CS6322: Information Retrieval
Sanda Harabagiu

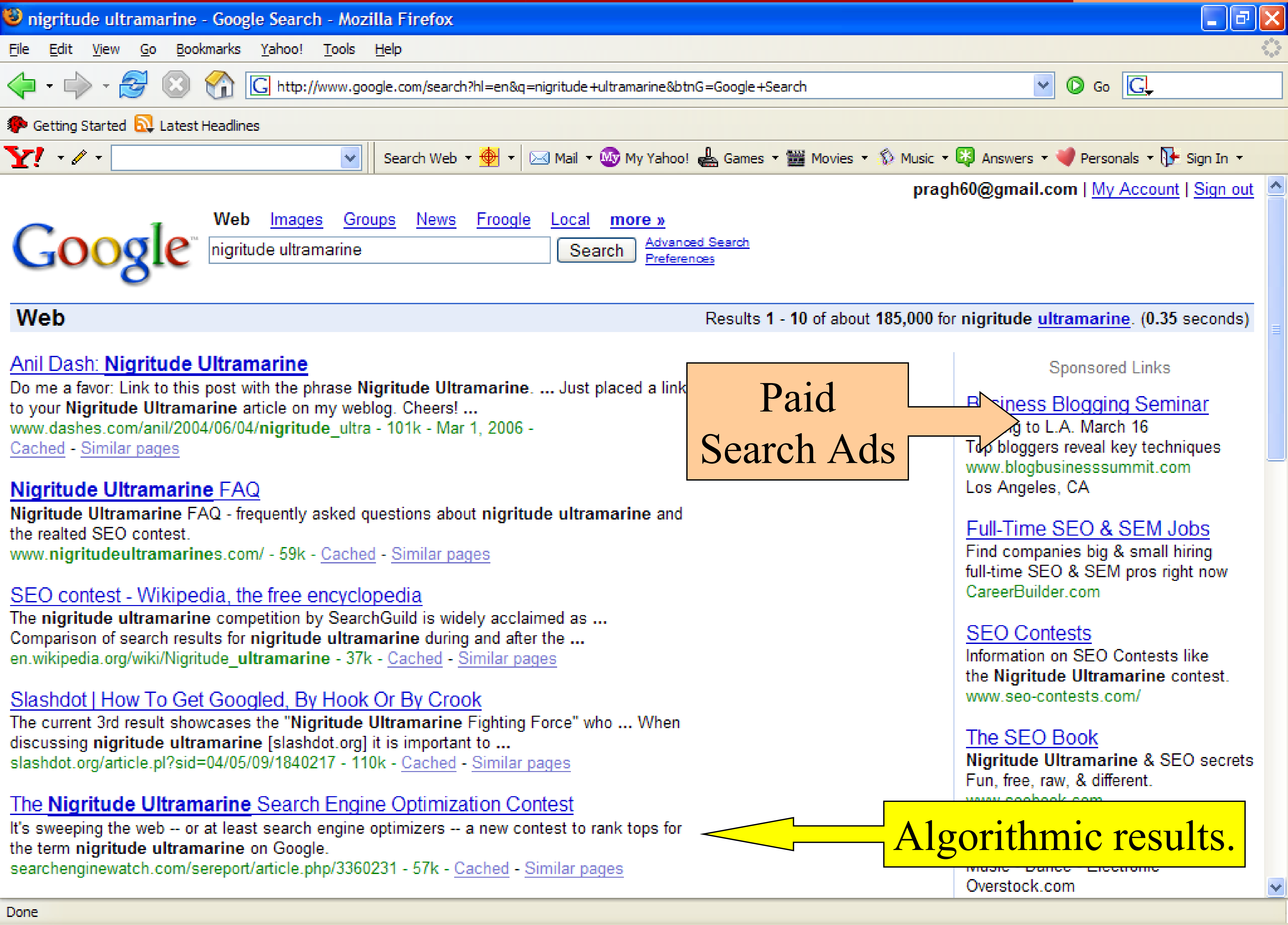
Lecture 8: Web search basics

Brief (non-technical) history

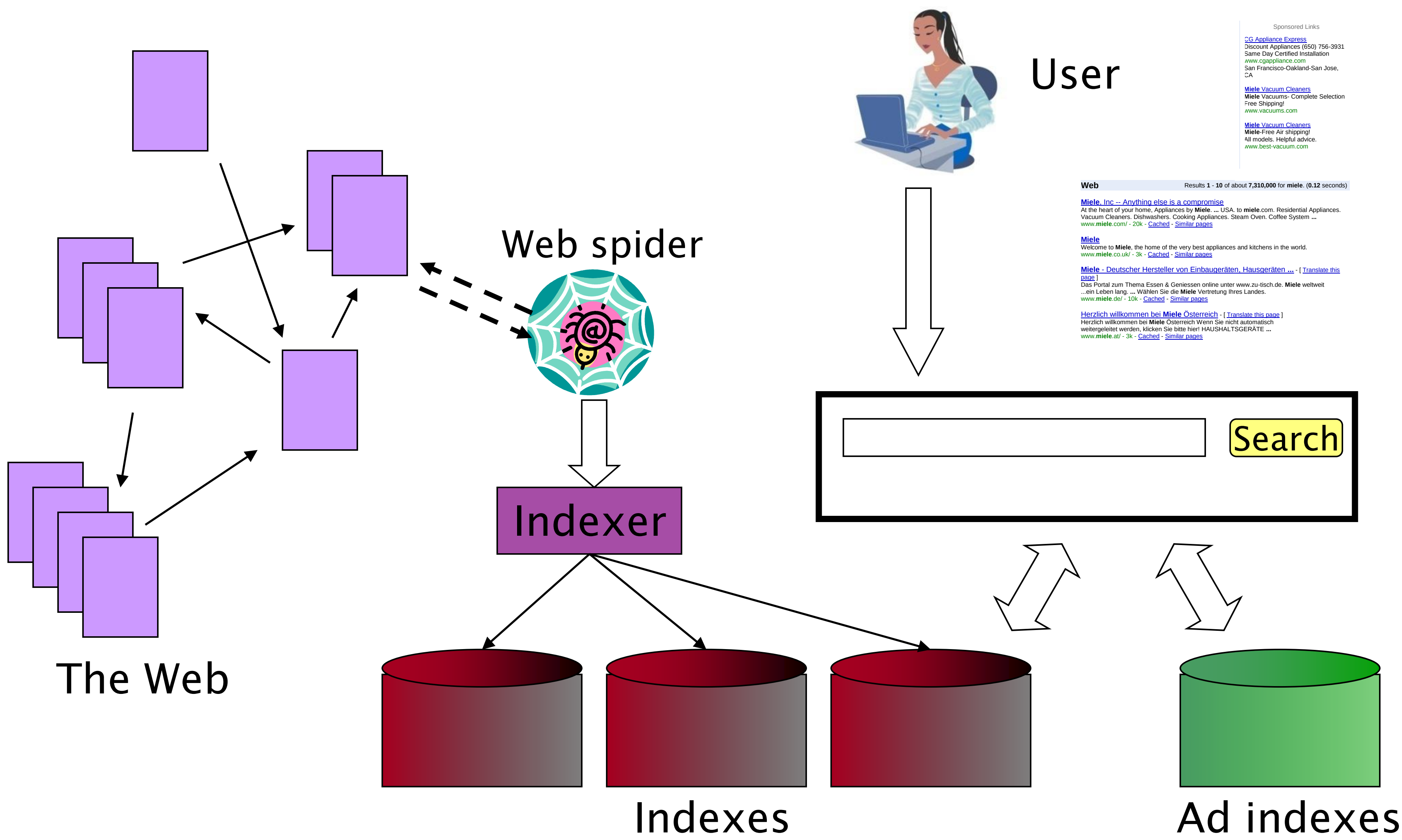
- Early keyword-based engines ca. 1995-1997
  - Altavista, Excite, Infoseek, Inktomi, Lycos
- Paid search ranking: Goto (morphed into Overture.com → Yahoo!)
  - Your search ranking depended on how much you paid
  - Auction for keywords: ***casino*** was expensive!

