Introduction to

Information Retrieval

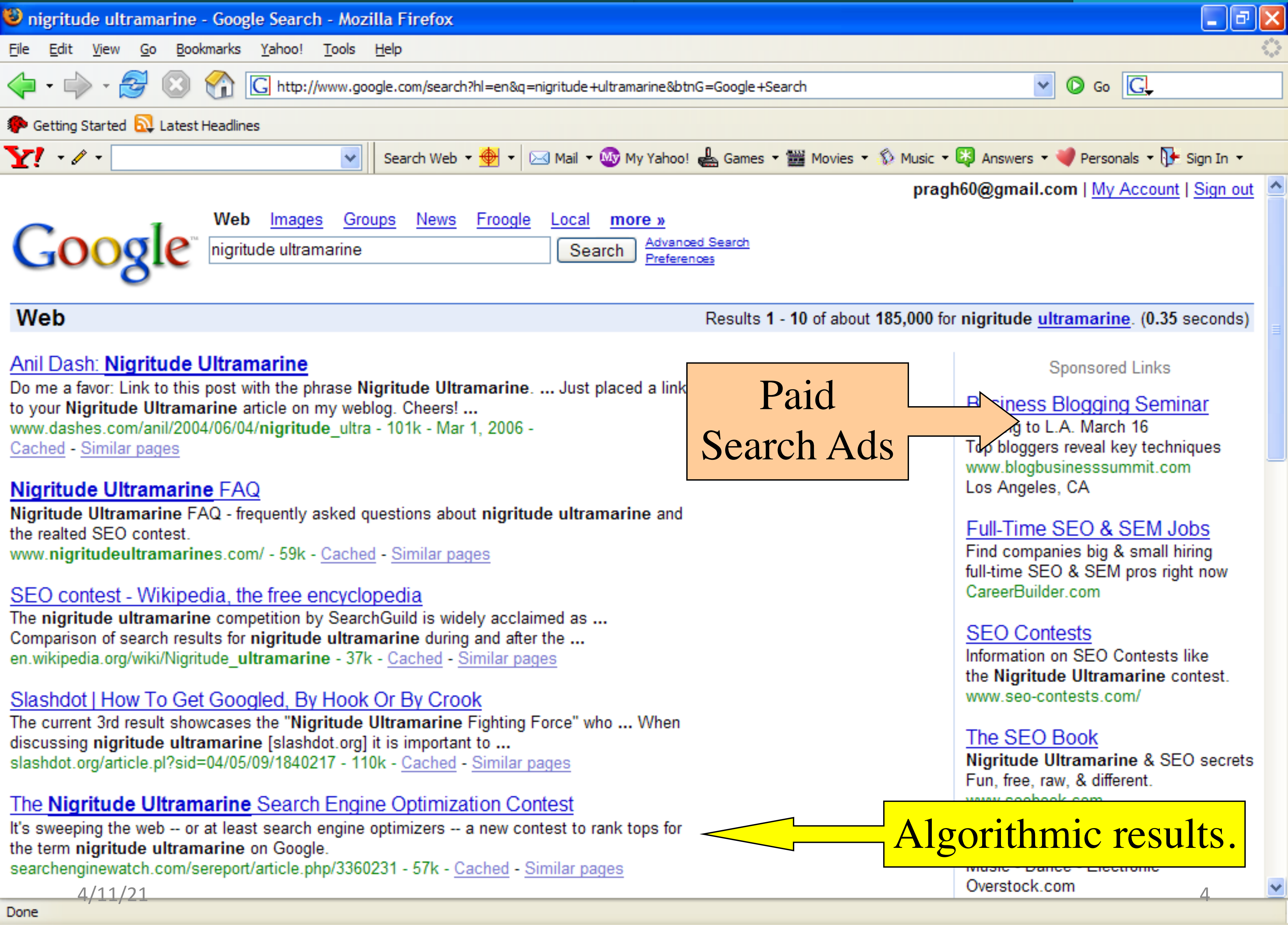
Lecture 16: Web search basics

Brief (non-technical) history

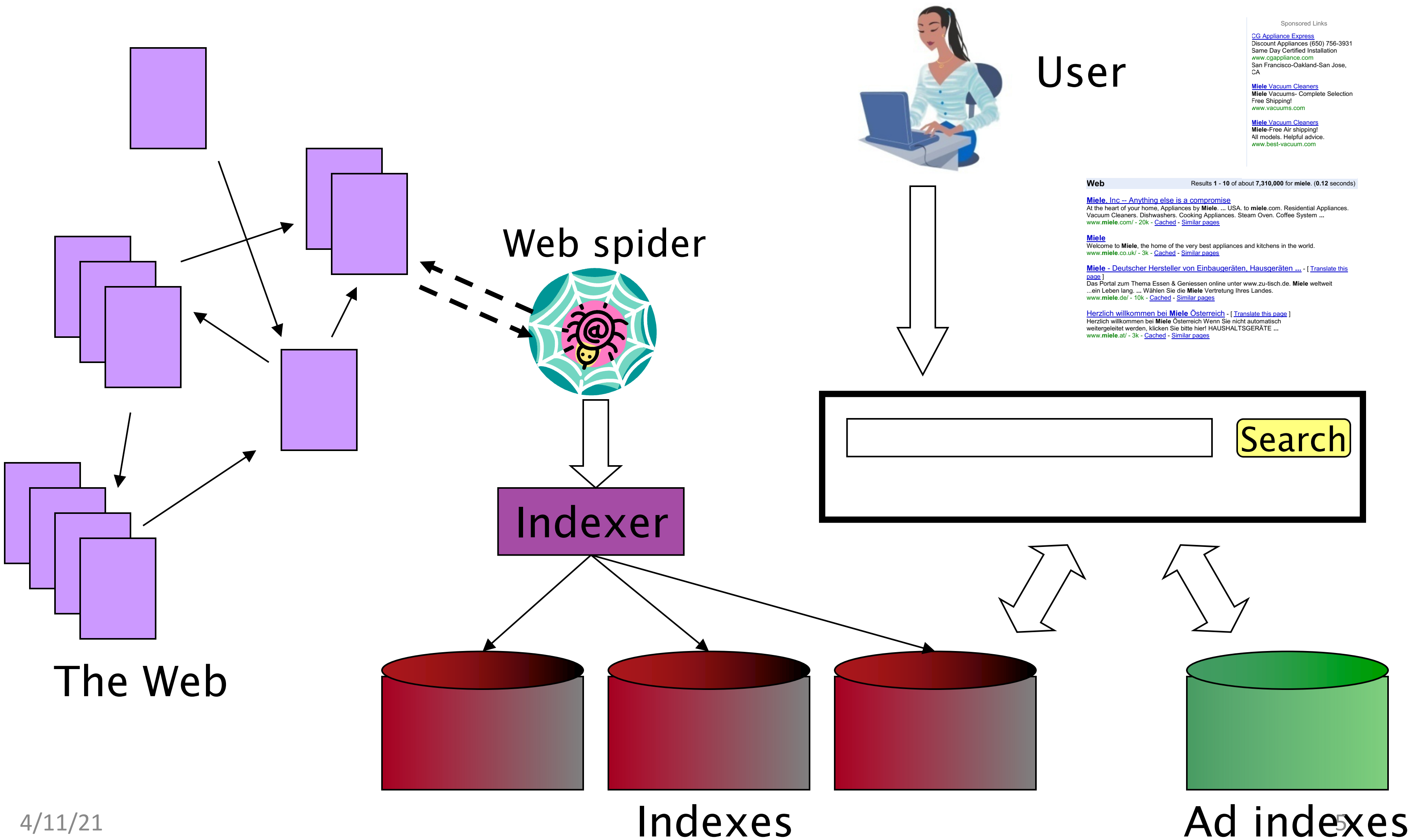
- Early keyword-based engines ca. 1995-1997
  - Altavista, Excite, Infoseek, Inktomi, Lycos
- Paid search ranking: Goto (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid
  - Auction for keywords: ***casino*** was expensive!

