**Information retrieval on the Web**

*Tools & algorithmic issues*

Andrei Broder & Monika Henzinger

_Compaq Systems Research Center_

November 1998

---

**What is this talk about?**

- **Topic:**
  - Algorithms for retrieving information on the Web

- **Non-Topics:**
  - Algorithmic issues in classic information retrieval (IR), e.g. stemming
  - String algorithms, e.g. approximate matching
  - Other algorithmic issues related to the Web:
    - networking & routing
    - caching
    - security
    - e-commerce
Preliminaries

- Each Web page has a unique URL
  - Page name: http://theory.stanford.edu/~focs98/tutorials.html
  - Access protocol
  - Host name = Domain name

- (Hyper) link = pointer from one page to another, loads second page if clicked on
- In this talk: document = (Web) page

Example of a query

- princess diana

Engine 1
- Relevant and high quality
- Re: Princess Diana Memorial Wedding

Engine 2
- Relevant but low quality
- Re: Lost in the shadow of Princess Diana

Engine 3
- Not relevant index pollution
- Re: Princess Diana: Queen of Hearts

Re: Princess Diana: Queen of Hearts
- Relevant but low quality
- Personal page: www.cnn.com
- Re: Princess Diana: Queen of Hearts
- Personal page: www.cnn.com
Open problems

Outline

- Classic IR vs. Web IR
- Some IR tools specific to the Web
  - For each type
    - Examples
    - Algorithmic issues
- Conclusions

Details on
- Ranking
- Duplicate elimination
- Measuring the Web

Classic IR

- **Input:** Document collection
- **Goal:** Retrieve documents or text with information content that is relevant to user’s information need
- **Two aspects:**
  1. Processing the collection
  2. Processing queries (searching)
- Reference Texts: SW’97, BF’92
Determining query results

“model” = strategy for determining which documents to return

- Logical model: String matches plus AND, OR, NOT
- Vector model:
  - Documents and query represented as vector of terms
  - Vector entry $i = \text{weight of term } i = \text{function of frequencies within document and within collection}$
  - Similarity of document & query = cosine of angle of their vectors
  - Query result: documents ordered by similarity
- Other models used in IR but not discussed here:
  - Probabilistic model
  - Cognitive model
  - ...

IR on the Web

- **Input**: The publicly accessible Web
- **Goal**: Retrieve high quality pages that are relevant to user's need
  - Static (files: text, audio, ...)  
  - Dynamically generated on request: mostly data base access
- **Two aspects**:
  1. Processing and representing the collection
     - Gathering the static pages
     - “Learning” about the dynamic pages
  2. Processing queries (searching)
What’s different about the Web?

Pages

- Bulk ....................... 350 M (7/98); grows at 20M/month
- Lack of stability......... Estimates: 1%/day -- 1%/week
- Heterogeneity
  - Type of documents .. Text, pictures, audio, scripts,…
  - Quality .................... From dreck to FOCS papers …
  - Language .................. 100+
- Duplication
  - Syntactic............... 30% (near) duplicates
  - Semantic.................. ??
- Non-running text........ many home pages, bookmarks, ...
- High linkage............... ≥ 8 links/page in the average

Typical home page: non-running text
**The big challenge**

Meet the user needs given the heterogeneity of Web pages

---

**What’s different about the Web?**

**Users**

- Make poor queries
  - Short (2.35 terms avg)
  - Imprecise terms
  - Sub-optimal syntax (80% queries without operator)
  - Low effort
- Wide variance in
  - Needs
  - Expectations
  - Knowledge
  - Bandwidth

- Specific behavior
  - 85% look over one result screen only
  - 78% of queries are not modified
  - Follow links
  - See various user studies in CHI, Hypertext, SIGIR, etc.
The bigger challenge

Meet the user needs given the heterogeneity of Web pages and the poorly made queries.

Why don’t the users get what they want?

Example

User need
I need to get rid of mice in the basement

User request (verbalized)
What’s the best way to trap mice alive?

