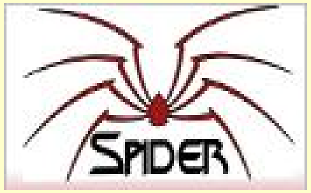
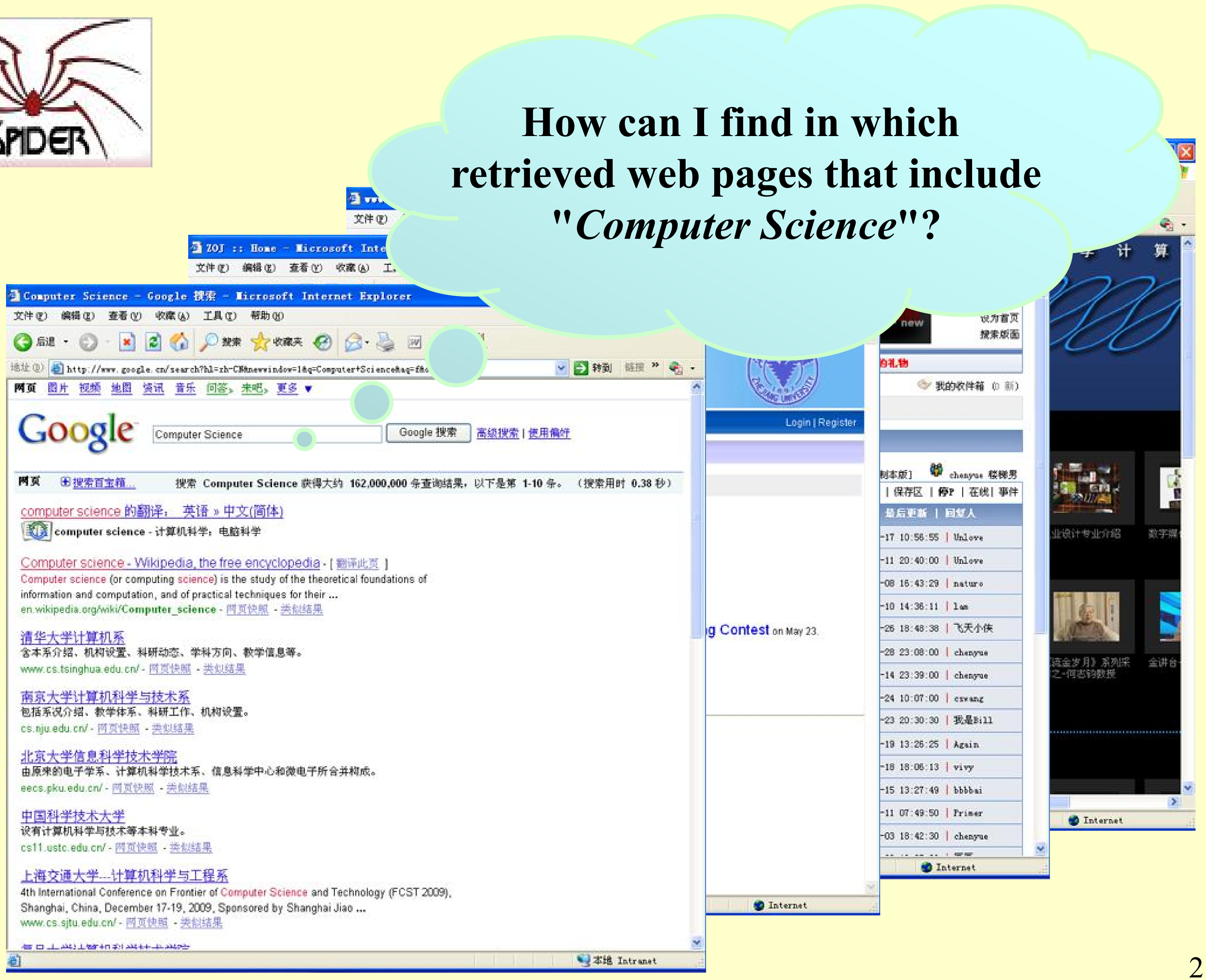
Inverted File Index

Inverted File Index





How can I find in which retrieved web pages that include "*Computer Science*"?

