The CMU machine learning protesters

http://www.flickr.com/photos/30686429@N07/3953914015/in/set-72157622330082619/
Web basics

David Kauchak

cs160

Fall 2009

adapted from:

http://www.stanford.edu/class/cs276/handouts/lecture13-webchar.ppt
Administrative

- CS lunch today!
- Unique hw5
  - reading
  - course feedback
- Schedule
Boolean queries

- c OR a AND f
- a AND f OR c
Outline

- Brief overview of the web
- Web Spam
- Estimating the size of the web
- Detecting duplicate pages
Brief (non-technical) history

- Early keyword-based engines
  - Altavista, Excite, Infoseek, Inktomi, ca. 1995-1997
- Sponsored search ranking: Goto.com (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid
  - Auction for keywords: casino was expensive!
Brief (non-technical) history

- 1998+: Link-based ranking pioneered by Google
  - Blew away all early engines save Inktomi
  - Great user experience in search of a business model
  - Meanwhile Goto/Overture’s annual revenues were nearing $1 billion
- Result: Google added paid-placement “ads” to the side, independent of search results
  - Yahoo followed suit, acquiring Overture (for paid placement) and Inktomi (for search)
Why did Google win?

- Relevance/link-based
- Simple UI
- Hardware – used commodity parts
  - inexpensive
  - easy to expand
  - fault tolerance through redundancy
- What’s wrong (from the search engine’s standpoint) of having a cost-per-click (CPC) model and ranking ads based only on CPC?
Web search basics

The Web

Web spider

Indexer

Indexes

Search

User

Ad indexes
User needs/queries

- Researchers/search engines often categorize user needs/queries into different types
- For example…?
User Needs

- **Need [Brod02, RL04]**
  - **Informational** – want to learn about something (~40%)
    - Low hemoglobin
  - **Navigational** – want to go to that page (~25%)
    - United Airlines
  - **Transactional** – want to do something (web-mediated) (~35%)
    - Access a service
    - Seattle weather
    - Downloads
    - Mars surface images
    - Shop
    - Canon S410
  - **Gray areas**
    - Find a good hub
      - Car rental Brasil
    - Exploratory search “see what’s there”
How far do people look for results?

“When you perform a search on a search engine and don’t find what you are looking for, at what point do you typically either revise your search, or move on to another search engine? (Select one)”

- 27% After reviewing more than 3 pages
- 25% After reviewing the first 3 pages
- 20% After reviewing the first 2 pages
- 16% After reviewing the first page
- 12% After reviewing the first few entries

(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)
Users’ empirical evaluation of results

- Quality of pages varies widely
  - Relevance is not enough
  - Other desirable qualities (non IR!!)
    - Content: Trustworthy, diverse, non-duplicated, well maintained
    - Web readability: display correctly & fast
    - No annoyances: pop-ups, etc

- Precision vs. recall
  - On the web, recall seldom matters
  - Recall matters when the number of matches is very small

- What matters
  - Precision at 1? Precision above the fold?
  - Comprehensiveness – must be able to deal with obscure queries

- User perceptions may be unscientific, but are significant over a large aggregate
The Web document collection

- No design/co-ordination
- Content includes truth, lies, obsolete information, contradictions …
- Unstructured (text, html, …), semi-structured (XML, annotated photos), structured (Databases)…
- Financial motivation for ranked results
- Scale much larger than previous text collections … but corporate records are catching up
- Growth – slowed down from initial “volume doubling every few months” but still expanding
- Content can be dynamically generated
Web Spam

http://blog.lib.umn.edu/wilsper/informationcentral/spam.jpg
The trouble with sponsored search …

- It costs money. What’s the alternative?
- **Search Engine Optimization:**
  - “Tuning” your web page to rank highly in the algorithmic search results for select keywords
  - Alternative to paying for placement
  - Intrinsically a marketing function
- Performed by companies, webmasters and consultants (“Search engine optimizers”) for their clients
- Some perfectly legitimate, some very shady
Simplest forms

- First generation engines relied heavily on *tf/idf*
- What would you do as an SEO?
- SEOs responded with dense repetitions of chosen terms
  - e.g., *maui resort maui resort maui resort*
  - Often, the repetitions would be in the same color as the background of the web page
    - Repeated terms got indexed by crawlers
    - But not visible to humans on browsers

Pure word density cannot be trusted as an IR signal
Variants of keyword stuffing

- Misleading meta-tags, excessive repetition
- Hidden text with colors, style sheet tricks, etc.

**Meta-Tags =**
“... London hotels, hotel, holiday inn, hilton, discount, booking, reservation, sex, mp3, britney spears, viagra, ...”
Spidering/indexing

Any way we can take advantage of this system?
Cloaking

- Serve fake content to search engine spider
More spam techniques

- **Doorway pages**
  - Pages optimized for a single keyword that re-direct to the real target page

- **Link spamming**
  - Mutual admiration societies, hidden links, awards – more on these later
  - *Domain flooding:* numerous domains that point or re-direct to a target page

- **Robots**
  - Fake query stream – rank checking programs
    - “Curve-fit” ranking programs of search engines
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)
- 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
  - Suspect pattern detection
More on spam

- Web search engines have policies on SEO practices they tolerate/block
- Adversarial IR: the unending (technical) battle between SEO’s and web search engines
- Research  http://airweb.cse.lehigh.edu/
Size of the web

What is the size of the web?