Polysemy
Results
mouse trap

Synonymy
computer supplies, software, etc
AltaVista output: mouse trap

1. Mouse Trap Powered Vehicle Challenge
   Welcome to Mouse Trap Powered Vehicle Challenge page!
   Schools from around the world have participated in this fun project. Can YOU design an MTPV to go the
   URL: www.scar.edu/elk.sh.usfom/sMTPV_Files/index.html
   Last modified 1-May-98 - page size 9K - in English
   [ Translate ]

2. Pan, Paper & Mouse, Inc., Mouse Trap - Final
   Mouse Trap - Free from the company Pan, Paper & Mouse, Inc.
   URL: www.pan.com/MouseTrap.htm
   Last modified 31-Jan-98 - page size 9K - in English
   [ Translate ]

   Mouse Trap - The Serpent, The Girl and Her Dog from the company Pan, Paper & Mouse, Inc.
   URL: www.pan.com/Ad_marg.htm
   Last modified 31-Jan-98 - page size 9K - in English
   [ Translate ]

4. Pan, Paper & Mouse, Inc., Mouse Trap - New Year's Eve
   Mouse Trap New Year's Eve from the company Pan, Paper & Mouse, Inc.
   URL: www.pan.com/Ad_newyear.htm
   Last modified 31-Jan-98 - page size 9K - in English
   [ Translate ]

5. Mouse-Trap Car
   Mouse-trap car instructions
   URL: www.mtcar.com/mtcar.htm
   Last modified 17-Aug-98 - page size 9K - in English
   [ Translate ]

6. Pan, Paper & Mouse, Inc., Mouse Trap - Chinese
   Mouse Trap Chinese from the company Pan, Paper & Mouse, Inc.
   URL: www.pan.com/Ad_chinese.htm
   Last modified 31-Jan-98 - page size 9K - in English
   [ Translate ]

AltaVista output: mice trap

1. Rat Trap kills rats, mice with harmless by electric shock
   Trap rat trap for Pest Control Operators, homeowners, facility managers. Kill rats and mice with harmless with
   environmentally friendly Rat Zapper traps
   URL: www.agr.com/agents Mt.htm
   Last modified 26-Jun-98 - page size 9K - in English
   [ Translate ]

2. Retrapping Mouse Trap Ingenious professional tool catches up to 20 mice at a sl
   Retrapping Mouse Trap Ingenious professional tool catches up to 20 mice at a time around the home.
   Last modified 26-Jun-98 - page size 9K - in English
   [ Translate ]

3. Mouse Trap kills mice, rodents with high technology trap
   Mouse Traps kills mice, rodents with high technology trap
   URL: www.agr.com/agents Mtr.htm
   Last modified 25-Aug-98 - page size 9K - in English
   [ Translate ]

4. PETA Spring 1997 Catalog - Caring Consumer Products
   Summer 1997 Caring Consumer Products: Humane 'Smart' Mousetrap. When you
   when you place the trap, turn the handle into a T.C.H.I. Plastic trap catches mice alive.
   [ Translate ]

5. House Mouse - Trapping Instruction and Tips.
   How to Get Rid of House Mice: House Mice
   URL: www.supplyhypernet.net/ tips.htm
   Last modified 6-Aug-98 - page size 9K - in English
   [ Translate ]
The bright side: Web advantages vs. classic IR

User
- Many tools available
- Personalization
- Timeliness
- Interactivity (refine the query if needed)

Collection/tools
- Redundancy
- Hyperlinks
- Statistics
  - Easy to gather
  - Large sample sizes
- Interactivity (make the users explain what they want)

Quantifying the quality of results
- How to evaluate different strategies?
- How to compare different search engines?
**Classic evaluation of IR systems**

We start from a human made relevance judgement for each (query, page) pair and compute:

- **Precision**: % of returned pages that are relevant.
- **Recall**: % of relevant pages that are returned.
- **Precision at 10**: % of top 10 pages that are relevant ("ranking quality")
- ...