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
to L.A. March 16
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
    - Downloads
    - Shop

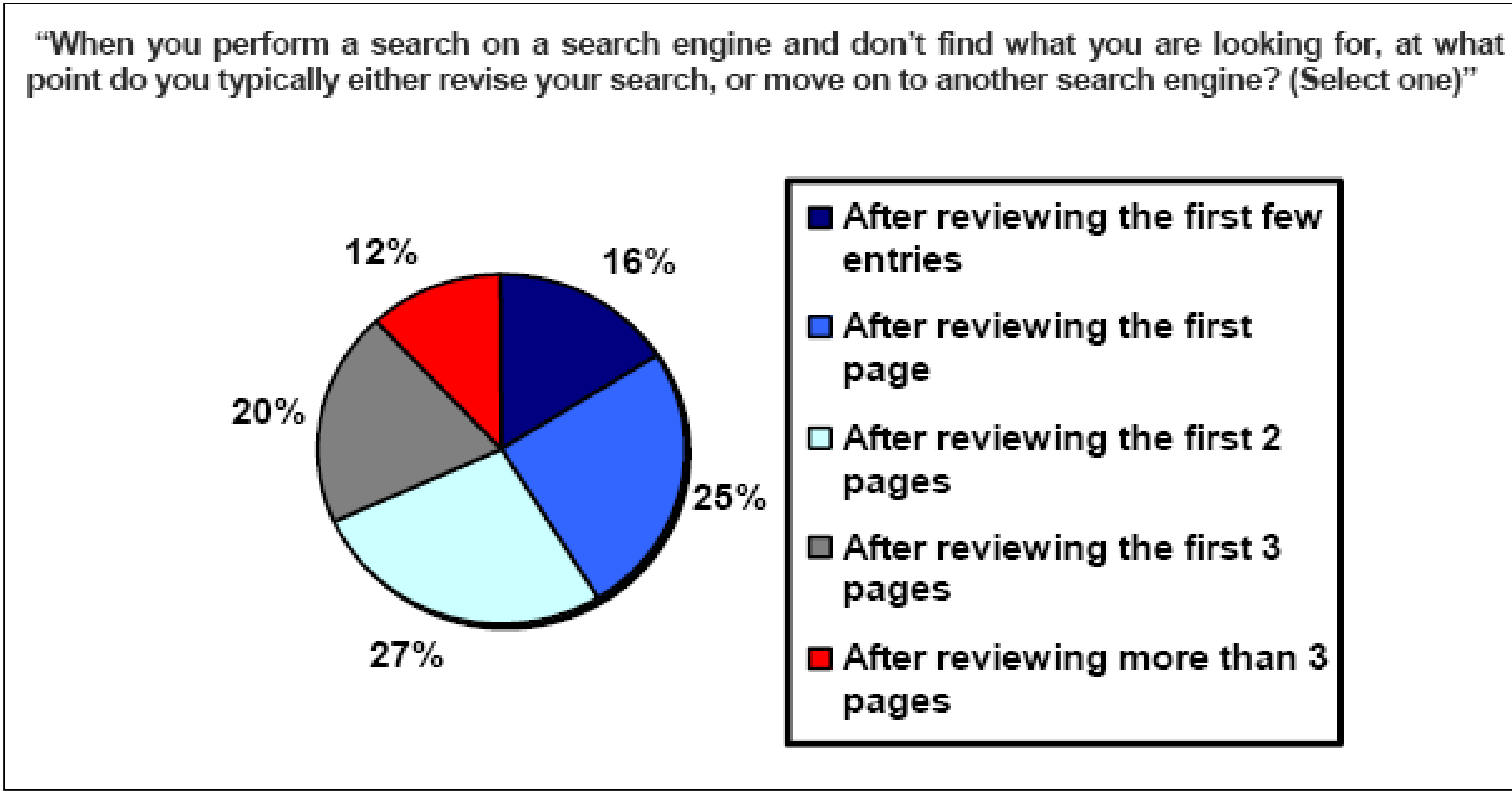
    Seattle weather

    Mars surface images

    Canon S410
  - **Gray areas**
    - Find a good hub
    - Exploratory search "see what's there"

    Car rental Brasil

How far do people look for results?



(Source: iprospect.com WhitePaper_2006_SearchEngineUserBehavior.pdf)

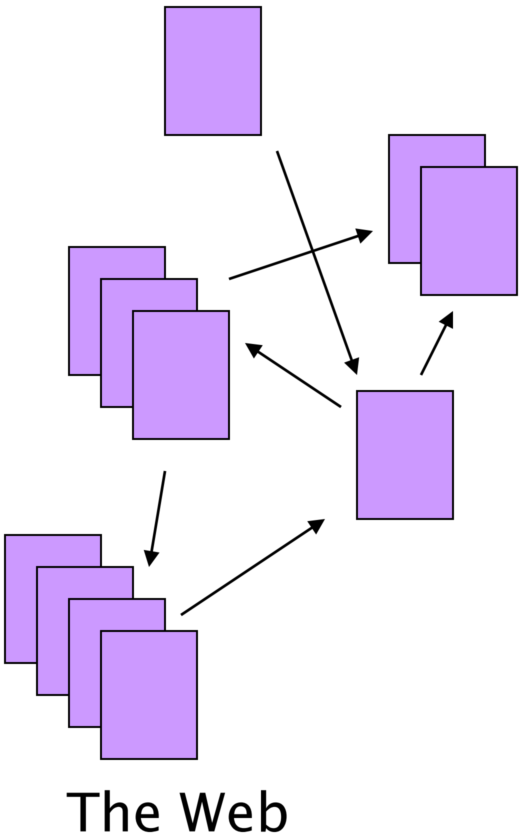
Users' empirical evaluation of results

- 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)

The trouble with paid search ads …

- 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

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 ☺

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

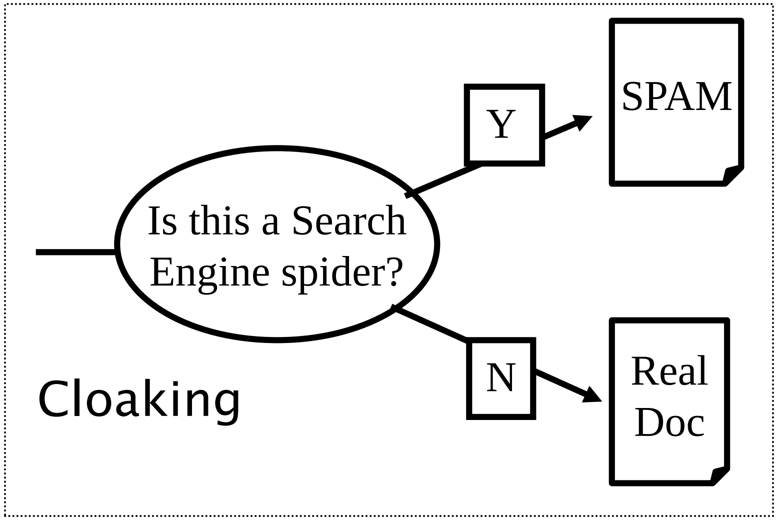
Variants of keyword stuffing

- 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, …"

