Plan for today

- Web size estimation
- Mirror/duplication detection
- Pagerank

Size of the web

What is the size of the web?

- Issues
  - The web is really infinite
  - Dynamic content, e.g., calendar
    - Soft 404: www.yahoo.com/anything is a valid page
  - Static web contains syntactic duplication, mostly due to mirroring (~20-30%)
  - Some servers are seldom connected
- Who cares?
  - Media, and consequently the user
  - Engine design
  - Engine crawl policy. Impact on recall

What can we attempt to measure?

- The relative size of search engines
- The notion of a page being indexed is still reasonably well defined.
  - Already there are problems
    - Document extension: e.g. Google indexes pages not yet crawled by indexing anchor text.
    - Document restriction: Some engines restrict what is indexed (first n words, only relevant words, etc.)
- The coverage of a search engine relative to another particular crawling process.

Statistical methods

- Random queries
- Random searches
- Random IP addresses
- Random walks
URL sampling via Random Queries

- Ideal strategy: Generate a random URL and check for containment in each index.
- Problem: Random URLs are hard to find!

Random queries [Bhar98a]

- Sample URLs randomly from each engine
  - 20,000 random URLs from each engine
  - Issue random conjunctive query with <200 results
- Test if present in other engines.
  - Query with 8 rarest words. Look for URL match
  - Compute intersection & size ratio
  \[
  \text{Intersection} = \frac{x}{100} \times E_1 = \frac{y}{100} \times E_2
  \]
  \[
  E_1/E_2 = \frac{y}{x}
  \]
- Issues
  - Random narrow queries may bias towards long documents
    (Verify with disjunctive queries)
  - Other biases induced by process

Random searches [Lawr98, Lawr99]

- 575 & 1050 queries from the NEC RI employee logs
- 6 Engines in 1998, 11 in 1999
- Implementation:
  - Restricted to queries with < 600 results in total
  - Counted URLs from each engine after verifying query match
  - Computed size ratio & overlap for individual queries
  - Estimated index size ratio & overlap by averaging over all queries
- Issues
  - Samples are correlated with source of log
  - Duplicates
  - Technical statistical problems (must have non-zero results, ratio average, use harmonic mean?)

Queries from Lawrence and Giles study

- adaptive access control
- neighborhood preservation topographic
- hamiltonian structures
- right linear grammar
- pulse width modulation neural
- unbalanced prior probabilities
- ranked assignment method
- internet explorer favourites importing
- karvel thornber
- zili liu
- softmax activation function
- bose multidimensional system theory
- gamma mlp
dvi2pdf
john oliensis
rieke spikes exploring neural
video watermarking
counterpropagation network
fat shattering dimension
abelson amorphous computing

Size of the Web Estimation [Lawr98, Bhar98a]

- Capture – Recapture technique
  - Assumes engines get independent random subsets of the Web

Bharat & Broder: 200 M (Nov 97), 275 M (Mar 98)
Lawrence & Giles: 320 M (Dec 97)
Random IP addresses [Lawr99]

- Generate random IP addresses
- Find, if possible, a web server at the given address
- Collect all pages from server
- Advantages
  - Clean statistics, independent of any crawling strategy

Random IP addresses [ONei97, Lawr99]

- HTTP requests to random IP addresses
  - Ignored: empty or authorization required or excluded
  - [Lawr99] Estimated 2.8 million IP addresses running crawlable web servers (16 million total) from observing 2500 servers.
  - OCLC using IP sampling found 8.7 M hosts in 2001
    - Netcraft [Netc02] accessed 37.2 million hosts in July 2002
  - [Lawr99] exhaustively crawled 2500 servers and extrapolated
    - Estimated size of the web to be 800 million
    - Estimated use of metadata descriptors:
      - Meta tags (keywords, description) in 34% of home pages
      - Dublin core metadata in 0.3%

Issues

- Virtual hosting
- Server might not accept http://102.93.22.15
- No guarantee all pages are linked to root page
- Power law for # pages/hosts generates bias

Random walks [Henz99, BarY00, Rusm01]

- View the Web as a directed graph from a given list of seeds.
- Build a random walk on this graph
  - Includes various “jump” rules back to visited sites
  - Converges to a stationary distribution
  - Time to convergence not really known
  - Sample from stationary distribution of walk
  - Use the “small results set query” method to check coverage by SE
    - “Statistically clean” method, at least in theory!