**Evaluation in the Web context**

- Quality of pages varies widely
- Relevance is not enough
- We need both relevance and high quality = value of page.
**Web IR tools**

- General-purpose search engines:
  - direct: AltaVista, Excite, Infoseek, Lycos, HotBot, ….
  - Indirect (Meta-search): MetaCrawler, DogPile, AskJeeves, …
- Hierarchical directories: Yahoo!, all portals.
- Specialized search engines:
  - Home page finder: Ahoy
  - Shopping robots: Jango, Junglee,…
  - Applet finders

**Web IR tools (cont...)**

- Search-by-example: Netscape’s “What’s related”, Excite’s “More like this”, …
- Collaborative filtering: Firefly, GAB, …
- Notification systems: Clipping services

- Meta-information:
  - Web statistics:
    - size, number of domains, geography, ….
  - Usage statistics
  - Query log statistics
  - …
General purpose search engines

- Search engines’ components:
  - Spider = Crawler -- collects the documents
  - Indexer -- process and represents the data
  - Search interface -- answers queries
- Example: AltaVista as of 7/98
  - Collects ~ 170,000,000 pages.
  - Indexes ~ 125,000,000 pages = 700 GB of text.
  - Roughly 35% of the entire indexable Web.

Algorithmic issues related to search engines

- Collecting documents
  - Priority
  - Load balancing
    - Internal
    - External
  - Trap avoidance
  - ...
- Processing and representing the data
  - Query-independent ranking
  - Graph representation
  - Index building
  - Duplicate elimination
  - Categorization
  - ...
- Processing queries
  - Query-dependent ranking
  - Duplicate elimination
  - Query refinement
  - Clustering
  - ...


**Ranking**

- **Goal:** order the answers to a query in decreasing order of value
  - Query-independent: assign an intrinsic value to a document, regardless of the actual query
  - Query-dependent: value is determined only wrt a particular query.
  - Mixed: combination of both valuations.

- **Examples**
  - Query-independent: length, vocabulary, publication data, number of citations (indegree), etc.
  - Query-dependent: cosine measure

---

**Some ranking criteria**

- **Content-based** techniques (variant of term vector model or probabilistic model)
- **Ad-hoc factors** (anti-porn heuristics, publication/location data, ...)
- **Human annotations**
- **Connectivity-based** techniques
  - Query-independent (PageRank [PBMW’98, BP’98], indegree [CK’97], ...)
  - Query-dependent (HITS [K’98], ...)

---
Connectivity analysis

- Idea: Mine hyperlink information of the Web
- Assumptions:
  - Links often connect related pages
  - A link between pages is a recommendation
- Classic IR work (citations = links) a.k.a. “Bibliometrics” [K’63, G’72, S’73, …]
- Many Web related papers build on this idea [PPR’96, AMM’97, S’97, CK’97, K’98, BP’98,…]

Graph representation for the Web

- Node for each page u
- Directed edge (u,v) if page u contains a hyperlink to page v.
Query-independent connectivity based ranking: PageRank [BP’98]

- Consider a random Web surfer:
  - Jumps to random page with probability $\varepsilon$
  - With probability $1-\varepsilon$, follows a random hyperlink of the current page
- Transition probability matrix is
  $$
  \varepsilon \times U + (1-\varepsilon) \times A
  $$
  
  where $U$ is the uniform distribution and $A$ is adjacency matrix (normalized)
- Query-independent rank = stationary probability for this Markov chain.

Output from Google!

 princess diana
Output from Google!: jobs

Query-dependent ranking: the neighborhood graph

- Subgraph associated to each query

An edge for each hyperlink, but no edges within the same host
HITS [K’98]

- **Goal**: Given a query find:
  - Good sources of content (authorities)
  - Good sources of links (hubs)

**Intuition**

- Authority comes from in-edges.
  Being a good hub comes from out-edges.

