

## Sources

---

- Andrei Broder, IBM
- Krishna Bharat, Google

## Information Retrieval and Text Mining

WS 2004/05, Jan 21, 2005  
Hinrich Schütze

## Topic Specific Pagerank [Have02]

---

- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
  - Selects a category (say, one of the 16 top level ODP categories) based on a query & user -specific distribution over the categories
  - Teleport to a page uniformly at random within the chosen category
- So we compute 16 pageranks instead of 1.

## Today's topics

---

- Topic-specific pagerank
- Behavior-based ranking
- Web crawling and corpus construction
- Search engine infrastructure

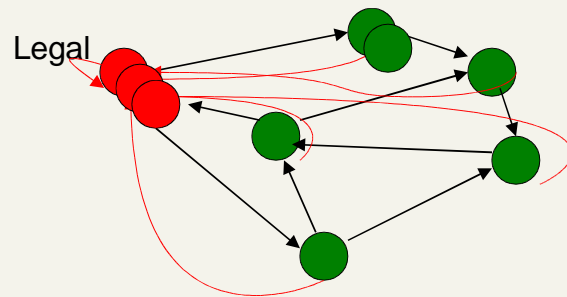
# Influencing PageRank (“Personalization”)

- Input:
  - Web graph  $W$
  - influence vector  $\mathbf{v}$   
 $\mathbf{v} : (\text{page} \rightarrow \text{degree of influence})$
- Output:
  - Rank vector  $\mathbf{r}$ : (page  $\rightarrow$  page importance wrt  $\mathbf{v}$ )
- $\mathbf{r} = \text{PR}(W, \mathbf{v})$

# Topic Specific PR: Implementation

- offline**: Compute pagerank distributions wrt to *individual* categories
  - Query independent model as before
  - Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category
- online**: Distribution of weights over categories computed by query context classification
  - Generate a dynamic pagerank score for each page – weighted sum of category-specific pageranks

# Non-uniform Teleportation

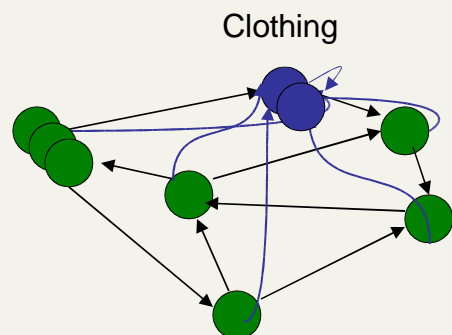


Teleport with 10% probability to a Legal page

# Distribute weight over categories

- Query: suit
- Context: legal firm
- Assign:
  - Legal  $\rightarrow$  0.9
  - Clothing  $\rightarrow$  0.1
  - All other categories  $\rightarrow$  0.0

## Interpretation



10% Clothing teleportation

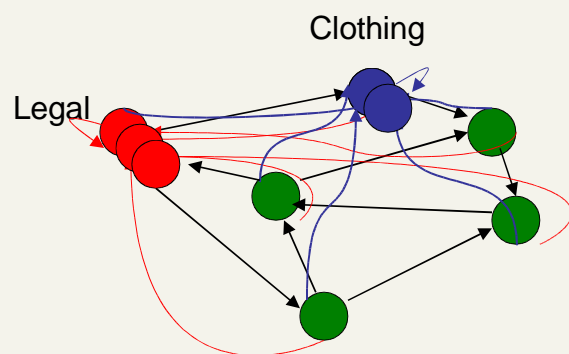
## Interpretation of Composite Score

- For a set of personalization vectors  $\{v_j\}$

$$\sum_j [w_j \cdot \text{PR}(W, v_j)] = \text{PR}(W, \sum_j [w_j \cdot v_j])$$

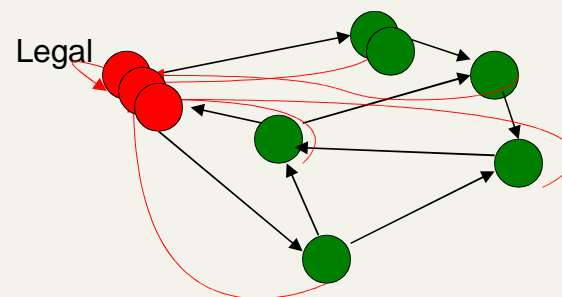
- Weighted sum of rank vectors itself forms a valid rank vector, because  $\text{PR}()$  is linear wrt  $v_j$