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 search "ads" to the side, independent of search results
  - Yahoo followed suit, acquiring Overture (for paid placement) and Inktomi (for search)
- 2005+: Google gains search share, dominating in Europe and very strong in North America
  - 2009: Yahoo! and Microsoft propose combined paid search offering


nigritude ultramarine - Google Search - Mozilla Firefox
File Edit View Go Bookmarks Yahoo! Tools Help
http://www.google.com/search?hl=en&q=nigritude+ultramarine&btnG=Google+Search
Go
Getting Started Latest Headlines
Search Web Mail My Yahoo! Games Movies Music Answers Personals Sign In
pragh60@gmail.com | My Account | Sign out
Web Images Groups News Froogle Local more »
nigritude ultramarine
Search
Advanced Search
Preferences
Web
Results 1 - 10 of about 185,000 for nigritude ultramarine. (0.35 seconds)
Anil Dash: Nigritude Ultramarine
Do me a favor: Link to this post with the phrase Nigritude Ultramarine. ... Just placed a link to your Nigritude Ultramarine article on my weblog. Cheers! ...
www.dashes.com/anil/2004/06/04/nigritude_ultra - 101k - Mar 1, 2006 -
Cached - Similar pages
Nigritude Ultramarine FAQ
Nigritude Ultramarine FAQ - frequently asked questions about nigritude ultramarine and the realted SEO contest.
www.nigritudeultramarines.com/ - 59k - Cached - Similar pages
SEO contest - Wikipedia, the free encyclopedia
The nigritude ultramarine competition by SearchGuild is widely acclaimed as ...
Comparison of search results for nigritude ultramarine during and after the ...
en.wikipedia.org/wiki/Nigritude_ultramarine - 37k - Cached - Similar pages
Slashdot | How To Get Googled, By Hook Or By Crook
The current 3rd result showcases the "Nigritude Ultramarine Fighting Force" who ... When discussing nigritude ultramarine [slashdot.org] it is important to ...
slashdot.org/article.pl?sid=04/05/09/1840217 - 110k - Cached - Similar pages
The Nigritude Ultramarine Search Engine Optimization Contest
It's sweeping the web -- or at least search engine optimizers -- a new contest to rank tops for the term nigritude ultramarine on Google.
searchenginewatch.com/sereport/article.php/3360231 - 57k - Cached - Similar pages
Sponsored Links
Top bloggers reveal key techniques
www.blogbusinesssummit.com
Los Angeles, CA
Full-Time SEO & SEM Jobs
Find companies big & small hiring full-time SEO & SEM pros right now
CareerBuilder.com
SEO Contests
Information on SEO Contests like the Nigritude Ultramarine contest.
www.seo-contests.com/
The SEO Book
Nigritude Ultramarine & SEO secrets
Fun, free, raw, & different.
Overstock.com
Done
Paid Search Ads
Algorithmic results.

