Inverted File Organization and Indexing Model

Reference:

James Allan, University of Massachusetts Amherst
Gerald Benoit, Simmons College
Pandu Nayak and Prabhakar Raghavan，Stanford University

Edt. By: Qingcai Chen, HITSZ

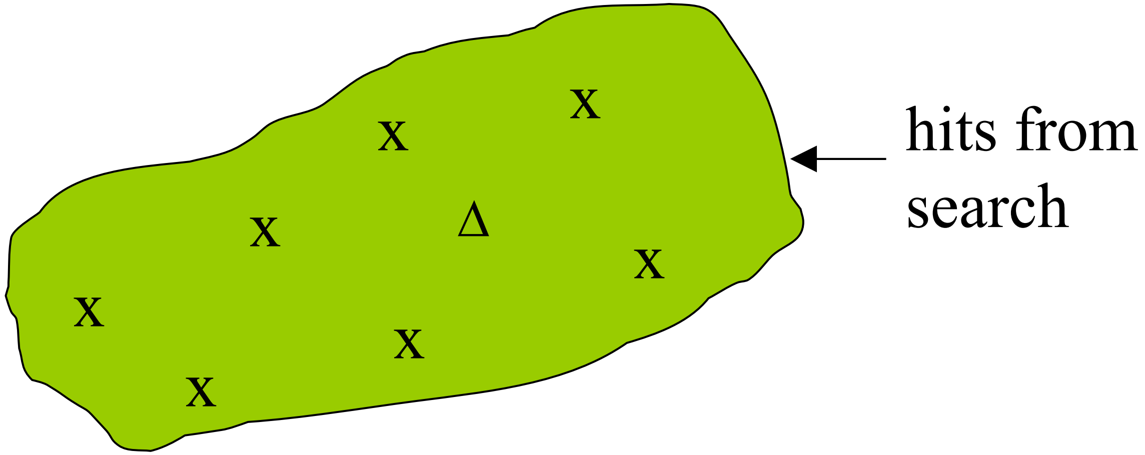
Some vocabulary

- File organizations or *indexes* are used to increase performance of system
    - Will talk about how to store indexes later
- Text ***indexing*** is the process of deciding what will be used to represent a given document
- These ***index terms*** are then used to build indexes for the documents
- The ***retrieval model*** described how the indexed terms are incorporated into a model
    - Relationship between retrieval model and indexing model

Content

- Brief Review of Vector Space Model
- Inverted Files
- Index Models (Feature Selection)

Results of a Search



x documents found by search

Δ query

Use of Inverted Files for Calculating Similarities

In the term vector space, if **q** is query and $\mathbf{d}_j$ a document, then **q** and $\mathbf{d}_j$ have no terms in common iff $\mathbf{q.d}_j = 0$, so we don' t need to waste resource on these documents.

1. To calculate all the **non-zero similarities**, find all the documents, $\mathbf{d}_j$, that contain **at least one term** in the query:

   Merge the inverted lists for each term t_i in the query, with a logical *OR*, to establish a set of hits, *R*.

   For each $\mathbf{d}_j \in R$, calculate *Similarity*(q, $\mathbf{d}_j$), using appropriate weights.

2. Return the elements of *R* in ranked order.

Content

- *Brief Review of Vector Space Model*
- Inverted Files
- Index Models (Feature Selection)

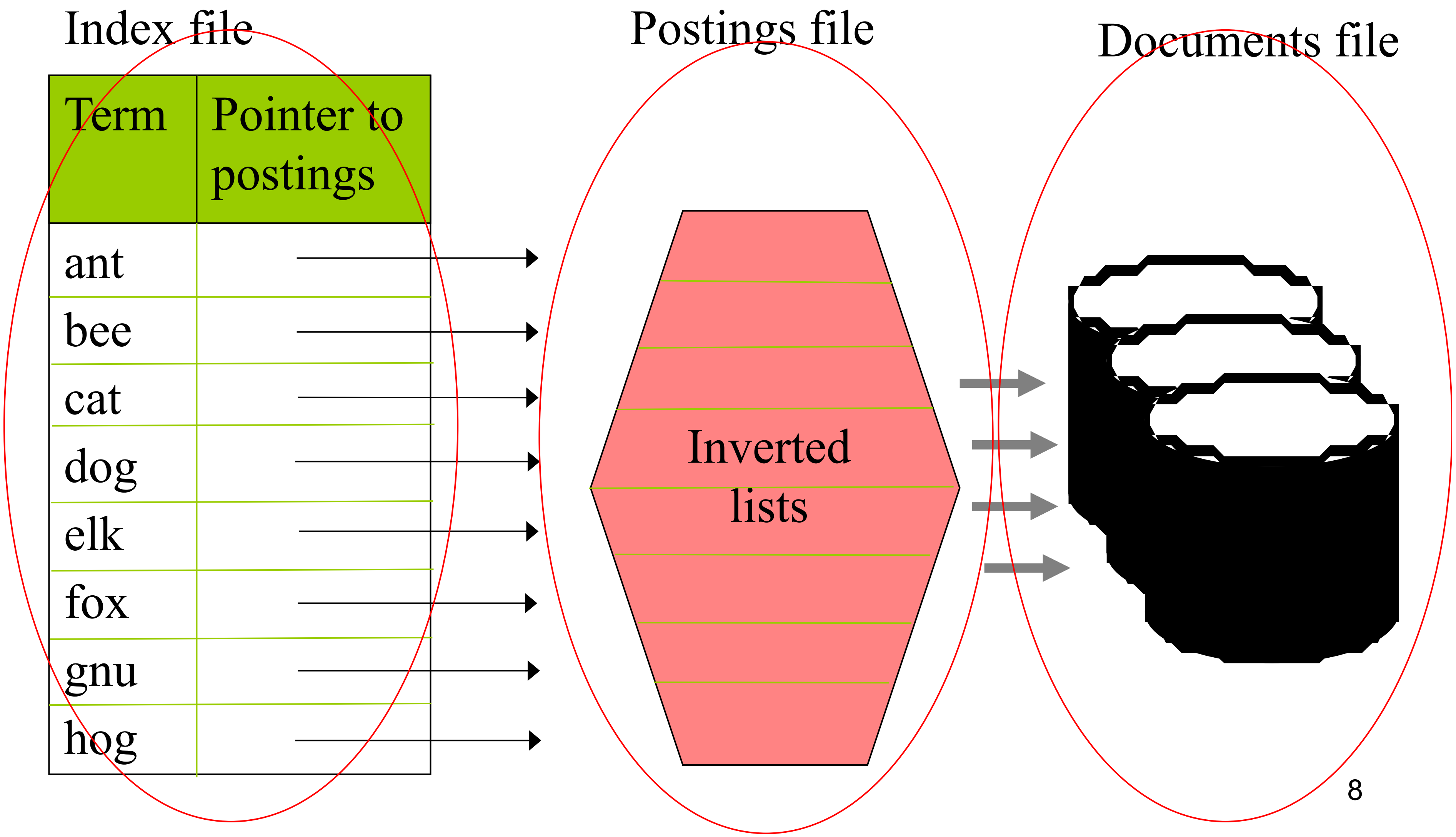
Representation of Inverted Files

Document file: Stores the documents. Important for user interface design.

Index (word list, vocabulary) file: Stores list of terms (keywords). Designed for searching and sequential processing, e.g., for range queries, (**lexicographic index**). Often held in memory.

Postings file: Stores an **inverted list** (postings list) of postings for each term. Designed for rapid merging of lists and calculation of similarities. Each list is usually stored sequentially.

Organization of Inverted Files



Index file

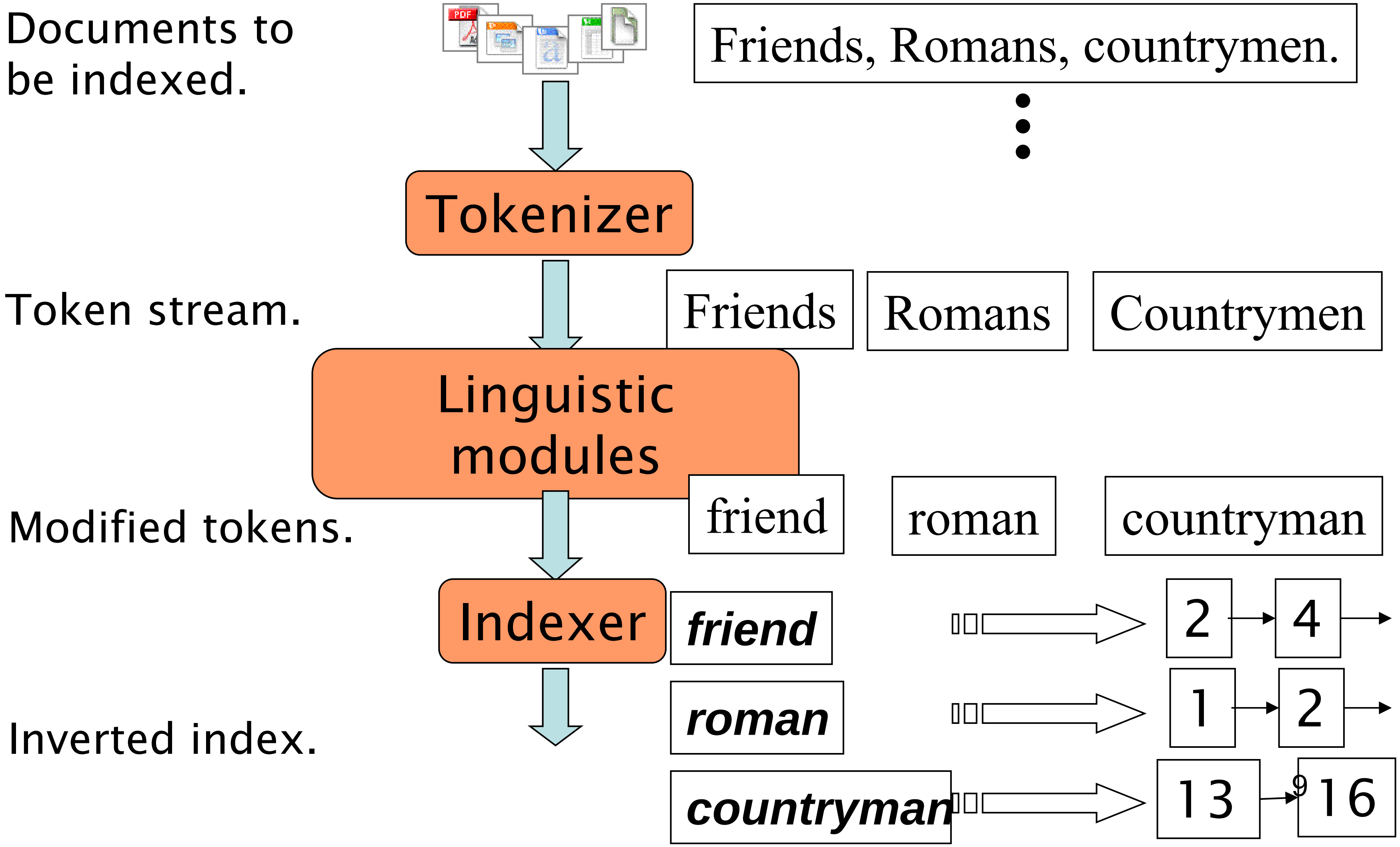
Term	Pointer to postings
ant	
bee	
cat	
dog	
elk	
fox	
gnu	
hog	

Postings file

Inverted lists

Documents file

The basic indexing pipeline



Documents to be indexed.