## Interpretation



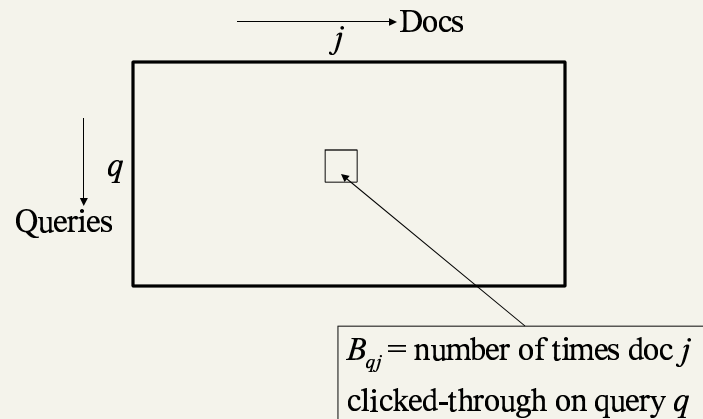
$pr = (0.9 \text{ PR}_{\text{legal}} + 0.1 \text{ PR}_{\text{clothing}})$  gives you:  
9% legal teleportation, 1% clothing teleportation

## Interpretation



10% Legal teleportation

## Query-doc popularity matrix $B$



When query  $q$  issued again, order docs by  $B_{qj}$  values.

## Web vs. hypertext search

- The WWW is full of free-spirited opinion, annotation, authority conferral
- Most other forms of hypertext are far more structured
  - enterprise intranets are regimented and templated
  - very little free-form community formation
  - no critical mass of links
  - web-derived link ranking doesn't quite work
- Other environments
  - Case law
  - Scientific literature

## Issues to consider

- Weighing/combining text- and click-based scores.
- What identifies a query?
  - Ferrari Mondial
  - Ferrari Mondial
  - Ferrari mondial
  - ferrari mondial
  - "Ferrari Mondial"
- Can use heuristics, but search parsing slowed.

## Behavior-Based Ranking

## Basic assumption

---

- Relevance can be directly measured by number of click throughs
- Valid?

## Vector space implementation

---

- Maintain a term-doc popularity matrix  $C$ 
  - as opposed to query-doc popularity
  - initialized to all zeros
- Each column represents a doc  $j$ 
  - If doc  $j$  clicked on for query  $q$ , update  $C_j \leftarrow C_j + \epsilon q$  (here  $q$  is viewed as a vector).
- On a query  $q'$ , compute its cosine proximity to  $C_j$  for all  $j$ .
- Combine this with the regular text score.

## Validity of Basic Assumption

---

- Click through to docs that turn out to be non-relevant: what does a click mean?
- Self-perpetuating ranking
- Spam
- All votes count the same

## Issues

---

- Normalization of  $C_j$  after updating
- Assumption of query compositionality
  - “white house” document popularity derived from “white” and “house”
- Updating – live or batch?

---

## Crawling, Corpus Construction

---

## Variants

- Time spent viewing page
  - Difficult session management
  - Inconclusive modeling so far
- Does user back out of page?
- Does user stop searching?
- Does user transact?

---

## Crawling and Corpus Construction

- Crawl order
- Filtering duplicates
- Mirror detection

---

## Raumaenderung

- Neuer Raum fuer die Uebung:
  - Phonetiklabor, Montags, 15:45 - 17:15
- Uebung faellt aus:
  - 14.02.

