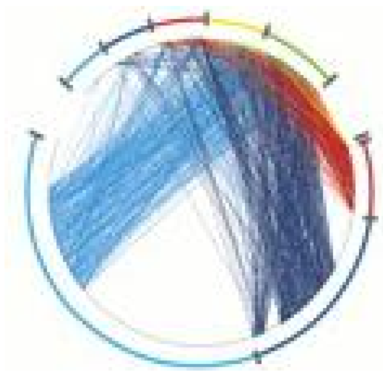
Models and Algorithms for Complex Networks

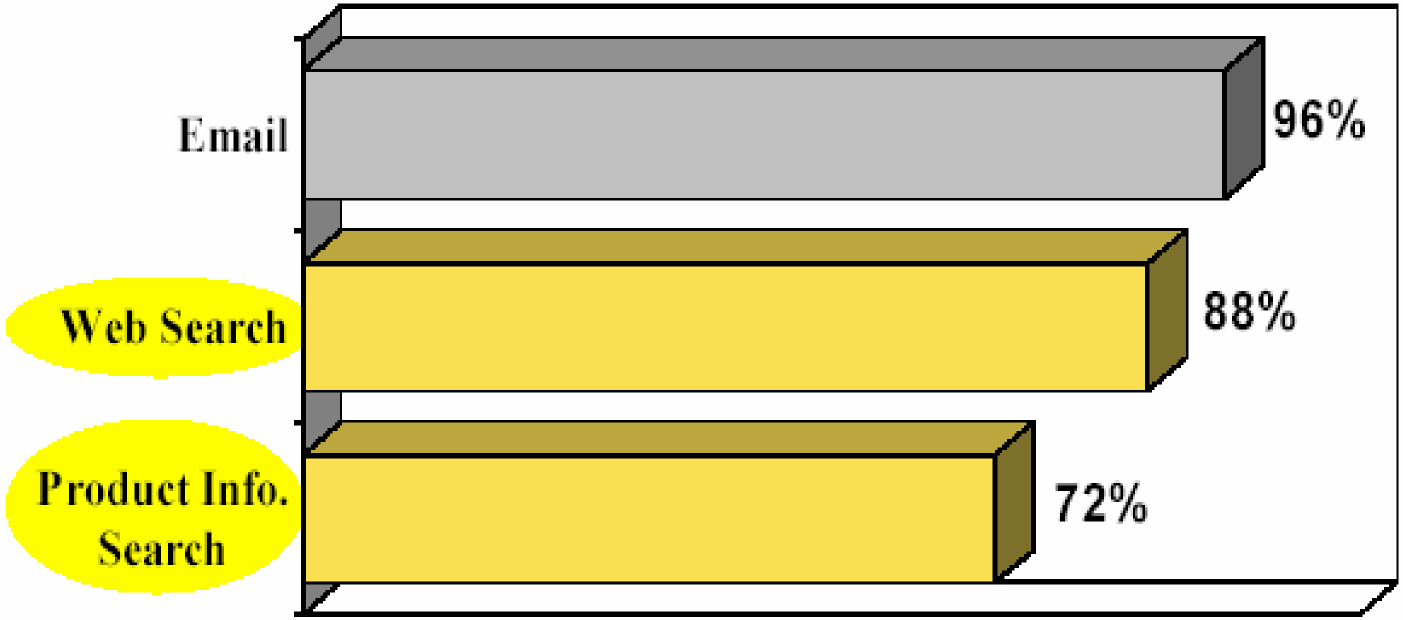
Searching the Web



Why Web Search?

- Search is the main motivation for the development of the Web
  - people post information because they want it to be found
  - people are conditioned to searching for information on the Web ("Google it")
  - The main tool is text search
    - directories cover less than 0.05% of the Web
    - 13% of traffic is generated by search engines
- Great motivation for academic and research work
  - Information Retrieval and data mining of massive data
  - Graph theory and mathematical models
  - Security and privacy issues

Top Online Activities



Feb 25, 2003: >600M queries per day

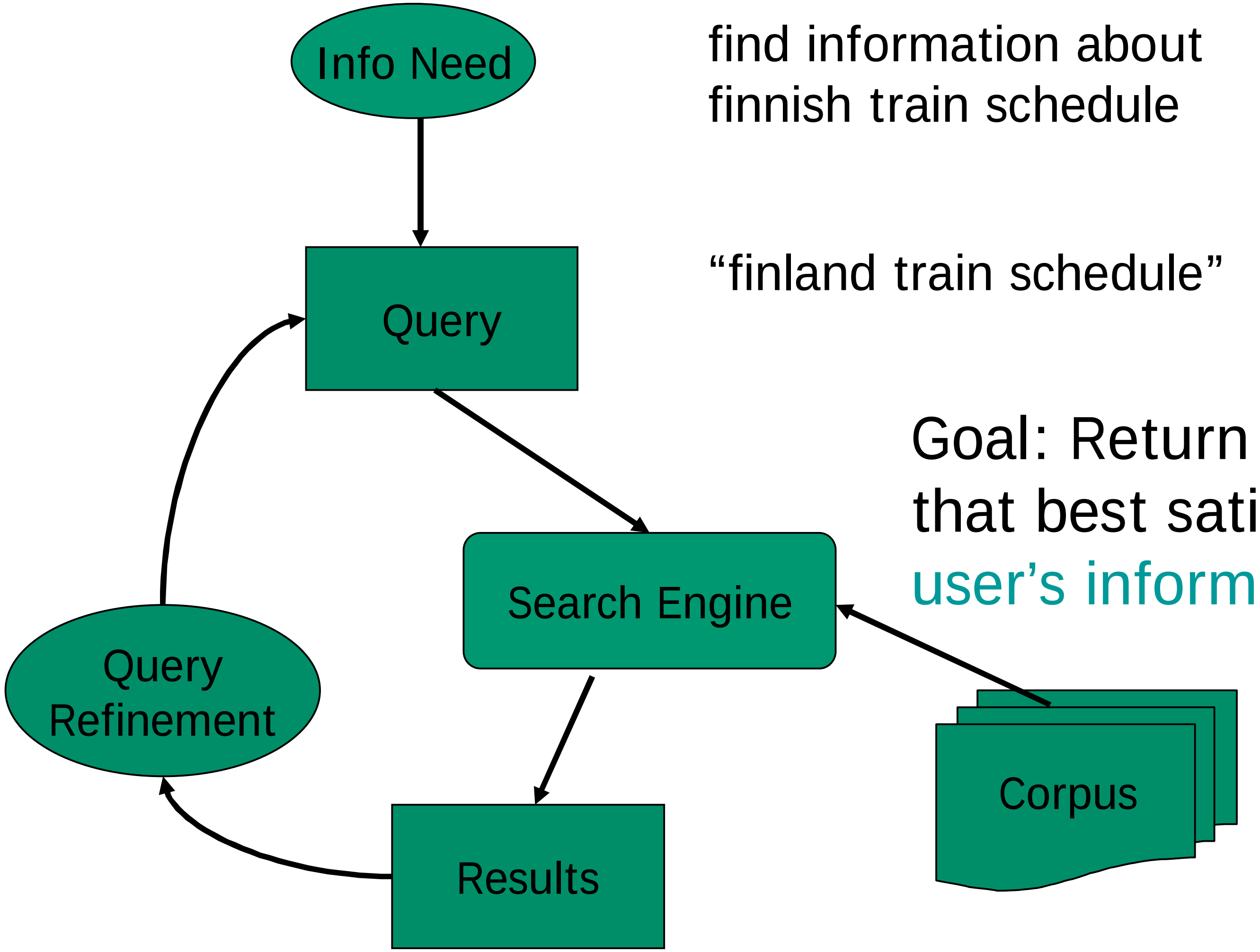
Outline

- Web Search overview
  - from traditional IR to Web search engines
- The anatomy of a search engine
  - Crawling, Duplicate elimination, indexing