Friends, Romans, countrymen.

Tokenizer

Token stream.

Friends Romans Countrymen

Linguistic modules

Modified tokens.

friend roman countryman

Indexer

Inverted index.

friend → 2 → 4

roman → 1 → 2

countryman → 13 → 16

Parsing a document

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?

Each of these is a classification problem.

But these tasks are often done heuristically …

Decisions in Building Inverted Files: What is a Term?

Underlying character set, e.g., printable ASCII, Unicode, UTF8.

Is there a controlled vocabulary? If so, what words are included?

List of stopwords. “的、因为、所以…..”

Rules to decide the beginning and end of words, e.g., spaces or punctuation. “长长长长长，行行行行行”

Character sequences not to be indexed, e.g., sequences of numbers. “12345, 2603, 3475?”

What’s the differences between Chinese and English?

TOKENS AND TERMS

Tokenization(断词,标记化)

- Input: “***Friends, Romans, Countrymen***”
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- A token is a sequence of characters in a document
- Each such token is now a candidate for an index entry, after further processing
  - Described below
- But what are valid tokens to emit?

Tokenization

- Issues in tokenization:
  - ***Finland’ s capital*** →
    Finland? Finlands? Finland’ s?
  - ***Hewlett-Packard*** → ***Hewlett*** and ***Packard*** as two tokens?
    - ***state-of-the-art***: break up hyphenated sequence.
    - ***co-education***
    - ***lowercase***, ***lower-case***, ***lower case*** ?
    - It can be effective to get the user to put in possible hyphens
  - ***San Francisco***: one token or two?
    - How do you decide it is one token?

Numbers

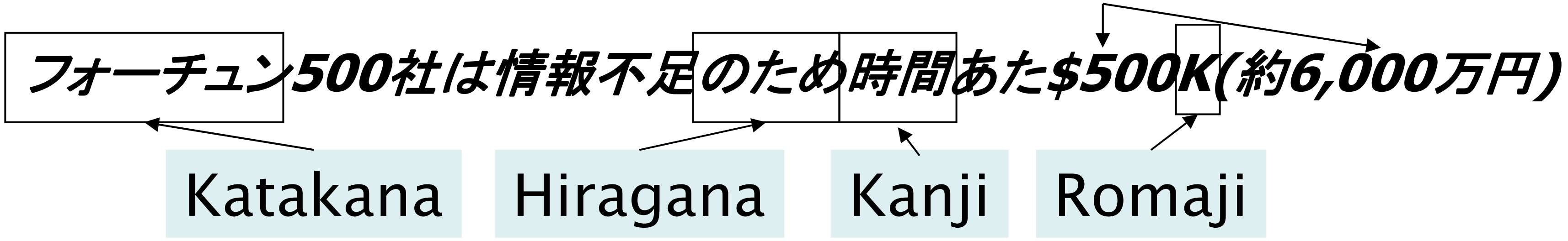
- ***3/12/91   Mar. 12, 1991   12/3/91***
- ***55 B.C.***
- ***B-52***
- ***My PGP key is 324a3df234cb23e***
- ***(800) 234-2333***
  - Often have embedded spaces
  - Older IR systems may not index numbers
    - But often very useful: think about things like looking up error codes/stacktraces on the web
  - Will often index “meta-data” separately
    - Creation date, format, etc.

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
      - Until at least 2003, it didn't on Google
        - » Internationalization!
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module
      - Can give a 15% performance boost for German

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana

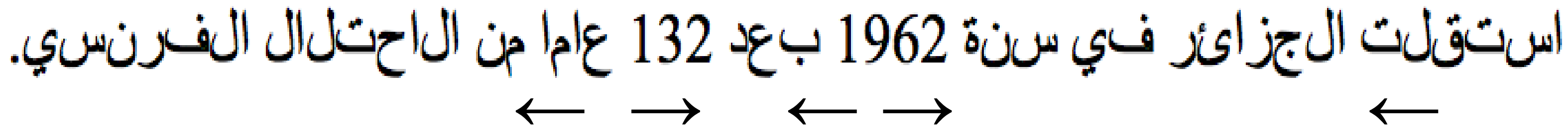
Hiragana

Kanji

Romaji

End-user can express query entirely in hiragana(平假名）

Tokenization: language issues

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures
- 

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

  start
- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’
- With Unicode, the surface presentation is complex, but the stored form is straightforward

Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words
- But the trend is away from doing this:
  - Good compression techniques means the space for including stopwords in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - You need them for:
    - Phrase queries: “King of Denmark”
    - Various song titles, etc.: “Let it be”, “To be or not to be”
    - “Relational” queries: “flights to London”

Normalization to terms

- We need to "normalize" words in indexed text as well as query words into the same form
  - We want to match U.S.A. and USA
- Result is terms: a term is a (normalized) word type, which is an entry in our IR system dictionary
- We most commonly implicitly define equivalence classes of terms by, e.g.,
  - deleting periods to form a term
    - U.S.A., USA ⎝ USA
  - deleting hyphens to form a term
    - anti-discriminatory, antidiscriminatory ⎝ antidiscriminatory

Normalization: other languages

- Accents(重音): e.g., French résumé vs. resume.
- Umlauts(元音变音): e.g., German: Tuebingen vs. Tübingen
  - Should be equivalent
- Most important criterion:
  - How are your users like to write their queries for these words?

- Even in languages that standardly have accents, users often may not type them
  - Often best to normalize to a de-accented term
    - Tuebingen, Tübingen, Tubingen ⎝ Tubingen

Normalization: other languages

- Normalization of things like date forms
  - 7月30日 vs. 7/30
  - Japanese use of kana(假名) vs. Chinese characters

- Tokenization and normalization may depend on the language and so is intertwined with language detection

Morgen will ich in MIT …

Is this German "mit"?

- Crucial: Need to "normalize" indexed text as well as query terms into the same form

Case folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence?
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail
  - Often best to lower case everything, since users will use lowercase regardless of ‘correct’ capitalization…

- Google example:
  - Query C.A.T.
  - #1 result was for “cat” (well, Lolcats) not Caterpillar Inc.

Normalization to terms

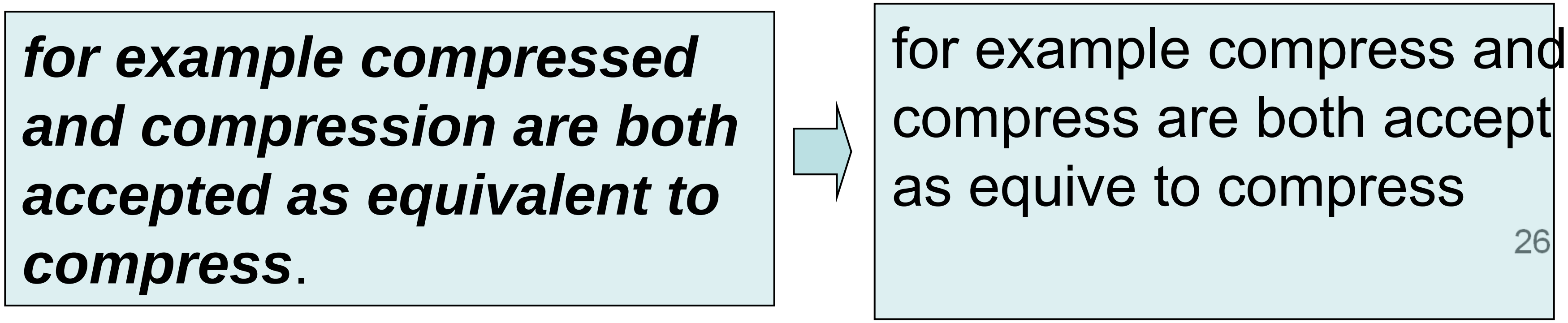
- An alternative to equivalence classing is to do asymmetric expansion
- An example of where this may be useful
  - Enter: window   Search: window, windows
  - Enter: windows   Search: Windows, windows, window
  - Enter: Windows   Search: Windows
- Potentially more powerful, but less efficient

Thesauri(辞典) and soundex*

- Do we handle synonyms(同义词) and homonyms(同形同音异义词)?
  - E.g., by hand-constructed equivalence classes
    - car = automobile color = colour
  - We can rewrite to form equivalence-class terms
    - When the document contains automobile, index it under car-automobile (and vice-versa)
  - Or we can expand a query
    - When the query contains automobile, look under car as well
- What about spelling mistakes?
  - One approach is soundex(探测法，返回同音字串), which forms equivalence classes of words based on phonetic heuristics

Stemming (词干分析)

- Reduce terms to their “roots” before indexing
- “Stemming” suggest crude affix chopping
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.