- 7,452,502,600,001 pages (as of yesterday)
- The web is really infinite
  - Dynamic content, e.g., calendar
  - Soft 404: www.yahoo.com/<anything> is a valid page
- What about just the static web… issues?
  - Static web contains syntactic duplication, mostly due to mirroring (~30%)
  - Some servers are seldom connected
Who cares about the size of the web?

- It is an interesting question, but beyond that, who cares and why?
- Media, and consequently the user
- Search engine designer (crawling, indexing)
- Researchers
What can we measure?

Besides absolute size, what else might we measure?

- Users interface is through the search engine
  - Proportion of the web a particular search engine indexes
  - The size of a particular search engine’s index
  - Relative index sizes of two search engines

Challenges with these approaches?

Biggest one: search engines don’t like to let people know what goes on under the hood
Although we can’t ask how big a search engine’s index is, we can often ask questions like “does a document exist in the index?”
Proportion of the web indexed

- We can ask if a document is in an index.
- How can we estimate the proportion indexed by a particular search engine?
Size of index A relative to index B

web

random sample

engine A

engine B

proportion of sample in index
Sampling URLs

- Both of these questions require us to have a random set of pages (or URLs)
- Problem: Random URLs are hard to find!
- Ideas?
- Approach 1: Generate a random URL contained in a given engine
  - Suffices for the estimation of relative size
- Approach 2: Random pages/ IP addresses
  - In theory: might give us a true estimate of the size of the web (as opposed to just relative sizes of indexes)
Random URLs from search engines

- Issue a random query to the search engine
  - Randomly generate a query from a lexicon and word probabilities (generally focus on less common words/queries)
  - Choose random searches extracted from a query log (e.g. all queries from Pomona College)
- From the first 100 results, pick a random page/URL
Things to watch out for

- Biases induced by random queries
  - Query Bias: Favors content-rich pages in the language(s) of the lexicon
  - Ranking Bias: Use conjunctive queries & fetch all
  - Checking Bias: Duplicates, impoverished pages omitted
  - Malicious Bias: Sabotage by engine
  - Operational Problems: Time-outs, failures, engine inconsistencies, index modification

- Biases induced by query log
  - Samples are correlated with source of log
Random IP addresses

Generate random IP

xxx.xxx.xxx.xxx

check if there is a web server at that IP

collect pages from server

randomly pick a page/URL
Random IP addresses

- [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
Random walks

- View the Web as a directed graph
- Build a random walk on this graph
  - Includes various “jump” rules back to visited sites
    - Does not get stuck in spider traps!
    - Can follow all links!
  - Converges to a stationary distribution
    - Must assume graph is finite and independent of the walk.
    - Conditions are not satisfied (cookie crumbs, flooding)
    - Time to convergence not really known
- Sample from stationary distribution of walk
- Use the “strong query” method to check coverage by SE
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
Duplicate detection

http://rlv.zcache.com/cartoon_man_with_balled_fist_postcard-p239288482636625726trdg_400.jpg
Duplicate documents

- The web is full of duplicated content
  - Redundancy/mirroring
  - Copied content
- Do we care?
- How can we detect duplicates?
- Hashing
  - Hash each document
  - Compares hashes
  - For those that are equal, check if the content is equal
Duplicate?
Near duplicate documents

- Many, many cases of near duplicates
  - E.g., last modified date the only difference between two copies of a page
- A good hashing function specifically tries not to have collisions
- Ideas?
  - Locality sensitive hashing – (http://www.mit.edu/~andoni/LSH/)
  - Similarity – main challenge is efficiency!
Computing Similarity

- We could use edit distance, but way too slow
- What did we do for spelling correction?
- compare word n-gram (shingles) overlap
  - *a rose is a rose is a rose* →
    - a_rose_is_a
    - rose_is_a_rose
    - is_a_rose_is
    - a_rose_is_a
- Use Jaccard Coefficient to measure the similarity between documents (A and B)/(A or B)
N-gram intersection

- Computing exact set intersection of n-grams between all pairs of documents is expensive/intractable
- How did we solve the efficiency problem for spelling correction?
  - Indexed words by character n-grams
  - AND query of the character n-grams in our query word
- Will this work for documents?
- Number of word n-grams for a document is too large!
Efficient calculation of JC

- Use a hash function that maps an n-gram to a 64 bit number

Doc A

n-grams

Doc A

Jaccard Coefficient
Efficient calculation of JC

- Use a hash function that maps an n-gram to a 64 bit number

What if we just compared smallest one of each?
Efficient calculation of JC

- Use a hash function that maps an n-gram to a 64 bit number

- Apply a permutation to each 64 bit number
- Compare smallest values
- Repeat some number of times (say 200)
Efficient JC

Document 1

- Start with 64-bit \( n \)-grams
- Permute on the number line with \( \pi_i \)
- Pick the min value
Test if Doc1 = Doc2

Document 1

Document 2

Are these equal?
Test if Doc1 = Doc2

The minimum values after the permutations will be equal with probability =

\[ \frac{\text{Size of intersection}}{\text{Size of union}} \]
Claim...

