

CS276

Information Retrieval and Web Search

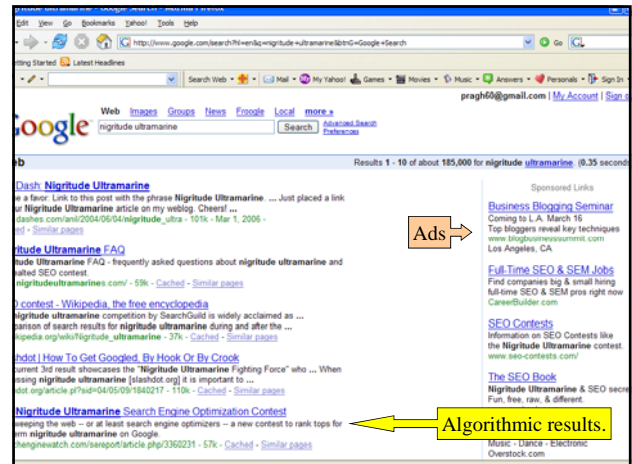
Lecture 13: Web search basics

Brief (non-technical) history

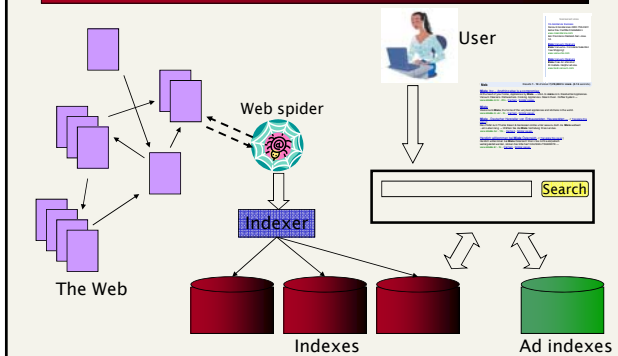
- Early keyword-based engines
 - Altavista, Excite, Infoseek, Inktomi, ca. 1995-1997
- Sponsored search ranking: Goto.com (morphed into Overture.com → Yahoo!)
 - Your search ranking depended on how much you paid
 - Auction for keywords: **casino** was expensive!

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)



Web search basics

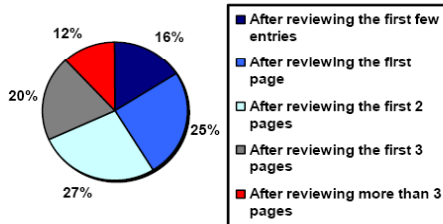


User Needs

- Need [Brod02, RL04]**
 - Informational** – want to learn about something (~40% / 65%)
 - Low hemoglobin
 - Navigational** – want to go to that page (~25% / 15%)
 - United Airlines
 - Transactional** – want to do something (web-mediated) (~35% / 20%)
 - 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)"



(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

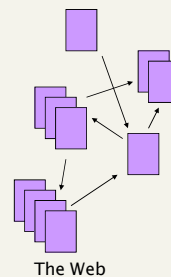
Users' empirical evaluation of results

- 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
- What matters
 - Precision at 1? Precision above the fold?
 - Comprehensiveness – must be able to deal with obscure queries
 - Recall matters when the number of matches is very small
- **User perceptions may be unscientific, but are significant over a large aggregate**

Users' empirical evaluation of engines

- Relevance and validity of results
- UI – Simple, no clutter, error tolerant
- Trust – Results are objective
- Coverage of topics for polysemic queries
- Pre/Post process tools provided
 - Mitigate user errors (auto spell check, search assist,...)
 - Explicit: Search within results, more like this, refine ...
 - Anticipative: related searches
- Deal with idiosyncrasies
 - Web specific vocabulary
 - Impact on stemming, spell-check, etc
 - Web addresses typed in the search box
 - ...

