
for exampl compress and compress ar both accept as equival to compress

中文重叠词还原——可视为“词干还原”

形容词(AB)	ABAB 式	AABB 式	A 里 AB 式
高兴	高兴高兴	高高兴兴	
明白	明白明白	明明白白	
热闹	热闹热闹	热热闹闹	
潇洒	潇洒潇洒	潇潇洒洒	
糊涂		糊糊涂涂	糊里糊涂
流气			流里流气
粘乎	粘乎粘乎	粘粘乎乎	
凉快	凉快凉快	凉凉快快	

形容词（A）	ABB 式	ABCD 式
黑	黑压压	黑不溜秋
白	白花花	白不吡咧
红	红彤彤	
亮	亮晶晶	
恶	恶狠狠	
香	香喷喷	
滑	滑溜溜	

文字拼写的自动校正


心经.docx - Word
文件
开始
插入
设计
页面布局
引用
邮件
审阅
视图
加载项
EndNote X6
登录
粘贴
剪贴板
黑体
四号
字体
AaBbCcDd
正文
AaBbCcDd
无间隔
AaBb
标题 1
样式
编辑
观自在菩萨，行深般若波罗蜜多时，照见五蕴皆空，度一切苦厄。
舍利子，色不异空，空不异色，色即是空，空即是色，受想行识，亦复如是。
舍利子，是诸法空相，不生不灭，不垢不净，不增不减。
是故空中无色，无受想行识，无眼耳鼻舌身意，无色声香味触法，无
第 1 页，共 1 页
312 个字
中文(中国)
160%

同义词/近义词提取

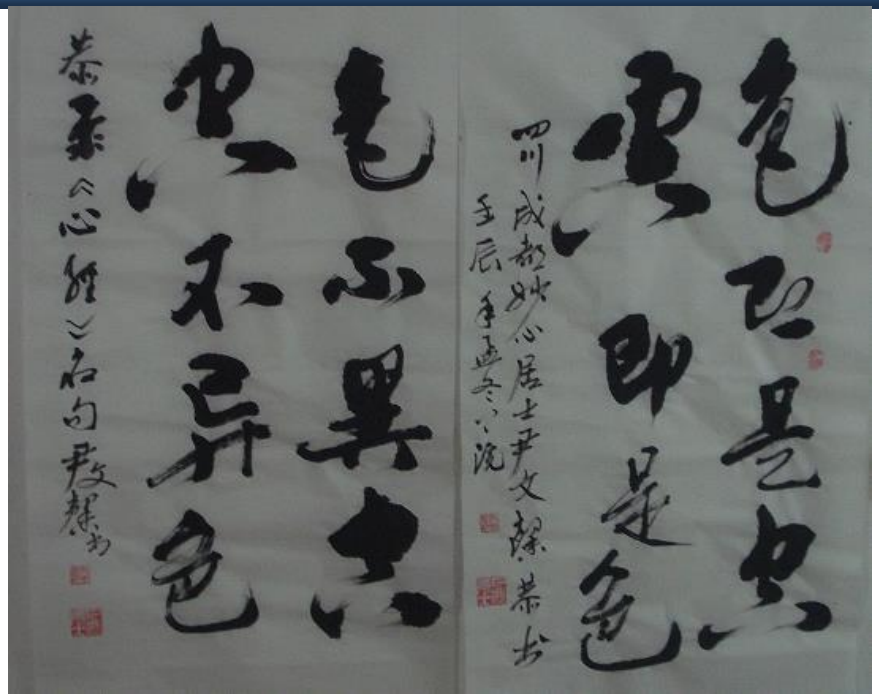
- 英文示例

歌曲名称：be want you wanna be
歌手：Darin Zanyar

- 中文示例
  - 《林教头风雪山神庙》（高中语文）中，写李小二看见两个人走进他的酒店，交头接耳地商量着什么，悄悄喊来老婆，说道：“大姐，这两个人来得不尴尬。”老婆道：“怎的不尴尬？”随后李小二找来林冲，告诉他：“却才东京来的尴尬人，在我这里清管营、差拨吃了半日酒。”同样的两个人，前作“不尴尬”，后作“尴尬人”，意思是相同还是相反？前之“不尴尬”，课文注释说：“不尴尬，鬼鬼祟祟，不正派。也作‘尴尬’，或者‘不尴不尬’。”后之“尴尬人”未注，从语境看，此处也该作“鬼鬼祟祟，不正派”讲，与前文“不尴尬”的意思相同。