ZOJ :: Home - Microsoft Inte
Computer Science - Google 搜索 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://www.google.cn/search?hl=zh-CN&newwindow=1&q=Computer+Science&aq=f&o
转到 链接
网页 图片 视频 地图 资讯 音乐 问答 来吧 更多
Google
Computer Science
Google 搜索
高级搜索 | 使用偏好
Login | Register
g Contest on May 23
Internet
本地 Intranet

网页 搜索百宝箱... 搜索 Computer Science 获得大约 162,000,000 条查询结果，以下是第 1-10 条。（搜索用时 0.38 秒）

computer science 的翻译： 英语 » 中文(简体)
computer science - 计算机科学；电脑科学

Computer science - Wikipedia, the free encyclopedia - [翻译此页]
Computer science (or computing science) is the study of the theoretical foundations of information and computation, and of practical techniques for their ...
en.wikipedia.org/wiki/Computer_science - 网页快照 - 类似结果

清华大学计算机系
含本系介绍、机构设置、科研动态、学科方向、教学信息等。
www.cs.tsinghua.edu.cn/ - 网页快照 - 类似结果

南京大学计算机科学与技术系
包括系况介绍、教学体系、科研工作、机构设置。
cs.nju.edu.cn/ - 网页快照 - 类似结果

北京大学信息科学技术学院
由原来的电子学系、计算机科学技术系、信息科学中心和微电子所合并构成。
eecs.pku.edu.cn/ - 网页快照 - 类似结果

中国科学技术大学
设有计算机科学与技术等本科专业。
cs11.ustc.edu.cn/ - 网页快照 - 类似结果

上海交通大学---计算机科学与工程系
4th International Conference on Frontier of Computer Science and Technology (FCST 2009), Shanghai, China, December 17-19, 2009, Sponsored by Shanghai Jiao ...
www.cs.sjtu.edu.cn/ - 网页快照 - 类似结果

☞ **Solution 1:** **Scan each page for the string "Computer Science".**

Wait till your next life !

Have more than 1 billion web pages Indexed

☞ Solution 2: Term-Document Incidence Matrix

〖Example〗 Document sets

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

	1	2	3	4
a	0	1	1	1
arrived	0	0	1	1
damaged	0	1	0	0
delivery	0	0	1	0
fire	0	1	0	0
gold	1	1	0	1
of	0	1	1	1
in	0	1	1	1
shipment	0	1	0	1
silver	1	0	1	0
truck	1	0	1	1

silver & truck

☞ Solution 3: Compact Version - Inverted File Index

【Definition】 **Index** is a mechanism for locating a given term in a text.

【Definition】 **Inverted file** contains a list of pointers (e.g. the number of a page) to all occurrences of that term in the text.

Doc	Text
1	Gold silv
2	Shipment of g damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

Index

No.	Term	Times; Documents
		2,3,4>
		1; 3>
5	fire	<1; 2>
6	gold	<3; 1,2,4>
7	of	<3; 2,3,4>
8	in	<3; 2,3,4>
9	shipment	<2; 2,4>
10	silver	<2; 1,3>
11	truck	<3; 1,3,4>

Inverted because it lists for a *term*, all documents that contain the term

Inverted File Index

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

Term Dictionary

Posting List

How to easily print the sentences which contain the words and highlight the words?

Why do we keep “times” (frequency)?

Index Generator

```
while ( read a document D ) {
    while ( read a term T in D ) {
        if ( Find( Dictionary, T ) == false )
            Insert( Dictionary, T );
        Get T's posting list;
        Insert a node to T's posting list;
    }
}
Write the inverted index to disk;
```

Token Analyzer
Stop Filter

Vocabulary
Scanner

Vocabulary
Insertor

Memory management

While reading a term ……

✂ ***Word Stemming***

Process a word so that only its stem or root form is left.

〖Example〗

$$\left.\begin{array}{l}\textbf{Process}\\ \textbf{processing}\\ \textbf{processes}\\ \textbf{processed}\end{array}\right\}\ \textbf{process}\qquad \left.\begin{array}{l}\textbf{says}\\ \textbf{said}\\ \textbf{saying}\end{array}\right\}\ \textbf{say}$$

✂ ***Stop Words***

Some words are so common that almost every document contains them, such as “a” “the” “it”. It is useless to index them. They are called *stop words*. We can eliminate them from the original documents.