The Web document collection



- No design/co-ordination
- Distributed content creation, linking, democratization of publishing
- Content includes truth, lies, obsolete information, contradictions ...
- Unstructured (text, html, ...), semi-structured (XML, annotated photos), structured (Databases)...
- 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*

Spam

Search Engine Optimization

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
 - Thus, 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*
 - The top-ranked pages for the query `maui resort` were the ones containing the most `maui`'s and `resort`'s
- 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 =

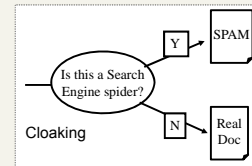
"... London hotels, hotel, holiday inn, hilton, discount, booking, reservation, sex, mp3, britney spears, viagra, ..."

Search engine optimization (Spam)

- Motives
 - Commercial, political, religious, lobbies
 - Promotion funded by advertising budget
- Operators
 - Contractors (Search Engine Optimizers) for lobbies, companies
 - Web masters
 - Hosting services
- Forums
 - E.g., Web master world (www.webmasterworld.com)
 - Search engine specific tricks
 - Discussions about academic papers ©

Cloaking

- Serve fake content to search engine spider
- DNS cloaking: Switch IP address. Impersonate



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
 - Millions of submissions via Add-Url

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
 - <http://help.yahoo.com/help/us/ysearch/index.html>
 - <http://www.google.com/intl/en/webmasters/>
- 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 ?

- 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 (~30%)
 - Some servers are seldom connected
- Who cares?
 - Media, and consequently the user
 - Engine design
 - Engine crawl policy. Impact on recall.

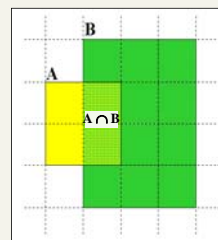
What can we attempt to measure?

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

New definition?

- (IQ is whatever the IQ tests measure.)
 - The statically indexable web is whatever search engines index.
- Different engines have different preferences
 - max url depth, max count/host, anti-spam rules, priority rules, etc.
- Different engines index different things under the same URL:
 - frames, meta-keywords, document restrictions, document extensions, ...

Relative Size from Overlap Given two engines A and B



Sample URLs randomly from A
Check if contained in B and vice versa

$$A \cap B = (1/2) * \text{Size A}$$

$$A \cap B = (1/6) * \text{Size B}$$

$$(1/2) * \text{Size A} = (1/6) * \text{Size B}$$

$$\therefore \text{Size A} / \text{Size B} =$$

$$(1/6) / (1/2) = 1/3$$

Each test involves: (i) Sampling (ii) Checking

Sampling URLs

- Ideal strategy: Generate a random URL and check for containment in each index.
- Problem: **Random URLs are hard to find!**
Enough to generate a random URL contained in a given Engine.
- Approach 1: Generate a random URL contained in a given engine
 - Suffices for the estimation of relative size
- Approach 2: Random walks / IP addresses
 - In theory: might give us a true estimate of the size of the web (as opposed to just relative sizes of indexes)

Statistical methods

- Approach 1
 - Random queries
 - Random searches
- Approach 2
 - Random IP addresses
 - Random walks

Random URLs from random queries

- Generate random query: how?
 - **Lexicon**: 400,000+ words from a web crawl Not an English dictionary
 - **Conjunctive Queries**: w_1 and w_2
e.g., vocalists AND rsi
- Get 100 result URLs from engine A
- Choose a random URL as the candidate to check for presence in engine B
- This distribution induces a probability weight $W(p)$ for each page.
- Conjecture: $W(SE_A) / W(SE_B) \sim |SE_A| / |SE_B|$

Query Based Checking

- **Strong Query** to check whether an engine B has a document D :
 - Download D . Get list of words.
 - Use 8 low frequency words as AND query to B
 - Check if D is present in result set.
- Problems:
 - Near duplicates
 - Frames
 - Redirects
 - Engine time-outs
 - Is 8-word query good enough?