文本自动翻译

- 色即是空，空即是色



- Word2013 ：All things of visible form and substance are empty, spatial is a color
- 百度翻译、金山词霸2012：Form is emptiness, emptiness is form
- Google翻译：Sex is zero, Kongjishise

一些词干还原工具:Lovins 1968

- **词干还原工具： Lovins**
  - **单遍扫描，最长后缀删除原则**
- 算法涉及如下部件：
  - ending，词后缀，共有294个
  - condition，词后缀去除条件，每个ending对应一个condition，共有294个
  - transformation，转换ending的方式，共有35个，详细列表见最后
- 算法分为两部：
  - 1.对英文词，根据ending列表，按照ending从长到短扫描，找到第一个符合condition的ending
  - 2.根据剩下的stem应用transformation，将ending转为恰当的形式

http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm

一些词干还原工具：Porter算法 1980

- **英文处理中最常用的词干还原算法。**
  - **经过实践证明是高效性的算法。**
  - 算法包括5个按照顺序执行的词项约简步骤：
  - 每个步骤都是按照一定顺序执行的
  - 每个步骤中包含了选择不同的规则的约定
  - 比如，从规则组中选择作用时词缀最长的那条规则

C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587).

The Porter Stemming Algorithm
http://tartarus.org/~martin/PorterStemmer/
The English (Porter2) stemming algorithm 2000
http://snowball.tartarus.org/algorithms/english/stemmer.html

一些词干还原工具：Porter算法中的典型规则

- *sses → ss* *caresses→ caress*
- *ies → i* *ponies →poini*
- *ational → ate* *national →nate*
- 在这些规则中经常要考虑词的“测度”这一概念
- *(m>1) EMENT →*
  - *replacement → replac*
  - *cement → cement*

determi，determinate，determination，determinations，determine，determined，determines，determining → determin

support, supported, supportable, supportance, supporter, supporters, supporting, supportor, supports → support

关于词干还原

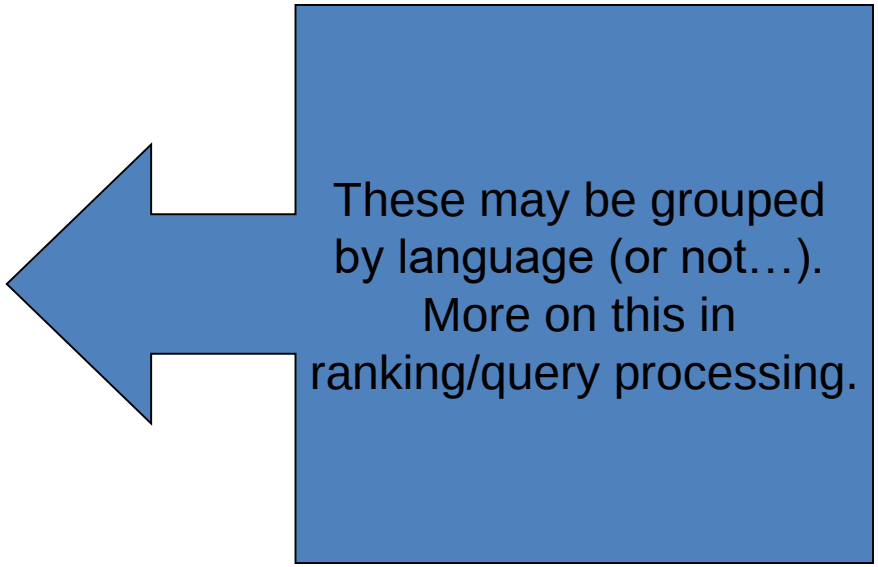
- 词干还原能够提高召回率，但是会降低准确率
  - e.g.：
  - operative ⇒ oper
- 词干还原对于芬兰语，西班牙语，德语，法语都有明显的作用，其中对芬兰语的提高达到30%（以MAP平均准确率来计算）。