... not so long ago

- Information Retrieval as a scientific discipline has been around for the last 40-50 years
- Mostly dealt with the problem of developing tools for librarians for finding relevant papers in scientific collections

Classical Information Retrieval



find information about
finnish train schedule

“finland train schedule”

Goal: Return the documents that best satisfy the user’s information need

Classical Information Retrieval

- Implicit Assumptions
  - fixed and well structured corpus of manageable size
  - trained cooperative users
  - controlled environment

Classic IR Goal

- Classic Relevance
  - For each query Q and document D assume that there exists a relevance score S(D,Q)
    - score average over all users U and contexts C
  - Rank documents according to S(D,Q) as opposed to S(D,Q,U,C)
    - Context ignored
    - Individual users ignored

IR Concepts

- Models
  - Boolean model: retrieve all documents that contain the query terms
    - rank documents according to some term-weighting scheme
  - Term-vector model: docs and queries are vectors in the term space
    - rank documents according to the cosine similarity
  - Term weights
    - tf × idf : (tf = term frequency, idf = log of inverse document frequency – promote rare terms)
- Measures
  - Precision: percentage of relevant documents over the returned documents
  - Recall: percentage of relevant documents over all existing relevant documents

IR Concepts - Boolean Model

- Boolean model: Data is represented as a 0/1 matrix
- Query: a boolean expression
  - the $\wedge$ world $\wedge$ war
  - the $\wedge$ (world$\vee$civil) $\wedge$war
- Return all the results that match the query
  - docs D_1 and D_2
- How are the documents ranked?

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	1	1	1	1
D_2	1	1	1	1
D_3	1	0	0	1

IR Concepts - Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	1	1	1
D_2	1	1	1	1
D_3	1	0	0	1

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	100	20	5	25
D_2	200	20	50	40
D_3	150	0	0	50

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
  - frequency of term i in document j
  - normalized by max

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- not all words are interesting
  - df_i = document frequency of term i

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

df	1	0.66	0.66	1

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- not all words are interesting
  - df_i = document frequency of term i
  - idf_i = inverse document frequency
    - $idf_i = \log(1/df_i)$

D_1

… … … … …
…the civil war
… world … …

D_2

… … … … …
the world war
… … civil …

D_3

… … … … …
… the war …
… …. …. ….

	the	civil	world	war
D_1	1	0.20	0.05	0.25
D_2	1	0.10	0.25	0.20
D_3	1	0	0	0.33

idf	0	0.17	0.17	0

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- idf_i = inverse document frequency
- $w_{ij} = tf_{ij} \times idf_i$

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

	the	civil	world	war
D_1	0	0.034	0.008	0
D_2	0	0.017	0.042	0
D_3	0	0	0	0

IR Concepts – Term weighting

- Assess the importance w_{ij} of term i in a document j
- tf_{ij} = term frequency
- idf_i = inverse document frequency
- $w_{ij} = tf_{ij} \times idf_i$
- Query: "the civil war"
  - document D_1 is more important

D_1	D_2	D_3
… … … … … …the civil war … world … …	… … … … … the world war … … civil …	… … … … … … the war … … …. …. ….

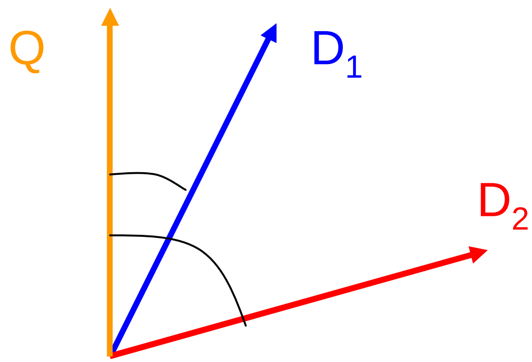
	the	civil	world	war
D_1	0	0.034	0.008	0
D_2	0	0.017	0.042	0
D_3	0	0	0	0

IR Concepts – Vector model

- Documents are vectors in the term space (weighted by w_{ij}), normalized on the unit sphere
- Query: "the civil war"
  - Q is a mini document - vector
- Similarity of Q and D is the cosine of the angle between Q and D
  - returns a set of ranked results

	the	civil	world	war
D_1	0	0.97	0.22	0
D_2	0	0.37	0.92	0
D_3	0	0	0	0

Q	0	1	1	0



IR Concepts – Measures

- There are A relevant documents to the query in our dataset.
- Our algorithm returns D documents.
- How good is it?

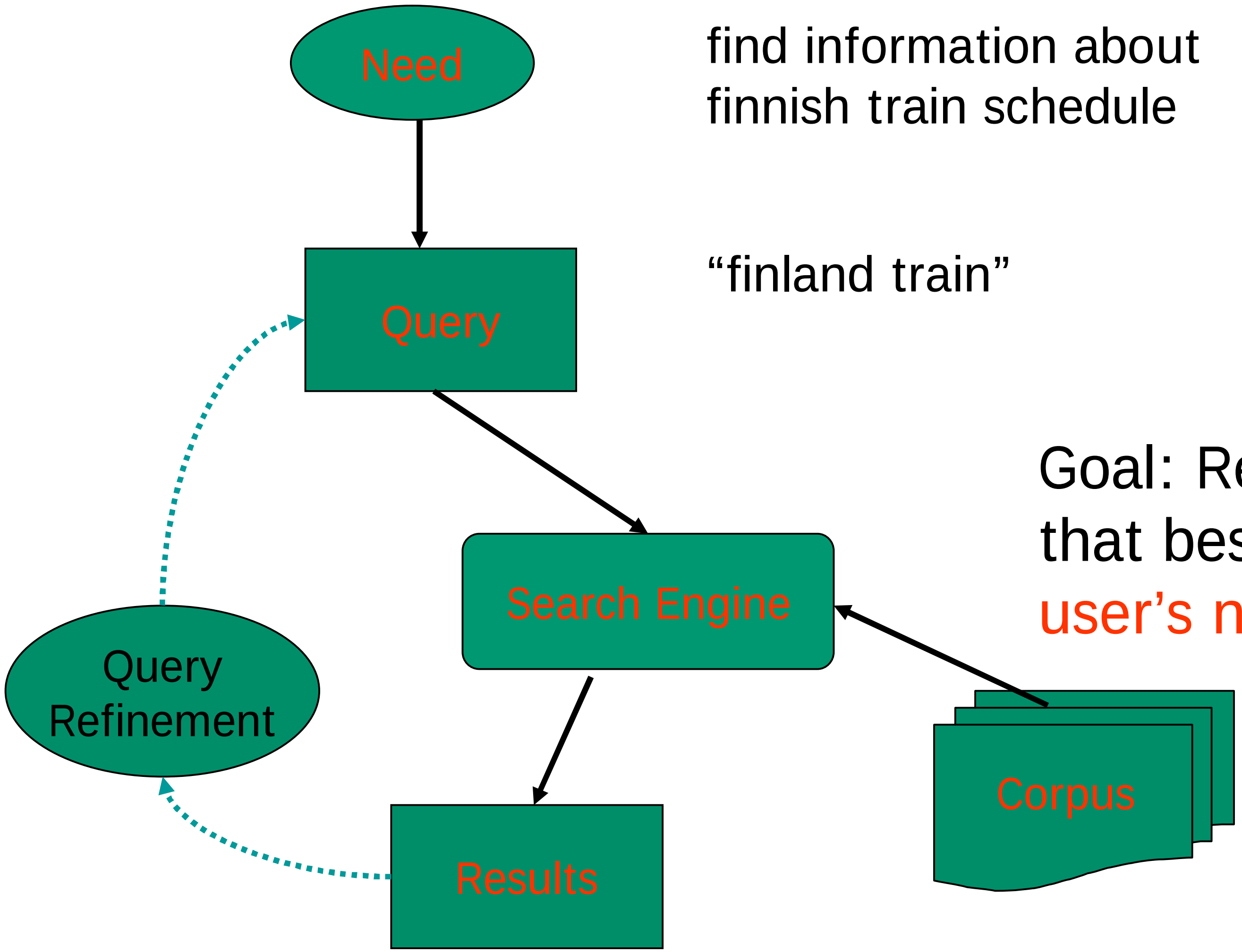
- Precision: Fraction of returned documents that are relevant