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

for example compress and compress are both accept as equive to compress

Lemmatization (词形还原)

- Reduce inflectional/variant forms to base form
- 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
- Lemmatization implies doing “proper” reduction to dictionary headword form

Document File

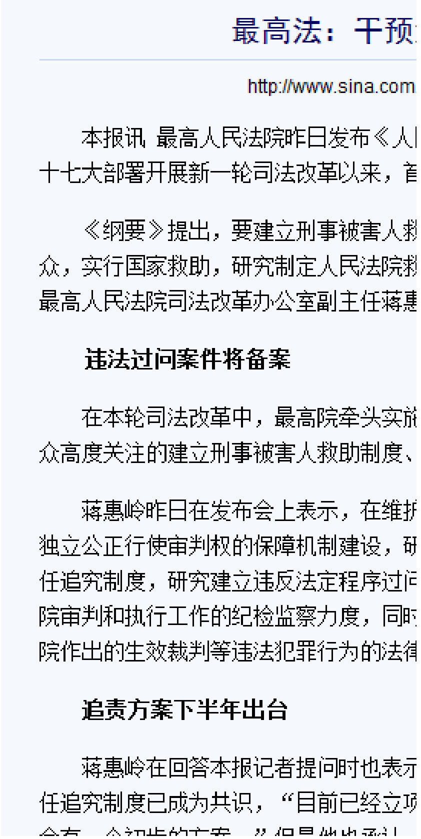
The documents file stores the documents that are being indexed. The documents may be:

- **primary documents**, e.g., electronic journal articles
- **surrogates (文档替代品)**, e.g., catalog records(目录) or abstracts(摘要)

Docs file for web search system (Example of docs)

A possible representation for purified web pages

- a document



最高法：干预

http://www.sina.com

本报讯 最高人民法院昨日发布《人

十七大部署开展新一轮司法改革以来，首

《纲要》提出，要建立刑事被害人救

众，实行国家救助，研究制定人民法院救

最高人民法院司法改革办公室副主任蒋惠

违法过问案件将备案

在本轮司法改革中，最高院牵头实施

众高度关注的建立刑事被害人救助制度、

蒋惠岭昨日在发布会上表示，在维护

独立公正行使审判权的保障机制建设，研

任追究制度，研究建立违反法定程序过问

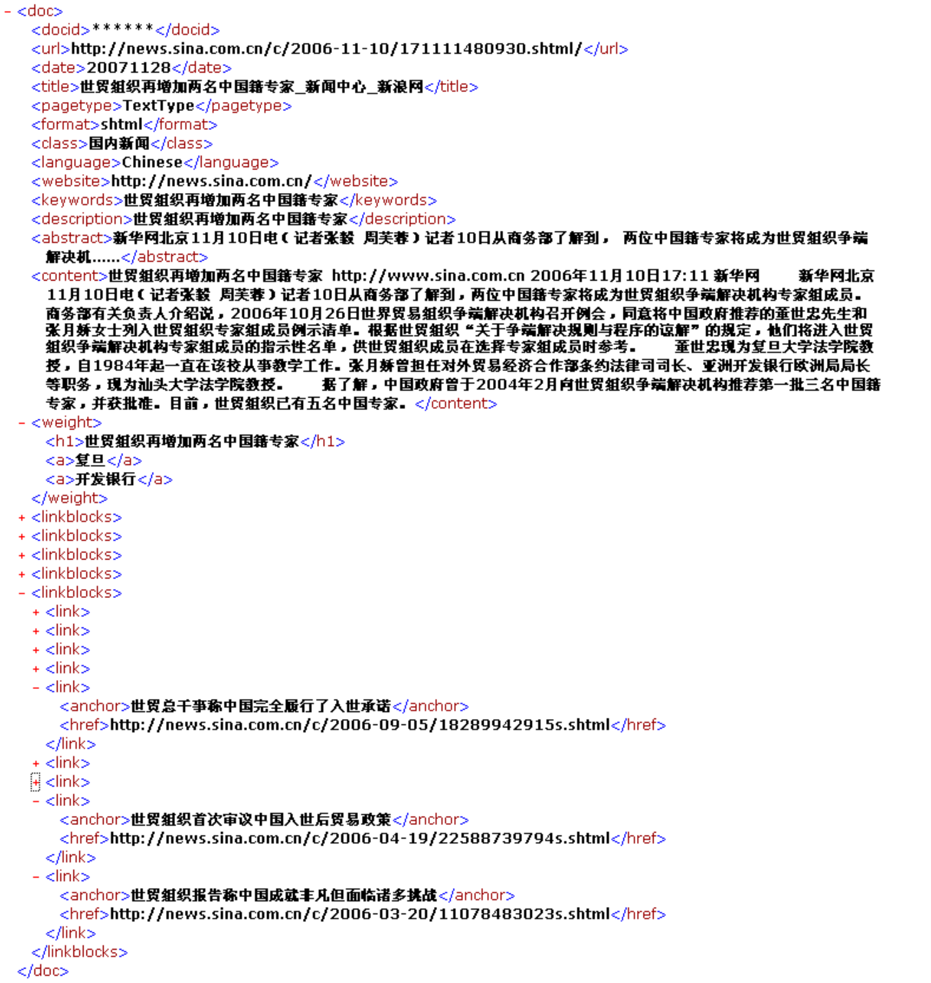
院审判和执行工作的纪检监察力度，同时

院作出的生效裁判等违法犯罪行为的法律

追责方案下半年出台

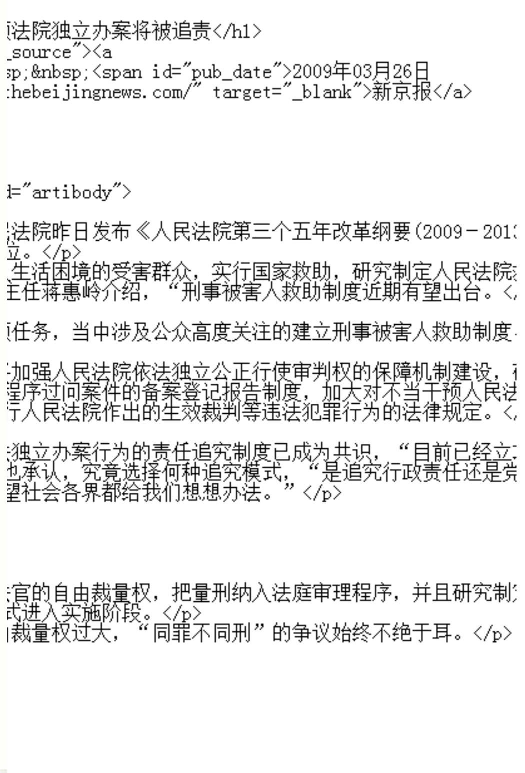
蒋惠岭在回答本报记者提问时也表示

任追究制度已成为共识，“目前已经立项



```
- <doc>
    <docid>******</docid>
    <url>http://news.sina.com.cn/c/2006-11-10/171111480930.shtml/</url>
    <date>20071128</date>
    <title>世贸组织再增加两名中国籍专家_新闻中心_新浪网</title>
    <pagetype>TextType</pagetype>
    <format>shtml</format>
    <class>国内新闻</class>
    <language>Chinese</language>
    <website>http://news.sina.com.cn/</website>
    <keywords>世贸组织再增加两名中国籍专家</keywords>
    <description>世贸组织再增加两名中国籍专家</description>
    <abstract>新华网北京11月10日电(记者张毅 周芙蓉)记者10日从商务部了解到， 两位中国籍专家将成为世贸组织争端
      解决机......</abstract>
    <content>世贸组织再增加两名中国籍专家 http://www.sina.com.cn 2006年11月10日17:11 新华网    新华网北京
      11月10日电(记者张毅 周芙蓉)记者10日从商务部了解到，两位中国籍专家将成为世贸组织争端解决机构专家组成员。
      商务部有关负责人介绍说，2006年10月26日世界贸易组织争端解决机构召开例会，同意将中国政府推荐的董世忠先生和
      张月姣女士列入世贸组织专家组成员例示清单。根据世贸组织“关于争端解决规则与程序的谅解”的规定，他们将进入世贸
      组织争端解决机构专家组成员的指示性名单，供世贸组织成员在选择专家组成员时参考。    董世忠现为复旦大学法学院教
      授，自1984年起一直在该校从事教学工作。张月姣曾担任对外贸易经济合作部条约法律司司长、亚洲开发银行欧洲局局长
      等职务，现为汕头大学法学院教授。    据了解，中国政府曾于2004年2月向世贸组织争端解决机构推荐第一批三名中国籍
      专家，并获批准。目前，世贸组织已有五名中国专家。</content>
  - <weight>
      <h1>世贸组织再增加两名中国籍专家</h1>
      <a>复旦</a>
      <a>开发银行</a>
    </weight>
  + <linkblocks>
  + <linkblocks>
  + <linkblocks>
  + <linkblocks>
  - <linkblocks>
    + <link>
    + <link>
    + <link>
    + <link>
    - <link>
        <anchor>世贸总干事称中国完全履行了入世承诺</anchor>
        <href>http://news.sina.com.cn/c/2006-09-05/18289942915s.shtml</href>
      </link>
    + <link>
    + <link>
    - <link>
        <anchor>世贸组织首次审议中国入世后贸易政策</anchor>
        <href>http://news.sina.com.cn/c/2006-04-19/22588739794s.shtml</href>
      </link>
    - <link>
        <anchor>世贸组织报告称中国成就非凡但面临诸多挑战</anchor>
        <href>http://news.sina.com.cn/c/2006-03-20/11078483023s.shtml</href>
      </link>
    </linkblocks>
</doc>
```

ie web page



```
法院独立办案将被追责</h1>
_source"><a
sp;&nbsp;<span id="pub_date">2009年03月26日
:hebeijingnews.com/" target="_blank">新京报</a>

d="artibody">

法院昨日发布《人民法院第三个五年改革纲要(2009－201
立。</p>
生活困境的受害群众，实行国家救助，研究制定人民法院
主任蒋惠岭介绍，“刑事被害人救助制度近期有望出台。</

任务，当中涉及公众高度关注的建立刑事被害人救助制度

加强人民法院依法独立公正行使审判权的保障机制建设，
程序过问案件的备案登记报告制度，加大对不当干预人民法
行人民法院作出的生效裁判等违法犯罪行为的法律规定。</

独立办案行为的责任追究制度已成为共识，“目前已经立
也承认，究竟选择何种追究模式，“是追究行政责任还是党
望社会各界都给我们想想办法。”</p>

官的自由裁量权，把量刑纳入法庭审理程序，并且研究制
式进入实施阶段。</p>
裁量权过大，“同罪不同刑”的争议始终不绝于耳。</p>
```