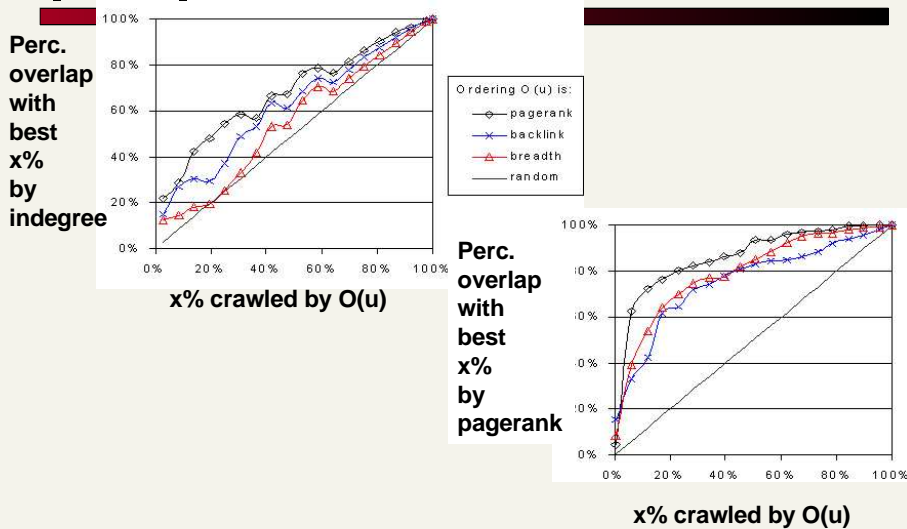
# Crawl Order

- Best pages first
  - Potential quality measures:
    - Final Indegree
    - Final Pagerank
  - Crawl heuristic:
    - BFS
    - Partial Indegree
    - Partial Pagerank
    - Random walk

# Crawling Issues

- How to crawl?
  - **Quality**: "Best" pages first
  - **Efficiency**: Avoid duplication (or near duplication)
  - **Etiquette**: Robots.txt, Server load concerns
- How much to crawl? How much to index?
  - **Coverage**: How big is the Web? How much do we cover?
  - **Relative Coverage**: How much do competitors have?
- How often to crawl?
  - **Freshness**: How much has changed?
  - How much has really changed? (why is this a different question?)

## Stanford Web Base (179K, 1998) [Cho98]



## /robots.txt: Example

The following example "/robots.txt" file specifies that no robots should visit any URL starting with "/cyberworld/map/" or "/tmp/", or /foo.html:

```
# robots.txt for http://www.example.com/
```

```
User-agent: *  
Disallow: /cyberworld/map/ # This is an infinite virtual URL space  
Disallow: /tmp/ # these will soon disappear  
Disallow: /foo.html
```

This example "/robots.txt" file specifies that no robots should visit any URL starting with "/cyberworld/map/", except the robot called "cybermapper":

```
# robots.txt for http://www.example.com/
```

```
User-agent: *  
Disallow: /cyberworld/map/ # This is an infinite virtual URL space  
  
# Cybermapper knows where to go.  
User-agent: cybermapper  
Disallow:
```

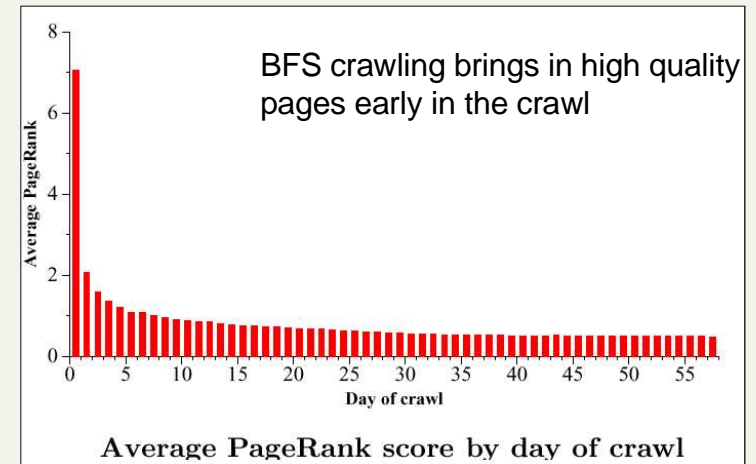
This example indicates that no robots should visit this site further:

```
# go away  
User-agent: *  
Disallow: /
```

# Adversarial IR (Spam)

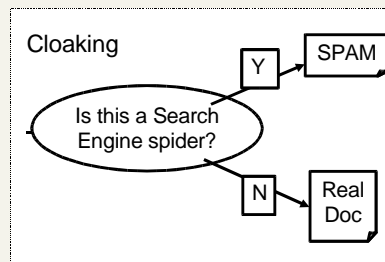
- Motives
  - Commercial, political, religious, lobbies
  - Promotion funded by advertising budget
- Operators
  - Contractors (Search Engine Optimizers) for lobbies, companies
  - Web masters
  - Hosting services
- Forum
  - Web master world ( [www.webmasterworld.com](http://www.webmasterworld.com) )
    - Search engine specific tricks
    - Discussions about academic papers ☺
- Not all spam is evil
  - Example: Terminix