语言的特殊性

- 词干还原和词形归并，都体现了不同语言之间的差异性，这些差异性包括：
  - 不同语言之间的差异
  - 特殊专业语言与一般语言的差异
  - 词干还原或者是词形归并往往通过在索引过程中增加插件程序的方式来实现
  - 商业软件
  - 开源软件

Dictionary entries – first cut

ensemble.french
時間.japanese
MIT.english
mit.german
guaranteed.english
entries.english
sometimes.english
tokenization.english



本节内容小结：如何建立词项词典？

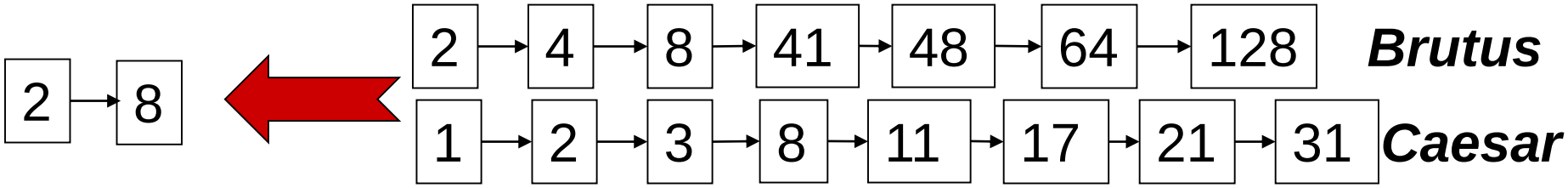
- 文档解析：格式？语言？编码方式？
- 词条化
  - 概念词条（Tokens）/词项（Terms）
  - 英文：连字符？空格？句号？数字？
  - 中文：Out of Vocabulary？Ambiguity？
  - 方法：针对不同的语言，采取不同策略
- 停用词：停用词表？查表法 or 基于文档频率
- 词项归一化：
  - 等价类？语言之间的区别？大小写转换？
  - 策略：建立同义词扩展表
- 词形归并：am, are, is→be
- 词干还原：去除单词两端词缀
  - Porter算法：规则
  - 提高召回率，但是会降低准确率

第3章 词典查找及扩展的倒排索引

- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询

合并算法（Postings Merges）回顾

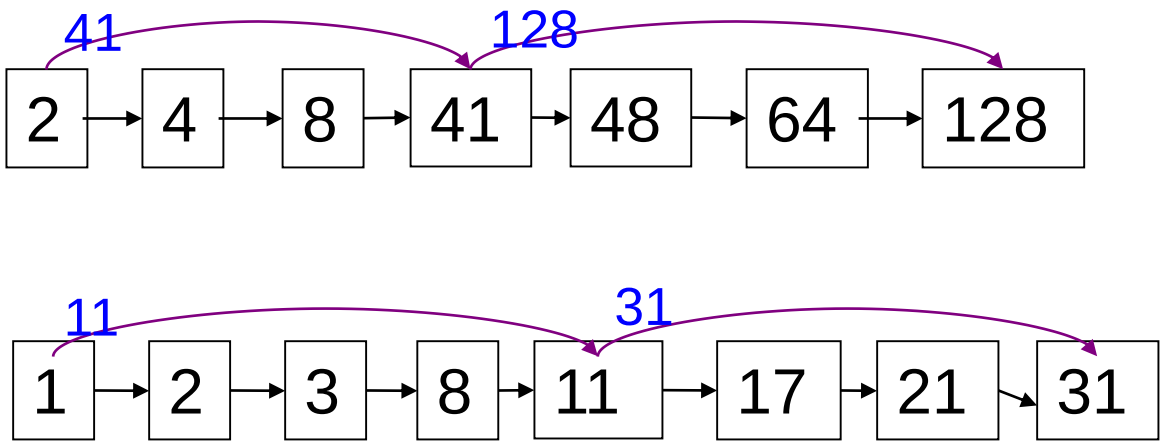
- 通过在二个倒排表之间同时移动指针来实现合并，此时的操作与线性表的总数成线性关系。