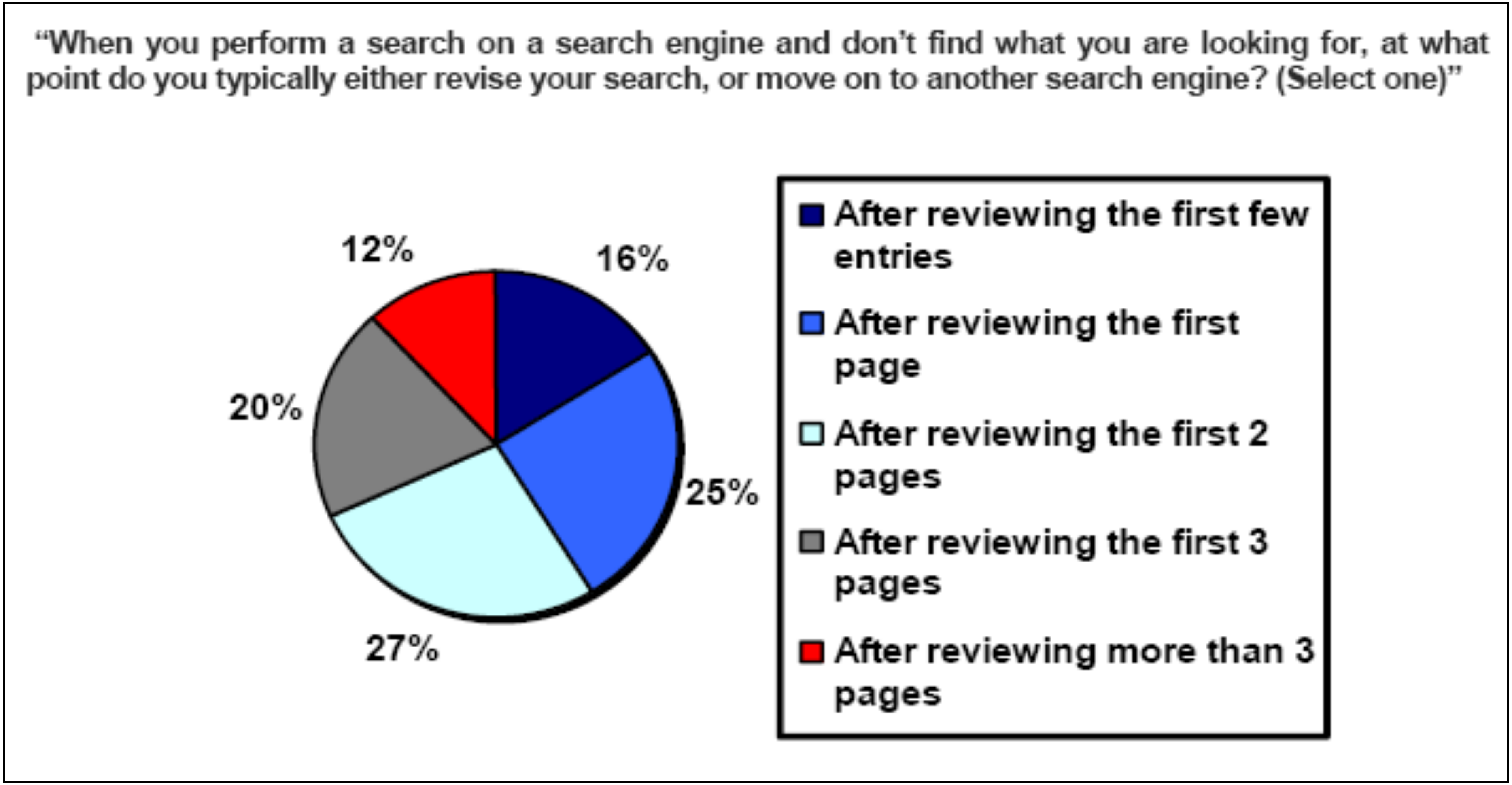
Web search basics



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?