$$P = \frac{|D \cap A|}{D}$$

- Recall: Fraction of all relevant documents that are returned

$$R = \frac{|D \cap A|}{A}$$

Web Search



find information about
finnish train schedule

“finland train”

Goal: Return the results that best satisfy the user’s need

The need behind the query

- Informational – learn about something (~40%)
  - "colors of greek flag", "haplotype definition"
- Navigational – locate something (~25%)
  - "microsoft", "Jon Kleinberg"
- Transactional – do something (~35%)
  - Access a service
    - "train to Turku"
  - Download
    - "earth at night"
  - Shop
    - "Nicon Coolpix"

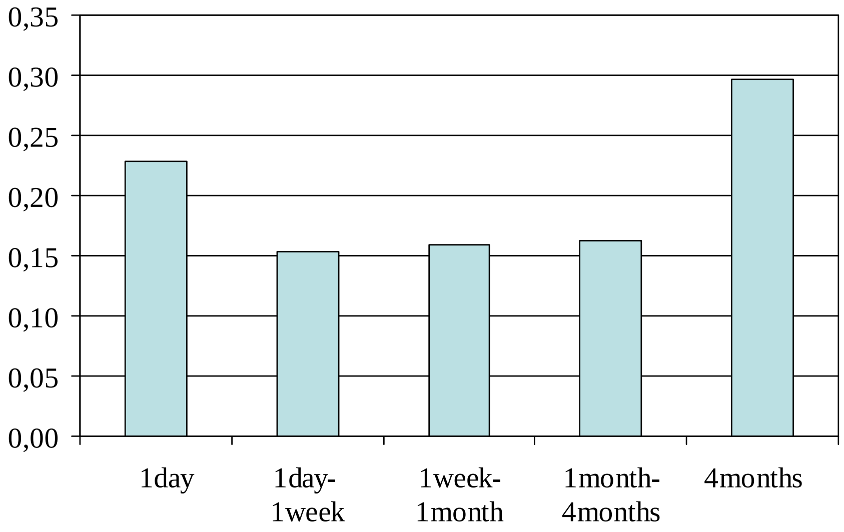
Web users

- They ask a lot but they offer little in return
  - Make ill-defined queries
    - short (2.5 avg terms, 80% <3 terms – AV, 2001)
    - imprecise terms
    - poor syntax
    - low effort
  - Unpredictable
    - wide variance in needs/expectations/expertise
  - Impatient
    - 85% look one screen only (mostly “above the fold”)
    - 78% queries not modified (one query per session)
- ...but they know how to spot correct information
  - follow “the scent of information”...

Web corpus

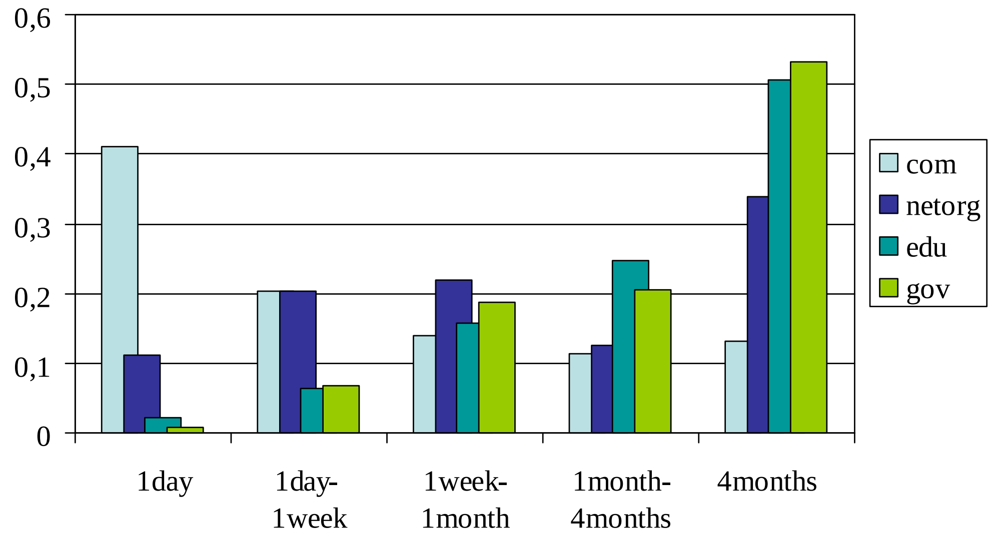
- Immense amount of information
  - 2005, Google: 8 Billion pages, Yahoo! : 20(!) Billion
  - fast growth rate (double every 8-12 months)
  - Huge Lexicon: 10s-100s millions of words
- Highly diverse content
  - many different authors, languages, encodings
  - different media (text, images, video)
  - highly un-structured content
- Static + Dynamic (“the hidden Web”)
- Volatile
  - crawling challenge

Rate of change [CGM00]