# Web Wide Crawl (328M pages, 2000) [Najo01]



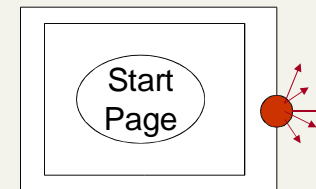
# A few spam technologies

- Cloaking
  - Serve fake content to search engine robot
  - **DNS cloaking:** Switch IP address. Impersonate
- Doorway pages
  - Pages optimized for a single keyword that re-direct to the real target page
- Keyword Spam
  - Misleading meta-keywords, excessive repetition of a term, fake "anchor text"
  - Hidden text with colors, CSS tricks, etc.
- Link spamming
  - Mutual admiration societies, hidden links, awards
  - **Domain flooding:** numerous domains that point or re-direct to a target page
  - Mixed editorial/commercial: goto-silicon-valley.com
- Robots
  - Fake click or query stream
  - Millions of submissions via Add-Url



Meta-Keywords =  
 "... London hotels, hotel, holiday inn, hilton,  
 discount, booking, reservation, sex, mp3,  
 britney spears, viagra, ..."

# BFS & Spam (Worst case scenario)

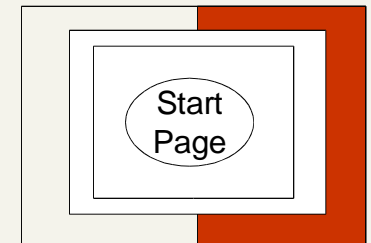


BFS depth = 2

Normal avg outdegree = 10

100 URLs on the queue including a spam page.

Assume the spammer is able to generate dynamic pages with 1000 outlinks



BFS depth = 3

2000 URLs on the queue  
 50% belong to the spammer

BFS depth = 4

1.01 million URLs on the queue  
 99% belong to the spammer

**FAQ: Cloaking & Stealth Technology**

### Tutorial: Cloaking and Stealth Technology

Featured as an ongoing multi part section newsletter, we are offering you all the stuff you to know, straight from the horse's mouth. Learn the secrets of the pros – subscription terminated anytime you wish.

**“Stealth, Cloaking, Phantom Tech”**

**fantomas spiderSpy™**  
The botBase

**Don't risk nasty surprises from spiders sneaking on your site under wraps!**

Sure, they tend to add and switch engines, IPs and User Agents almost all the time, and keeping up with their antics is a grueling task at best. But it's also a fact that professional traffic evaluation, stealthing technology and even page submission management depend on reliable search engine reference data, if you don't want to waste your valuable resources on inventing the wheel over and over again.

**Search Engine Optimization II**  
Tutorial on Cloaking & Stealth Technology

**FAQ**

- [What are Ghost Pages?](#)
- [What are Doorway Pages, then?](#)
- [And Hallway Pages?](#)
- [How are cloaked pages submitted?](#)
- [How about changing stealth pages?](#)
- [What are the mechanics of cloaking?](#)
- [What's a key switch?](#)
- [Isn't this really simple redirection technique?](#)
- [What about penalization?](#)

## Can you trust words on the page?

auctions.hitsoffice.com/

**Auctions**

**Pornographic Content**

www.ebay.com/

Examples from July 2002

## The war against spam

- Quality signals – Prefer authoritative pages based on:
  - Votes from authors (linkage signals)
  - Votes from users (usage signals)
- Policing of URL submissions
  - Anti robot test
- Limits on meta-keywords
- Robust link analysis
  - Ignore statistically implausible linkage (or text)
  - Use link analysis to detect spammers (guilt by association)

**DAMN3**

**internet.com**

Roll over to see the hidden advantages of WebSphere

Download Tools • Software Reviews • Book Reviews • Discussion

The latest tips.

### New Search Engine Marketing Practices

by *David Gikandi*

A study by Berrier Associates indicates that people who spend five or more hours a week online spend about 71% of their time searching for information. That goes to show the power search engines still wield over traffic. To keep you up to date on what online marketing professionals are now doing to win the search engine wars here is a brief look at some of the latest strategies being employed. August 2, 2000

**Search Engine Optimization I**  
Adversarial IR  
("search engine wars")

...got a COMPUTER QUESTION?