如果倒排表的长度分别是m和n，那么合并算法需要操作o（m+n）次。

我们能否做的更好？←上节课的问题

基于跳表（Skip List）的倒排记录表快速合并算法

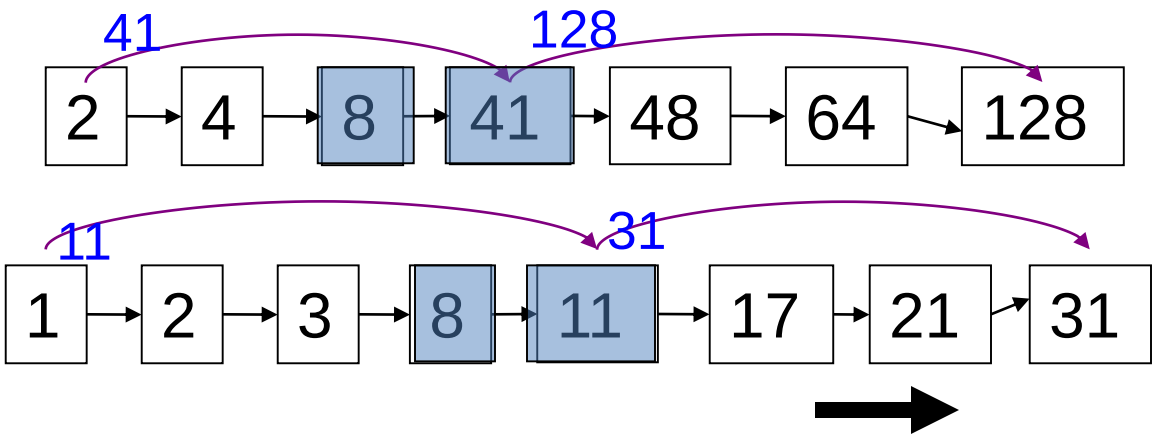


跳表指针能够跳过那些不可能出现在检索结果中的记录项。

构建跳表的二个主要问题：

- 如何利用跳表指针进行快速合并？
- 在什么位置设置跳表指针？

带有跳表指针的查询处理过程



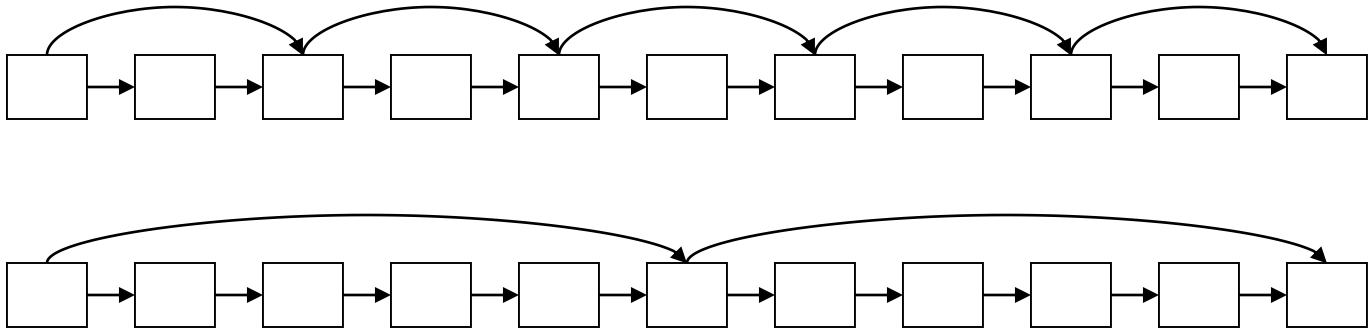
1. 假定我们在进行遍历一直发现到了共同的记录8，将结果8放入结果表中之后，我们继续移动二个表的指针。
2. 假定第一个表指针移到41，第二个表的指针移到11。
3. 由于11比41小，因此，上面的表不需要继续移动，只需移动下面的表，跳到31。
4. 这样就跳过了17,21。

