

# **A Survey On Web Information Retrieval Technologies**

Lan Huang

Computer Science Department

State University of New York

Stony Brook, NY 11790-4400

lanhuang@cs.sunysb.edu

# Outline

- **Introduction**
- **Search Engines**
  - **architecture (Google)**
  - **implementation issues**
  - **algorithms**
- **Hierarchical Directories**
  - **automatic categorization (TAPER)**
  - **automatic categorization (ODP/OpenGrid)**
- **Measuring the Web**

## Web Information Retrieval vs Classic Information Retrieval

- **Bulk:** the size of Internet is over 350 M documents as of July 1998
- **Dynamic Internet:** changing everyday
- **Heterogeneity:** pictures, audio, text and scripts etc.
- **Variety of Languages:** more than 100 types
- **Duplication:** nearly 30% are duplicates
- **High Linkage:** more than 8 links per page

## Web Information Retrieval vs Classic Information Retrieval (cont'd)

- **Ill-formed queries:** single word, imprecise word, low efforts
- **Wide Variance in Users:** Web users varies in their needs, expectations and knowledge
- **Specific Behavior:** 85% users only look at the first screen of the returned results from search engines, 78% never modify their queries

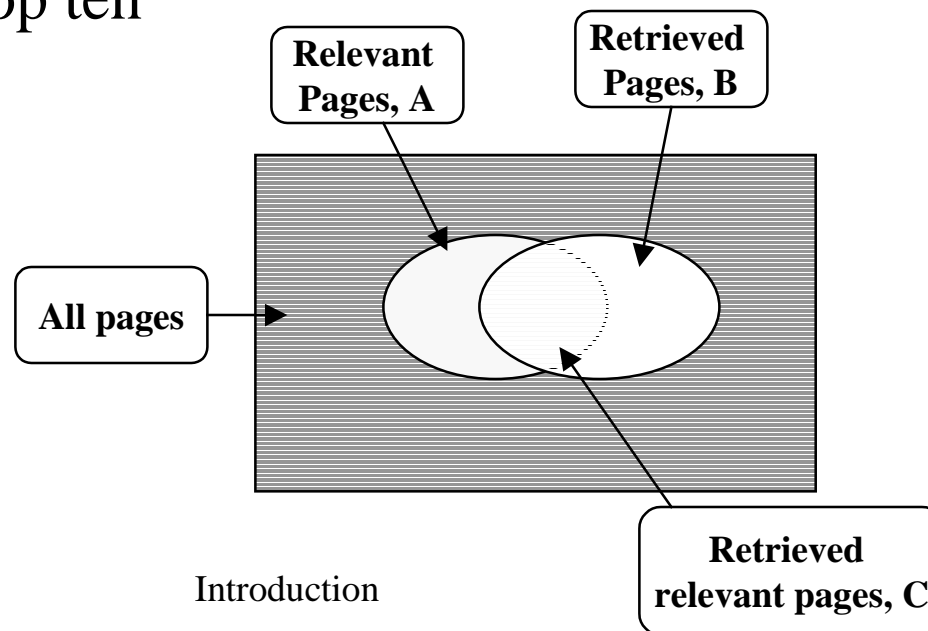
# The Goal

- **Classic IR**

- recall  $C/A$
- precision  $C/B$
- precision of top ten pages

- **Web IR**

- Both high-relevance and high-quality pages

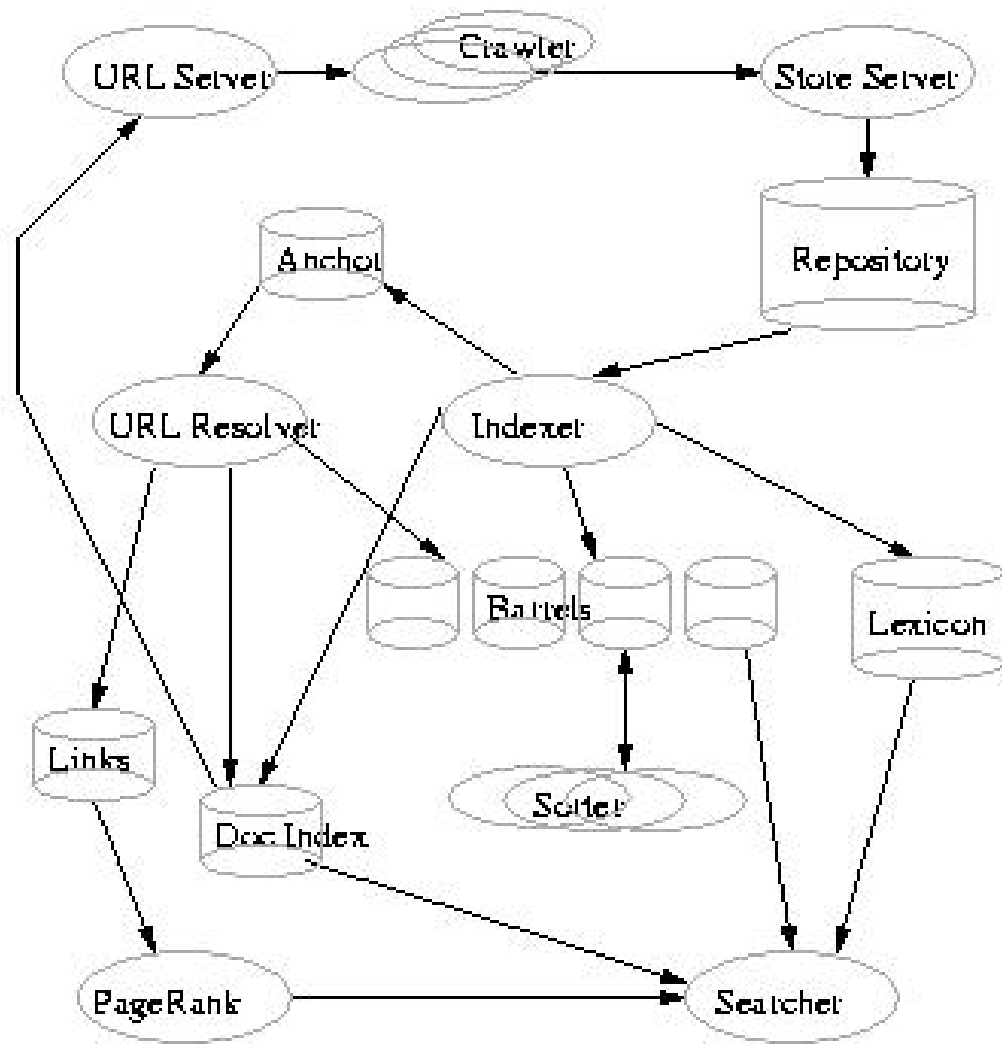


# General-Purpose Search Engines

- **Current Status of Search Engines**