Postings File

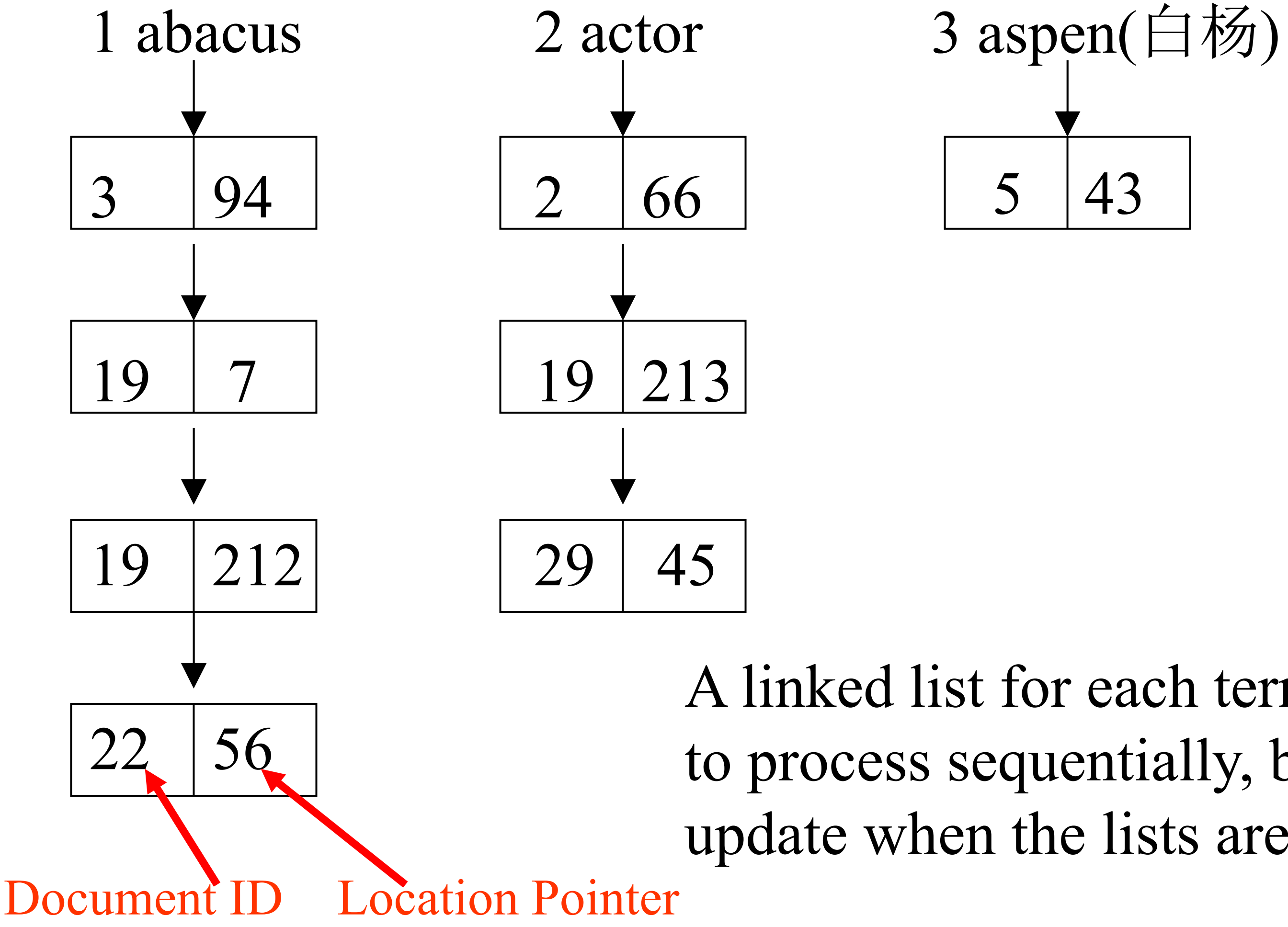
The **postings file** stores the elements of a sparse matrix(稀疏矩阵), the term assignment matrix.

It is stored as a separate **inverted list** for each column, i.e., a list corresponding to each term in the index file.

Each element in an inverted list is called a **posting (记录)**, i.e., the occurrence on a term in a document

Each list consists of one or many individual postings.

Postings File: A Linked List for Each Term



A linked list for each term is convenient to process sequentially, but slow to update when the lists are long.

Data for Calculating Term Weights

The calculation of term weights requires extra data to be held in the inverted file system.

For each term, t_j and document, d$_i$

f_{ij} number of occurrences of t_j in $\mathbf{d}_i$

For each term, t_j

n_j number of documents containing t_j

For each document, d$_i$

m_i maximum frequency of any term in $\mathbf{d}_i$

For the entire document file

n total number of documents

Index File: Individual Records for Each Term

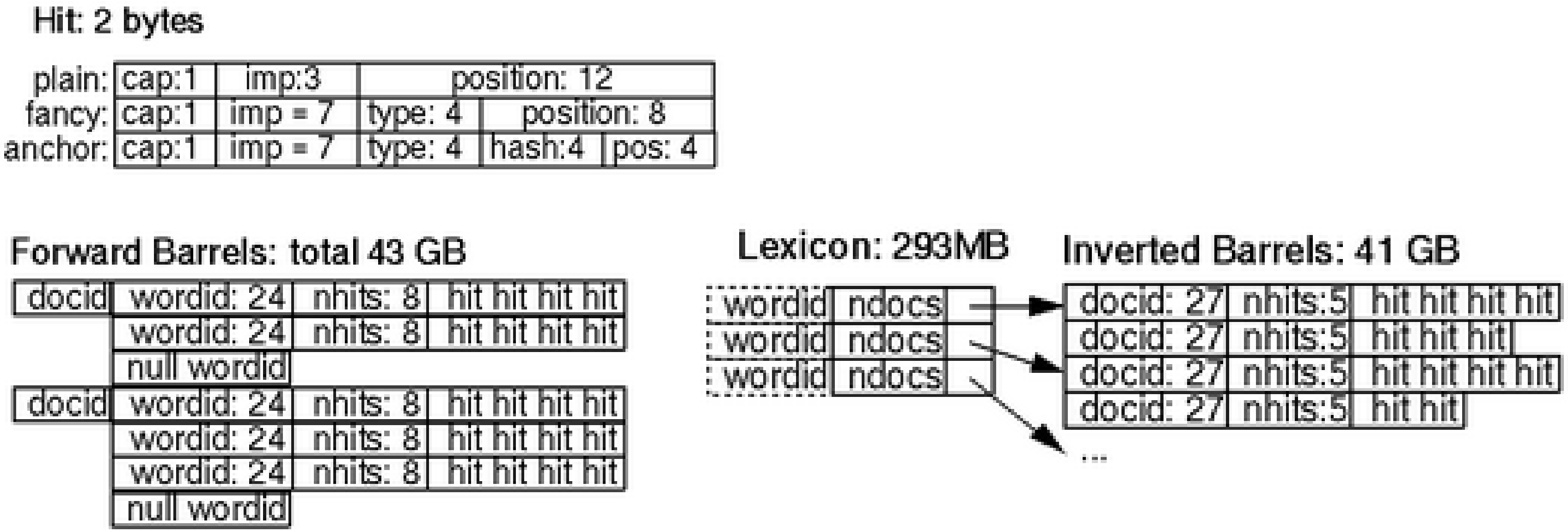
The record for term j in the index file contains:

term j
pointer to inverted (postings) list for term j
number of documents in which term j occurs (n_j)

Inverted Files: Google’s Example

Ref: Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, 1998

- It’s the earliest version of Google (before 1998)
- For plain text: font size is represented relative to the rest of the document using 3 bits


Hit: 2 bytes
plain: cap:1 imp:3 position: 12
fancy: cap:1 imp = 7 type: 4 position: 8
anchor: cap:1 imp = 7 type: 4 hash:4 pos: 4
Forward Barrels: total 43 GB
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
docid wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
wordid: 24 nhits: 8 hit hit hit hit
null wordid
Lexicon: 293MB
wordid ndocs
wordid ndocs
wordid ndocs
Inverted Barrels: 41 GB
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit hit
docid: 27 nhits:5 hit hit hit hit
docid: 27 nhits:5 hit hit
...

Length of Postings File

For a common term there may be very large numbers of postings for a given term.

Example:

1,000,000,000 documents
1,000,000 distinct words
average length 1,000 words per document
How many postings we may get?
10^{12} postings

Postings File

Merging inverted lists is the most computationally intensive task in many information retrieval systems.

Since inverted lists may be long, it is important to match postings efficiently.

Usually, the inverted lists will be held on disk and paged into memory for matching. Therefore algorithms for matching postings process the lists sequentially.

For efficient matching, the inverted lists should all be sorted in the same sequence.

Inverted lists are commonly cached to minimize disk accesses.

Structure of Index File

Index File Structures: Linear Index

Advantages

Can be searched quickly, e.g., by binary search, O(log *n*)

Good for lexicographic processing, e.g., *comp**

Convenient for batch updating

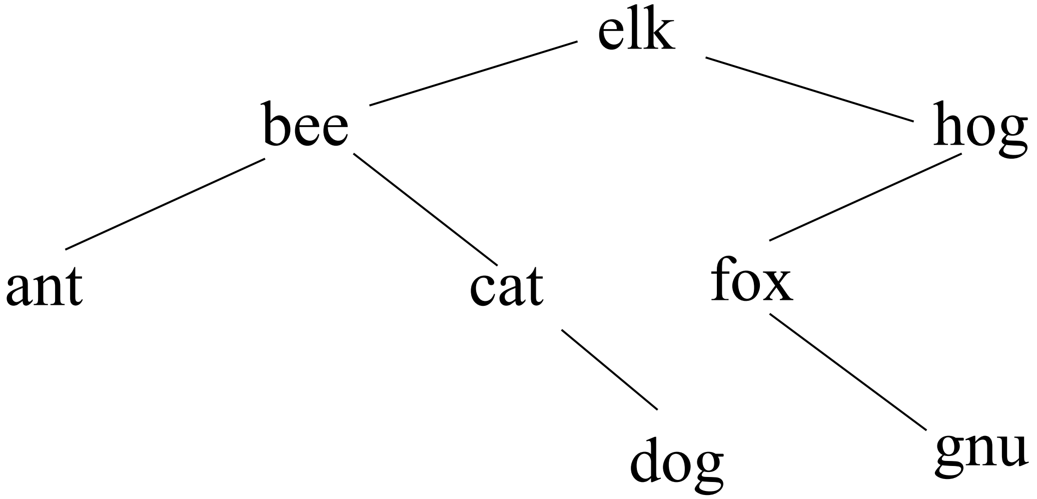
Economical use of storage

Disadvantages

Index must be rebuilt if an extra term is added

Index File Structures: Binary Tree

Input: elk, hog, bee, fox, cat, gnu, ant, dog



Binary Tree

Advantages

Can be searched quickly

Convenient for batch updating

Easy to add an extra term

Economical use of storage

Disadvantages

Less good for lexicographic processing, e.g., *comp**

Tree tends to become unbalanced

If the index is held on disk, important to optimize the number of disk accesses

Binary Tree

Calculation of maximum depth of tree.

Worst case: depth = n

$O(n)$

Ideal case (Balanced tree): depth = $\log(n + 1)/\log 2$

$O(\log n)$

Illustrates importance of balanced trees.

Right Threaded Binary Tree（右索二叉树）

Threaded tree:

A binary search tree in which each node uses an otherwise-empty left child link to refer to the node ‘s in-order predecessor (前导节点) and an empty right child link to refer to its in-order successor(后续节点).