While accessing a term ……

☞ **Solution 1:** Search trees (*B- trees, B+ trees, Tries, …*)

☞ **Solution 2:** Hashing

Discussion 3:

What are the pros and cons of using hashing, comparing to using search trees?

While not having enough memory ……

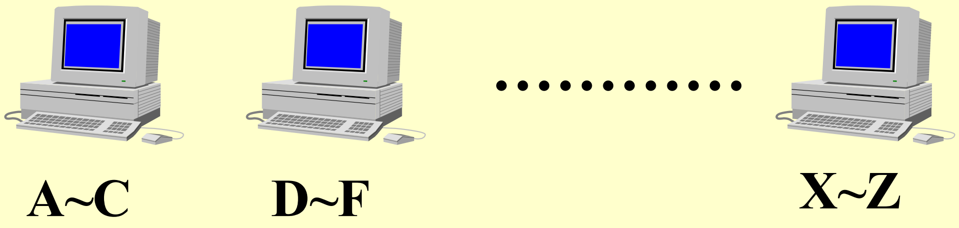
```
while ( read a document D ) {
  while ( read a term T in D ) {
    if ( Find( Dictionary, T ) == false )
      Insert( Dictionary, T );
    Get T's posting list;
    Insert a node to T's posting list;
  }
}
```

Sorted

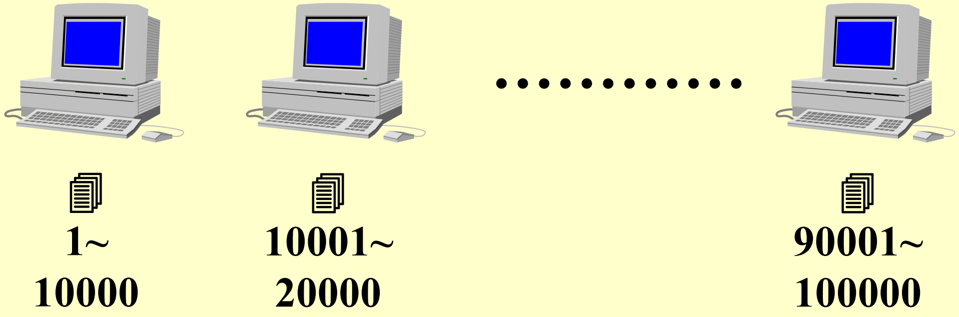
Distributed indexing (for web-scale indexing — don't try this at home!)