(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

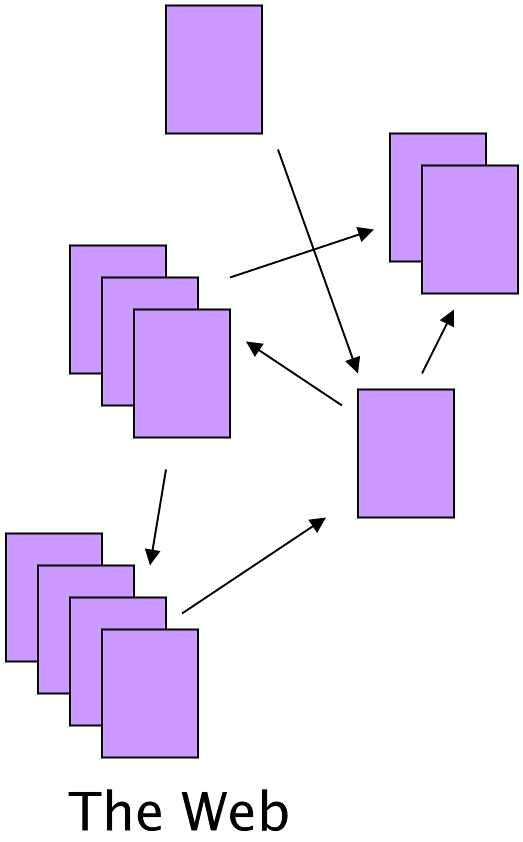
Users' empirical evaluation of results

- 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 first, the last, the best and the worst …"

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)

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?

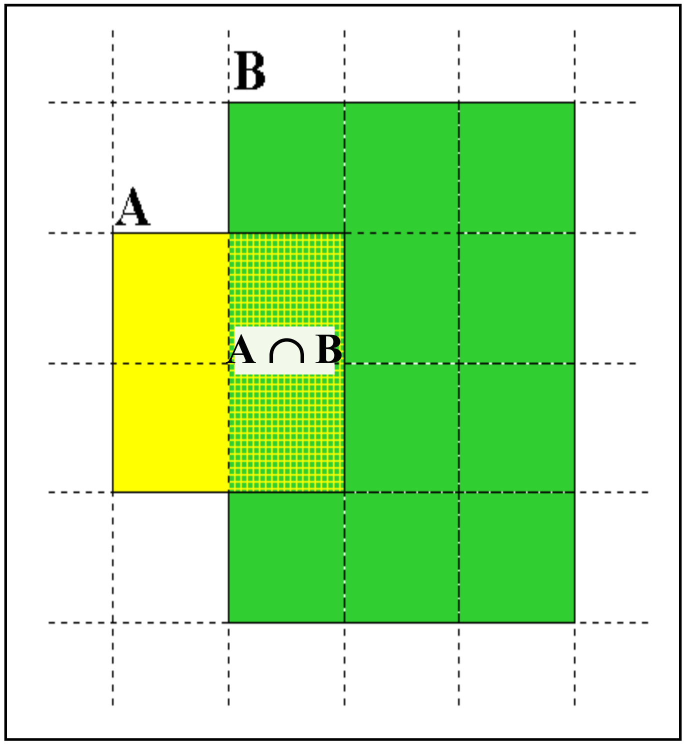
- The relative sizes of search engines
  - The notion of a page being indexed is still *reasonably* well defined.
  - Already there are problems
    - Document extension: e.g. engines index pages not yet crawled, by indexing anchortext.
    - Document restriction: All engines restrict what is indexed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

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
  - **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

Not an English dictionary

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 result 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

- *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*

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 pages
  - 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