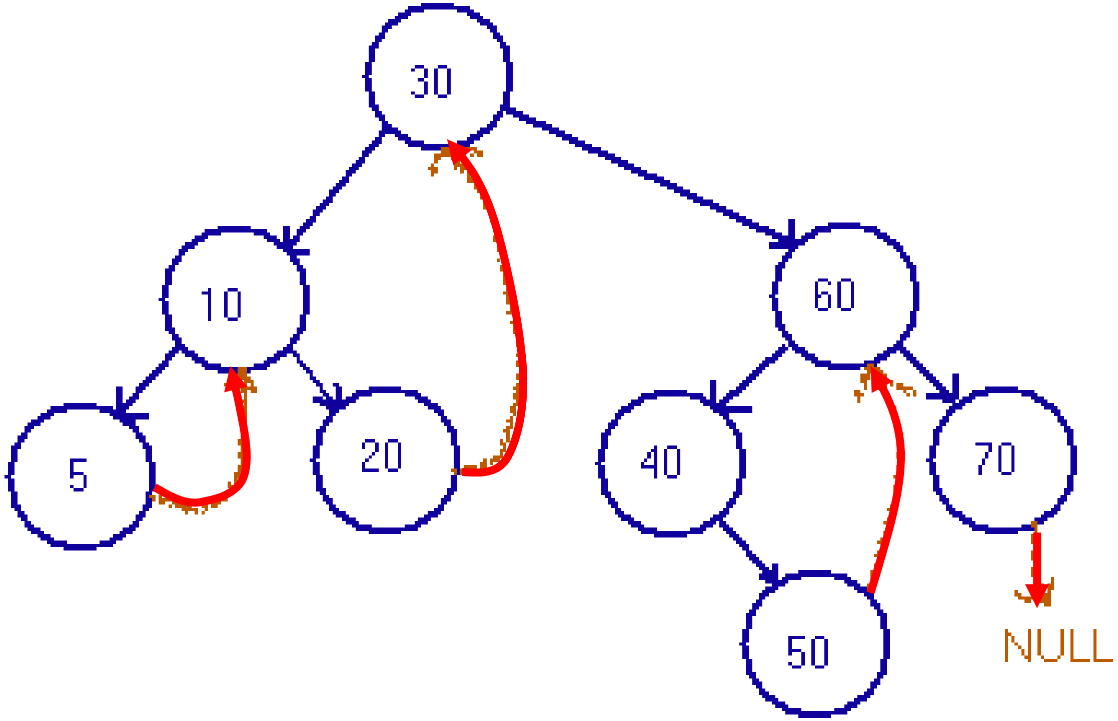
Right-threaded tree:

A variant of a threaded tree in which only the right thread, i.e. link to the successor, of each node is maintained. Can be used for lexicographic processing.

A good data structure when index held in memory

Knuth vol 1, 2.3.1, page 325.

Right Threaded Binary Tree



From: Robert F. Rossa

B-trees

B-tree of order m:

A balanced, multiway search tree (均衡、多道搜索树):

Each node stores many keys

Root has between 1 and 2*m* keys.
All other internal nodes have between *m* and 2*m* keys.

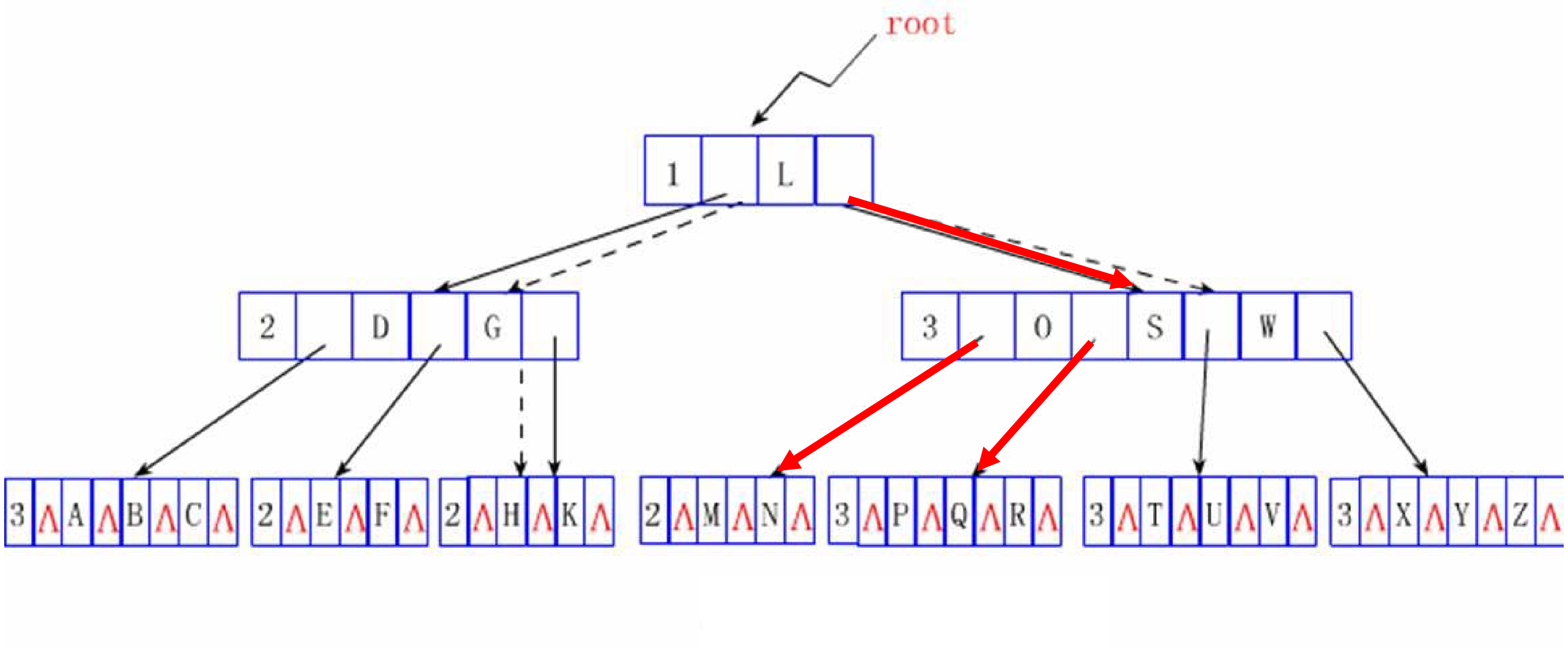
If k_i is the i^{th} key in a given internal node, then

-> all keys in the $(i-1)^{th}$ child are smaller than k_i
-> all keys in the i^{th} child are bigger than k_i

All leaves are at the same depth

B-trees

B-tree example (order 2)



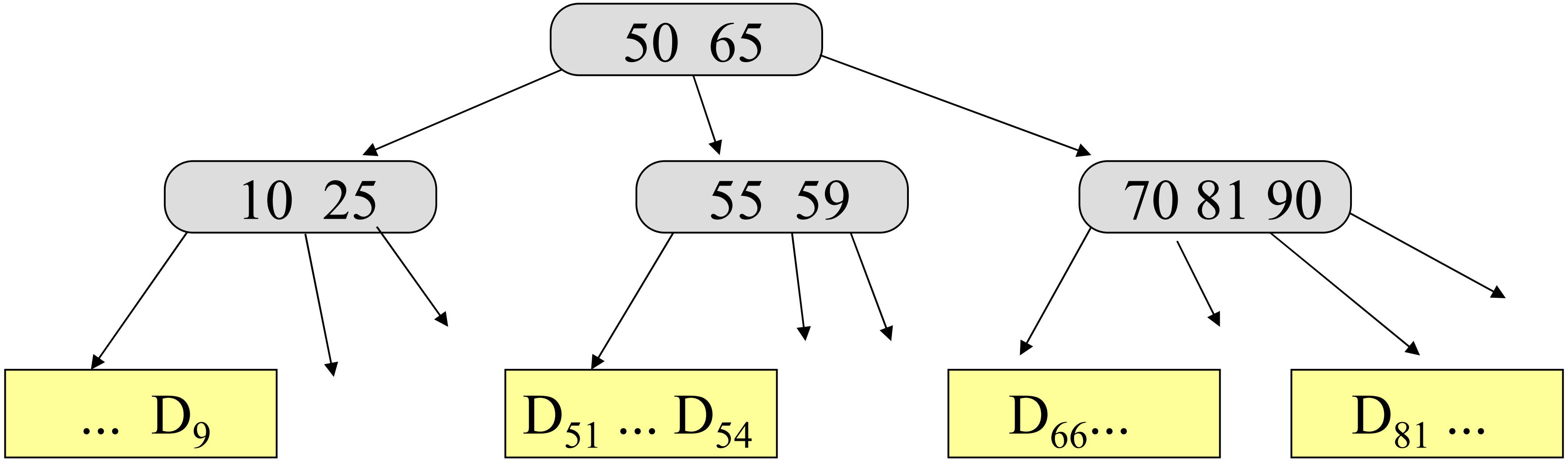
Every arrow points to a node containing between 2 and 4 keys.
A node with *k* keys has *k* + 1 pointers.

B^+-tree

Example: B^+-tree of order 2, bucket size 4

A B-tree is used as an index

Data is stored in the leaves of the tree, known as *buckets*



Tries: Search for Substring

In some information retrieval applications, any substring can be a search term.

Tries (ReTrieval) (搜索树), using **suffix trees**, provide lexicographical indexes for all the substrings in a document or set of documents.

String Matching

Find File: Find all files whose name includes the string **q**.

Simple algorithm: Build an inverted index of all substrings of the file names of the form *f,

Example: if the file name is foo.txt, search terms are:

foo.txt
oo.txt
o.txt
.txt
txt
xt
t

Lexicographic processing allows searching by any **q**.

Tries: Search for Substring

- **Concept**
- semi-infinite strings, or **sistrings (**半无限长字串)
  - Each sistring has a starting position in the text, and continues to the right until it is unique.
  - The sistrings are stored in (the leaves of) a tree, the **suffix tree**. Common parts are stored only once.
  - Each sistring can be associated with a location within a document where the sistring occurs.
  - Subtrees below a certain node represent all occurrences of the substring represented by that node.
- Suffix trees have a size of the same order of magnitude as the input documents.