JOBS.webdeveloper.com

## Computing Near Similarity

---

- Features:
  - Segments of a document (natural or artificial breakpoints) [Brin95]
  - **Shingles** (Word N-Grams) [Brin95, Brod98]  
“a rose is a rose is a rose” =>  
a\_rose\_is\_a  
rose\_is\_a\_rose  
is\_a\_rose\_is
- Similarity Measure
  - TFIDF [Shiv95]
  - Jaccard [Brod98]  
(Specifically,  $\text{Size\_of\_Intersection} / \text{Size\_of\_Union}$ )

## Shingles + Set Intersection

---

- Computing exact set intersection of shingles between all pairs of documents is expensive and infeasible
- Approximate using a cleverly chosen subset of shingles from each (a **sketch**)

## The war against spam

---

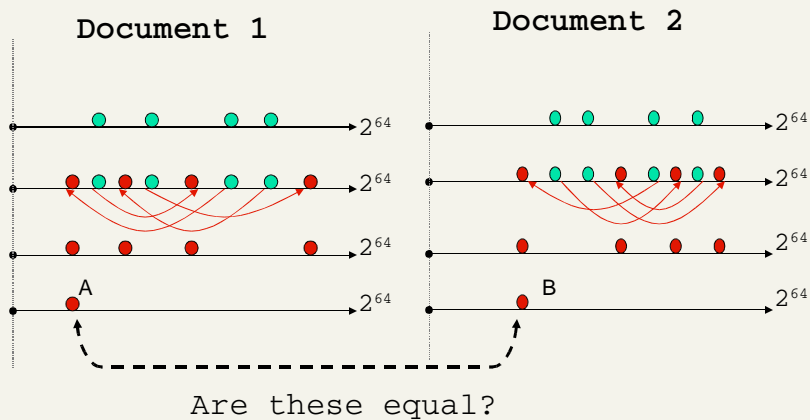
- Spam recognition by machine learning
  - Training set based on known spam
- Family friendly filters
  - Linguistic analysis, general classification techniques, etc.
  - For images: flesh tone detectors, source text analysis, etc.
- Editorial intervention
  - Blacklists
  - Top queries audited
  - Complaints addressed

## Duplicate/Near-Duplicate Detection

---

- **Duplication**: Exact match with fingerprints
- **Near-Duplication**: Approximate match
  - Overview
    - Compute syntactic similarity with an edit-distance measure
    - Use similarity threshold to detect near-duplicates
      - E.g., Similarity > 80% => Documents are “near duplicates”
      - Not transitive though sometimes used transitively

## Test if Doc1.Sketch[i] = Doc2.Sketch[i]

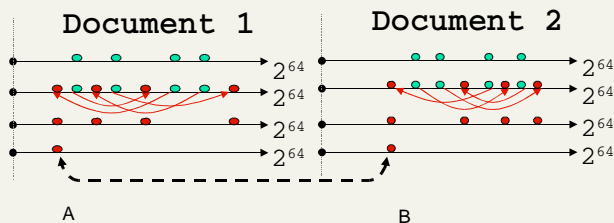


Test for 200 random permutations:  $\pi_1, \pi_2, \dots, \pi_{200}$

## Shingles + Set Intersection

- Estimate Jaccard:  $\text{size\_of\_intersection} / \text{size\_of\_union}$  based on a short sketch ([Brod97, Brod98])
  - Create a “sketch vector” (e.g., of size 200) for each document
  - Documents which share more than  $t$  (say 80%) corresponding vector elements are **similar**
  - For doc  $D$ , sketch[  $i$  ] is computed as follows:
    - Let  $f$  map all shingles in the universe to  $0..2^m$  (e.g.,  $f = \text{fingerprinting}$ )
    - Let  $\pi_i$  be a specific random permutation on  $0..2^m$
    - Pick sketch[  $i$  ] :=  $\text{MIN } \pi_i ( f(s) )$  over all shingles  $s$  in  $D$