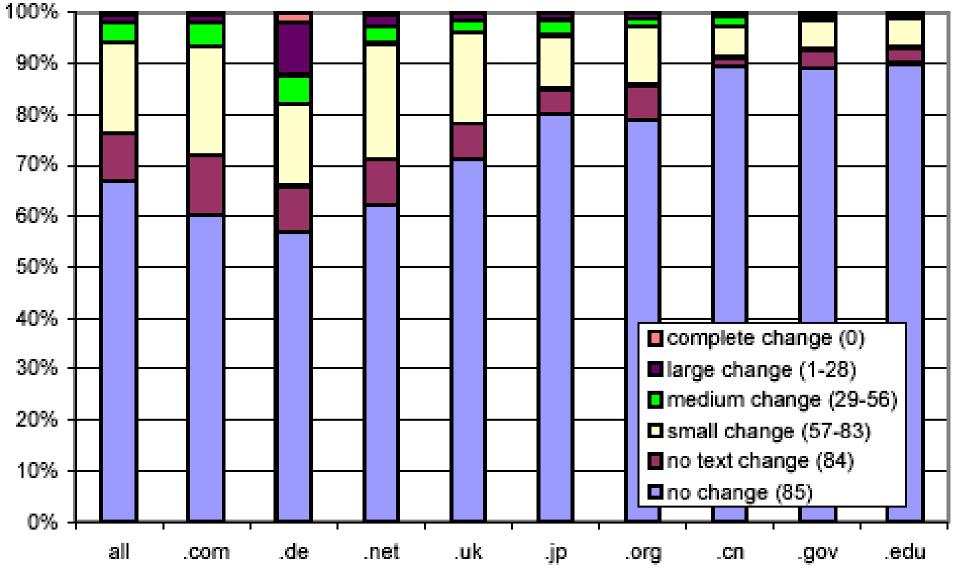


average rate of change

average rate of change
per domain

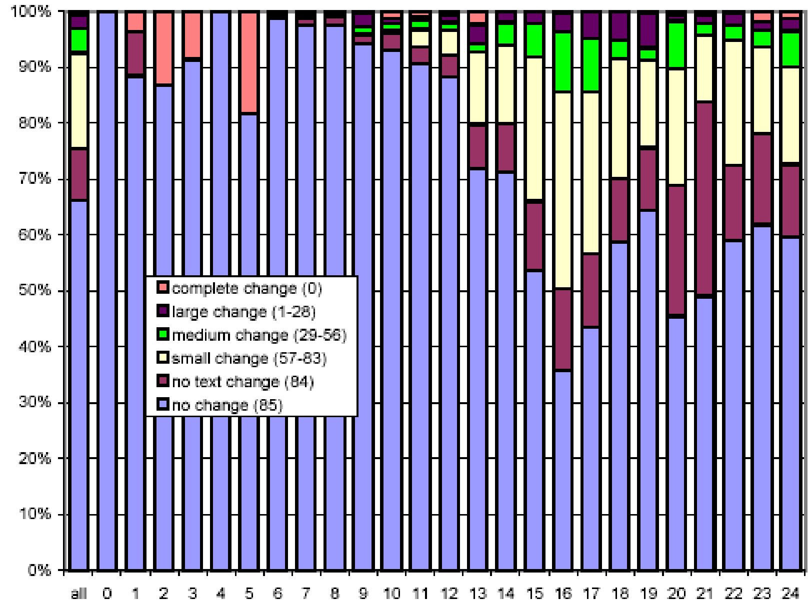


Rate of Change [FMNW03]



Rate of change per domain. Change between two successive downloads

Rate of change as a function of document length



Other corpus characteristics

- Links, graph topology, anchor text
  - this is now part of the corpus!
- Significant amount of duplication
  - ~30% (near) duplicates [FMN03]
- Spam!
  - 100s of million of pages
  - Add-URL robots

Query Results

- Static documents
  - text, images, audio, video,etc
- Dynamic documents (“the invisible Web”)
  - dynamic generated documents, mostly database accesses
- Extracts of documents, combinations of multiple sources
  - www.googlism.com

Googlism

Googlism for: tsaparas

tsaparas is president and ceo of prophecy entertainment inc
tsaparas is the only person who went to the college of the holy cross
tsaparas is to be buried in thessaloniki this morning following his death late on thursday night at the age of 87

Googlism for: athens

athens is the home of the parthenon
athens is the capital of greece and the country's economic
athens is 'racing against time'
athens is a hometown guy

The evolution of Search Engines

- First Generation – text data only
  - word frequencies, tf × idf
- Second Generation – text and web data
  - Link analysis
  - Click stream analysis
  - Anchor Text
- Third Generation – the need behind the query
  - Semantic analysis: what is it about?
  - Integration of multiple sources
  - Context sensitive
    - personalization, geographical context, browsing context

1995-1997: AltaVista Lycos, Excite

1998 - now : Google leads the way

Still experimental

First generation Web search

- Classical IR techniques
  - Boolean model
  - ranking using tf × idf relevance scores

- good for informational queries
- quality degraded as the web grew
- sensitive to spamming

Second generation Web search

- Boolean model
- Ranking using web specific data
  - HTML tag information
  - click stream information (DirectHit)
    - people vote with their clicks
  - directory information (Yahoo! directory)
  - anchor text
  - link analysis

Link Analysis Ranking

- Intuition: a link from q to p denotes endorsement
  - people vote with their links
- Popularity count
  - rank according to the incoming links
- PageRank algorithm
  - perform a random walk on the Web graph. The pages visited most often are the ones most important.

$$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|F(q)|} + (1 - \alpha)\frac{1}{n}$$

Second generation SE performance

- Good performance for answering navigational queries
  - “finding needle in a haystack”
- ... and informational queries
  - e.g “oscar winners”
- Resistant to text spamming
- Generated substantial amount of research
- Latest trend: specialized search engines

Result evaluation

- recall becomes useless
- precision measured over top-10/20 results
- Shift of interest from "relevance" to "authoritativeness/reputation"
- ranking becomes critical

Second generation spamming

- Online tutorials for "search engine persuasion techniques"
  - "How to boost your PageRank"
- Artificial links and Web communities
- Latest trend: "Google bombing"
  - a community of people create (genuine) links with a specific anchor text towards a specific page. Usually to make a political point

Google Bombing

Google Bombing

- Try also the following
  - “weapons of mass destruction”
  - “french victories”