Tries: Sistrings

A binary example

String: 01 100 100 010 111

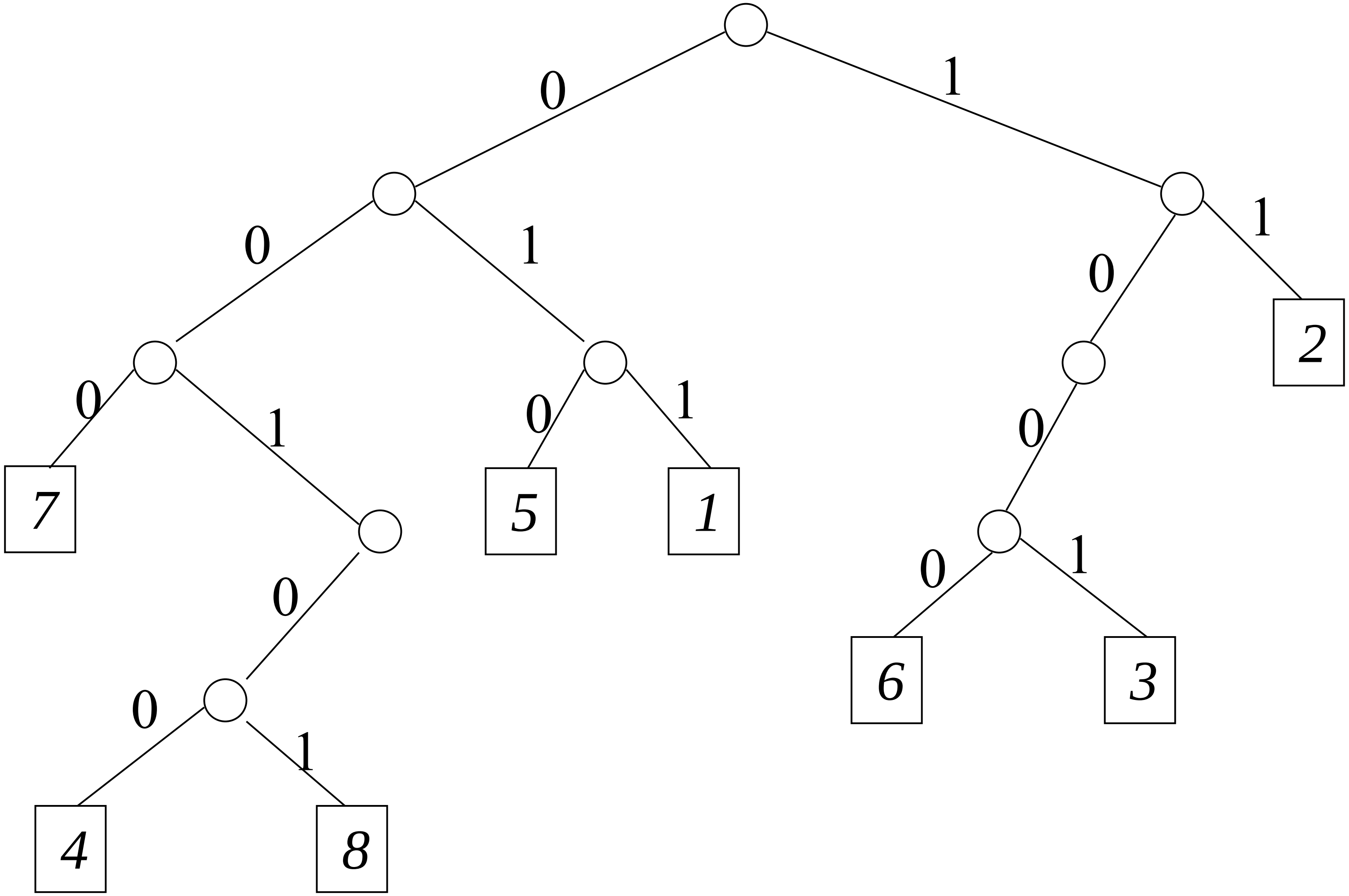
Sistrings:	*1*	01 100 100 010 111
	2	11 001 000 101 11
	3	10 010 001 011 1
	4	00 100 010 111
	5	01 000 101 11
	6	10 001 011 1
	7	00 010 111
	8	00 101 11

Tries: Lexical Ordering

$\boxed{7}$	00 010 111
$\boxed{4}$	00 100 010 111
$\boxed{8}$	00 101 11
$\boxed{5}$	01 000 101 11
$\boxed{1}$	01 100 100 010 111
$\boxed{6}$	10 001 011 1
$\boxed{3}$	10 010 001 011 1
$\boxed{2}$	11 001 000 101 11

Unique string indicated in blue

Trie: Basic Concept



Trie: Exercise

- Build the Trie of below string, the minimum length of a suffix is 6 characters(5 Minutes):
  - SFDSSSRTFSGEP

Suffix tree properties

- For a string S of length *n*, there are *n* leaves and at most *n* internal nodes.
  - therefore requires only linear space
- Each leaf represents a unique suffix.
- Concatenation of edge labels from root to a leaf spells out the suffix.
- Each internal node represents a distinct common prefix to at least two suffixes.

Tries: Implementation

- Double Array Trie
  - Theppitak Karoonboonyanan. ***An Implementation of Double-Array Trie.***
  - http://linux.thai.net/~thep/datrie/datrie.html

Content

- *Brief Review of Vector Space Model*
- *Inverted Files*
- * Index Models (Feature Selection) (self study)

Feature Selection (indexing model)

- Basic Issue: Which terms should be used to index (describe) a document?
- Different focus than retrieval model, but related
- Sometimes seen as *term weighting*
- Some approaches
  - TF·IDF
  - Term Discrimination model
  - 2-Poisson model
  - Clumping model
  - Language models

Index models

Think for a bit

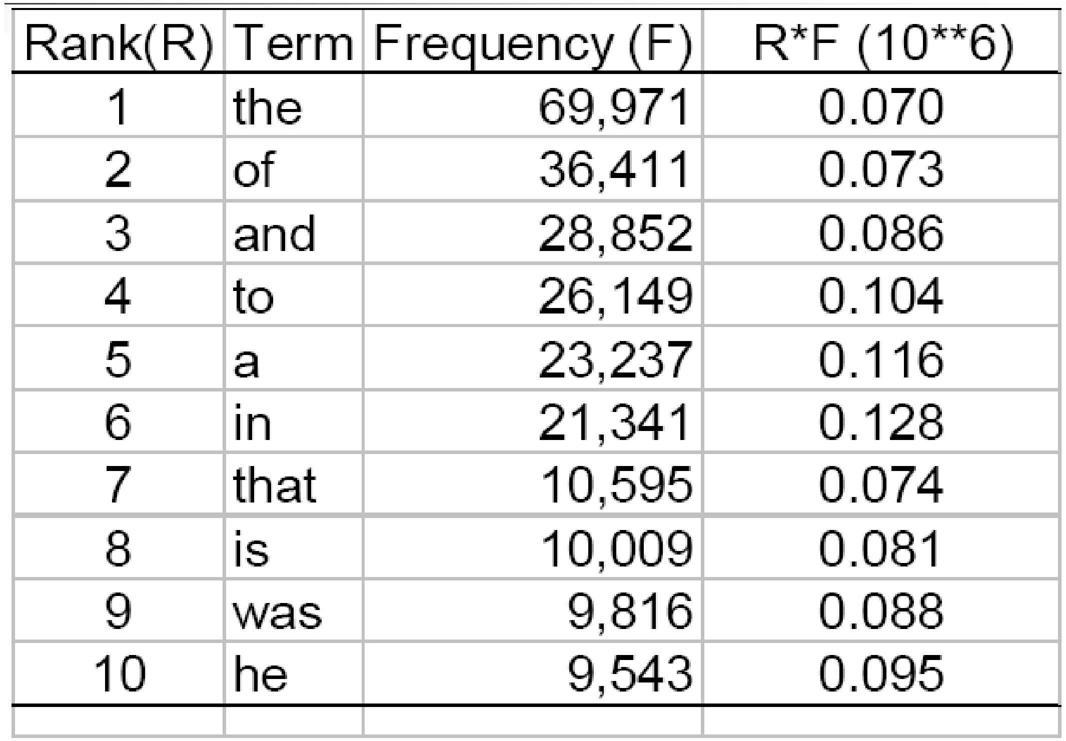
- What makes a term good for indexing?
  - Trying to represent “key” concepts in a document
- What makes an index term good for a query?

TF·IDF, just TF

- Standard weighting approach for many IR systems
  - many different variations of exactly how it is calculated
- TF component - the more often a term occurs in a document, the more important it is in describing that document ?

Zipf’ s Law (1949)
Word Frequency vs. Rank

$$f \propto \frac{1}{r} \qquad f \cdot r = k \ \ (\text{for constant } k)$$



Rank(R)	Term	Frequency (F)	R*F (10**6)
1	the	69,971	0.070
2	of	36,411	0.073
3	and	28,852	0.086
4	to	26,149	0.104
5	a	23,237	0.116
6	in	21,341	0.128
7	that	10,595	0.074
8	is	10,009	0.081
9	was	9,816	0.088
10	he	9,543	0.095

- Pro and Con
  - Remove stop words can greatly decrease the size of Inverted File
  - The most useful words are not occurred very frequently

Zipf's Law in Music (2011)

XH Yang, QC Chen, XL Wang, A Dictionary Based Indexing Approach for Music Information Retrieval, IJCPL, 2011.05