## However...

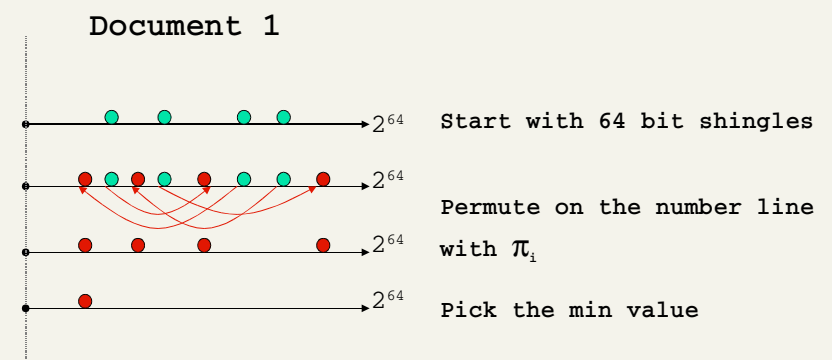


$A = B$  iff the shingle with the MIN value in the union of Doc1 and Doc2 is common to both (i.e., lies in the intersection)

This happens with probability:

$$\text{Size\_of\_intersection} / \text{Size\_of\_union}$$

## Computing Sketch[i] for Doc1



# Mirror Detection example

- <http://www.elsevier.com/> and <http://www.elsevier.nl/>
- Structural Classification of Proteins
  - <http://scop.mrc-lmb.cam.ac.uk/scop>
  - <http://scop.berkeley.edu/>
  - <http://scop.wehi.edu.au/scop>
  - <http://pdb.weizmann.ac.il/scop>
  - <http://scop.protres.ru/>

# Question

- Document  $D1=D2$  iff  $\text{size\_of\_intersection}=\text{size\_of\_union}$  ?

# Repackaged Mirrors

The image shows two side-by-side browser windows. The left window is Auctions.msn.com and the right is Auctions.lycos.com. Both display an 'Antiques' category page with a list of featured items. The items listed are nearly identical on both sites, including:

- ~Flow Blue Cake Plate With Pedestal~Gorgeous!!!
- ~Flow Blue Tauren With Soup Spoon~Gorgeous~ All Porcelain~\*
- Vintage Swiss Silver Case Pocket Watch by Remontoir
- One Nina & Three Rara Kuyu Paintings
- 0b2150502 / GORGEOUS HANDICRAFT TEAKWOOD ELEPHANT NCS152
- 0b2151103 / BEAUTIFUL HAND MADE TEAKWOOD ELEPHANT NCS152

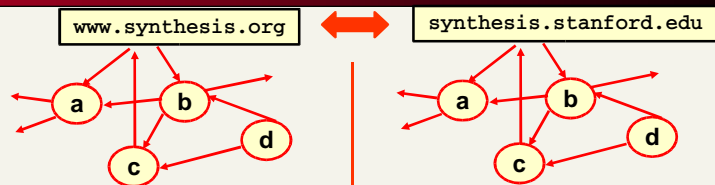
The listings show current bids and auction end dates, demonstrating that the content is mirrored across different hosts.

Aug  
2001

# Mirror Detection

- Mirroring is systematic replication of web pages across hosts.
  - Single largest cause of duplication on the web
- **Host1/α** and **Host2/β** are mirrors iff
  - For all (or most) paths  $p$  such that when  $\text{http://Host1/ } \alpha / p$  exists  $\text{http://Host2/ } \beta / p$  exists as well with identical (or near identical) content, and vice versa.

# Can we use URLs to find mirrors?



www.synthesis.org/Docs/ProjAbs/synsys/synalysis.html  
 www.synthesis.org/Docs/ProjAbs/synsys/visual-semi-quant  
 www.synthesis.org/Docs/annual.report96.final.html  
 www.synthesis.org/Docs/cicee-berlin-paper.html  
 www.synthesis.org/Docs/myr5  
 www.synthesis.org/Docs/myr5/cicee/bridge-gap.html  
 www.synthesis.org/Docs/myr5/cs/cs-meta.html  
 www.synthesis.org/Docs/myr5/mech/mech-intro-mechatron  
 www.synthesis.org/Docs/myr5/mech/mech-take-home.html  
 www.synthesis.org/Docs/myr5/synsys/experiential-learning  
 www.synthesis.org/Docs/myr5/synsys/mm-mech-dissec.htm  
 www.synthesis.org/Docs/yr5ar  
 www.synthesis.org/Docs/yr5ar/assess  
 www.synthesis.org/Docs/yr5ar/cicee  
 www.synthesis.org/Docs/yr5ar/cicee/bridge-gap.html  
 www.synthesis.org/Docs/yr5ar/cicee/comp-integ-analysis.h