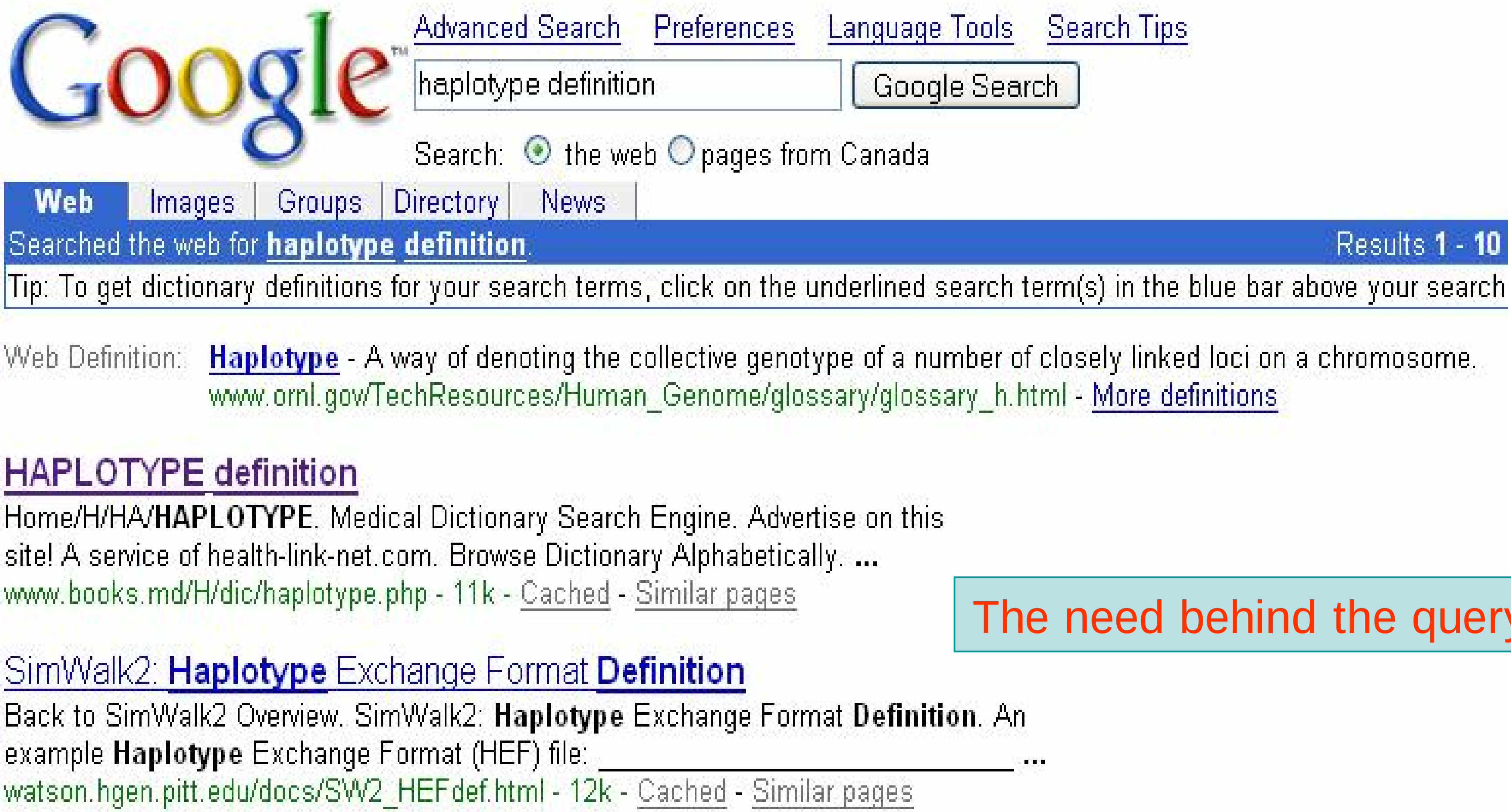
- Do Google bombs capture an actual trend?
- How sensitive is Google to such bombs?

Spamming evolution

- Spammers evolve together with the search engines. The two seem to be intertwined.