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

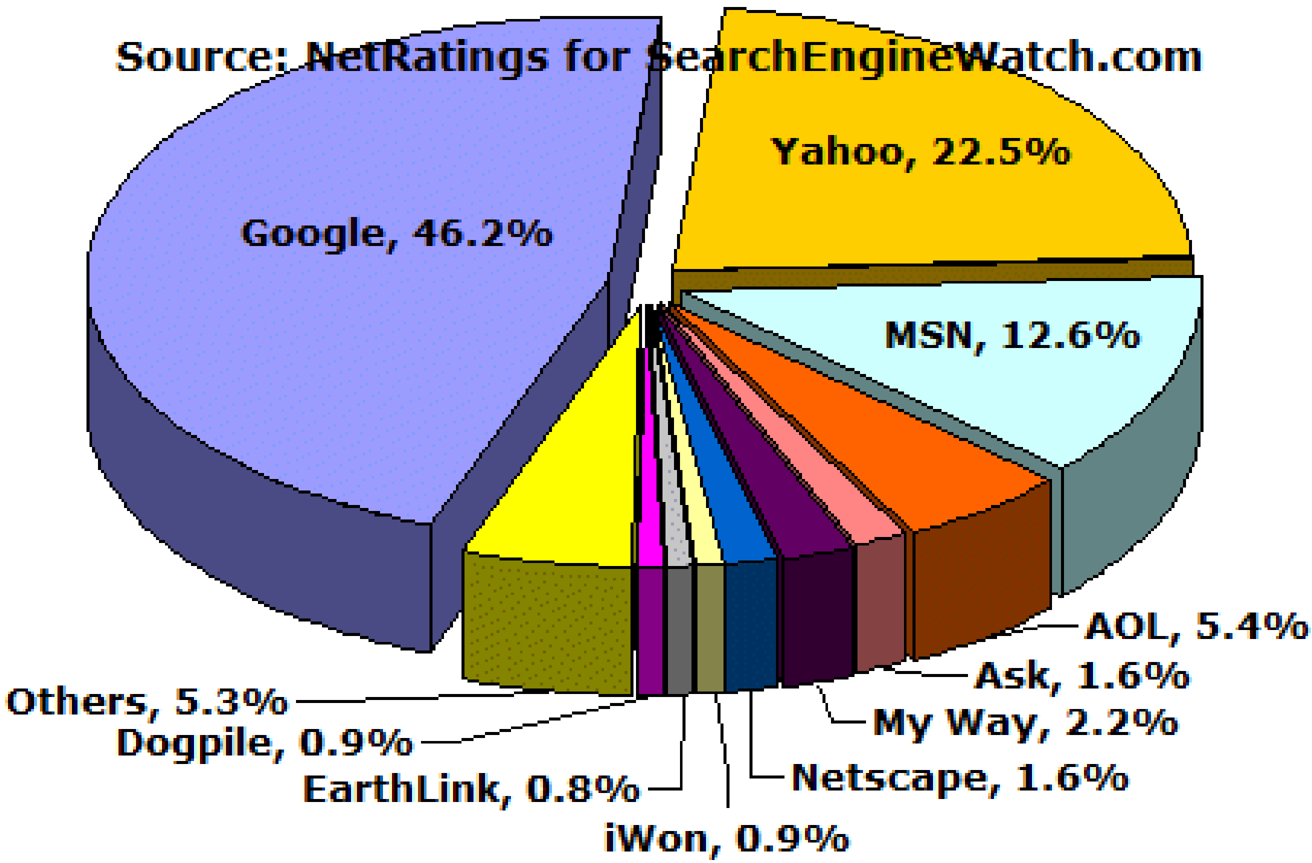
- Web search engines have policies on SEO practices they tolerate/block
  - http://help.yahoo.com/help/us/ysearch/index.html
  - http://www.google.com/intl/en/webmasters/
- Adversarial IR: the unending (technical) battle between SEO's and web search engines
- Research http://airweb.cse.lehigh.edu/

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.

Nielsen NetRatings Search Engine Ratings - July 2005



The Nielsen NetRatings MegaView Search reporting service measures the search behavior of more than a million people worldwide. These web surfers have real-time meters on their computers which monitor the sites they visit.

More statistics on searchenginewatch.com

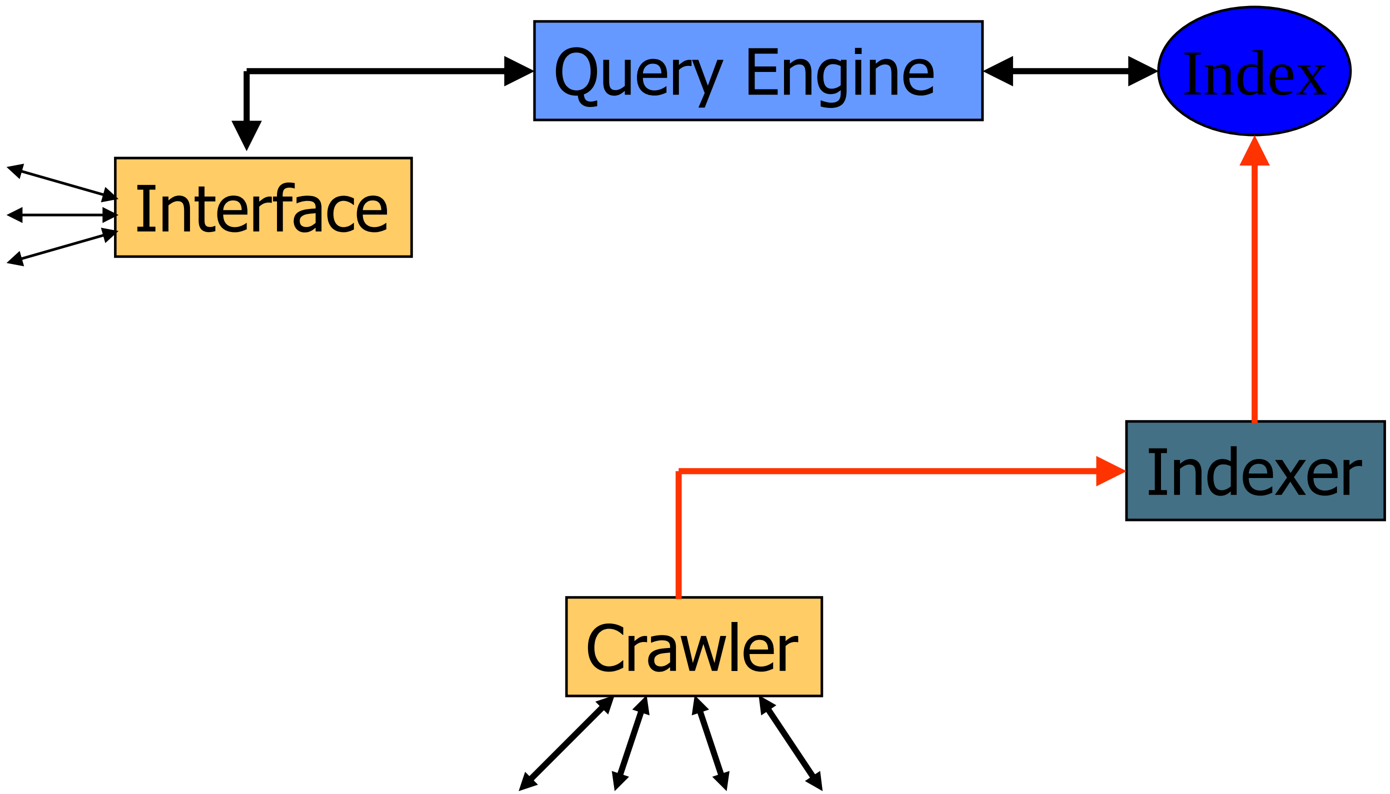
Web Challenges

- Distributed data
- High percentage of volatile data
- Large volume
- Unstructured and redundant data
- Quality of data
- Heterogeneous data