synthesis.stanford.edu/Docs/ProjAbs/deliv/high-tech-...  
 synthesis.stanford.edu/Docs/ProjAbs/mech/mech-enhanced-...  
 synthesis.stanford.edu/Docs/ProjAbs/mech/mech-intro-...  
 synthesis.stanford.edu/Docs/ProjAbs/mech/mech-mm-case-...  
 synthesis.stanford.edu/Docs/ProjAbs/synsys/quant-dev-new-...  
 synthesis.stanford.edu/Docs/annual.report96.final.html  
 synthesis.stanford.edu/Docs/annual.report96.final\_fn.html  
 synthesis.stanford.edu/Docs/myr5/assessment  
 synthesis.stanford.edu/Docs/myr5/assessment/assessment-...  
 synthesis.stanford.edu/Docs/myr5/assessment/mm-forum-kiosk-...  
 synthesis.stanford.edu/Docs/myr5/assessment/neato-ucb.html  
 synthesis.stanford.edu/Docs/myr5/assessment/not-available.html  
 synthesis.stanford.edu/Docs/myr5/cicee  
 synthesis.stanford.edu/Docs/myr5/cicee/bridge-gap.html  
 synthesis.stanford.edu/Docs/myr5/cicee/cicee-main.html  
 synthesis.stanford.edu/Docs/myr5/cicee/comp-integ-analysis.html

# Motivation: Why detect mirrors?

- Smart crawling
  - Fetch from the fastest or freshest server
  - Avoid duplication
- Better connectivity analysis
  - Combine inlinks
  - Avoid double counting outlinks
- Redundancy in result listings
  - “If that fails you can try: <mirror>/samepath”
  - Repeat the search with the omitted results inc.
- Proxy caching

# Top Down Mirror Detection [Bhar99, Bhar00c]

- E.g.,
  - [www.synthesis.org/Docs/ProjAbs/synsys/synalysis.html](http://www.synthesis.org/Docs/ProjAbs/synsys/synalysis.html)
  - [synthesis.stanford.edu/Docs/ProjAbs/synsys/quant-dev-new-teach.html](http://synthesis.stanford.edu/Docs/ProjAbs/synsys/quant-dev-new-teach.html)
- What features could indicate mirroring?
  - Hostname similarity:
    - word unigrams and bigrams: { www, www.synthesis, synthesis, ... }
  - Directory similarity:
    - Positional path bigrams { 0:Docs/ProjAbs, 1:ProjAbs/synsys, ... }
  - IP address similarity:
    - 3 or 4 octet overlap
    - Many hosts sharing an IP address => virtual hosting by an ISP
  - Host outlink overlap
  - Path overlap
    - Potentially, path + sketch overlap

# Bottom Up Mirror Detection [Cho00]

- Maintain clusters of subgraphs
  - Initialize clusters of trivial subgraphs
    - Group near-duplicate single documents into a cluster
  - Subsequent passes
    - Merge clusters of the same cardinality and corresponding linkage
- 
- Avoid decreasing cluster cardinality
  - To detect mirrors we need:
    - Adequate path overlap
    - Contents of corresponding pages within a small time range

# Connectivity Server

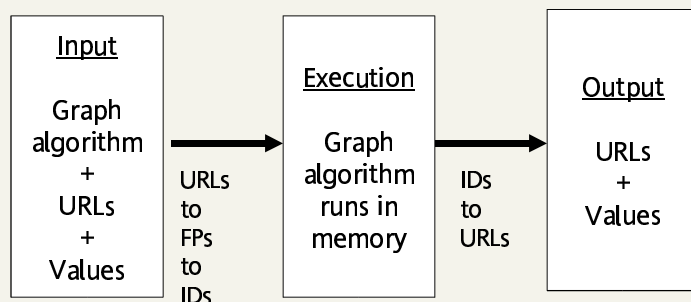
[CS1: Bhar98b, CS2 & 3: Rand01]

- Fast web graph access to support connectivity analysis
- Stores mappings in memory from
  - URL to outlinks, URL to inlinks
- Applications
  - HITS, Pagerank computations
  - Crawl simulation
  - Graph algorithms: web connectivity, diameter etc.
    - more on this later
  - Visualizations