基于跳表指针的倒排记录合并算法

```
INTERSECTWITHSKIPS(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 if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
 9                     then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10                          do p1 ← skip(p1)
11                     else p1 ← next(p1)
12                else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
13                     then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14                          do p2 ← skip(p2)
15                     else p2 ← next(p2)
16  return answer
```

在什么位置设置跳表指针？

- 策略：
  - 设置较多的指针→较短的步长⇒更多的跳跃机会
- 更多的指针比较次数和更多的存储空间
  - 设置较少的指针→较少的指针比较次数，但是需要设置较长的步长⇒较少的连续跳跃



设置跳表指针

- 放置跳表指针的一个简单的启发式策略是：

  如果倒排表的长度是L，那么在每个$\sqrt{L}$处均匀放置跳表指针

- 该策略没有考虑到查询词项的分布
- 如果索引相对固定的话，建立有效的跳表指针比较容易，如果索引需要经常的更新，建立跳表指针就相对困难点。
- 硬件参数对索引构建有一定的影响
  - CPU速度
  - 磁盘访问速度

第3章 词典查找及扩展的倒排索引

- 第一部分：如何建立词项词典？
  - 文档解析
  - 词条化
  - 停用词
  - 词项归一化
  - 词形归并
  - 词干还原
- 第二部分：如何实现倒排记录表？
  - 快速合并算法：带跳表的倒排记录表
  - 包含位置信息的倒排记录表以及短语查询
    - 方法1：二元词索引
    - 方法2：位置信息索引

短语查询（Phrase Query）

- 用户希望将类似“stanford university” “中国科学技术大学”的查询中的二个词看成是一个整体。
- 类似“I want to university at stanford”这样的文档是不会被匹配的。
  - 大部分的搜索引擎都支持双引号的短语查询，这种语法很容易理解并被用户成功使用。
  - 有很多查询在输入时 没有加双引号，其实都是隐式的短语查询（如人名）。
  - 要支持短语查询，只记录<*term : docs*>这样的条目是不能满足用户需要的。

第一种方法：二元词索引（Biword indexes）

- 将文档中每个连续词对看成一个短语
- 例如，文本“Friends, Romans, Countrymen”将生成如下的二元连续词对：
  - ***friends romans***
  - ***romans countrymen***
  - 其中的每一个二元词对都将作为词典中的词项
  - 经过上述的处理，此时可以处理二个词构成的短语查询

更长的短语查询

- 更长的短语查询可以分成多个短查询来处理
- 例如，文本“***stanford university palo alto***”将分解成如下的二元词对布尔查询：

stanford university AND university palo AND palo alto

- 对于该布尔查询返回的文档，我们不能确定其中是否真正包含最原始的四词短语。

很难避免伪正例的出现！

扩展的二元词索引（Extended Biword）

- 名词和名词短语构成的查询具有相当特殊的地位。
  - 首先对文本进行词条化，然后进行词性标注
  - 把每个词项分为名词（N）、虚词（X，冠词和介词和其他词。
  - 将形式为N*XN非词项序列看成一个扩展的二元词
  - 每个这样的扩展的二元次对应一个词项
- 例如：*catcher in the rye*(书名： 麦田守望者)