- Better authority comes from in-edges from good hubs.
  Being a better hub comes from out-edges to good authorities.
Repeat until $\text{HUB}$ and $\text{AUTH}$ converge:

Normalize $\text{HUB}$ and $\text{AUTH}$

$\text{HUB}[v] := \sum \text{AUTH}[u_i]$ for all $u_i$ with $\text{Edge}(v, u_i)$

$\text{AUTH}[v] := \sum \text{HUB}[w_i]$ for all $w_i$ with $\text{Edge}(w_i, v)$

Output from HITS: jobs

(start set from AltaVista)

- www.ajb.dni.us
- www.britnet.co.uk/jobs.htm
- www.monster.com
- www.careermosaic.com
- plasma-gate.weizmann.ac.il/Job...
- www.jobtrak.com
- www.occ.com
- www.jobserve.com
- www.allny.com/jobs.html
- www.commarts.com/bin/ca/be_jl
- www.jobcenter.com
- www.topjobs.co.uk
Problems & solutions

- Some edges are “wrong” -- not a recommendation:
  - multiple edges from same author
  - automatically generated
  - spam, etc.

  Solution: Weight edges to limit influence

- Topic drift
  - Query: jaguar AND cars
  - Result: pages about cars in general

  Solution: Analyze content and assign topic scores to nodes

Modified HITS algorithms

Repeat until $\text{HUB}$ and $\text{AUTH}$ converge:

- Normalize $\text{HUB}$ and $\text{AUTH}$

\[
\text{HUB}[v] := \sum \text{AUTH}[u] \cdot \text{TopicScore}[u,v] \cdot \text{weight}[v,u]
\]

for all $u$ with $\text{Edge}(v, u)$

\[
\text{AUTH}[v] := \sum \text{HUB}[w] \cdot \text{TopicScore}[w,v] \cdot \text{weight}[w,v]
\]

for all $w$ with $\text{Edge}(w, v)$
User study [BH’98]

Valuable pages within 10 top answers (averaged over 28 topics)

Open problems

- Compare performance of query-dependent and query-independent connectivity analysis
- Exploit order of links on the page (see e.g. [CDGKRRT’98])
- Both Google! and HITS compute principal eigenvector. What about non-principal eigenvector? ([K’98])
- Derive other graphs from the hyperlink structure …
More on graph representation

- Graphs derived from the hyperlink structure of the Web:
  - Node = page
  - Edge (u,v) iff pages u and v are related in a specific way (directed or not)
- Examples of edges:
  - iff u has hyperlink to v
  - iff there exists a page w pointing to both u and v
  - iff u is often retrieved within x seconds after v
  - …

Graph representation usage

- Ranking algorithms
  - PageRank
  - HITS
  - …
- Market research
  - Lycos “LinkAlert”
- Categorization of Web pages
  - [CDI’98]
- Visualization/Navigation
  - Mapuccino [MJSUZB’97]
  - Microsoft’s Site Mapping tool
  - …
- Structured Web query tools
  - WebSQL [AMM’97]
  - WebQuery [CK’97]
Directed edges = Hyperlinks

- **Goal:** Support two basic operations for all URLs collected by AltaVista
  - `InEdges(URL u, int k)`
    - Return k URLs pointing to u
  - `OutEdges(URL u, int k)`
    - Return k URLs that u points to

- **Difficulties:**
  - Memory usage (~180 M nodes, 1B edges)
  - Preprocessing time (days …)
  - Query time (~ 0.0001s/result URL)

---

**URL database**

Sorted list of URLs is 8.7 GB (= 48 bytes/URL)

- Delta encoding reduces it to 3.8 GB (= 21 bytes/URL)
Graph data structure

Node Table

URL Database

Inlist Table

Outlist Table

Graph compression: How much compression possible without significant run-time penalty?
- Efficient algorithms to find frequently repeated small structures (e.g. wheels, \(K_{2,2}\))
- Asymptotic complexity?

External memory graph algorithms: How to assign the graph representation to pages so as to reduce paging? (see [NGV’96, AAMVV’98])

Stringology: Less space for URL database? Faster algorithms for URL to node translation?

Dynamic data structures: How to make updates efficient at the same space cost?
Algorithmic issues related to search engines

- Collecting documents
  - Priority
  - Load balancing
    - Internal
    - External
  - Trap avoidance
  - ...

- Processing and representing the data
  - Query-independent ranking
  - Graph representation
    - Index building
    - Duplicate elimination
    - Categorization
    - ...