# Implementation

- Phase I – Candidate Pair Detection
  - Find features that pairs of hosts have in common
  - Compute a list of host pairs which might be mirrors
- Phase II – Host Pair Validation
  - Test each host pair and determine extent of mirroring
  - Check if 20 paths sampled from Host1 have near-duplicates on Host2 and vice versa
  - Use transitive inferences:
    - IF Mirror(A,x) AND Mirror(x,B) THEN Mirror(A,B)
    - IF Mirror(A,x) AND !Mirror(x,B) THEN !Mirror(A,B)
- Evaluation
  - 140 million URLs on 230,000 hosts (1999)
  - Best approach combined 5 sets of features
    - Top 100,000 host pairs had precision = 0.57 and recall = 0.86

# Usage



Translation Tables on Disk

URL text: 9 bytes/URL (compressed from ~80 bytes )

FP(64b) -> ID(32b): 5 bytes

ID(32b) -> FP(64b): 8 bytes

ID(32b) -> URLs: 0.5 bytes

# WebIR Infrastructure

- Connectivity Server
  - Fast access to links to support for link analysis
- Term Vector Database
  - Fast access to document vectors to augment link analysis

# Adjacency List Compression – II

- Inter List Compression
  - Basis: Similar URLs may share links
    - Close in ID space => adjacency lists may overlap
  - Approach
    - Define a representative adjacency list for a block of IDs
      - Adjacency list of a reference ID
      - Union of adjacency lists in the block
    - Represent adjacency list in terms of deletions and additions **when it is cheaper to do so**
  - Measurements
    - Intra List + Starts: 8–11 bits per link (580M pages/16GB RAM)
    - Inter List: 5.4–5.7 bits per link (870M pages/16GB RAM.)

# ID assignment

E.g., HIGH IDs:

Max(indegree, outdegree) > 254

- Partition URLs into 3 sets, sorted lexicographically
  - **High:** Max degree > 254
  - **Medium:** 254 > Max degree > 24
  - **Low:** remaining (75%)
- IDs assigned in sequence (densely)

ID	URL
...	
9891	www.amazon.com/
9912	www.amazon.com/jobs/
...	
9821878	www.geocities.com/
...	
40930030	www.google.com/
...	
85903590	www.yahoo.com/

## Adjacency lists

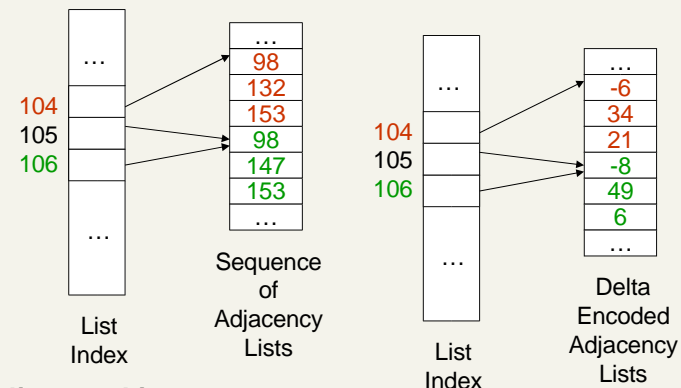
- In memory tables for Outlinks, Inlinks
- List index maps from a Source ID to start of adjacency list

# Term Vector Database

[Stat00]

- Fast access to 50 word term vectors for web pages
  - Term Selection:
    - Restricted to middle 1/3<sup>rd</sup> of lexicon by document frequency
    - Top 50 words in document by TF.IDF.
  - Term Weighting:
    - Deferred till run-time (can be based on term freq, doc freq, doc length)
- Applications
  - Content + Connectivity analysis (e.g., Topic Distillation)
  - Topic specific crawls
  - Document classification
- Performance
  - Storage: 33GB for 272M term vectors
  - Speed: 17 ms/vector on AlphaServer 4100 (latency to read a disk block)

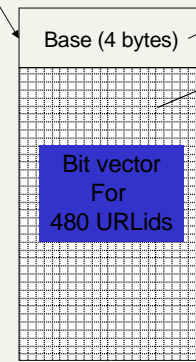
# Adjacency List Compression – I



- **Adjacency List:**
  - Smaller delta values are exponentially more frequent (80% to same host)
  - Compress deltas with variable length encoding (e.g., Huffman)
- **List Index pointers:** 32b for high, Base+16b for med, Base+8b for low
  - Avg = 12b per pointer

# Architecture

URLid \* 64 / 480



URLid to Term Vector  
Lookup