Several architectures for search engines

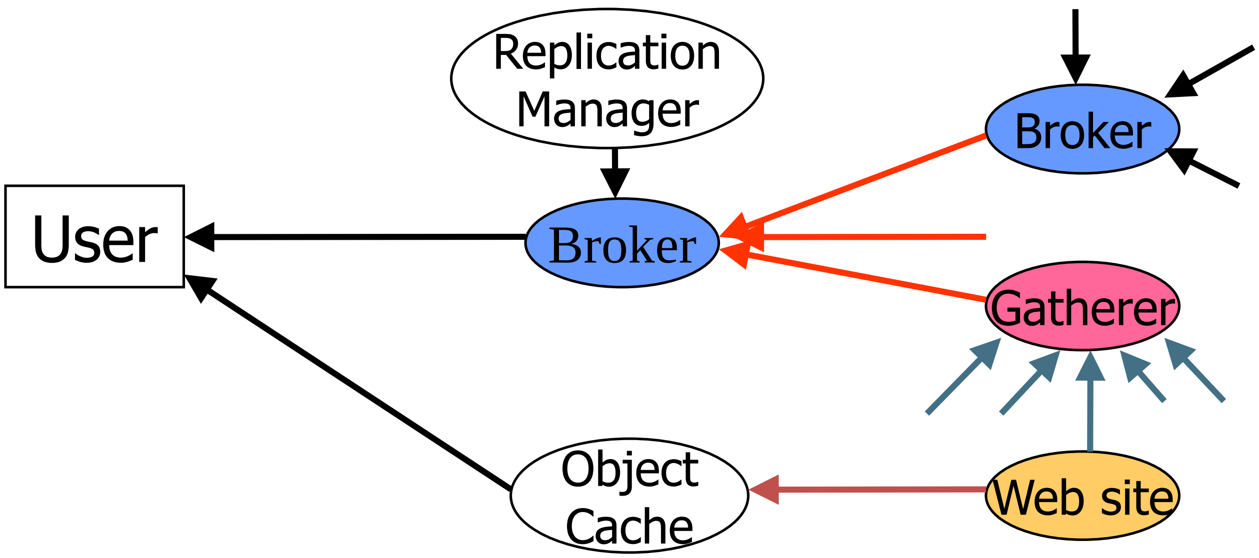
→ difference from standard IR systems: all queries must be answered <u>without</u> accessing the text.

Centralized Architecture



Distributed Architecture

- The Harvest distributed architecture requires the coordination of several Web servers



Gatherers = collect and extract indexing information from one or more web servers

Brokers = provide the indexing information and the query interface to the data gathered

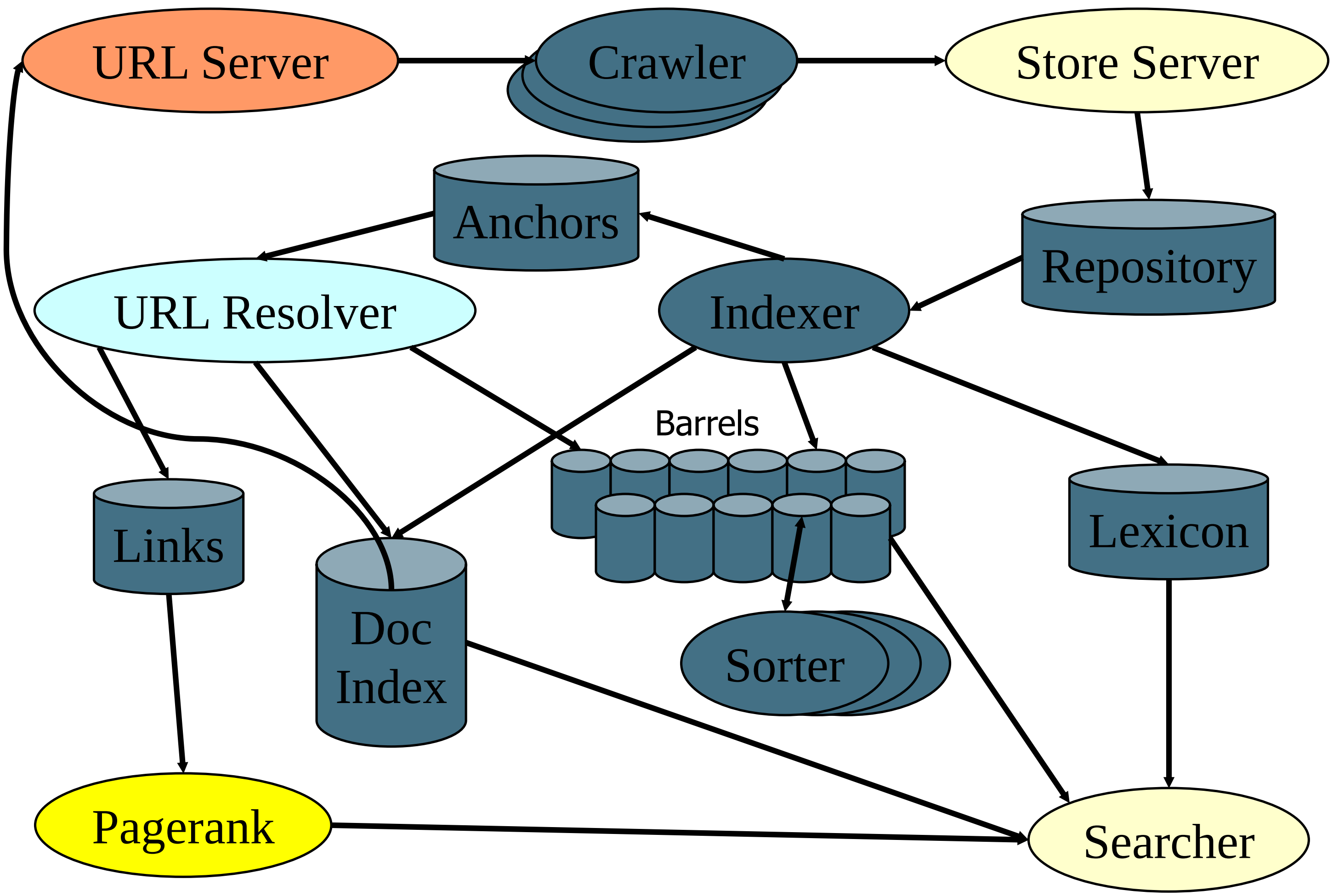
Search Engines

- Google case (1997)
  - Name is a common spelling of *googol* 10^{100}
  - Goal: build a very large-scale search engine
  - Index : 1997 100million; 2002 2 billion
  - In 1997 their forecast was that in 2000 they will have 1 billion page index (they had 2 billion pages!!!)
- World Wide Web Worm (1994)
  - Index 1994 110,000 web pages