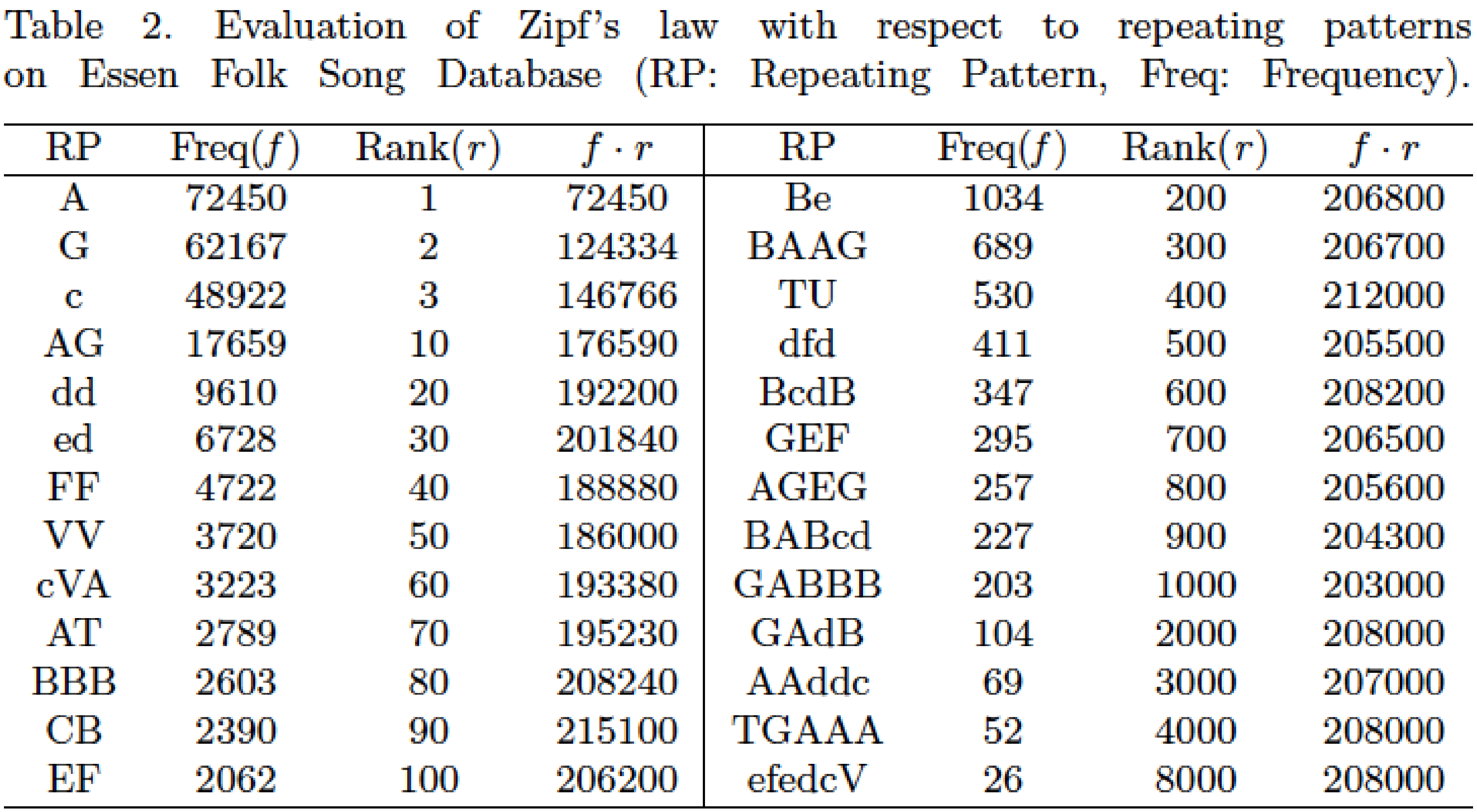
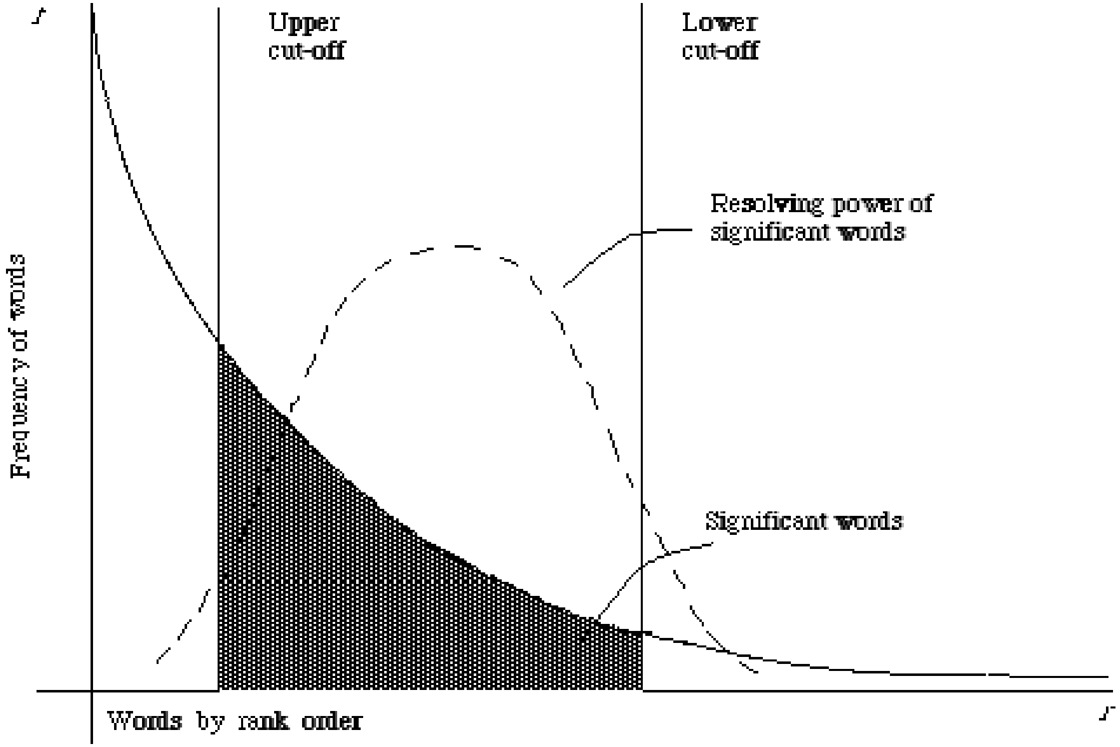


Table 2. Evaluation of Zipf's law with respect to repeating patterns on Essen Folk Song Database (RP: Repeating Pattern, Freq: Frequency).

RP	Freq(f)	Rank(r)	$f \cdot r$	RP	Freq(f)	Rank(r)	$f \cdot r$
A	72450	1	72450	Be	1034	200	206800
G	62167	2	124334	BAAG	689	300	206700
c	48922	3	146766	TU	530	400	212000
AG	17659	10	176590	dfd	411	500	205500
dd	9610	20	192200	BcdB	347	600	208200
ed	6728	30	201840	GEF	295	700	206500
FF	4722	40	188880	AGEG	257	800	205600
VV	3720	50	186000	BABcd	227	900	204300
cVA	3223	60	193380	GABBB	203	1000	203000
AT	2789	70	195230	GAdB	104	2000	208000
BBB	2603	80	208240	AAddc	69	3000	207000
CB	2390	90	215100	TGAAA	52	4000	208000
EF	2062	100	206200	efedcV	26	8000	208000

Term Frequency and Indexing Significance

- Luhn: term frequency is useful for determining significance for retrieval.



- from Chapter 2 of Information Retrieval, C.J. van Rijsbergen

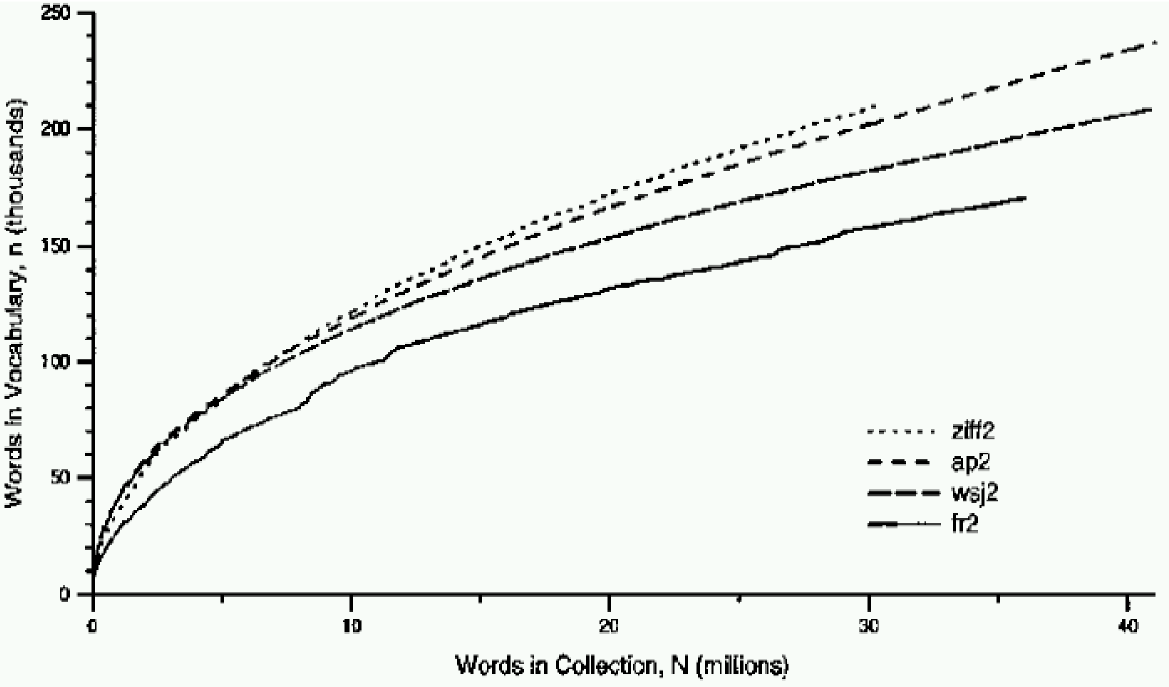
A question for estimation index scale

- How many terms we may get for a given scale document collection?

Heap’ s Law
Lexicon Size vs. Document Collection Size

$$n = |V| = K \cdot N^{\beta} \text{ with constants } K,\ 0 < \beta < 1$$

- Typical constants
  - K≈ 10−100
  - β ≈ 0.4−0.6 (approx. square-root)



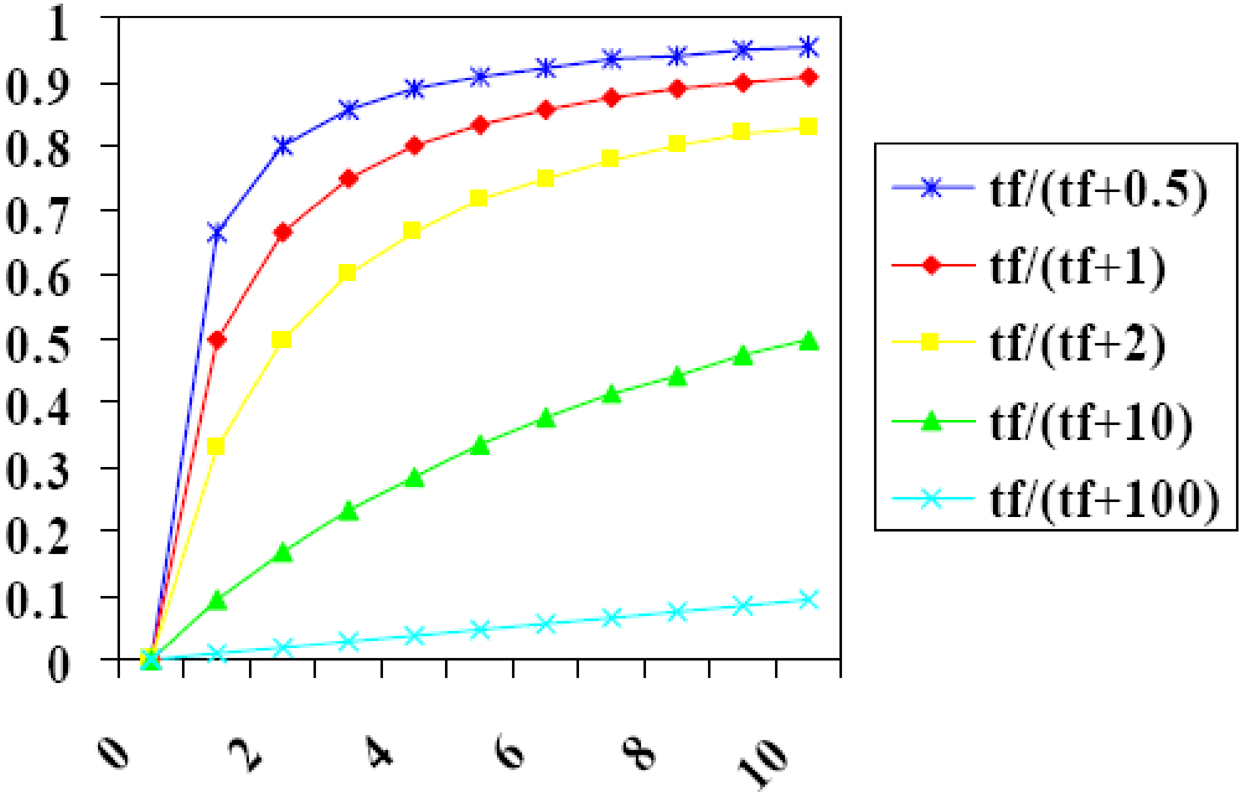
Robertson TF weight

- Often called “Okapi TF”
- Based on a set of simple criteria loosely connected to the 2-Poisson model[1]
- Basic formula is *tf/*(*k*+*tf*) where *k* is a constant (approx. 1-2)
- Document length introduced as a *verbosity* factor (冗长因子)
  - same topic but more words
  - e.g. function used in INQUERY:

$$\frac{tf}{tf + 0.5 + 1.5\dfrac{doc\_length}{avg\_doc\_length}}$$

[1] S.E. Robertson, S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In Proceedings of SIGIR-94, pages 232--241, 1994.