- **Architecture**
- **Engineering Details**
  - **Crawling the Web**
  - **Caching Query Results**
  - **Incremental Updates to Inverted Index**
- **Algorithmic Issues**
  - **Ranking (PageRank, HITS)**
  - **Duplicate Elimination**

## Current Status of Search Engines (as of Nov. 1998)

- **Google:** 90-100 M pages. Excellent page ranking.
- **Alta Vista:** 275 M pages. Advanced Search is powerful.
- **Northern Light:** 250M pages. Good at academic and business domains.
- **Infoseek:** 90 M pages. Nice *Search within results* features.
- **FastSearch:** 250M pages. Fastest Engine.



## Google architecture

# Data Structure of Google

- **Repository:** it contains compressed full HTML texts of every web page.
- **Document Index:** various information for a document--pointers, checksum, statistics.
- **Lexicon:** 14 million words.
- **Hit Lists:** a list of occurrences of a particular word in a particular document.
- **Forward Index:** it holds a number of barrels which holds a range of wordID's with hit lists followed.
- **Inverted Index:** a list of sorted barrels.

# Forward, Inverse Index, Lexicon

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

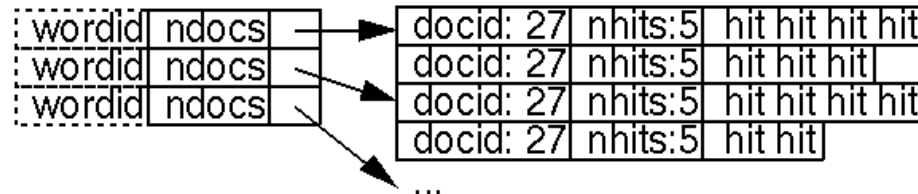
Forward Barrels: total 43 GB

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

Inverted Barrels: 41 GB



# Engineering Issues

- **Crawling the Web**
- **Caching the Query Results**
- **Incremental Updates to Inverted Index**