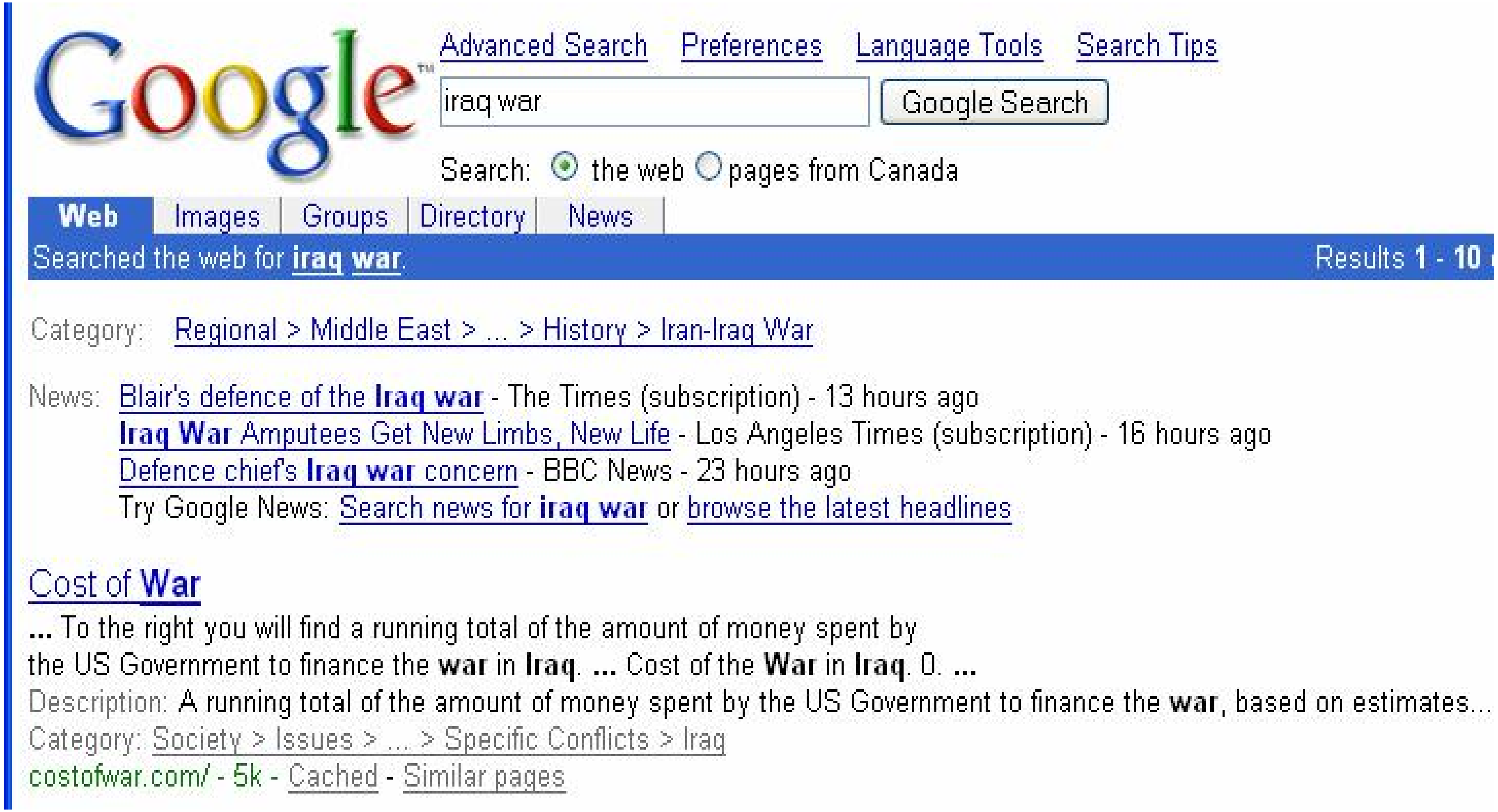
Adversarial Information Retrieval

Third generation Search Engines: an example



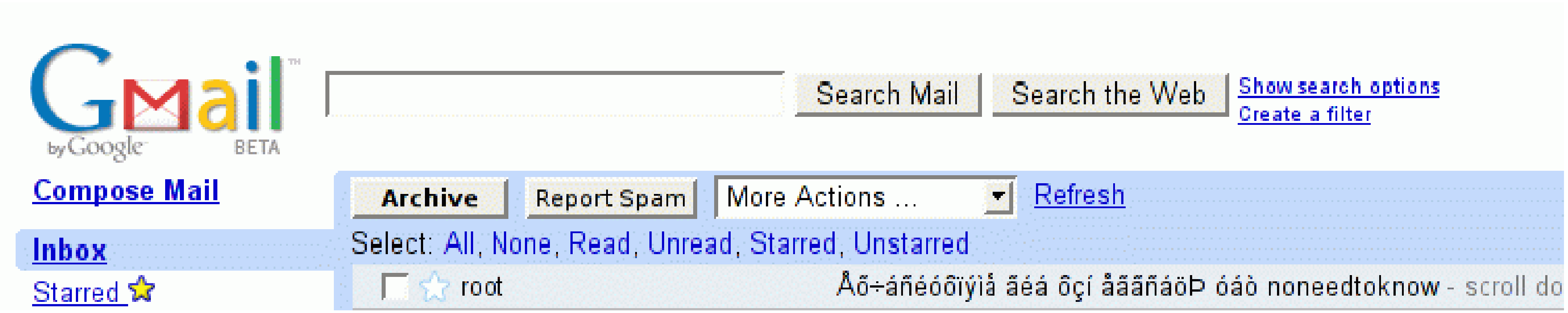
The need behind the query

Third generation Search Engines: another example

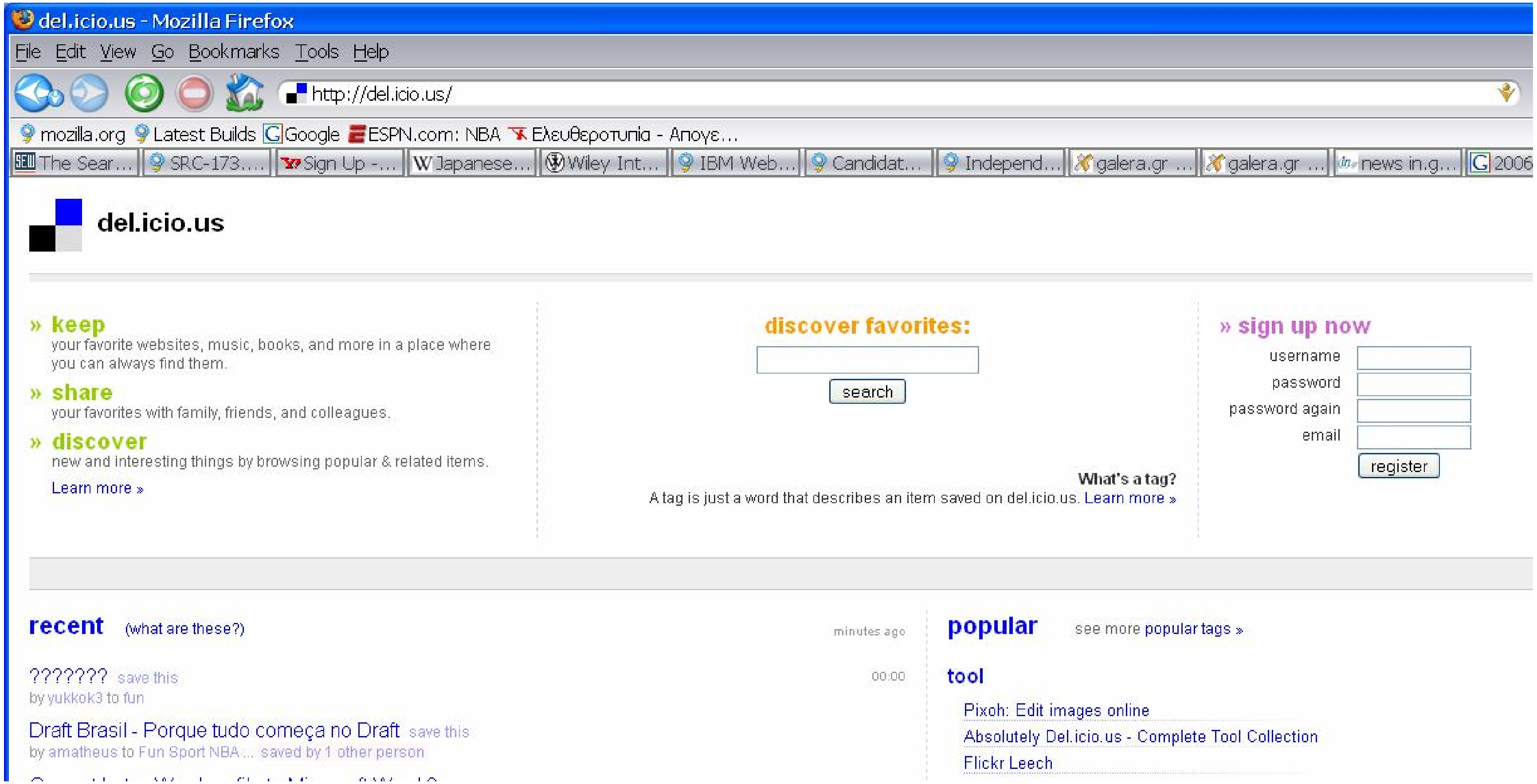


Third generation Search Engines: another example

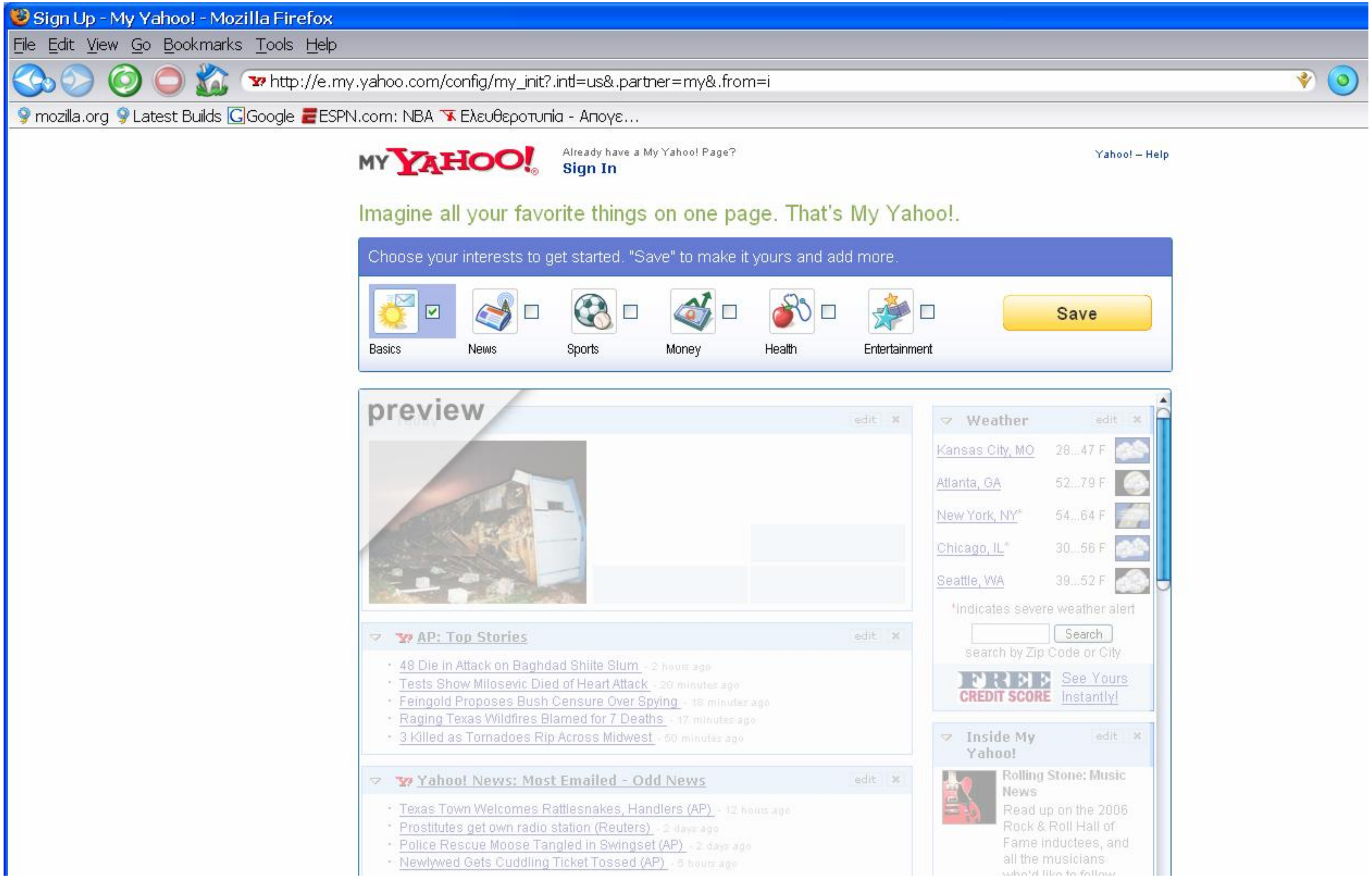
Integration of Search and Mail?