- Processing queries
  - Query-dependent ranking
  - Duplicate elimination
  - Query refinement
  - Clustering
  - ...

Performance

- Indexes about 0.8TB of text
- Every word is indexed! (No stop words.)
- About 37 million queries on weekdays
- Mean response time = 0.6 sec (elapsed Palo Alto time)
- Median response time, as observed in NJ = 0.9 sec [LG’98b]
**Why is AltaVista so fast?**

- Conceptually simple algorithms
- Good algorithms engineering
- Good software engineering
- Lots of fast hardware, lots of RAM

---

**AltaVista search index**

- View all documents as concatenated into one document (Special token to separate original documents)
- Basic data structure: flat inverted file [See e.g. BF’92]
  - Store for each word, a delta-encoded list of all locations where word occurs
  - Order words lexicographically and compress common word prefixes
- \[\Rightarrow\] Size of full-text index \(\approx 30\% \) of input size
Extended Boolean model:
- **matches**: exact, prefix, phrase
- **operators**: AND, OR, AND NOT, NEAR

To support this: **only one abstract data type**

**Index Stream Reader (ISR)** = the ordered list of locations that match a given query.

For a given ISR:
- `loc()`: return current location
- `next()`: advance to next location
- `seek(int k)`: advance to first location $\geq k$

---

**Translating a query into an ISR**

Each query is decomposed into these elementary operations:

- **CREATE**: `WORD \rightarrow ISR` ............... Create a list
- **OR**: `ISR \times ISR \rightarrow ISR` ............... Merge lists
- **AND NOT**: `ISR \times ISR \rightarrow ISR` ....... Intersect lists
- **AND**: `ISR \times ISR \rightarrow ISR` ...................."  
- **NEAR**: `ISR \times ISR \rightarrow ISR` .................... "
- **PHRASE**: `ISR \times ISR \times \ldots \times ISR \rightarrow ISR` .... "

The final resulting ISR is translated back into list of documents and ranked
The search happens on

- ~ 20 64-bit machines
- Typical configuration
  - 10 CPU each (625 MHz)
  - 12 GB RAM
  - 300 GB RAID

Reasons for duplicate filtering

- Proliferation of almost but not quite equal documents on the Web:
  - Legitimate: Mirrors, local copies, updates, etc.
  - Malicious: Spammers, spider traps, dynamic URLs
  - Mistaken: Spider errors
- Costs:
  - RAM and disks
  - Unhappy users
- Approximately 30% of the pages on the Web are (near) duplicates. [BGMZ'97,SG'98]
Basic mechanism needed

- Must filter both duplicate and near-duplicate documents
- Computing pair-wise edit distance would take forever
- Preferably to store only a short sketch for each document.

The basics of a solution

[B’97],[BGMZ’97],[B’98]

1. Reduce the problem to a set intersection problem
2. Estimate intersections by sampling minima.
Shingling

- Shingle = Fixed size sequence of w contiguous words
  
  a rose is a rose is a rose
  a rose is a
  rose is a rose
  is a rose is
  a rose is a
  rose is a rose

Set of 64 bit fingerprints

Defining resemblance

\[
\text{resemblance} = \frac{|S_1 \Delta S_2|}{|S_1 \cup S_2|}
\]
**Sampling minima**

- Apply a random permutation $\sigma$ to the set $[0..2^{64}]$.
- Crucial fact
  
  Let $\alpha = \min(\sigma(S_1))$  \quad $\beta = \min(\sigma(S_2))$

  ![Diagram](Diagram.png)

  $\Pr(\alpha = \beta) = \frac{|S_1 \oplus S_2|}{|S_1 \oplus S_2|}$

**Implementation**

- Choose a set of $t$ random permutations of $U$.
- For each document keep a sketch $S(D)$ consisting of $t$ minima = samples.
- Estimate resemblance of $A$ and $B$ by counting common samples.
- The permutations should be from a min-wise independent family of permutations. See [BCFM’97] for the theory of mwi permutations.
If we need only high resemblance

Sketch 1: [sketch representation]
Sketch 2: [sketch representation]