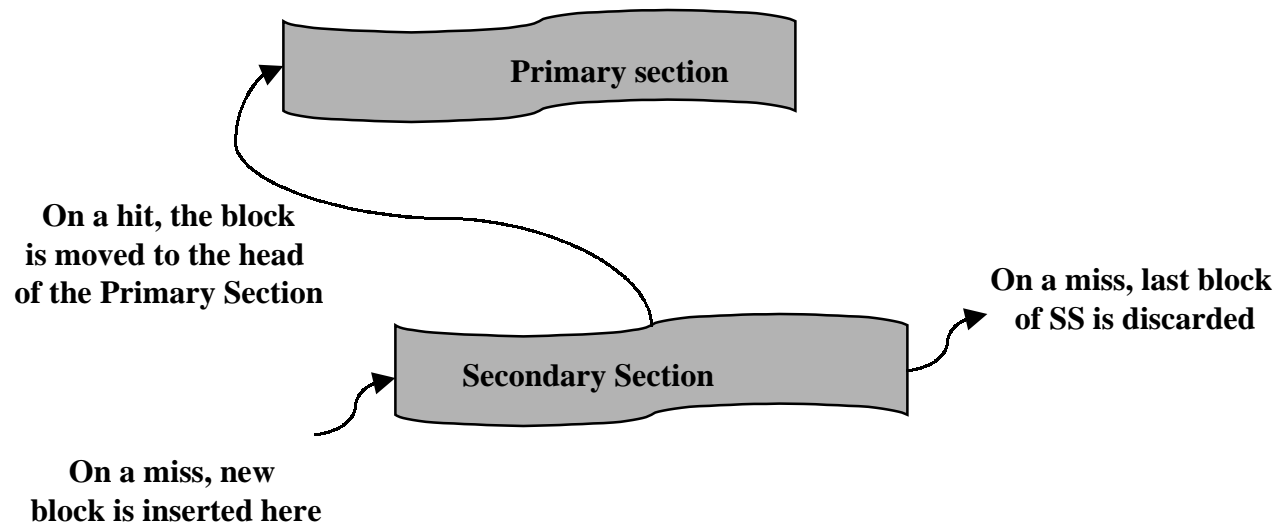
# Crawling the Web

- **In what order to scan the URLs in the queue**
- **Asynchronous I/O to manage events**
- **Multiple queues (500) : all URLs from same server goes to the same queue.**

# Caching Query Results

- **LRU-2S**
  - **The blocks in the cache are ordered in the same fashion as a cache using LRU replacement policy.**
  - **The cache is divided into two parts: primary section, and secondary section**
  - **On a miss, the last block of the secondary section is replaced, and the new block is put on the head of the secondary section.**
  - **On a hit, the block is moved to the head of the primary section .**

## Caching Query Results (cont'd)



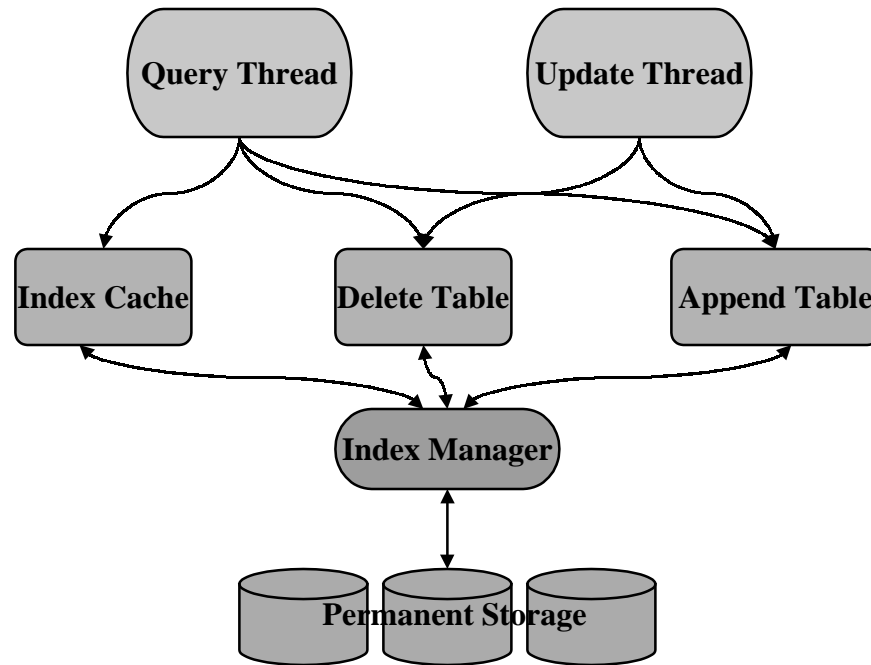
## Conclusions for Caching

- **Medium-sized caches (hundred Mbytes) can easily exploit the locality and serve a significant percentage of the submitted queries**
- **Effective Cache Replacement Policies should take into account both recency and frequency.**