Issues

- List of seeds is a problem.
- Practical approximation might not be valid: Non-uniform distribution, subject to link spamming
- Still has all the problems associated with “strong queries”

Conclusions

- No sampling solution is perfect.
- Lots of new ideas ...
  ....but the problem is getting harder
- Quantitative studies are fascinating and a good research problem
Duplicates and mirrors

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

Computing 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]
    - Set intersection [Brod98]
      (Specifically, Size_of_Intersection / Size_of_Union)

Shingles + Set Intersection

- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
- Approximate using a cleverly chosen subset of shingles from each (a sketch)
- Estimate size_of_intersection / 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 = fingerprinting)
  - Let π_i be a specific random permutation on 0..2^m
  - Pick MIN_π_i(f(s)) over all shingles s in D

Computing Sketch[i] for Doc1

Document 1

Start with 64 bit shingles
Permute on the number line with π_i
Pick the min value

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

Document 1

Document 2

Are these equal?

Test for 200 random permutations: π_1, π_2, ..., π_200
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:
\[
\frac{\text{Size of intersection}}{\text{Size of union}}
\]

Why? See minhash slides on class website.

Mirror Detection

- Mirroring is systematic replication of web pages across hosts.
- Single largest cause of duplication on the web
- Host1/a and Host2/b are mirrors if
  - For all (or most) paths p such that when http://Host1/ a / p exists http://Host2/ b / p exists as well
  - with identical (or near identical) content, and vice versa.
- E.g.,
  - Structural Classification of Proteins
    - http://scop.mrc-lmb.cam.ac.uk/scop
    - http://scop.berkeley.edu/
    - http://pdb.weizmann.ac.il/scop
    - http://scop.protres.ru/

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”
  - Proxy caching

Repackaged Mirrors

- Auctions.msn.com
- Auctions.lycos.com

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

Can we use URLs to find mirrors?

- www.synthesis.org
- synthesis.stanford.edu
Top Down Mirror Detection  [Bhar99, Bhar00c]

- E.g.,
  - www.synthesis.org/Docs/ProjAbs/qsnsys/qsnsys.html
  - synthesis.stanford.edu/Docs/ProjAbs/qsnsys/qsnsys.html
- What features could indicate mirroring?
  - Hostname similarity:
    - word unigrams and bigrams: { www, www.synthesis, synthesis, ... }
  - Directory similarity:
    - Positional path bigrams { Docs/ProjAbs, 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

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 = 0.89

Link Analysis on the Web

Citation Analysis

- Citation frequency
- Co-citation coupling frequency
  - Cocitations with a given author measures “impact”
  - Cocitation analysis [Mcca90]
  - Convert frequencies to correlation coefficients, do multivariate analysis/joining, validate conclusions
  - E.g., cocitation in the “Geography and GIS” web shows communities [Lars96]
- Bibliographic coupling frequency
  - Articles that co-cite the same articles are related
- Citation indexing
  - Who is a given author cited by? (Garfield [Garf72])
    - E.g., Science Citation Index ( http://www.isinet.com/)
    - CiteSeer ( http://citeseer.ist.psu.edu ) [Lawr99a]

Query-independent ordering

- First generation: using link counts as simple measures of popularity.
- Two basic suggestions:
  - Undirected popularity:
    - Each page gets a score = the number of in-links plus the number of out-links (3+2=5).
  - Directed popularity:
    - Score of a page = number of its in-links (3).