- Divide sketch into $k$ groups of $s$ samples ($t = k \times s$)
- Fingerprint each group → feature
- Two documents are fungible if they have more than $r$ common features.
- Want
  Fungibility ⇔ Resemblance above fixed threshold $\rho$

Real implementation

- $\rho = 90\%$. In a 1000 word page with shingle length = 8 this corresponds to
  - Delete a paragraph of about 50-60 words.
  - Change 5-6 random words.
- Sketch size $t = 84$, divide into $k = 6$ groups of $s = 14$ samples
- 8 bytes fingerprints → we store only $6 \times 8 = 48$ bytes/document
- Threshold $r = 2$
**Probability that two documents are deemed fungible**

Two documents with resemblance $\rho$

- Using the full sketch
  \[ P = \sum_{i=t}^{k \cdot s} \binom{k \cdot s}{i} \rho^i (1 - \rho)^{k \cdot s - i} \]

- Using features
  \[ P = \sum_{i=r}^{k} \binom{k}{i} \rho^s \cdot i (1 - \rho^s)^{k - i} \]

---

**Features vs. full sketch**

Probability that two pages are deemed fungible

![Graph showing probability vs. resemblance with two lines: one for using full sketch and one for using features.]

- Using full sketch
- Using features
Related work

- Similarity detection [M’94, BDG’95, SG’95, H’96, BGMZ’97]
- Mathematics of resemblance [B’97, B’98]
- Min-wise independent permutations [BCFM’97, BCM’98]
- Duplication on the Web [SG’98]
- Efficient ways of finding data base elements that share many attributes. [FSGMU’98]

Open problems

- Best way of grouping samples for a given threshold and/or for multiple thresholds?
- Efficient ways to find in a data base pairs of records that share many attributes. Best approach?
- Min-wise independent permutations -- lots of open questions.
- Connections with the Hamming distance nearest neighbor problem. (See [IM’98]).
- Other applications possible (images, sounds, ...) -- need translation into set intersection problem.
Algorithmic issues related to search engines

- Collecting documents
  - Priority
  - Load balancing
    - Internal
    - External
  - Trap avoidance
    - ...

- Processing and representing the data
  - Query-independent ranking
  - Graph representation
  - Index building
  - Duplicate elimination
    - Categorization
    - ...

- Processing queries
  - Query-dependent ranking
  - Duplicate elimination
    - Query refinement
    - Clustering
    - ...

Adding pages to the index

Crawling process

1. Get link at top of queue
2. Fetch page
3. Index page and parse links
4. Add to queue
5. Add URL
6. Queue of links to explore
7. Expired pages from index
**Queuing discipline**

- Standard graph exploration:
  - Random
  - BFS
  - DFS (+ depth limits)
- **Goal**: get “best” pages for a given index size
  - Priority based on query-independent ranking
  - Visit the page that has the highest potential PageRank [CGP’98]
- **Goal**: keep index fresh
  - Priority based on rate of change [CLW’97]

**Load balancing**

- **Internal** -- can not handle too much retrieved data simultaneously, but
  - Response time is unpredictable
  - Size of answers is unpredictable
  - There are additional system constraints (# threads, # open connections, etc.)
- **External**
  - Should not overload any server or connection
  - A well-connected crawler can saturate the entire outside bandwidth of some small countries
  - Any queuing discipline must be acceptable to the community
Web IR Tools

- General-purpose search engines
  - Hierarchical directories
    - Automatic categorization
  - Specialized search engines
    - (dealing with heterogeneous data sources)
      - Shopping robots
      - Home page finder [SLE’97]
      - Applet finders
      - …
  - Search-by-example
  - Collaborative filtering
  - Notification systems
  - Meta-information

Dealing with heterogeneous sources

- Modern life problem:
  - Given information sources with various capabilities, query all of them and combine the output.
  - Hot issue in current data base research
  - Red hot issue in the industry.
- Examples
  - Inter-business e-commerce e.g. **www.industry.net**
  - Meta search engines
  - Shopping robots
- Issues
  - Determining relevant sources -- the “identification” problem
  - Merging the results -- the “fusion” problem
Standards and tools