  N X X N

- 利用这样的扩展二元词索引处理查询，
  - 将查询拆分成N和X
  - 将查询划分成扩展的二元词
  - 最后在索引中进行查找

二元词索引的问题：

- 会出现伪正例子
- 增加词汇表的大小
  - 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

- 二元词索引并非是标准的解决方案，混合索引机制（二元词索引和位置索引进行合并）可以更加完美的解决短语查询的问题。

第二种方法：位置信息索引（Positional indexes）

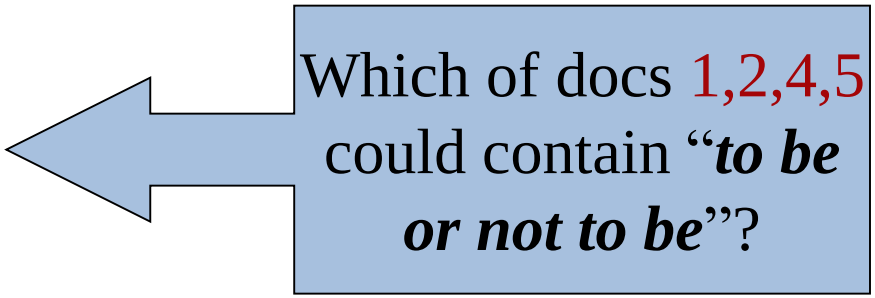
- 在这种索引中，对每个词项，采取以下方式存储倒排表记录：

〈词项，词项频率；
文档1：位置1，位置2……；
文档2：位置1，位置2……；
……〉

<***be***: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>

位置信息索引例子

<***be***: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>



•对于短语查询，仍然采用合并算法，查找符合的文档
•不只是简单的判断二个词是否出现在同一文档中，还需要检查他们出现的位置情况

通过位置信息可以判断，第一个be和最后一个be之间，应该相差4个位置，由此我们可以判断，文档4, 5都有可能是符合要求的文档

短语查询过程

- 例子：

查询词：“to be or not to be”

倒排表：

- *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; . . .

1. 考虑to和be的倒排表的合并，查找同时包含to和be的文档
2. 检查表中，看看是否某个be的前面的一个位置上正好出现to

65

短语查询示例

习题 **2-9** [*] 下面给出的是一个位置索引的一部分，格式为：

词项: 文档 1: （位置 1, 位置 2, …）; 文档 2: （位置 1, 位置 2, …）;

angels: 2: （36,174,252,651）; 4: （12,22,102,432）; 7: （17）;

fools: 2: （1,17,74,222）; 4: （8,78,108,458）; 7: （3,13,23,193）;

fear: 2: （87,704,722,901）; 4: （13,43,113,433）; 7: （18,328,528）;

in: 2: （3,37,76,444,851）; 4: （10,20,110,470,500）; 7: （5,15,25,195）;

rush: 2: （2,66,194,321,702）; 4: （9,69,149,429,569）; 7: （4,14,404）;

to: 2: （47,86,234,999）; 4: （14,24,774,944）; 7: （199,319,599,709）;

tread: 2: （57,94,333）; 4: （15,35,155）; 7: （20,320）;

where: 2: （67,124,393,1001）; 4: （11,41,101,421,431）; 7: （16,36,736）;

那么哪些文档和以下的查询匹配？其中引号内的每个表达式都是一个短语查询。

a. “fools rush in”;

b. “fools rush in” AND “angels fear to tread”。

邻近查询（Proximity queries）

- Employ me/3 place,表示从左边或右边相距在k个词之类
- 显然，位置索引能够用于邻近搜索，而二元词搜索则不能

- 临近查询在上一节课中有示例

回顾示例：WestLaw

Westlaw BETTER RESULTS FASTER.

http://www.westlaw.com/

- 最大的收费的法律搜索服务提供商
- 几十T的数据；700,000多用户
- 大多数用户仍然使用布尔查询
- 查询的例子：
  - **What is the statute of limitations in cases involving the federal tort claims act?**
  - **LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM**
    - **/3 = within 3 words, /S = in same sentence**
- 许多专业的用户通常喜欢使用布尔查询
  - 因为用户精确的知道自己会得到什么
- 但是这并不意味着能得到更好的结果