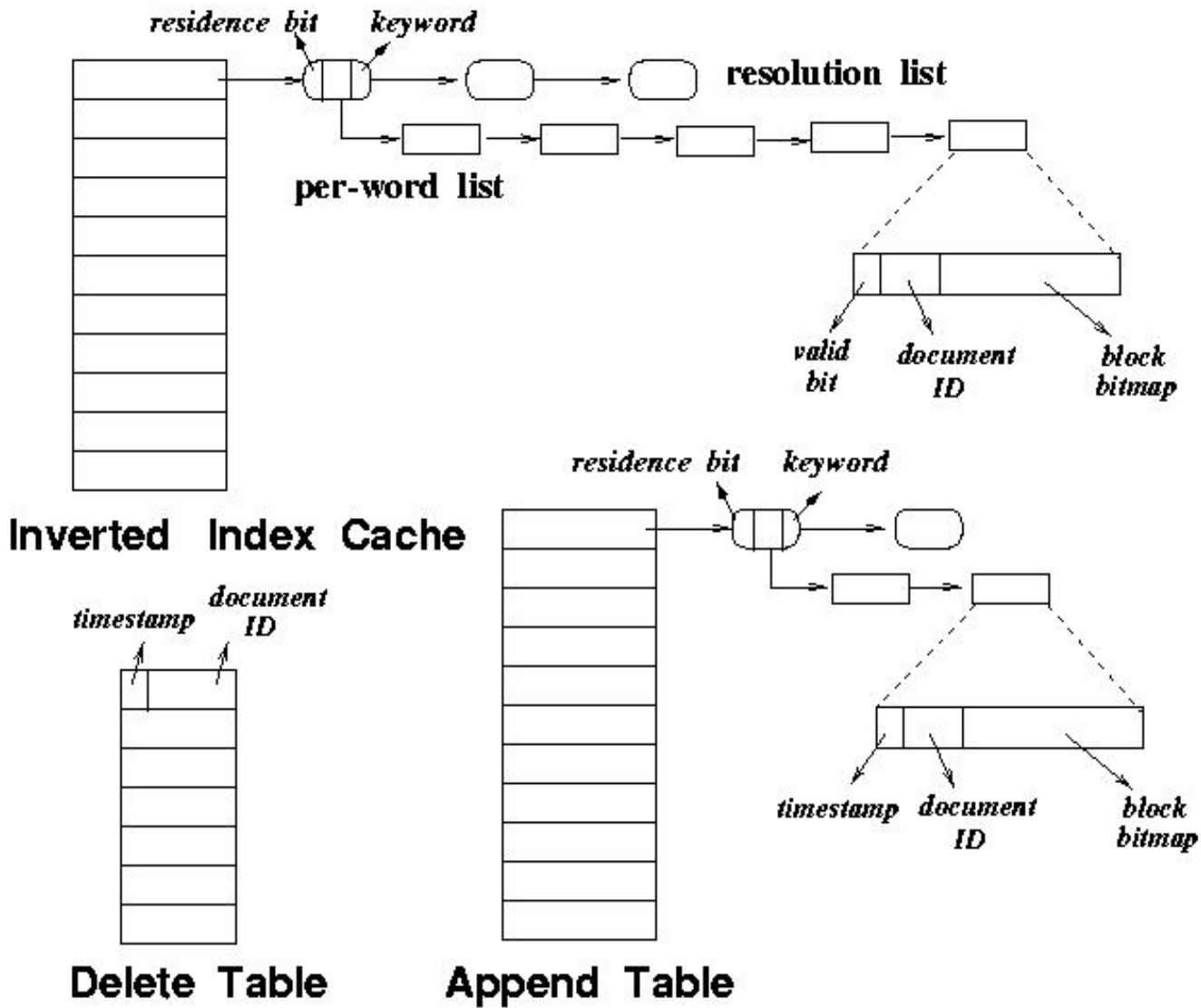
# Incremental Updates to Inverted Index

- **Solution used in existing search engines -- double sets of inverted indices**
- **Incremental updates research:**
  - **HYBRID Scheme: chains together chunks of the inverted lists and provides a number of parameters to control the size of the chunks and the length of the chains.**
  - **Disadvantages: extra storage and long freezing period**
- **Our System and Proposal: Codir**
  - **Updates and Queries are serviced simultaneously on the same set of inverted lists by using shared memory**

# Codir System



# Data Structure in Codir



## How does Codir work...

- **On a deletion request:** the docID is inserted into the *Deletion Table*;
- **On an insertion request:** the inverted index lists are inserted into *Append Table*;
- **On a query request:** query engine will return the results based on *Append, Deletion, Index Cache* tables;

# Algorithms in Search Engines

- **Ranking**
  - **content-based (vector model)**
  - **human annotations**
  - **connectivity-based**
    - Query independent: PageRank from Stanford
    - Query dependent: HITS by Jon Kleinberg
- **Duplicates Elimination**
  - **Growth Strategy from Stanford**

## PageRank Algorithm (Page and Brin, Google)

- **Assumptions:**
  - links often connects related pages
  - a link suggests recommendation
- Page A has pages T1... Tn which point to it. Damping factor d is set to 0.85. C(A) is defined as the number of links going out of page A. PageRank of a page A is defined as follows:

$$\mathbf{PR}(A) = (1-d) + d * (\mathbf{PR}(T1)/C(T1) + \dots + \mathbf{PR}(Tn)/C(Tn))$$

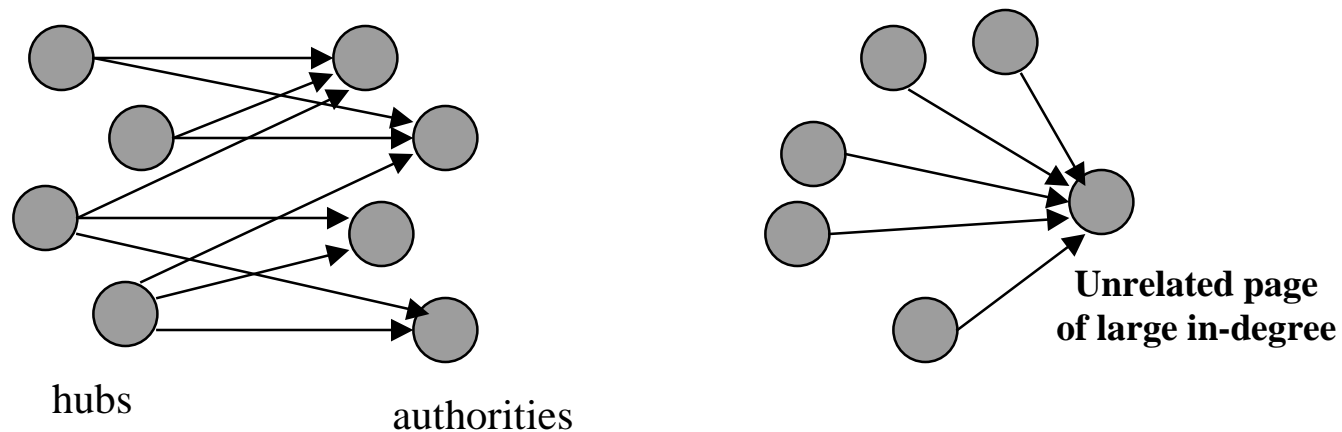
$$\mathbf{P} = (1-d) * \mathbf{U} + (d) * \mathbf{A}$$