- Some standards:
  - STARTS → Stanford Protocol Proposal for [distributed] Internet Retrieval and Search
  - EDI → Electronic Data Interchange
  - XML → Extensible Markup Language
    - http://www.w3.org/TR/1998/REC-xml-19980210

- Some tools:
  - WebL → scripting language for automating tasks on the Web
  - WebSQL → SQL like query language [M’96, AMM’97]

Example: a shopping robot

- Input: A product description in some form
- Find: Merchants for that product on the Web
- Jango [DEW’97]
  - preprocessing: Store vendor URLs in database; learn for each vendor:
    - the URL of the search form
    - how to fill in the search form and
    - how the answer is returned
  - request processing: fill out form at every vendor and test whether the result is a success
  - range of products is predetermined
**Jango input example**

![Excite Product Finder](image1)

**Jango output**

![Excite Product Finder](image2)
Direct query to K&L Wine

What price decadence?

1983 Pichon Lalande

1961 Palmer, Margaux

1996 Chambertin, Jean Raphet

1993 Haut-Marbuzet, St-Estephe
Open problems

- Going beyond the lowest common capability
- Learning problem: automatic understanding of new interfaces
- Efficient ways to determine which DB is relevant to a particular query
- “Partial knowledge indexing”: indexer has limited access to the full DB

Web IR Tools

- General-purpose search engines
- Hierarchical directories
- Specialized search engines (dealing with heterogeneous data sources)
  - Search-by-example
  - Collaborative filtering
  - Notification systems
  - Meta-information
Search-by-example

- **Given:** set $S$ of URLs
- **Find:** URLs of similar pages
- **Algorithms:**
  - **Connectivity-based:** build graph consisting of neighborhood of $S$ and run HITS [K'98]
  - **Usage based:** related pages are pages visited frequently after $S$ [Netscape “What’s related”]
  - **Query refinement** [Excite’s “More like this”]

Netscape example
Web IR Tools

- General-purpose search engines:
- Hierarchical directories
- Specialized search engines:
  - Search-by-example
  - Collaborative filtering
  - Notification systems
  - Meta-information

Collaborative filtering

User input

Collected input

prediction phase

analysis phase

Suggestions

Model

Explicit preferences
Lots of projects

- Collaborative filtering seldom used in classic IR, big revival on the Web. Projects:
  - PHOAKS -- ATT labs → Web pages recommendation based on Usenet postings
  - Firefly -- MIT → share profiles without infringing privacy
  - GAB -- Bellcore → Web browsing
  - GroupLens -- U. Minnesota → Usenet newsgroups
  - EachToEach -- Compaq SRC → rating movies
  - ...

See http://sims.berkeley.edu/resources/collab/

Why do we care?

- The ranking schemes that we discussed are also a form of collaborative ranking!
  - Connectivity = people vote with their links
  - Usage = people vote with their clicks
- These schemes are used only for a global model building. Can it be combined with per-user data?
  Ideas:
  - Consider the graph induced by the user’s bookmarks.
  - Profile the user -- see www.globalbrain.net
  - Deal with privacy concerns!
Web IR Tools

- General-purpose search engines:
- Hierarchical directories
- Specialized search engines:
- Search-by-example
- Collaborative filtering
- Meta-information
  - Web Statistics: Size, Growth, ...
  - Usage Statistic
  - Query Log Statistics

Size of the Web

Arachnometry: The art and science of measuring the world wide Web, in particular the coverage of various search engines.

- How big is the Web?
- How fast does the Web grow?
- What is the proportion of pages in French, XML, Postscript, etc.? ?
- How do various search engines compare?
Comparing Search Engine Coverage

- Naïve Approaches
  - Get a list of URLs from each search engine and compare
    - Not practical or reliable.
  - Result Set Size Comparison
    - Reported sizes are approximate.
- …
- Better Approach
  - Statistical Sampling

URL sampling

- Ideal strategy: Generate a random URL and check for containment in each index.
- Random URLs are hard to generate:
  - Random walks methods
    - Graph is directed
    - Stationary distribution is non-uniform
    - Must prove rapid mixing.
  - Pages in cache, query logs [LG’98a], etc.
    - Correlated to the interests of a particular group of users.
- A simple way: collect all pages on the Web and pick one at random.
Sampling via queries [BB’98]