Advantages & disadvantages

- Statistically sound under the induced weight.
- Biases induced by random query
 - Query Bias: Favors content-rich pages in the language(s) of the lexicon
 - Ranking Bias: *Solution*: Use conjunctive queries & fetch all
 - Checking Bias: Duplicates, impoverished pages omitted
 - Document or query restriction bias: engine might not deal properly with 8 words conjunctive query
 - Malicious Bias: Sabotage by engine
 - Operational Problems: Time-outs, failures, engine inconsistencies, index modification.

Random searches

- Choose random searches extracted from a local log [Lawrence & Giles 97] or build "random searches" [Notess]
 - Use only queries with small results sets.
 - Count normalized URLs in result sets.
 - Use ratio statistics

Advantages & disadvantages

- Advantage
 - Might be a better reflection of the human perception of coverage
- Issues
 - Samples are correlated with source of log
 - Duplicates
 - Technical statistical problems (must have non-zero results, ratio average not statistically sound)

Random searches

- 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

Queries from Lawrence and Giles study

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

Random IP addresses

- Generate random IP addresses
- Find a web server at the given address
 - If there's one
- Collect all pages from server
 - From this, choose a page at random

Random IP addresses

- 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%

Advantages & disadvantages

- Advantages
 - Clean statistics
 - Independent of crawling strategies
- Disadvantages
 - Doesn't deal with duplication
 - Many hosts might share one IP, or not accept requests
 - No guarantee all pages are linked to root page.
 - Eg: employee pages
 - Power law for # pages/hosts generates bias towards sites with few pages.
 - But bias can be accurately quantified IF underlying distribution understood
 - Potentially influenced by spamming (multiple IP's for same server to avoid IP block)

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

Advantages & disadvantages

- Advantages
 - “Statistically clean” method at least in theory!
 - Could work even for infinite web (assuming convergence) under certain metrics.
- Disadvantages
 - List of seeds is a problem.
 - Practical approximation might not be valid.
 - Non-uniform distribution
 - Subject to link spamming

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

Duplicate documents

- The web is full of duplicated content
- Strict duplicate detection = exact match
 - Not as common
- But many, many cases of near duplicates
 - E.g., Last modified date the only difference between two copies of a page

Duplicate/Near-Duplicate Detection

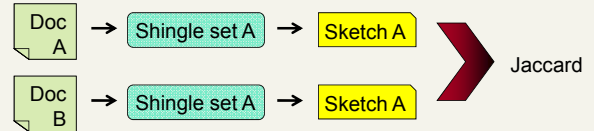
- *Duplication*: Exact match can be detected with fingerprints
- *Near-Duplication*: Approximate match
 - Overview
 - Compute syntactic similarity with an edit-distance measure
 - Use similarity threshold to detect near-duplicates
 - E.g., Similarity > 80% => Documents are “near duplicates”
 - Not transitive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

Computing Similarity

- Features:
 - Segments of a document (natural or artificial breakpoints)
 - Shingles (Word N-Grams)
 - a rose is a rose is a rose →
 - a_rose_is_a
 - rose_is_a_rose
 - is_a_rose_is
 - a_rose_is_a
- Similarity Measure between two docs (= sets of shingles)
 - Set intersection
 - Specifically (Size_of_Intersection / Size_of_Union)

Shingles + Set Intersection

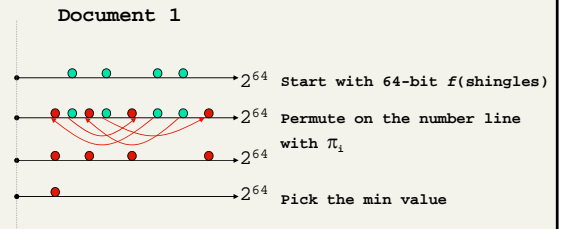
- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
 - Approximate using a cleverly chosen subset of shingles from each (a *sketch*)
 - Estimate (size_of_intersection / size_of_union) based on a short sketch