where A is the adjacency matrix, U is unit matrix

- **Static ranking**

# HITS Algorithm (J. Kleinberger)

- **Authorities: good sources of content**
- **Hubs: good sources of links**
- **Mutually reinforcing relationship:** A good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs.



## HITS Algorithm (cont'd)

- We would like to focus on a collection  $S$  of pages which is **small, rich in relevant pages and contains many of the significant authorities.**

- The procedure to find such a set of pages:

$\sigma$ : a query string

$\varepsilon$ : a text-based search engine

$t, d$ : natural numbers

Let  $R \sigma$  denote the top  $t$  results of  $\varepsilon$  on  $\sigma$

Set  $S \sigma = R \sigma$

For each page  $p \in R \sigma$

    Let  $\Gamma(p)^+$  denote the set of all pages  $p$  points to.

    Let  $\Gamma(p)^-$  denote the set of all pages pointing to  $p$ .

    Add all pages in  $\Gamma(p)^+$  to  $S \sigma$

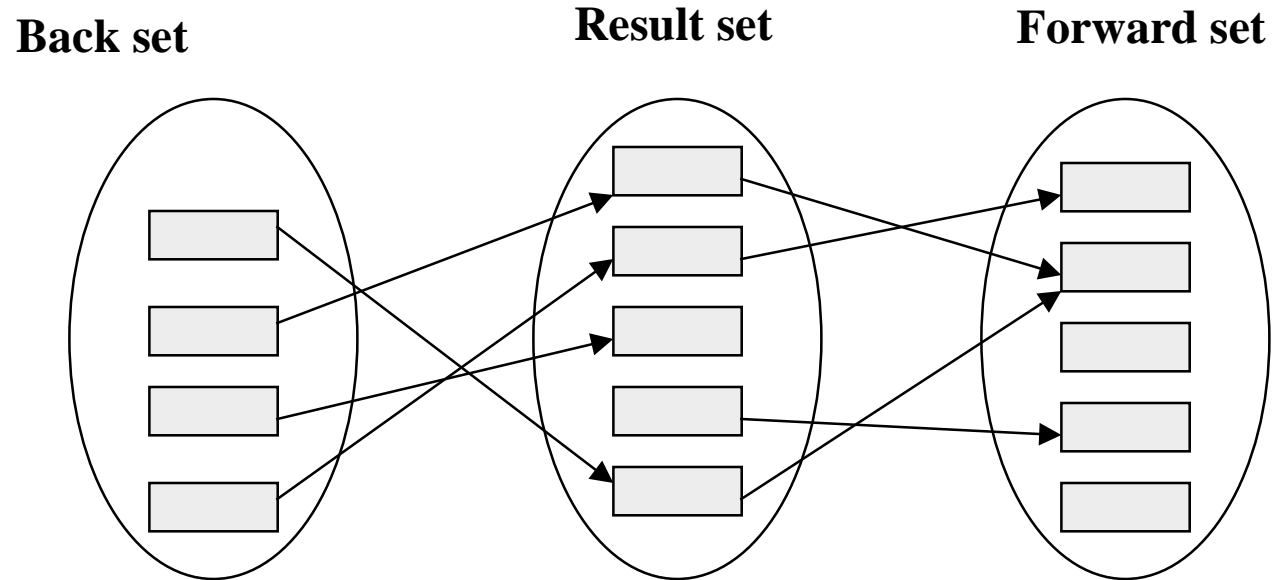
    if  $|\Gamma(p)^-| < d$  then Add pages in  $|\Gamma(p)^-|$  to  $S \sigma$