Integration of Search Engines and Social Networks



Integration of Search Engines and Social Networks



Personalization

- Use information from multiple sources about the user to offer a personalized search experience
  - bookmarks
  - mail
  - toolbar
  - social network

More services

- Google/Yahoo maps
- Google Earth
- Mobile Phone Services
- Google Desktop

- The search engines war: Google, Yahoo, MSN
  - a very dynamic time for search engines

- Search Engine Economics: How do the search engines produce income?
  - advertising (targeted advertising)
  - privacy issues?

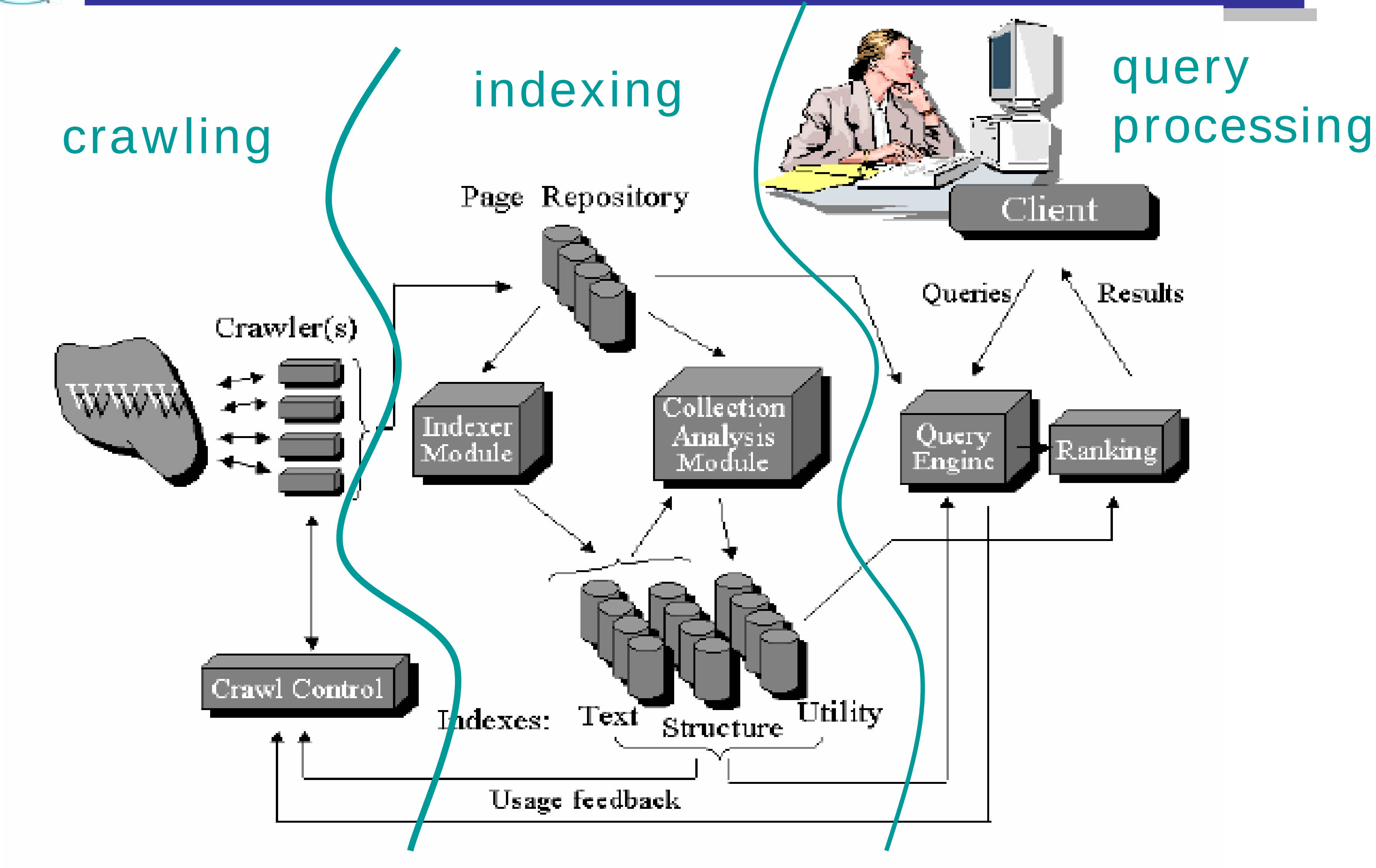
The future of Web Search?

EPIC

Outline

- Web Search overview
  - from traditional IR to Web search engines
- The anatomy of a search engine
  - Crawling, Duplicate elimination, Indexing

The anatomy of a Search Engine



Crawling

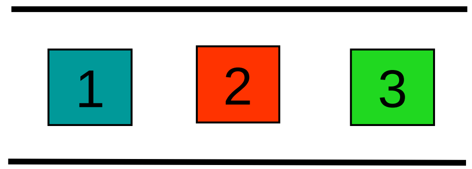
- Essential component of a search engine
  - affects search engine quality
- Performance
  - 1995: single machine – 1M URLs/day
  - 2001: distributed – 250M URLs/day
- Where do you start the crawl from?
  - directories
  - registration data
  - HTTP logs
  - etc...