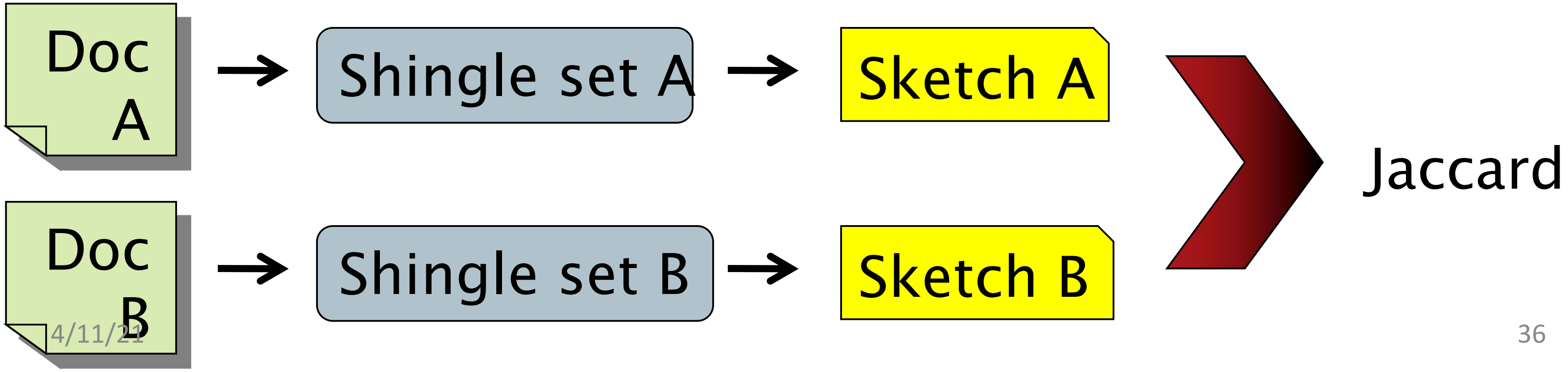
    rose_is_a_rose

    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

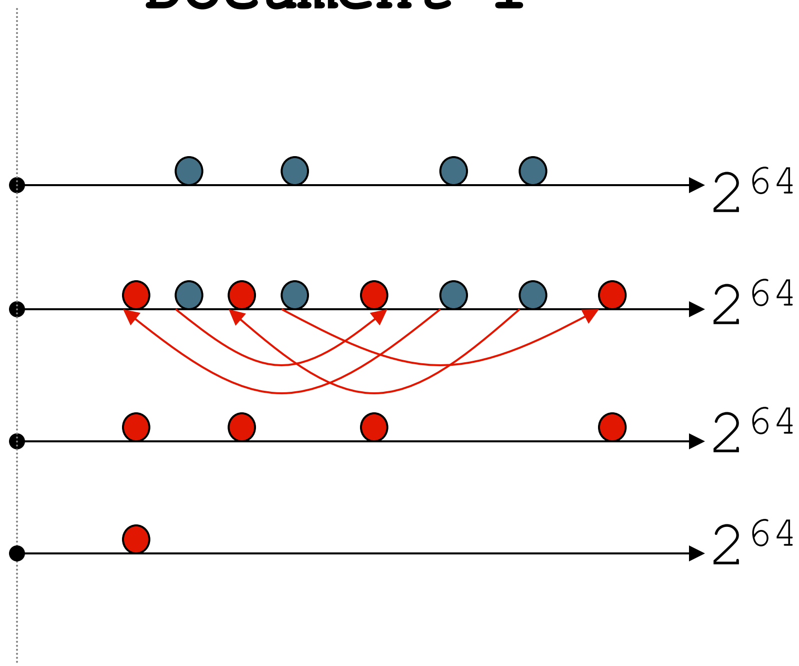


Sketch of a document

- 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, $\text{sketch}_D[\, i \,]$ is as follows:
    - Let f map all shingles in the universe to $[0, 2^m-1]$ (e.g., f = fingerprinting)
    - Let π_i be a *random permutation* on $[0, 2^m-1]$
    - Pick MIN $\{\pi_i(f(s))\}$ over all shingles *s* in *D*