    Else Add an arbitrary set of  $d$  pages from  $|\Gamma(p)^-|$  to  $S \sigma$

End

Return  $S \sigma$

# HITS Algorithm (cont'd)

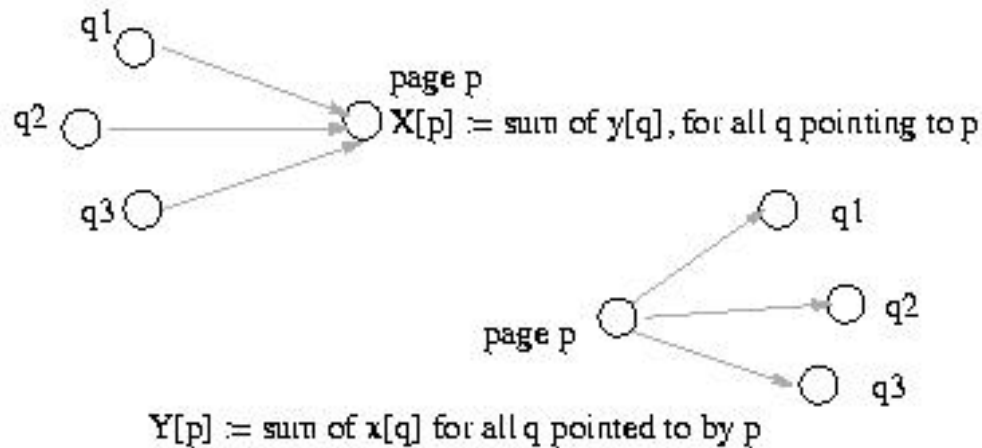


## HITS Algorithm (cont'd)

- **The Basic Operation:**

**authority weight**  $x^{<p>} \leftarrow \sum_{q:(q,p) \in E} y^{<q>} \text{ -- } I \text{ operation}$

**hub weight**  $y^{<p>} \leftarrow \sum_{q:(p,q) \in E} x^{<q>} \text{ -- } O \text{ operation}$



## HITS Algorithm (cont'd)

- **Iterate(G,k)**

**G**: a collection of  $n$  linked pages

**k**: a natural number

Let  $z$  denote the vector  $(1,1,1,\dots,1) \in \mathbb{R}^n$

Set  $x_0 := z, y_0 := z$

For  $i = 1, 2, \dots, k$

    Apply the  $I$  operation to  $(x_{i-1}, y_{i-1})$ , obtaining new  $x$ -weights  $x_i'$

    Apply the  $O$  operation to  $(x_{i-1}, y_{i-1})$ , obtaining new  $y$ -weights  $y_i'$

    Normalize  $x_i'$ , obtaining  $x_i$ .

    Normalize  $y_i'$ , obtaining  $y_i$ .

End

Return  $(x_k, y_k)$ .

## HITS Algorithm (cont'd)

- **Problems with HITS**
  - **Mutually Reinforcing Relationships: undue weight to one single author**
  - **Automatically Generated Links: Web authoring tools insert unrelated links into Web pages**
  - **Non-related Nodes: topic drift e.g. Mango Fruit will find result in fruit instead of Mango**

## Improved HITS Algorithm

- **Fractional weight to edges to avoid undue weight to single author (Weight)**
- **Eliminate non-relevant nodes from graph**
- **Regulating the influence of a node based on its relevance ( TopicScore)**

$$\text{HUB}[v] = \sum \text{AUTH}[u_i] \text{TopicScore}[u_i] \text{Weight}[v, u_i]$$

for all  $u_i$  with edge( $v, u_i$ )

$$\text{AUTH}[v] = \sum \text{HUB}[w_i] \text{TopicScore}[w_i] \text{Weight}[w_i, v]$$

for all  $w_i$  with edge( $w_i, v$ )

# Duplicate Elimination

- **Around 30% of the Web pages are duplicates**
  - unhappy users, extra RAM, storage cost...
- **Difficulties in detecting replicated collections:**
  - Update Frequency
  - Mirror Partial Coverage
  - Different Formats
  - Partial Crawls
- **Growth Strategy (omitted here)**