邻近搜索中两个倒排记录的合并

算法寻找两个词项在k个词之内出现的情形，返回一个三元组<文档ID，词项在p_1中的位置，词项在p_2中的位置>的列表

```
POSITIONALINTERSECT(p1, p2, k)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then l ← ⟨ ⟩
 5             pp1 ← positions(p1)
 6             pp2 ← positions(p2)
 7             while pp1 ≠ NIL
 8             do while pp2 ≠ NIL
 9                do if |pos(pp1) − pos(pp2)| ≤ k
10                      then ADD(l, pos(pp2))
11                      else if pos(pp2) > pos(pp1)
12                              then break
13                   pp2 ← next(pp2)
14                while l ≠ ⟨ ⟩ and |l[0] − pos(pp1)| > k
15                do DELETE(l[0])
16                for each ps ∈ l
17                do ADD(answer, ⟨docID(p1), pos(pp1), ps⟩)
18                pp1 ← next(pp1)
19             p1 ← next(p1)
20             p2 ← next(p2)
21        else if docID(p1) < docID(p2)
22                then p1 ← next(p1)
23                else p2 ← next(p2)
24  return answer
```

位置信息索引的讨论

- 采用位置索引会大大增加倒排记录表的存储空间，即使采用后面讨论的压缩方法也无济于事。
- 由于用户期望能够进行短语查询和邻近查询，所以还是得采取这种索引方式。

- 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

位置信息索引的大小

- 位置索引需要对词项的每次出现保留一个条目，因此索引的大小取决于文档的平均长度。
- 网页的平均长度不超过1000个词项。
- 但是某些文件，（SEC 股票文件）很容易就达到1000，000个词项。
- 假设，平均1000个词项中每个词项的频率都是1

Document size	Postings	Positional postings
1000	1	1
100,000	1	100

索引空间相差两个数量级。

71

经验法则(English-like)

- 位置索引大概是非位置索引大小的2—4倍
- 位置索引的大小大约是原始文档的30%--50%
- 提醒：上述经验规律适用于英语及类英语的语言

72

混合索引机制

- 二元词索引和位置索引二种策略可以进行有效的合并。假如用户通常只查询特定的短语，如Michael Jackson，那么基于位置索引的倒排记录表合并方式效率很低。
- 一个混合策略是：对某些查询使用短语索引或只使用二元词索引，而对其他短语查询则采用位置索引。短语索引所收录的那些较好的查询可以根据用户最近的访问行为日志统计得到，也就是说，它们往往是那些高频常见的查询。当然，这并不是唯一的准则。
- Williams等人（2004）评估了更复杂的混合索引机制，（引入后续词索引方法）。
  - 对于一个典型的web短语混合查询，其完成时间大概是只使用位置索引的1/4
  - 比只使用位置索引增加26%的空间

73

本节内容小结

- 带跳表的倒排记录表
  - 跳表（Skip List）
  - 跳表指针：位置？个数？
  - 如果索引需要经常的更新?
- 包含位置信息的倒排记录表
  - 短语查询→二元词索引
    - 二元词索引→扩展的二元词索引：词性标注
    - 增加词汇表的大小
  - 短语查询→位置信息索引
    - 位置信息索引→邻近查询
    - 大大增加倒排记录表
  - 短语查询→混合索引机制

本章内容小结

- 如何建立词项词典？
  - 文档解析：格式？语言？编码方式？
  - 词条化：词条（Tokens）/词项（Terms）
  - 停用词：停用词表？查表法 or 基于文档频率
  - 词项归一化：等价类←→同义词扩展表
  - 词形归并：am, are, is →be
  - 词干还原：去除单词两端词缀、Porter算法
- 如何实现倒排记录表？
  - 跳表:跳表指针(位置、个数、更新问题)
  - 短语查询
    - 二元词索引→扩展的二元词索引：词性标注
    - 位置信息索引→邻近查询
    - 增加倒排记录表
    - →混合索引机制