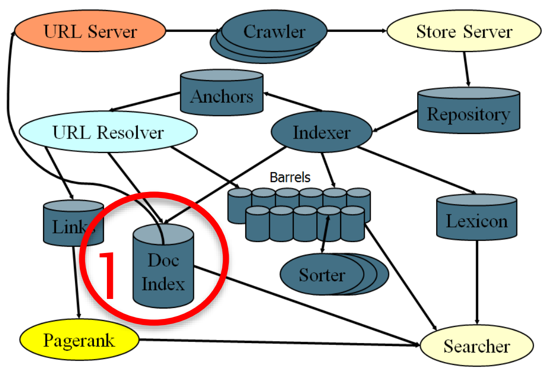
Web Search Engines: **Google**

- *Hyperlinks* represent new forms of information
- *Hypertexts* have existed and been studied for a long time
- New → a large number of hyperlinks created by independent individuals
- Hyperlinks provide a valuable source of information for Web IR → Link analysis

Google Anatomy



Google Anatomy

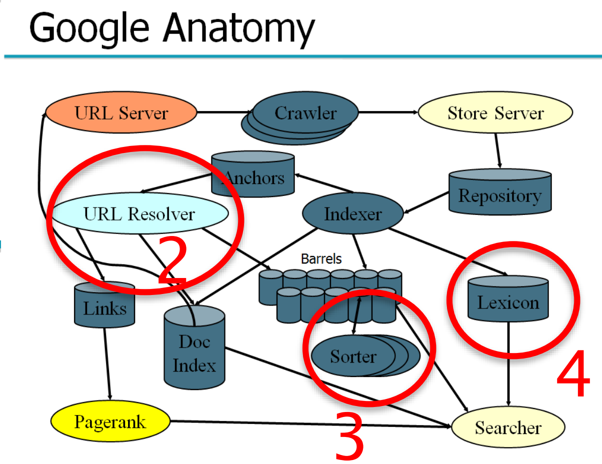


Google Indexer

Functions

- Reads the repository
- Uncompress the documents, parse them
- Each document is converted into a set set of word occurrences →hits
  - 1 hit records [word, position in document, font size, capitalization]
- The hits are distributed in a set of "barrels" partially sorted forward index
- Parses all links and stores info about them in anchors file:
  - <*where the links point from*, *to*, *text of link*>

URL Resolver for *Google*



Reads the anchors file

- Converts relative URLs in absolute URLs
  - Translate them in docIDs
- Puts the anchor text into the forward index, associated with the docID that the anchor points to
- Generates a database of links = pairs of docIDs

Sorter – takes the barrels (sorted by docID) and resorts them by wordID to generate the inverted index.

- Sorting in place → little temporary space is needed.

Dump Lexicon – a program that takes the inverted list and the lexicon produced by the indexer → generates new lexicon from searcher.

Major Data Structures in Google

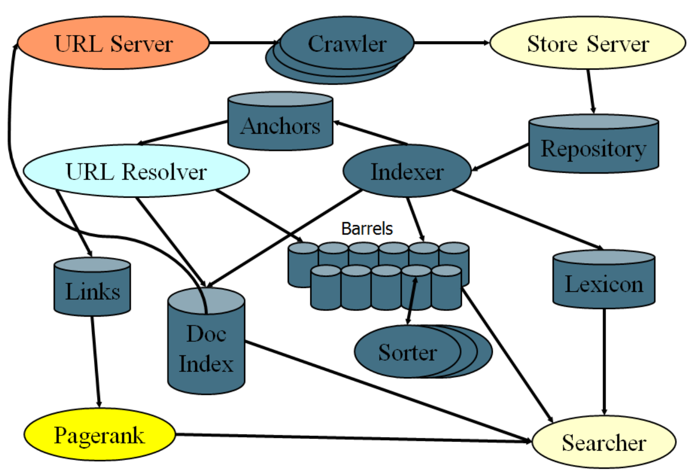
Optimized for crawling/indexing/searching large document collections.

Trick → avoid disk seeks (10ms per seek)

→ influences the design of data structures

- Big files
- Repository
- Document Index
- Lexicon
- Hit list
- Forward index
- Inverted index

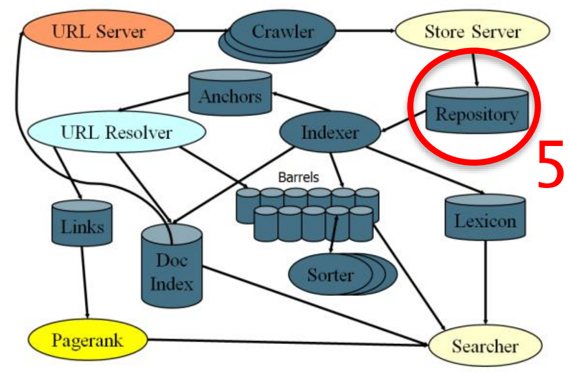
Google Anatomy



Big Files

- Virtual files spanning multiple file systems → addressable by 64 bit integers
- Allocation among multiple file systems is handled automatically
  - Allocates and de-allocates descriptors
  - Compression options

Google Anatomy



Repository

- Contains the full HTML of every web page
- Compression using zlib

Repository: 53.5 GB = 147.8 GB uncompressed

sync	length	Compressed packet
sync	length	Compressed packet

…

docid	ecode	urllen	pagelen	url	page

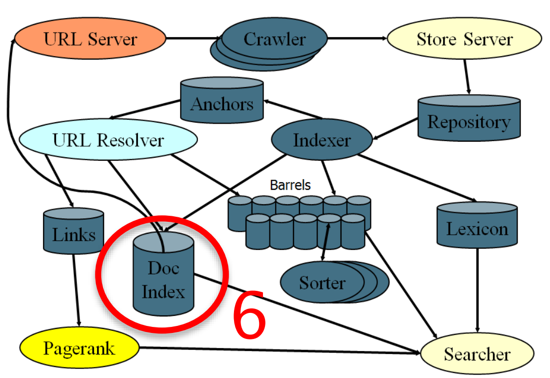
Packet (stored compressed in repository)

Document Index

- Information about each document
- Solution: fixed width ISAM (Index sequential access mode) index, ordered by docID
- Each query:

current document status	Pointer to repository	Document checksum	statistics

Google Anatomy



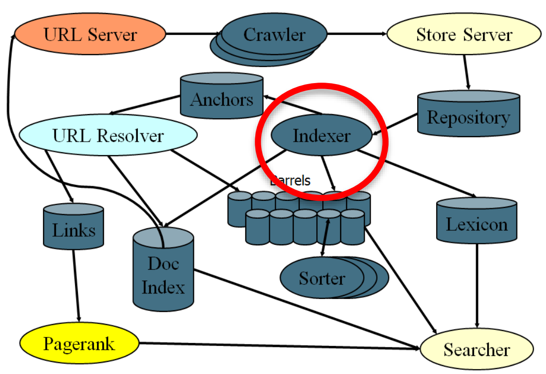
Lexicon: 14 million of words

- ❑ List of words + hash table of pointers
- ❑ __Sanda_Harabagiu_research__

Hit Lists

- Account for most of the space used in both the forward and inverted indices
- It contains 2 forms of information:
  1. documents where a word occurred,
  2. position in document and font info
- Encoding position, font, capitalization
  - Simple encoding (3 integers)
  - Compact encoding (hand-optimized allocation of bits)
  - Huffman encoding
- Select – hand optimized compact encoding
  - Less space than simple encoding
  - Less bit manipulation than Huffman coding

Google Anatomy



Google Compact Coding

- Uses 2 bytes for every hit
- There are two types of hits: fancy hits and plain hits:
  1. Fancy lists → hits occurring in a URL, title, anchor text or meta tag
  2. Plain hits → all the other