Query processing

- First retrieve all pages meeting the text query (say venture capital).
- Order these by their link popularity (either variant on the previous page).
Spamming simple popularity

- Exercise: How do you spam each of the following heuristics so your page gets a high score?
- Each page gets a score = the number of in-links plus the number of out-links.
- Score of a page = number of its in-links.

Pagerank scoring

- Imagine a browser doing a random walk on web pages:
  - Start at a random page
  - At each step, go out of the current page
  - "In the steady state" each page has a long-term visit rate - use this as the page's score.

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.

Teleporting

- At a dead end, jump to a random web page.
- At any non-dead end, with probability 10%, jump to a random web page.
  - With remaining probability (90%), go out on a random link.
  - 10% - a parameter.

Result of teleporting

- Now cannot get stuck locally.
- There is a long-term rate at which any page is visited (not obvious, will show this).
- How do we compute this visit rate?

Markov chains

- A Markov chain consists of \( n \) states, plus an \( n \times n \) transition probability matrix \( P \).
- At each step, we are in exactly one of the states.
- For \( 1 \leq i, j \leq n \), the matrix entry \( P_{ij} \) tells us the probability of \( j \) being the next state, given we are currently in state \( i \).
  - \( P_{ii} > 0 \) is OK.
Markov chains
- Clearly, for all $i$, $\sum_{j=1}^{n} p_{ij} = 1$.
- Markov chains are abstractions of random walks.
- **Exercise**: represent the teleporting random walk from 3 slides ago as a Markov chain, for this case:

Ergodic Markov chains
- A Markov chain is **ergodic** if
  - you have a path from any state to any other
  - you can be in any state at every time step, with non-zero probability.

Ergodic Markov chains
- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
  - **Steady-state distribution**.
  - Over a long time-period, we visit each state in proportion to this rate.
  - It doesn’t matter where we start.

Probability vectors
- A probability (row) vector $x = (x_1, \ldots, x_n)$ tells us where the walk is at any point.
  - E.g., $(000\ldots1\ldots000)$ means we’re in state $i$.
  - More generally, the vector $x = (x_1, \ldots, x_n)$ means the walk is in state $i$ with probability $x_i$.

Change in probability vector
- If the probability vector is $x = (x_1, \ldots, x_n)$ at this step, what is it at the next step?
- Recall that row $i$ of the transition prob. Matrix $P$ tells us where we go next from state $i$.
- So from $x$, our next state is distributed as $xP$.

Steady state example
- The steady state looks like a vector of probabilities $a = (a_1, \ldots, a_n)$:
  - $a_i$ is the probability that we are in state $i$.

For this example, $a_1=1/4$ and $a_2=3/4$. 
How do we compute this vector?

- Let \( a = (a_1, \ldots, a_n) \) denote the row vector of steady-state probabilities.
- If we our current position is described by \( a \), then the next step is distributed as \( aP \).
- But \( a \) is the steady state, so \( a = aP \).
- Solving this matrix equation gives us \( a \).
  - So \( a \) is the (left) eigenvector for \( P \).
  - (Corresponds to the “principal” eigenvector of \( P \) with the largest eigenvalue.)
  - Transition probability matrices always have larges eigenvalue 1.

One way of computing \( a \)

- Recall, regardless of where we start, we eventually reach the steady state \( a \).
- Start with any distribution (say \( x = (10 \ldots 0) \)).
- After one step, we’re at \( xP \).
- After two steps at \( xP^2 \), then \( xP^3 \) and so on.
- “Eventually” means for “large” \( k \), \( xP^k = a \).
- Algorithm: multiply \( x \) by increasing powers of \( P \) until the product looks stable.

Pagerank summary

- Preprocessing:
  - Given graph of links, build matrix \( P \).
  - From it compute \( a \).
  - The entry \( a_i \) is a number between 0 and 1: the pagerank of page \( i \).
- Query processing:
  - Retrieve pages meeting query.
  - Rank them by their pagerank.
  - Order is query-independent.

The reality

- Pagerank is used in google, but so are many other clever heuristics.
- more on these heuristics later.

Resources