—— Each node contains index of a <u>subset</u> of collection

☞ **Solution 1: Term-partitioned index**



☞ **Solution 2: Document-partitioned index**

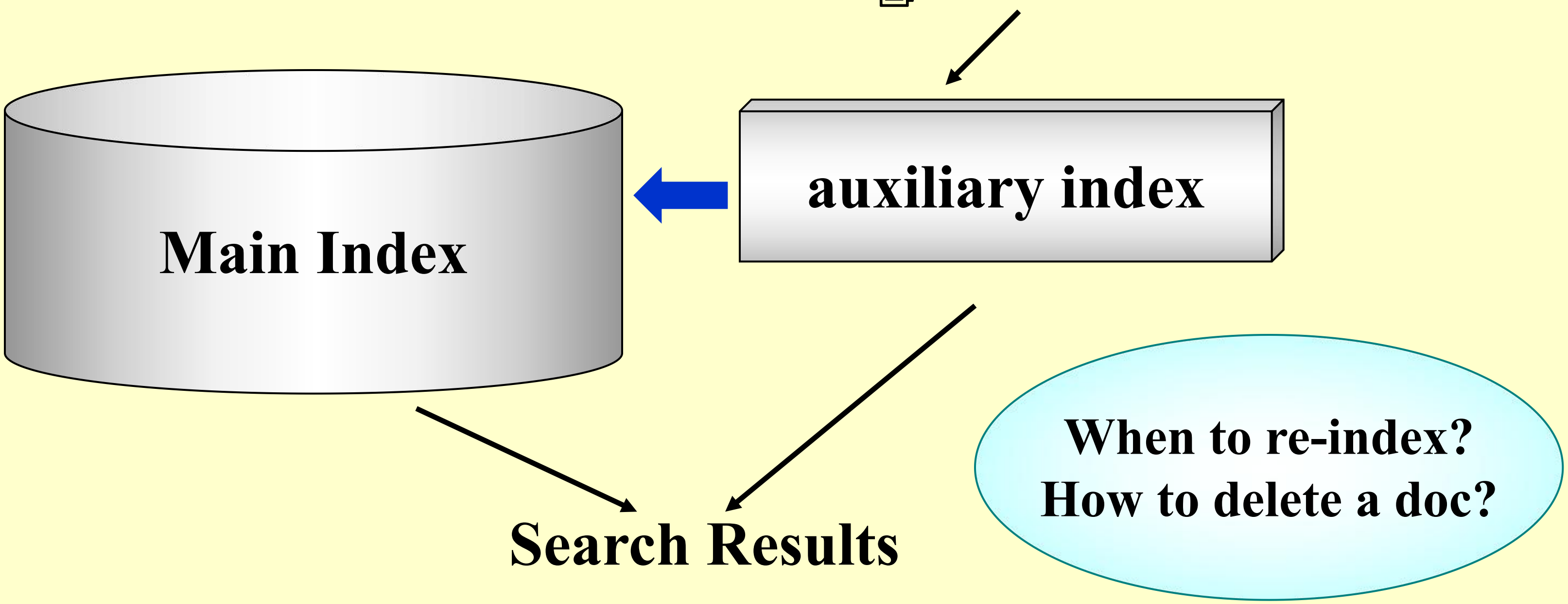


Dynamic indexing

☞ Docs come in over time

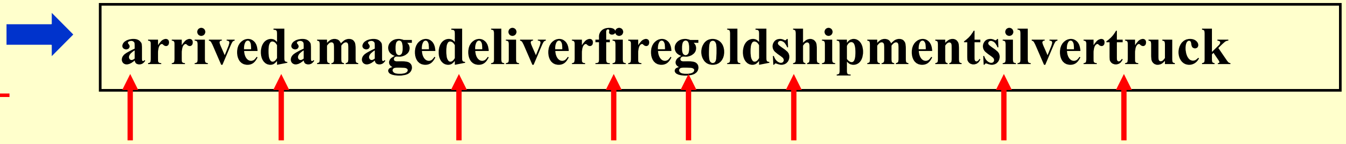
- postings updates for terms already in dictionary
- new terms added to dictionary

☞ Docs get deleted



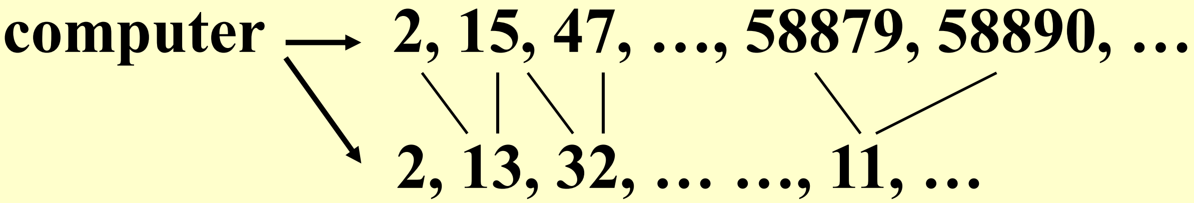
Compression

Term
~~a~~
arrive
damage
deliver
fire
gold
~~of~~
~~in~~
shipment
silver
truck



arrivedamagedeliverfiregoldshipmentsilvertruck

Posting List:



computer → 2, 15, 47, …, 58879, 58890, …

2, 13, 32, … …, 11, …

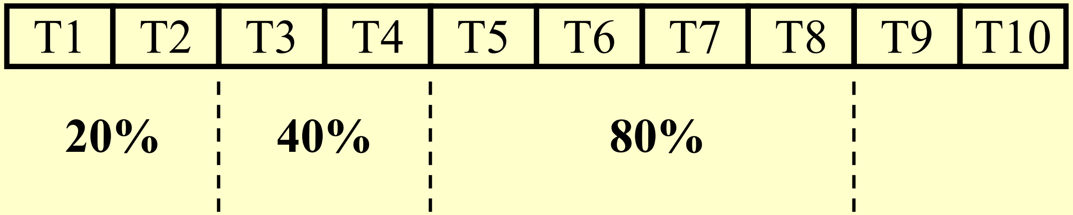
☝ Most gaps can be encoded with far fewer than 20 bits