The codification:

A plain hit has

- 1 capitalization bit
- font size (3 bits)
- 12 bits of word position in document (positions higher than 4096 are labeled 4096)
- For font size → only 7 values, since 111 signals a fancy bit

Google Compact Coding - cont

A fancy hit has

- 1 capitalization bit
- Font size set to 7
- 4 bits encode the type of fancy hit
- 8 bits for position
- For anchor hits the 8 bits for position are split:
  - 4 bits for position anchor
  - 4 bits for hash of the docID the anchor occurs in

Hit: 2 bytes

Cap:1	Imp:3	Position:12		
Cap:1	Imp=7	Type:4	Position:8	
Cap:1	Imp=7	Type:4	Hash:4	Pos:4

Forward Index

- It is partially sorted
- Stored in 64 barrels. Each barrel holds a range of word Ids
- If a document contains words that fall in a particular barrel, the docID is recorded into the barrel followed by a list of word Ids with hit-lists corresponding to the words

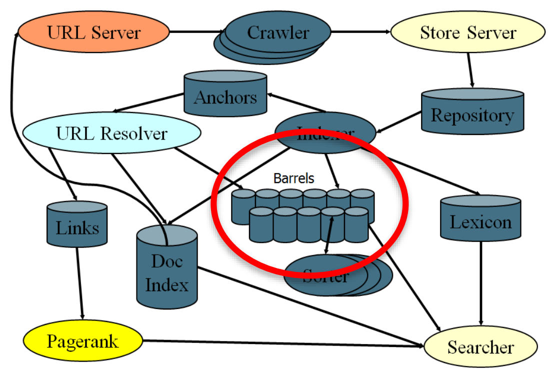
Forward barrels: total 43 GB

docid	Wordid:24	Nhits:8	Hit hit hit hit
	Wordid:24	Nhits:8	Hit hit hit hit
	Null wordid		
docid	Wordid:24	Nhits:8	Hit hit hit hit
	Wordid:24	Nhits:8	Hit hit hit hit
	Wordid:24	Nhits:8	Hit hit hit hit
	Null wordid		

Forward Index -cont

- DocIds are duplicated –reasonable for 64 buckets but saves considerable time and coding complexity
- Instead of storing actual word Id, store relative difference to minimum word ID in the barrel. Use only 24 bits for word ID.
- 8 bits → hit list length

Google Anatomy



Inverted Index

- On the same barrels as forward index
- For every valid wordID – there is a pointer in the lexicon to the barrel that the wordID falls into:
  - Barrel points to a doclist of docIDs + corresponding hit lists.

- Issue: in what order the docIDs should appear in the doclist ?
  - ***Solution 1***: store them sorted by docID – quick mapping for multiple word queries
  - ***Solution 2***: store them sorted by a ranking of occurrence of the word in each document – merging is difficult,
  - ***Solution (Google):*** compromise: 2 sets of inverted barrels:
    - 1 set for hit lists which include title and anchor hits
    - 1 set for the other hit lists.
    - Check the first set – if not enough matches, check the 2nd set

Crawling the Web

Running a Web Crawler is a challenging task

- Crawling is the most fragile application – it involves interacting with hundreds of thousands of Web servers and various name servers → beyond the control of the system
- Google solution: fast distributed crawling system
  - The URL server serves lists of URLs to 3-5 crawlers (both server and crawlers are implemented in Python)
  - Each crawler keeps 300 connections open at once to retrieve Web pages at a fast enough pace.
  - 100 web pages/sec ≈ 600k/sec

Google crawler

- Performance stress: DNS lookup
- Each crawler maintains its own DNS cache (it does not need to do a DNS lookup before crawling each document)
- Each of the 300 connections can be in a different state: Lookup DNS, Connect to host, Send request or Receive response
- **The Robot Exclusion Standard**

  (devised in 1994 by Martin Kostner)

  - Declares that a Web server administrator should create a document accessible at the relative URL /robots.txt
  - E.g.

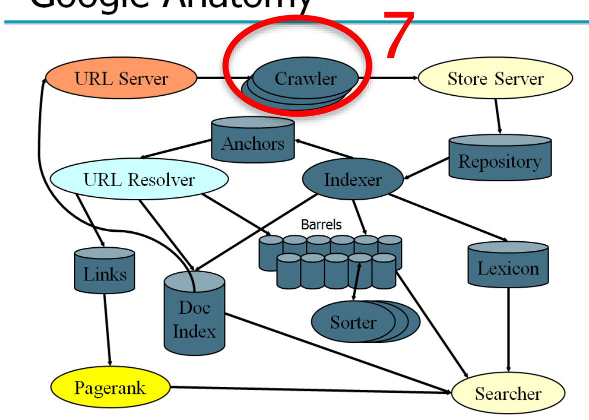
    User-agent: *

    Disallow: /cgi

    Disallow: /stats

  Try: www.utdallas.edu/robots.txt

Google Anatomy



The robots.txt file

- 3 basic directives can be in robots.txt: User-agent, Allow and Disallow
- If the robots.txt file contains:

```
User-agent: *
Disallow:   /
```

the administrator wants to shut out all robots from the entire site

- Multiple User-agents can be specified:

```
User-agent: friendly indexes
User-agent: search-thingy
Disallow: /cgi-bin/
Allow: /
```

- Another example:

```
User-agent: *
Disallow: /
User-agent: search-thingy
Allow: /
```

➔ all robots should go away, except the search-thingy robot

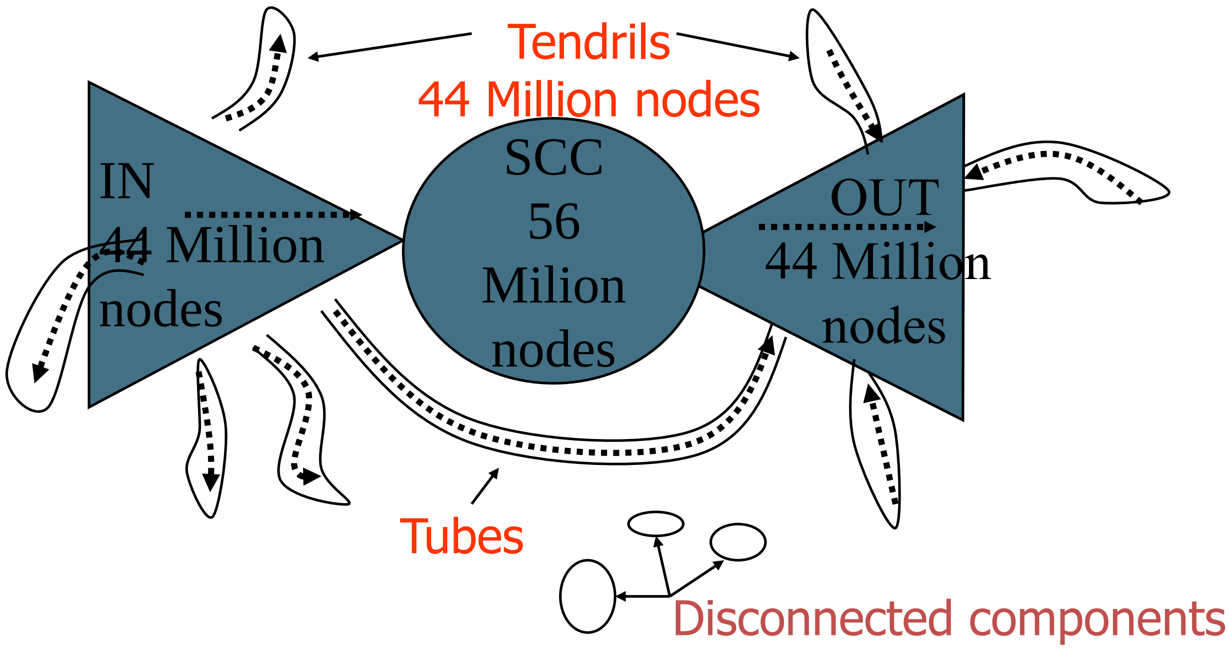
Google Initial Performance

Storage Statistics	
Total size of Fetched Pages	147.8 GB
Compressed Repository	53.5 GB
Short Inverted Index	4.1 GB
Full Inverted Index	37.2 GB
Lexicon	293 MB
Temporary anchor data (not in total)	6.6 GB
Document Index Incl. Variable Width Data	9.7 GB
Links Database	3.9 GB
Total Without Repository	55.2 GB
Total With Repository	108.7 GB