Computing Sketch[i] for Doc1

Document 1

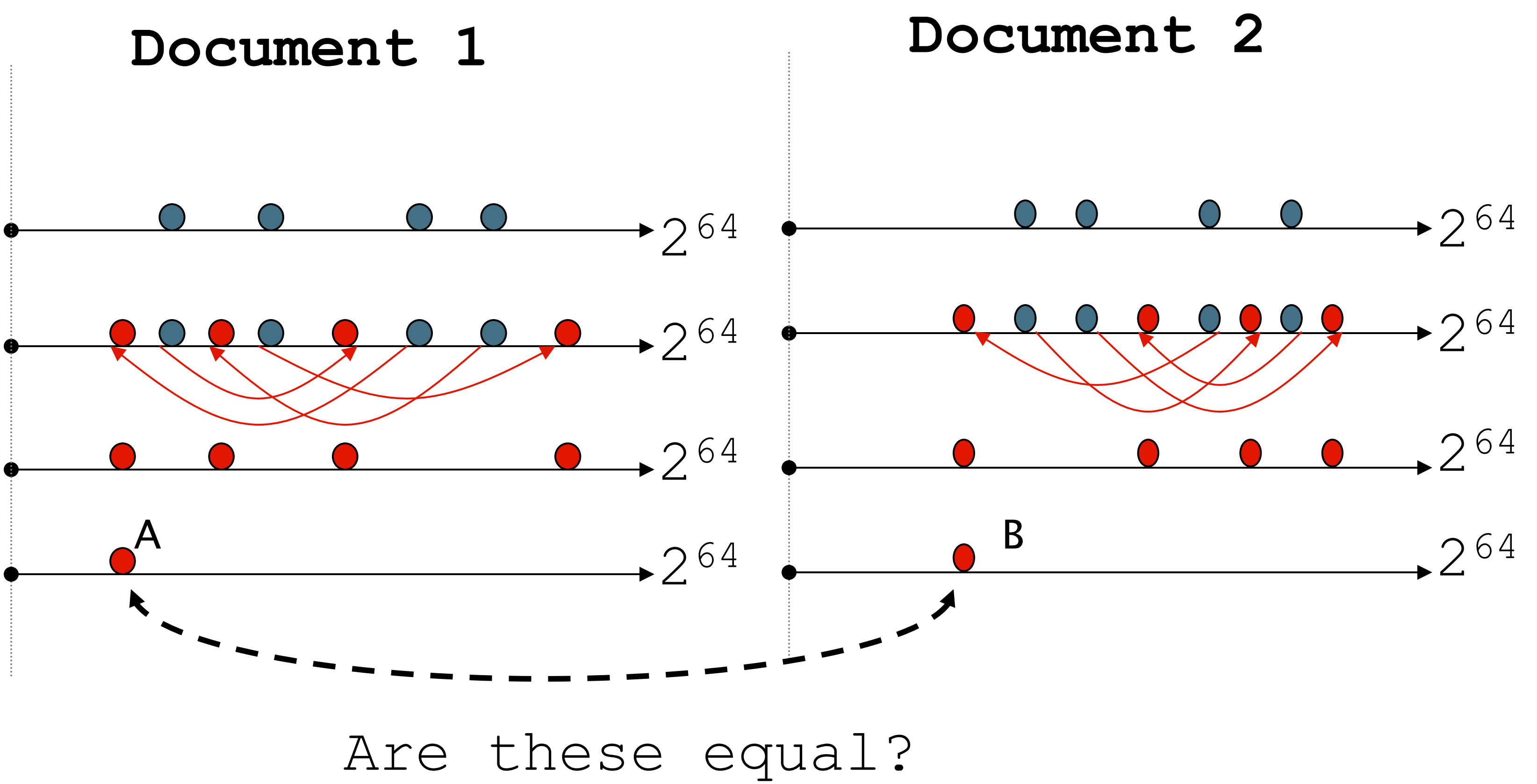


Start with 64-bit *f*(shingles)