Thresholding

☞ **Document:** only retrieve the top *x* documents where the documents are ranked by weight

- 👎 Not feasible for Boolean queries
- 👎 Can miss some relevant documents due to truncation

☞ **Query:** Sort the query terms by their frequency in ascending order; search according to only some percentage of the original query terms



T1 T2 T3 T4 T5 T6 T7 T8 T9 T10

20% 40% 80%

Measures for a search engine

☞ **How fast does it index**

- Number of documents/hour

☞ **How fast does it search**

- Latency as a function of index size

☞ **Expressiveness of query language**

- Ability to express complex information needs
- Speed on complex queries

☞ **User happiness ?**

- **Data Retrieval Performance Evaluation** (after establishing correctness)

  > Response time

  > Index space

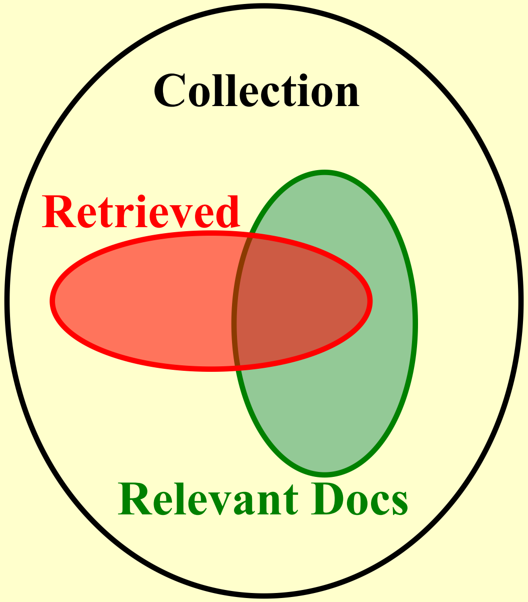
- **Information Retrieval Performance Evaluation**

  > + How *relevant* is the answer set?

Relevance **measurement requires 3 elements:**

1. **A benchmark document collection**
2. **A benchmark suite of queries**
3. **A binary assessment of either Relevant or Irrelevant for each query-doc pair**

	Relevant	Irrelevant
Retrieved	R_R	I_R
Not Retrieved	R_N	I_N

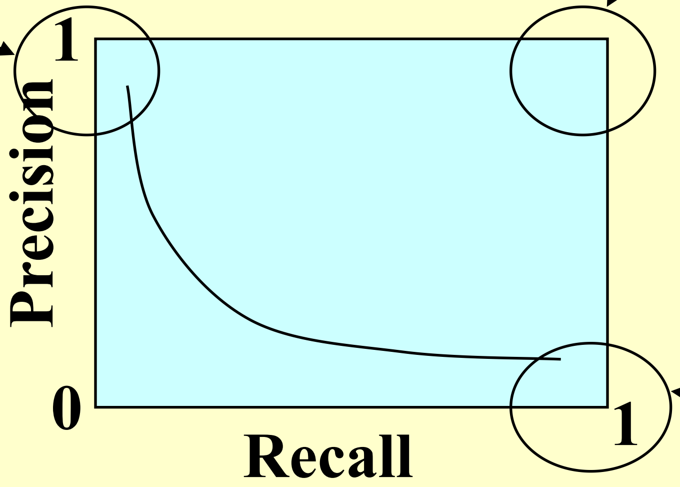


Precision $P = R_R / (R_R + I_R)$

Recall $R = R_R / (R_R + R_N)$

Returns relevant documents but misses many useful ones too

The ideal



Returns most relevant documents but includes lot of junk

Discussion 4:

How to improve the *relevancy* of search results?

Reference:

Download "InvertedFileIndex.zip".

- **The Google File System.pdf**
- **Building an Inverted Index.pdf**
- **Inverted Index Construction(ppt).pdf**
- **Compression.pdf**