Web Page Statistics	
Number of Web Pages Fetched	24 million
Number of URLs seen	76.5 million
Number of Email addresses	1.7 million
Number of 404's	1.6 million

The Web Macroscopic Structure

- AltaVista experiments



90% are in a single, giant Strongly Connected Component (SCC)
IN = pages that can reach SCC, but cannot be reached from it
OUT = pages reached from SCC, but cannot access SCC
TENDRILS = pages that cannot reach SCC and cannot be reached by SCC
All 4 sets have roughly the same size ➔ big surprise !

More Data

- The diameter of the central core of SCC is at least 28, the diameter of the graph is > 500
- The probability that a path exists from the source to the destination 0.24
  - If a directed path exists, its average length is 16
- The Power law for in-degree
  - The probability that a node has in-degree I is proportional to $1/i^x$ for some $x > 1$
    - $x = 2.1$ in the AltaVista May '99 Crawl

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.)
- The coverage of a search engine relative to another particular crawling process.

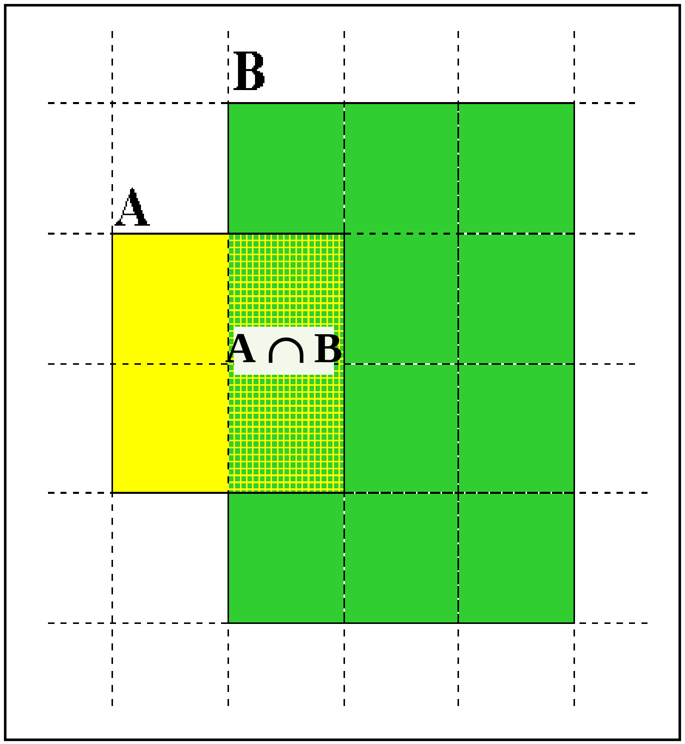
New definition?

(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
  - **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: $W(SE_A) / W(SE_B) \sim |SE_A| / |SE_B|$

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

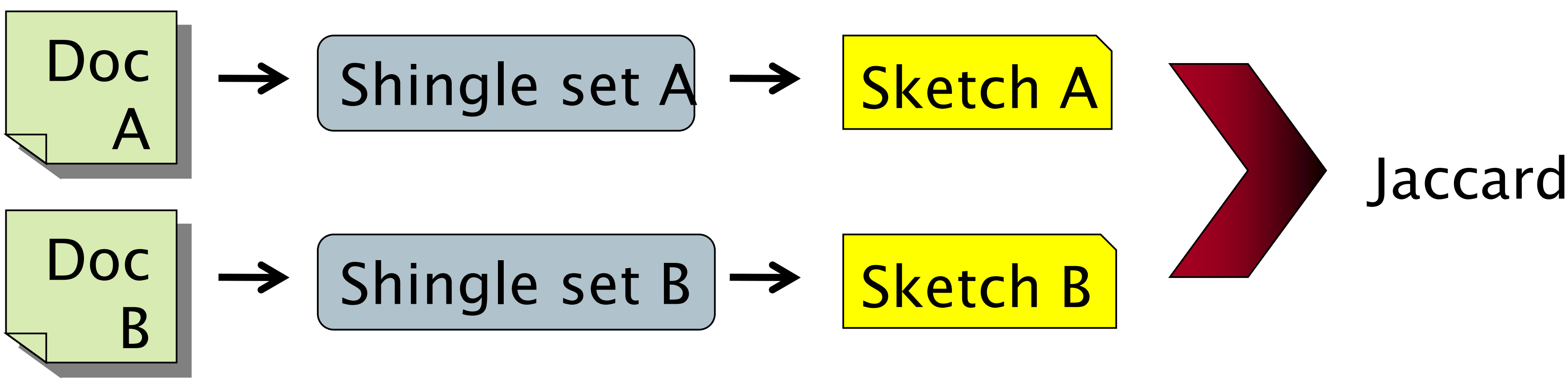
    rose_is_a_rose

    is_a_rose_is

    a_rose_is_a
- Similarity Measure between two docs (= <u>sets of shingles</u>)
  - Set intersection
  - Specifically (Size_of_Intersection / Size_of_Union)

Shingles + Set Intersection

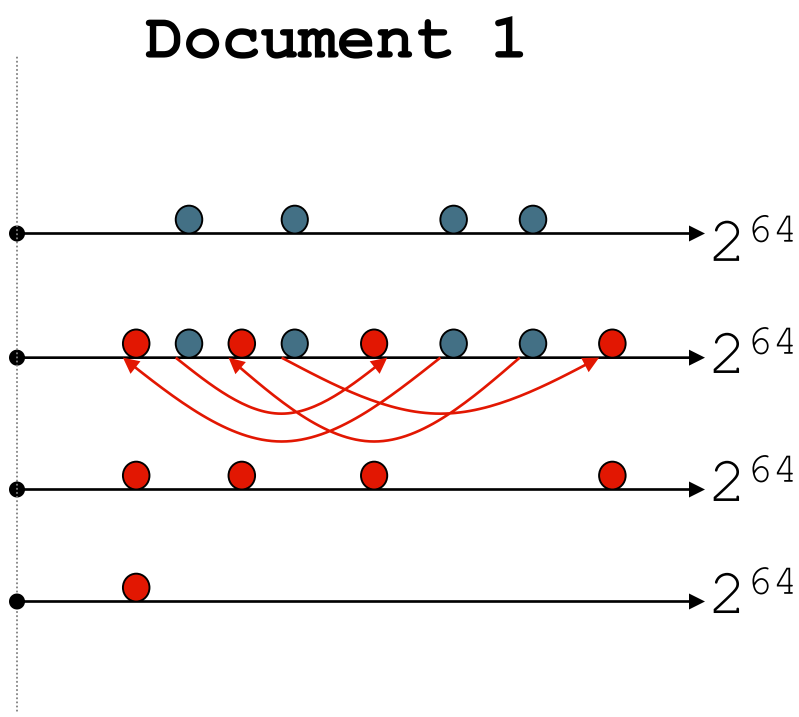
- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
  - Approximate using a cleverly chosen subset of shingles from each (a *sketch*)
- Estimate (size_of_intersection / size_of_union) based on a short sketch