- Repeat this, say 200 times, with different permutations
- Measure the number of times they’re equal
- This is a reasonable estimate for the JC
All signature pairs

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a single pair of documents.
- But we still have to estimate $N^2$ coefficients where $N$ is the number of web pages.
  - Still slow
- Need to reduce the set of options
  - locality sensitive hashing (LSH)
  - sorting (Henzinger 2006)
Cool search engines

- What do you think will be the most important feature(s) in next-generation search algorithms?
- Is it better to have a broad, general search engine or one that is tailored to your needs?
- What new markets can be explored using a search engine?
- Some of these search engines are niche-specific sites and others are search aggregators. Is web search diverging in the direction of many topic-specific sites or converging to one large find-everything site? Is one of these better? What should we be aiming for?
- What are the benefits of live updating searches (Collecta) vs. previously indexed content (Google)?
- How do you think Collecta is able to find results so quickly?
- The article mentions “inserting a human element into search.” What exactly does this mean? How can a web search include human power? Is that useful?
Set Similarity of sets $C_i$, $C_j$

\[
\text{Jaccard}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}
\]

- View sets as columns of a matrix $A$; one row for each element in the universe. $a_{ij} = 1$ indicates presence of item $i$ in set $j$
- Example

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Jaccard($C_1, C_2$) = $\frac{2}{5} = 0.4$
Key Observation

- For columns $C_i$, $C_j$, four types of rows

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$C_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Overload notation: $A = \# \text{ of rows of type } A$

- Claim

$$\text{Jaccard}(C_i, C_j) = \frac{A}{A + B + C}$$
“Min” Hashing

- Randomly permute rows
- Hash \( h(C_i) = \) index of first row with 1 in column \( C_i \)
- Surprising Property
  \[
  P[h(C_i) = h(C_j)] \approx \text{Jaccard}(C_i, C_j)
  \]
- Why?
  - Both are \( A/(A+B+C) \)
  - Look down columns \( C_i, C_j \) until first non-Type-D row
  - \( h(C_i) = h(C_j) \iff \) type A row
Min-Hash sketches

- Pick $P$ random row permutations
- MinHash sketch

$\text{Sketch}_D = \text{list of } P \text{ indexes of first rows with 1 in column } C$

- Similarity of signatures
  - Let $\text{sim}[\text{sketch}(C_i),\text{sketch}(C_j)] = \text{fraction of permutations where MinHash values agree}$
  - Observe $\mathbb{E}[\text{sim}(\text{sig}(C_i),\text{sig}(C_j))] = \text{Jaccard}(C_i,C_j)$
Example

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R₂</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>R₃</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R₄</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R₅</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Signatures

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perm 1 = (12345)</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Perm 2 = (54321)</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Perm 3 = (34512)</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Similarities

<table>
<thead>
<tr>
<th></th>
<th>1-2</th>
<th>1-3</th>
<th>2-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col-Col</td>
<td>0.00</td>
<td>0.50</td>
<td>0.25</td>
</tr>
<tr>
<td>Sig-Sig</td>
<td>0.00</td>
<td>0.67</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Implementation Trick

- **Permuting** universe even once is prohibitive

- **Row Hashing**
  - Pick $P$ hash functions $h_k: \{1, \ldots, n\} \rightarrow \{1, \ldots, O(n)\}$
  - Ordering under $h_k$ gives random permutation of rows

- **One-pass Implementation**
  - For each $C_i$ and $h_k$, keep "slot" for min-hash value
  - **Initialize** all $\text{slot}(C_i, h_k)$ to infinity
  - **Scan rows** in arbitrary order looking for 1’s
    - Suppose row $R_j$ has 1 in column $C_i$
    - For each $h_k$,
      - if $h_k(j) < \text{slot}(C_i, h_k)$, then $\text{slot}(C_i, h_k) \leftarrow h_k(j)$
**Example**

<table>
<thead>
<tr>
<th></th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>( C_1 ) slots</th>
<th>( C_2 ) slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>1</td>
<td>0</td>
<td>h(1) = 1</td>
<td>1</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>0</td>
<td>1</td>
<td>g(1) = 3</td>
<td>3</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>1</td>
<td>1</td>
<td>h(2) = 2</td>
<td>1</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>1</td>
<td>0</td>
<td>g(2) = 0</td>
<td>3</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>0</td>
<td>1</td>
<td>h(3) = 3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>g(3) = 2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>h(4) = 4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>g(4) = 4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>h(5) = 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>g(5) = 1</td>
<td>2</td>
</tr>
</tbody>
</table>

\( h(x) = x \mod 5 \)

\( g(x) = 2x + 1 \mod 5 \)
Comparing Signatures

- **Signature Matrix S**
  - Rows = Hash Functions
  - Columns = Columns
  - Entries = Signatures

- Can compute – Pair-wise similarity of any pair of signature columns