Robertson TF weight (Cont’d)



TF·IDF

- Inverse document frequency (IDF) weight proposed by Spark Jones in 1972
  - This version of IDF is log (*N*/*df*) + 1
  - *N* is the number of documents in the collection, *df* is the number of documents the term occurs in
  - IDF is -log p, the entropy of a term
  - sometimes normalized when used in TF.IDF combination
  - e.g. for INQUERY:

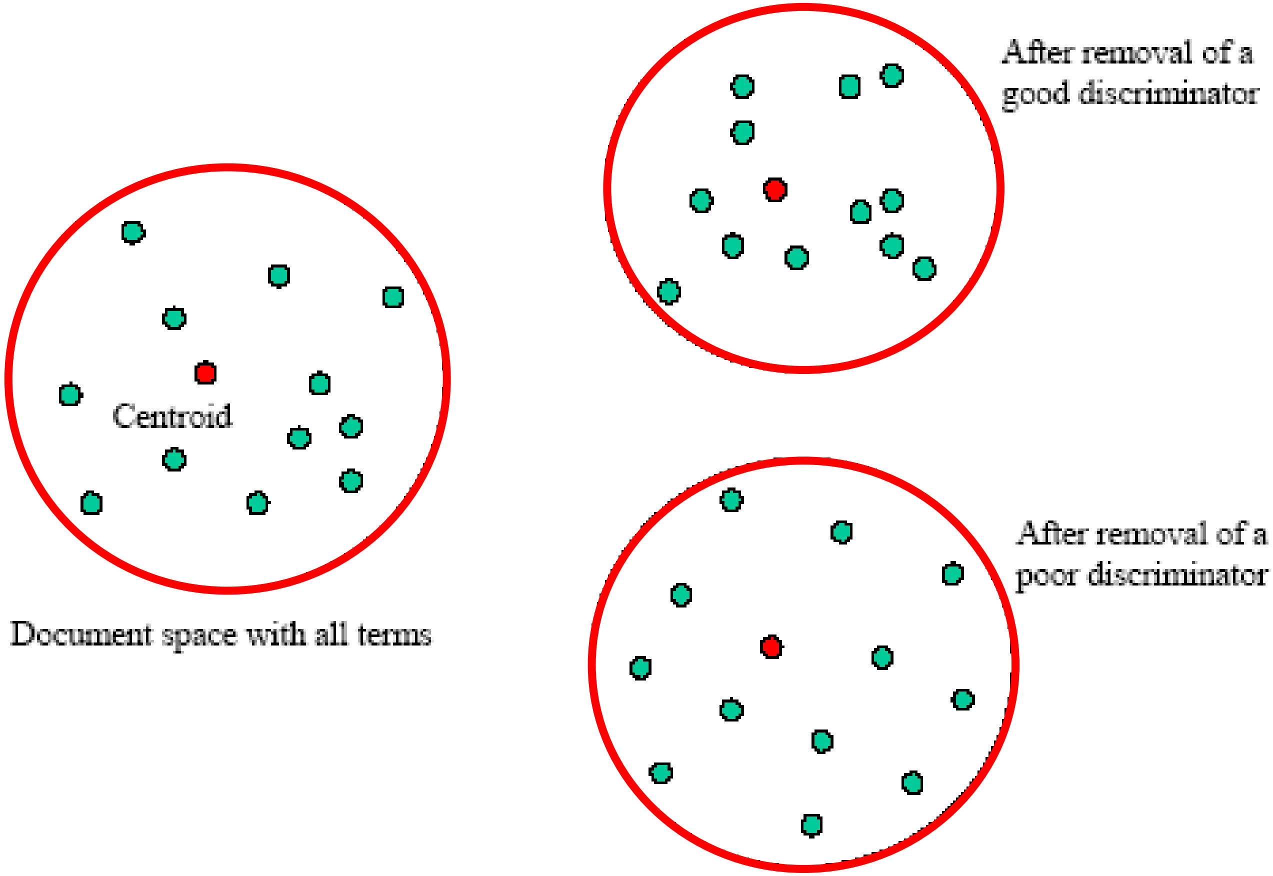
$$\frac{\log(\frac{N+0.5}{df})}{\log(N+1.0)}$$

- TF and IDF are combined using multiplication
- Regression has also been used to develop more complex ways of combining this information
- No satisfactory model behind these combinations

Term Discrimination Model

- Proposed by Salton in 1975
- Based on vector space model
  - documents and queries are vectors in an *n*-dimensional space for *n* terms
- Compute *discrimination value* of a term
  - degree to which use of the term will help to distinguish documents
- Compare average similarity of documents both with and without an index term

Term Discrimination Model



Term Discrimination Model (Cont’d)

- Compute average similarity or “density” of document space

$$AVGSIM = K\sum_{i=1}^{n}\sum_{i \neq j}\sum_{j=1}^{n} similar(D_i, D_j)$$

  - *AVGSIM* is the density
  - where K is a normalizing constant (e.g., 1/n(n-1))
  - *similar()* is a similarity function such as cosine correlation
- Can be computed more efficiently using an average document or *centroid*
  - frequencies in the centroid vector are average of frequencies in document vectors
  -

$$AVGSIM = K\sum_{i=1}^{n} similar(\overline{D}, D_i)$$

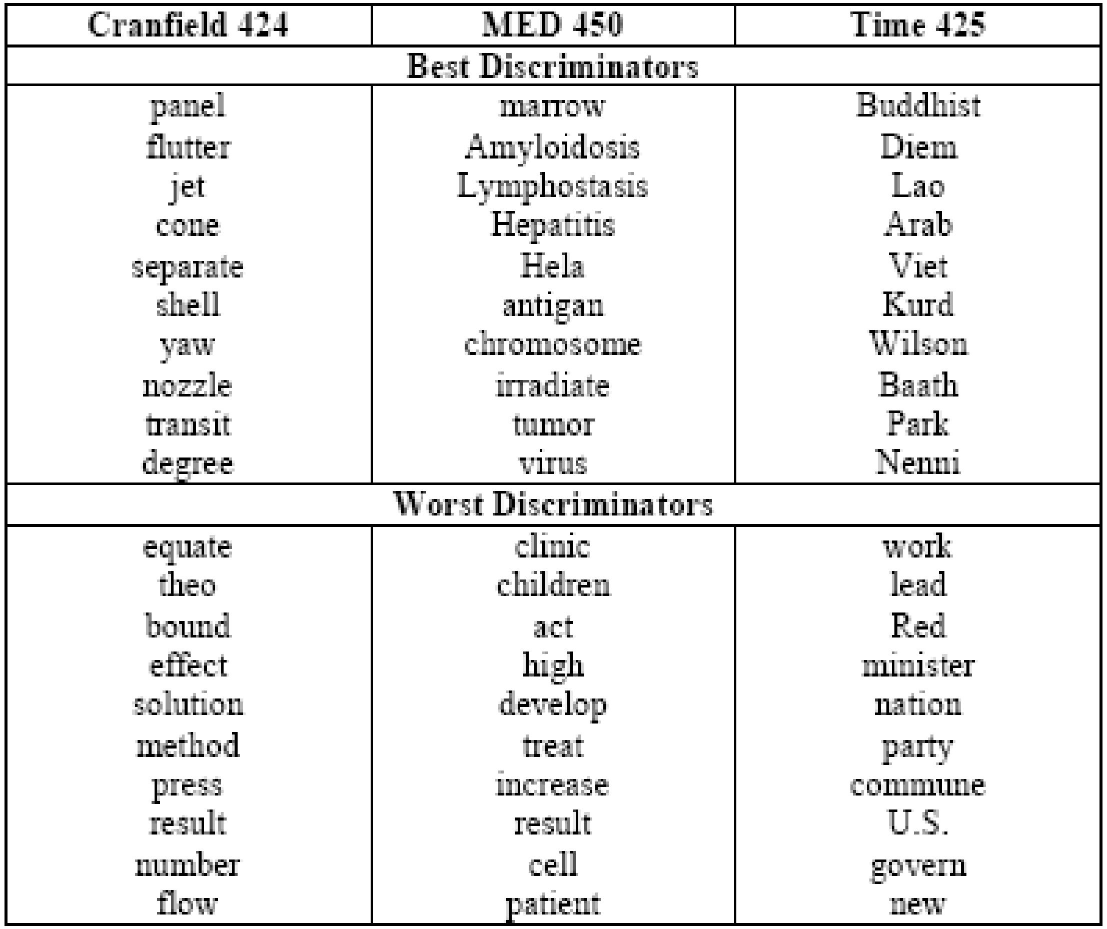
Term Discrimination Model (Cont’d)

- Let $(AVGSIM)_k$ be density with term *k* removed from documents
- Discrimination value for term k is

$$DISCVALUE_k = (AVGSIM)_k - AVGSIM$$

- Good discriminators have positive $DISCVALUE_k$
  - introduction of term decreases the density (moves some docs away)
  - tend to be medium frequency
- Indifferent discriminators have *DISCVALUE* near zero
  - introduction of term has no effect
  - tend to be low frequency
- Poor discriminators have negative *DISCVALUE*
  - introduction of term increases the density (moves all docs closer)
  - tend to be high frequency
- Obvious criticism is that discrimination of *relevant* and *nonrelevant* documents is the important factor

Best and Worst Discriminators for 3 Collections



Cranfield 424	MED 450	Time 425
Best Discriminators		
panel	marrow	Buddhist
flutter	Amyloidosis	Diem
jet	Lymphostasis	Lao
cone	Hepatitis	Arab
separate	Hela	Viet
shell	antigan	Kurd
yaw	chromosome	Wilson
nozzle	irradiate	Baath
transit	tumor	Park
degree	virus	Nenni
Worst Discriminators		
equate	clinic	work
theo	children	lead
bound	act	Red
effect	high	minister
solution	develop	nation
method	treat	party
press	increase	commune
result	result	U.S.
number	cell	govern
flow	patient	new

Summary

- Inverted File
  - Documents are indexed by term or word based posting lists
  - Index File structure includes linear, B-tree or Hash table. Usually B-trees are more efficient in searching.
- Feature Selection
  - To determine the most important or representative words or terms to be applied for document indexing.
  - Feature selection approaches: TF, IDF and their variations
  - Term discrimination model can be applied to select index terms

Question?