- Search engines have the best crawlers -- why not exploit them?
- Method:
  - Sample from each engine in turn
  - Estimate the relative sizes of two search engines
  - Compute absolute sizes from a reference point

Estimate relative sizes

Select pages randomly from A (resp. B)
Check if page contained in B (resp. A)

\[ |A \cap B| = (1/2) \cdot |A| \]
\[ |A \cap B| = (1/6) \cdot |B| \]
∴ \[ |B| = 3 \cdot |A| \]

Two steps: (i) Selecting   (ii) Checking
Selecting a random page

- Generate random query
  - Build a lexicon of words that occur on the Web
  - Combine random words from lexicon to form queries
- Get the first 100 query results from engine A
- Select a random page out of the set
- Distribution is biased -- the conjecture is that

\[
\sum_{D \in A \cap B} \frac{p(D)}{\sum_{D \in A} p(D)} \sim \frac{|A \cap B|}{|A|}
\]

where \( p(D) \) is the probability that D is picked by this scheme.

Checking if an engine has a page

- Create a “unique query” for the page:
  - Use 8 rare words.
  - E.g., for the Systems Research Center Home Page:
Results of the BB’98 study

Status as of July ‘98
- Web size: 350 million pages
- Growth: 25M pages/month
- AltaVista is the biggest: 35%
- Six largest engines cover: 2/3
- Small overlap: 3M pages

Crawling strategy is very different!

Exclusive listings in millions of pages

- Lycos: 5
- Northern Light: 20
- AltaVista: 50
- HotBot: 45
- Excite: 13
- Infoseek: 8
Open problems

- Random page generation via random walks
- Better “sampling via queries” method
- Cryptography based approach: want random pages from each engine but no cheating! (page should be chosen u.a.r. from the actual index)
  - Each search engine can commit to the set of pages it has without revealing it
  - Need to ensure that this set is the same as the set actually indexed
  - Need efficient oblivious protocol to obtain random page from search engine
  - See [NP’98] for possible solution

How often do people view a page?

- Problems:
  - Web caches interfere with click counting
  - cheating pays (advertisers pay by the click)
- Solutions:
  - naïve: forces caches to re-fetch for every click.
    - Lots of traffic, annoyed Web users
  - extend HTML with counters [ML’97]
    - requires compliance, down caches falsify results.
  - use sampling [P’97]
    - force refetches on random days
    - force refetches for random users and IP addresses
  - cryptographic audit bureaus [NP’98a]
- Commercial providers: 100hot, Media Matrix, Relevant Knowledge, …
Query Log Statistics [SHMM’98]

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 non-empty requests
  – ~575 000 000 non-empty queries
  – ~153 000 000 unique non-empty 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

Lots of things we didn’t even touch...

• Clustering = group similar items (documents or queries)
together ↔ unsupervised learning
• Categorization = assign items to predefined categories ↔ supervised learning
• Classic IR issues that are not substantially different in
the Web context:
  – Latent semantic indexing -- associate “concepts” to
    queries and documents and match on concepts
  – Summarization: abstract the most important parts of
text content. (See [TS’98] for the Web context)
• Many others …
Final conclusions

- We talked mostly about IR methods and tools that
  - take advantage of the Web particularities
  - mitigate some of the difficulties
- Web IR offers plenty of interesting problems…
  … but not on a silver platter
- Almost every area of algorithms research is relevant:
  - data structures
  - cryptography
  - on-line scheduling
  - random walks
  - compression
  - load balancing
  - learning
  - graph algorithm
  - etc
  - stringology
  - external memory algorithms
  - algorithms

Great need for algorithms engineering

Thanks!

- Krishna Bharat
- Moses Charikar
- Edith Cohen
- Jeff Dean
- Uri Feige
- Puneet Kumar
- Hannes Marais
- Michael Mitzenmacher
- Prabhakar Raghavan
- Lyle Ramshaw