Sketch of a document

- Create a “sketch vector” (of size ~200) for each document
  - Documents that share ≥ *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 $0..2^m$ (e.g., f = fingerprinting)
    - Let π_i be a *random permutation* on $0..2^m$
    - Pick MIN $\{\pi_i(f(s))\}$ over all shingles *s* in *D*

Computing Sketch[i] for Doc1

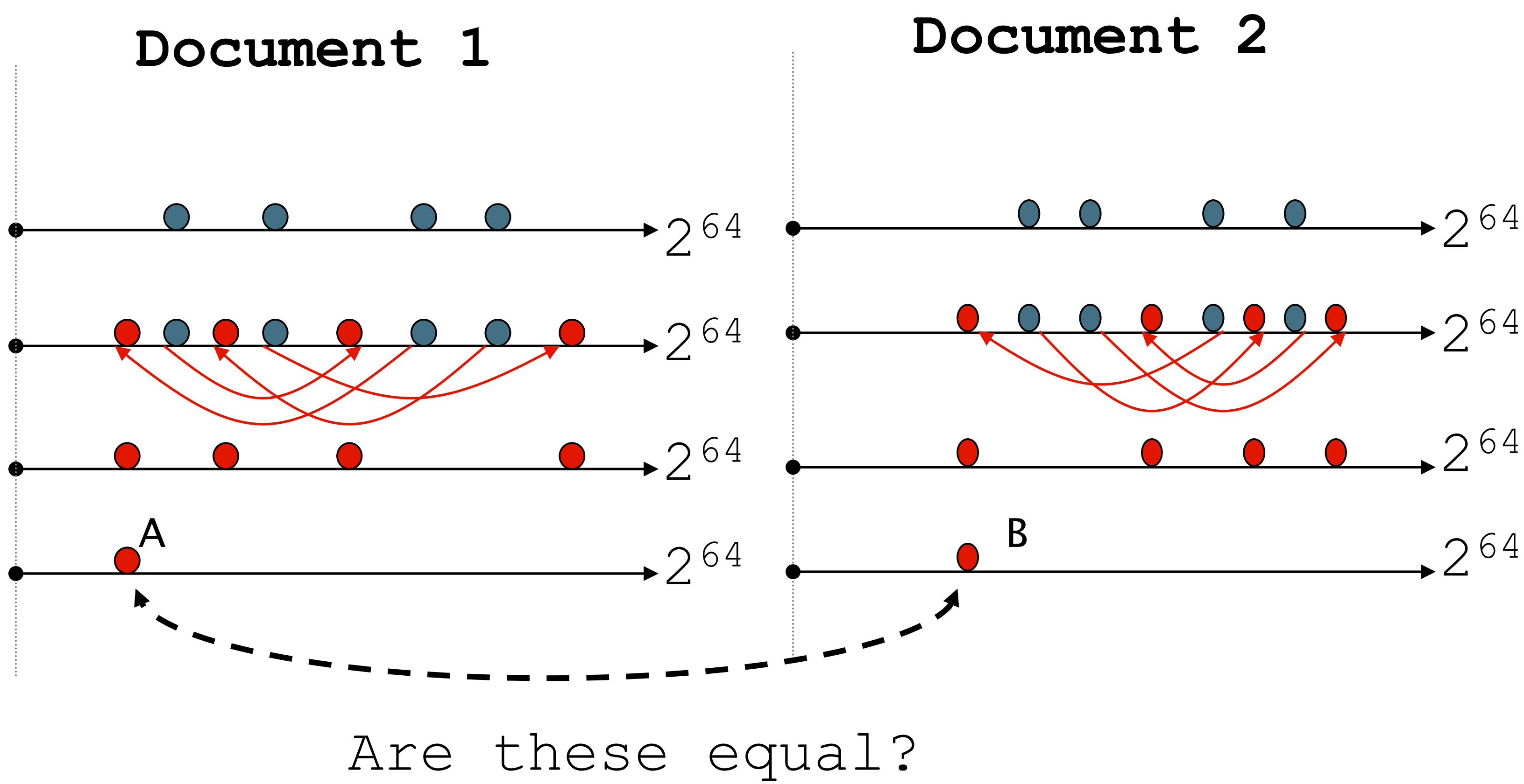


Start with 64-bit *f*(shingles)

Permute on the number line with π_i

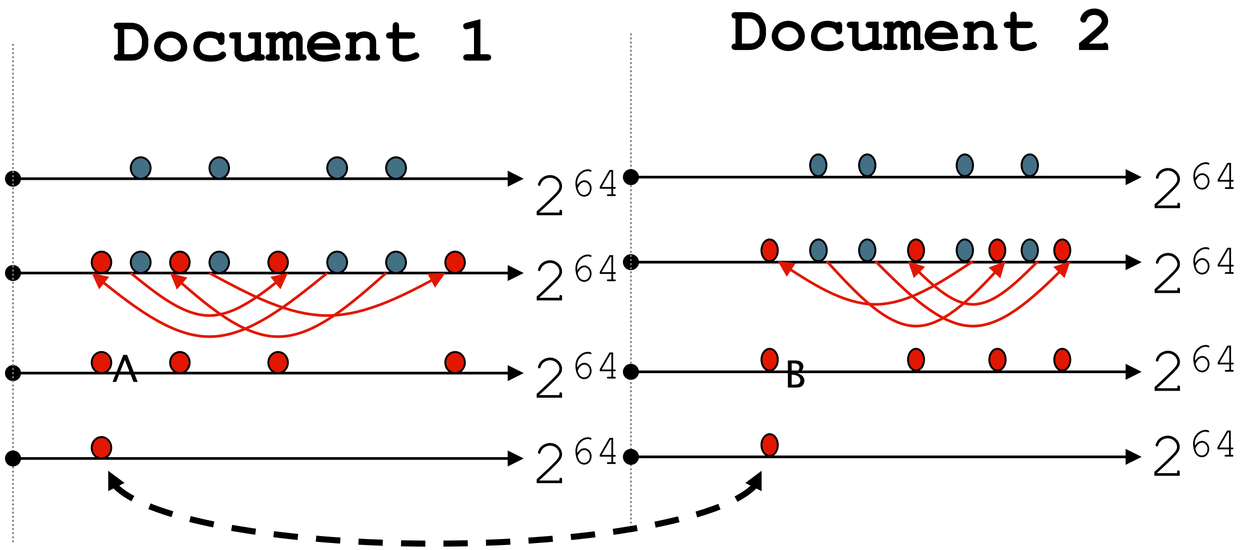
Pick the min value

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
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

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

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

Min-Hash sketches

- Pick P random row permutations
- MinHash sketch

Sketch$_D$ = list of P indexes of first rows with 1 in column C

- 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)

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	0.60	0.4
Sig-Sig	0.00	0.67	0.00

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 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
R_1	1	0
R_2	0	1
R_3	1	1
R_4	1	0
R_5	0	1

	C_1 slots	C_2 slots
$h(1) = 1$	1	-
$g(1) = 3$	3	-
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	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