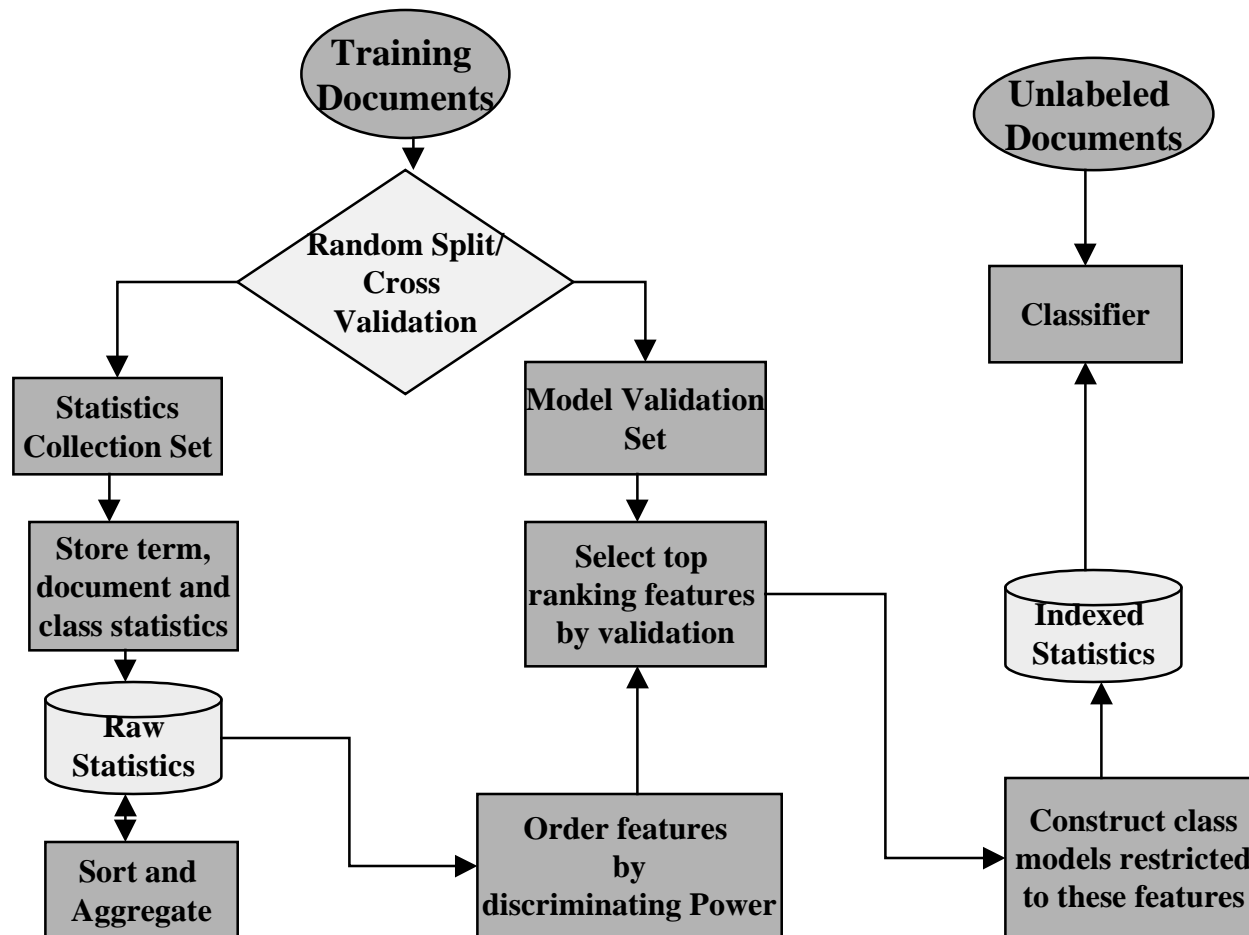
## Hierarchical Directories

- **Yahoo!:** 1million pages. Biggest and most famous directories around.
- **Infomine:** 16,000 pages. Compiled by academic librarians.
- **Britannica Web's Best:** 150,000 pages. Hand picked sites.
- **Librarian's Index:** 5000 pages. Highest quality sites only. Good annotations.
- **Galaxy:** 300,000 pages. Good annotations.
- **Problem: Cannot keep up with the fast growth of the Web.**

## **Automatic Categorization-- Taper**

- **Goal is to scale with the growth of the Web**
- **Feature is a function of context and document itself**
- **Utilizing the known information of the neighbors**
- **Best reported error rate is as high as 21%**

# Automatic Categorization-- Taper



## **Automatic Categorization -- ODP**

- **Volunteers manually categorizing the web pages**
- **Size:1,450,725 sites - 21,821 editors - 211,771 categories**
- **Sites using ODP data: Altavista, Lycos, Hotbot ...**
- **Ranking: cool pages and others**
- **A centralized system**
- **Cannot keep up with the growth of Web either??**

## Automatic Categorization -- OpenGrid

- **Extension to the HTML standard: tags enabling categorization and ranking-- cat, rank ...**  
*<a href="http://www.somenews.foo/" cat="News/Business" rank = "-30%">*
- **Rank/Categorization computation is done by a central search engine system. Various ranking algorithms can be applied to the computation.**
- **A distributed system: commentators, storage are world-wide**

# Measuring the Web

- **How big is the web?**
- **How fast does the Web grow?**
- **How do various search engines compare?**