Permute on the number line with π_i

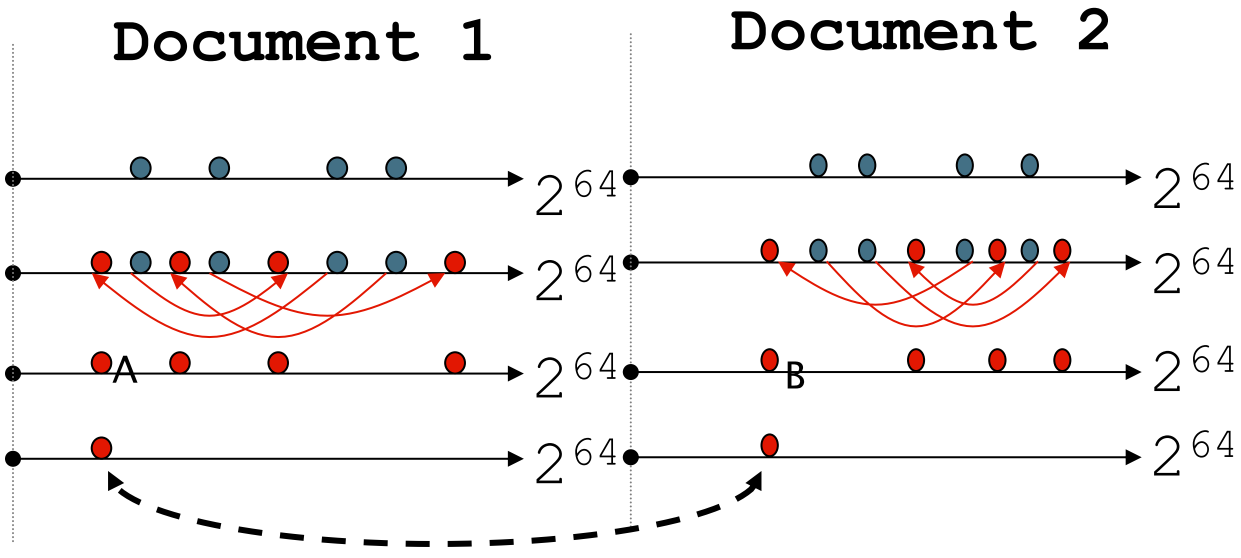
Pick the min value

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



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

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)

Why?

Claim: This happens with probability

Size_of_intersection / Size_of_union

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
0	1
1	0
1	1
0	0
1	1
0	1

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
type A	1	1
type B	1	0
type C	0	1
type 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

$$\Pr\left[\, h(C_i) = h(C_j) \,\right] = \text{Jaccard}\left(C_i, C_j\right)$$

- Why?
  - Both are A/(A+B+C)
  - Look down columns C_i, C_j until first non-Type-D row
  - $h(C_i) = h(C_j) \leftarrow\rightarrow$ type A row

Min-Hash sketches

- Pick *P* random row permutations
- MinHash sketch

sketch(C) = list of *k* indexes of first rows with 1 in column C

this is a random variable

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

Practical Implementation

- Random permutation is hard to obtain; simulate them using universal hashing instead
  - h: {0, 1, 2, …, U} → {0, 1, 2, …, M}
  - h(x) = ((a*x + b) mod P) mod M
  - where
    - P >> U and is a prime number
    - a, b are two randomly chosen integers modulo P and a != 0
  - $\text{sketch}(C) = \{ \text{argmin}_{e \in C} \{ h_i(e) \} \mid 1 \leq i \leq k \}$

Example

	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

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

Example Using the Universal Hashing

$h(x) = (7x+1 \bmod 31) \bmod 9$
$g(x) = (17x+8 \bmod 31) \bmod 9$

	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

$S_1 = \{R_1, R_3, R_4\}$
$h(e) = \{8, 4, 2\}$ ➔ $min\_elem = R_4$
$g(e) = \{7, 1, 5\}$ ➔ $min\_elem = R_3$
$sketch(S1) = \{R_4, R_3\}$

$S_2 = \{R_2, R_5\}$
$h(e) = \{6, 5\}$ ➔ $min\_elem = R_5$
$g(e) = \{2, 0\}$ ➔ $min\_elem = R_5$
$sketch(S1) = \{R_5, R_5\}$

Therefore, estimated similarity between S_1 and S_2 is $0/2 = 0.0$

Note: this example results in different sketches from the previous slide

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