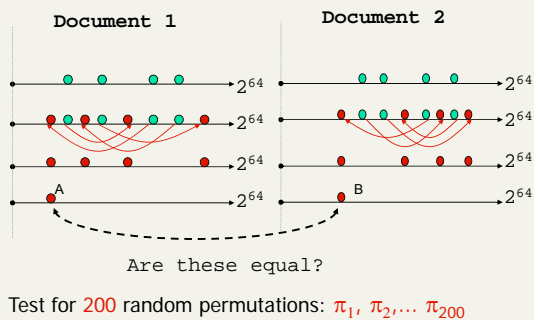
Sketch of a document

- Create a “sketch vector” (of size ~200) for each document
 - Documents that share $\geq t$ (say 80%) corresponding vector elements are **near duplicates**
 - For doc D , sketch $_D[i]$ is as follows:
 - Let f map all shingles in the universe to $0..2^m$ (e.g., f = fingerprinting)
 - Let π_i be a *random permutation* on $0..2^m$
 - Pick $\text{MIN} \{ \pi_i(f(s)) \}$ over all shingles s in D

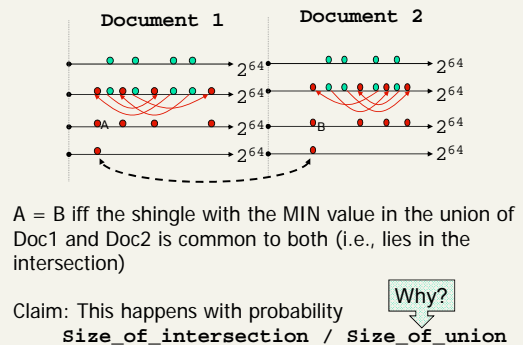
Computing Sketch[i] for Doc1



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



However...



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

	C_1	C_2
R_1	0	1
R_2	1	0
R_3	1	1
R_4	0	0
R_5	1	1
R_6	0	1

$$\text{Jaccard}(C_1, C_2) = 2/5 = 0.4$$

Key Observation

- For columns C_i, C_j , four types of rows

	C_i	C_j
A	1	1
B	1	0
C	0	1
D	0	0

- Overload notation: $A = \#$ 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

$$P[h(C_i) = h(C_j)] = \text{Jaccard}(C_i, C_j)$$
- Why?
 - Both are $A/(A+B+C)$
 - Look down columns C_i, C_j until first non-Type-D row
 - $h(C_i) = h(C_j) \leftrightarrow$ type A row

Min-Hash sketches

- Pick P random row permutations
- MinHash sketch

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

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

Example

		Signatures			
		S_1	S_2	S_3	
		Perm 1 = (12345)	1	2	1
		Perm 2 = (54321)	4	5	4
		Perm 3 = (34512)	3	5	4

		Similarities		
		1-2	1-3	2-3
Col-Col		0.00	0.50	0.25
Sig-Sig		0.00	0.67	0.00

	C_1	C_2	C_3
R_1	1	0	1
R_2	0	1	1
R_3	1	0	0
R_4	1	0	1
R_5	0	1	0

Implementation Trick

- Permuting universe even once is prohibitive
- Row Hashing
 - Pick P hash functions $h_k: \{1, \dots, n\} \rightarrow \{1, \dots, O(n)\}$
 - 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

	C_1	C_2		C_1 slots	C_2 slots
R_1	1	0	$h(1) = 1$	1	-
R_2	0	1	$g(1) = 3$	3	-
R_3	1	1	$h(2) = 2$	1	2
R_4	1	0	$g(2) = 0$	3	0
R_5	0	1	$h(3) = 3$	1	2
			$g(3) = 2$	2	0
			$h(4) = 4$	1	2
			$g(4) = 4$	2	0
			$h(5) = 0$	1	0
			$g(5) = 1$	2	0

$$h(x) = x \bmod 5$$
$$g(x) = 2x+1 \bmod 5$$

Comparing Signatures

- Signature Matrix S
 - Rows = Hash Functions
 - Columns = Columns
 - Entries = Signatures
- Can compute – Pair-wise similarity of any pair of signature columns

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
- One solution: locality sensitive hashing (LSH)
- Another solution: sorting (Henzinger 2006)

More resources

- IIR Chapter 19