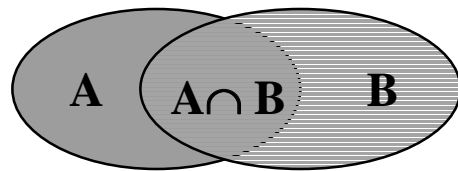
# Measuring the Web

- **Simple set comparison**

$$\Pr(A \cap B | A) = \text{size}(A \cap B) / \text{size}(A)$$

$$\Pr(A \cap B | B) = \text{size}(A \cap B) / \text{size}(B)$$

$$\text{size}(A) / \text{size}(B) = \Pr(A \cap B | B) / \Pr(A \cap B | A)$$



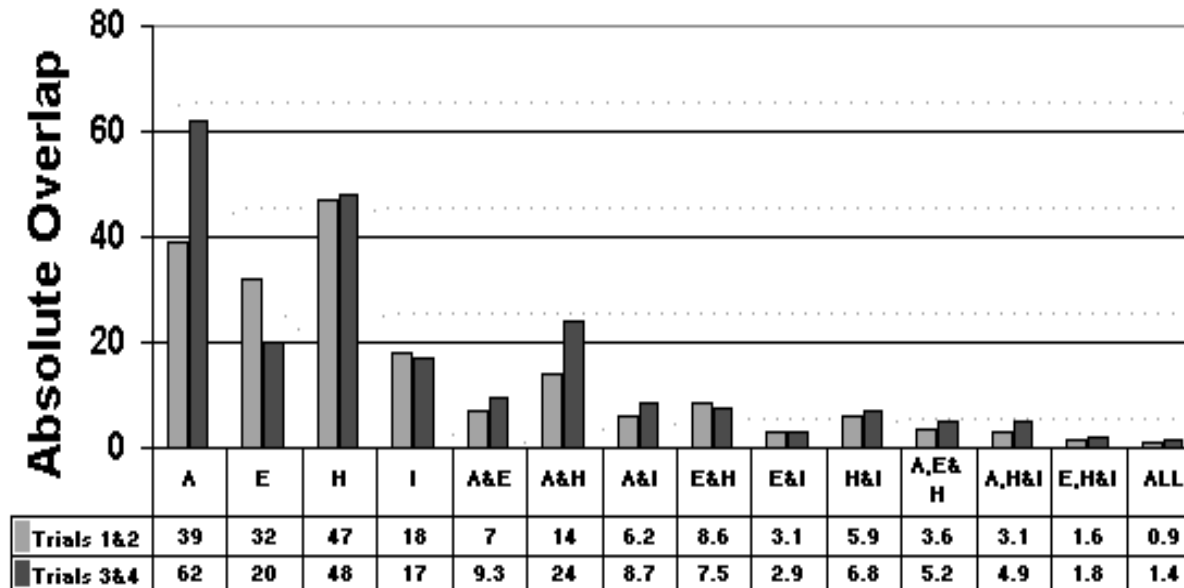
# Measuring the Web

- **Sampling**
  - randomly picking one from the all the indexes
  - firing a random query to a search engine, then select a random URL from the result set
- **Checking**
  - Full URL comparison
  - High similarity
  - Weak URL Comparison

# Measuring the Web: Query Log Statistics

- **request = new queries or new result screen of old query**
- **session = a series of requests by one user close together in time**
- **analyzed ~1B AltaVista requests consisting of:**
  - **840 000 000 nonempty requests**
  - **575 000 000 nonempty queries**
  - **153 000 000 unique nonempty queries**
  - **285 000 000 user sessions**
- **average number of terms: 2.35 (max 393)**
- **average number of operators: 0.41**
- **78% of sessions only 1 query asked**
- **64% of queries occur only once**

## Measuring the Web: (Nov. 1997)



**Figure: Normalized estimates for all intersections (expressed as a percentage of total joint coverage) where A-Alta Vista, I-Infoseek, E-Excite, H-HotBot**

# Measuring the Web: (Nov. 1997)

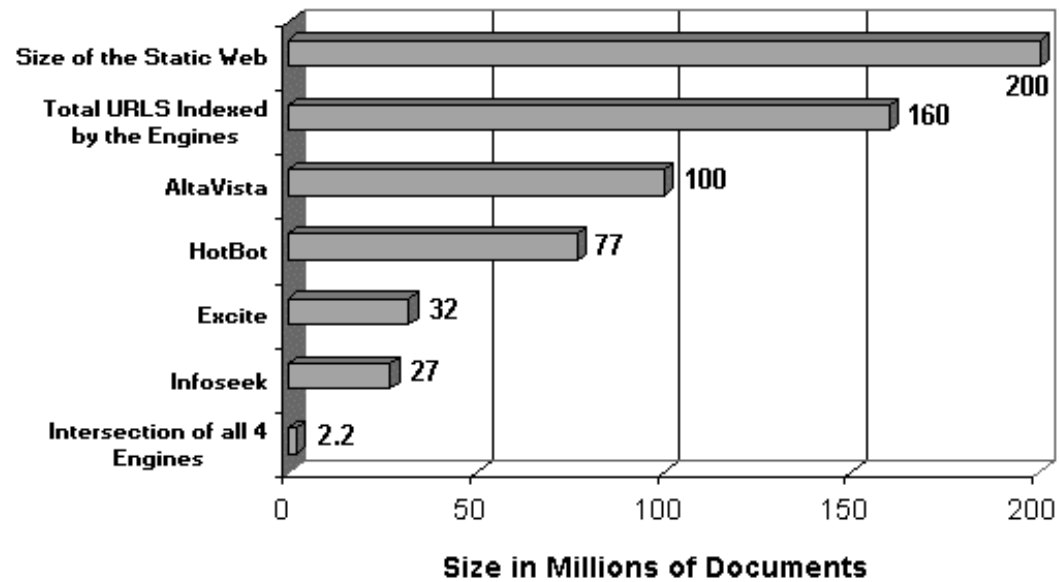


Figure: Absolute size estimates for Nov. 1997

## Conclusion

- **We look at two major Web IR tools: search engine and hierarchical directories**
- **We cover algorithms and implementation details:**
  - data structures, caching, incremental update
  - ranking(PageRank, HITS) , duplicate elimination
  - measuring the Web
- **We propose an incremental update scheme (codir) supporting immediate update visible to users.**
- **We did not touch:**
  - tools -- special robots, collaborative system,
  - algos -- query refinement, clustering...