Algorithmic issues

- Politeness
  - do not hit a server too often (robots.txt)
- Freshness
  - how often to refresh and which pages?
- Crawling order
  - in which order to download the URLs
- Coordination between distributed crawlers
- Avoiding spam traps
- Duplicate elimination
- Research: focused crawlers

Poor man's crawler

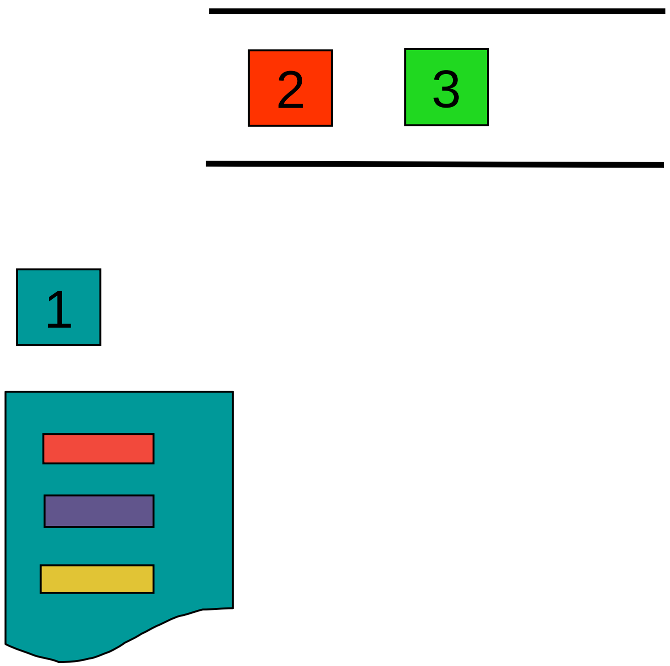
- A home-made small-scale crawler



start with a queue of URLs to be processed

Poor man's crawler

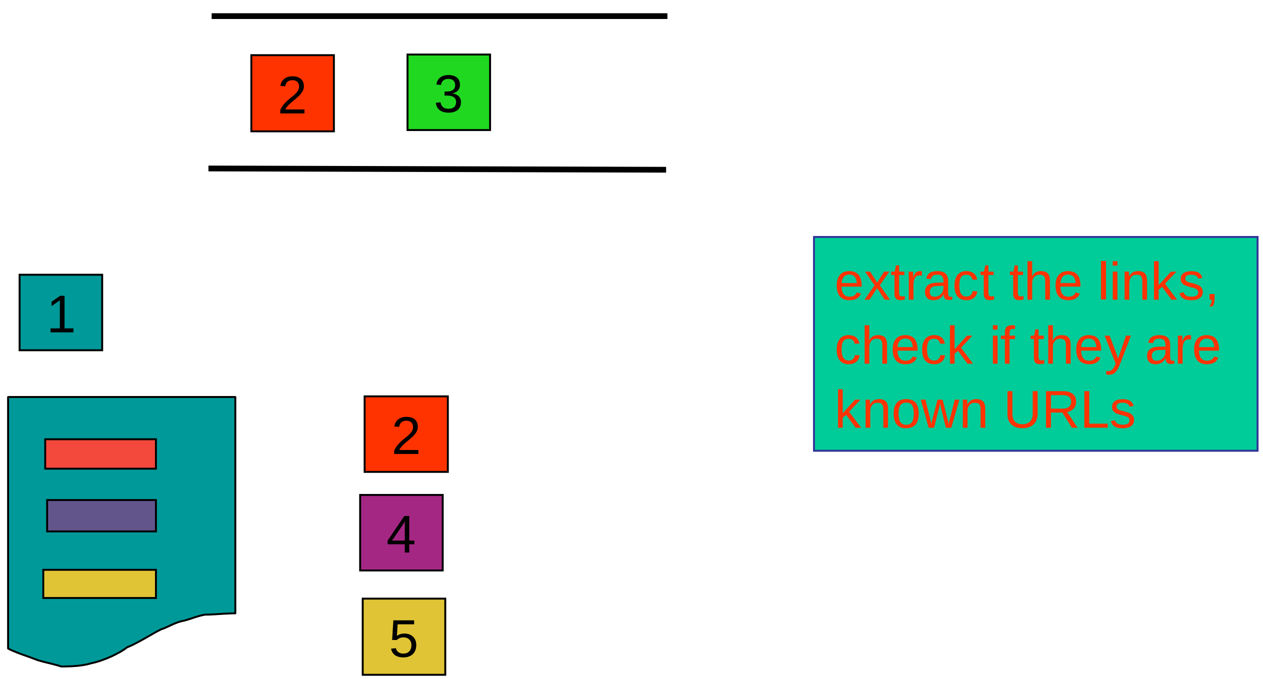
- A home-made small-scale crawler



fetch the first page to be processed

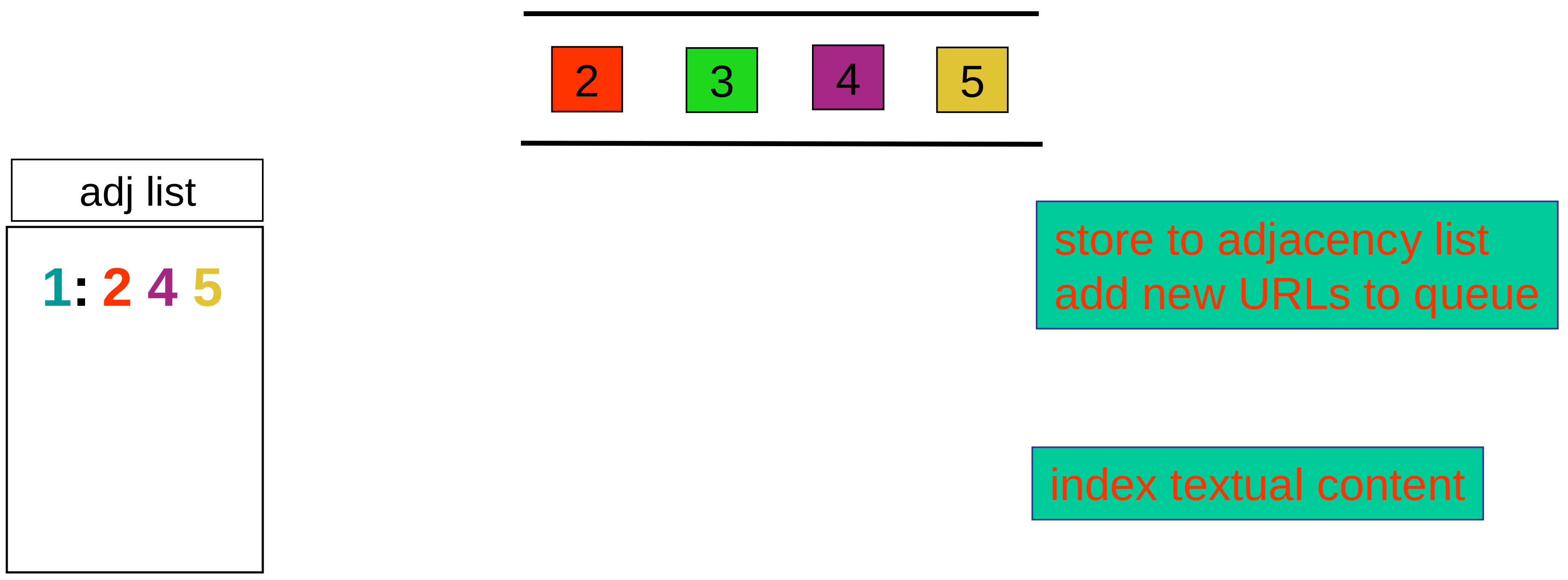
Poor man's crawler

- A home-made small-scale crawler



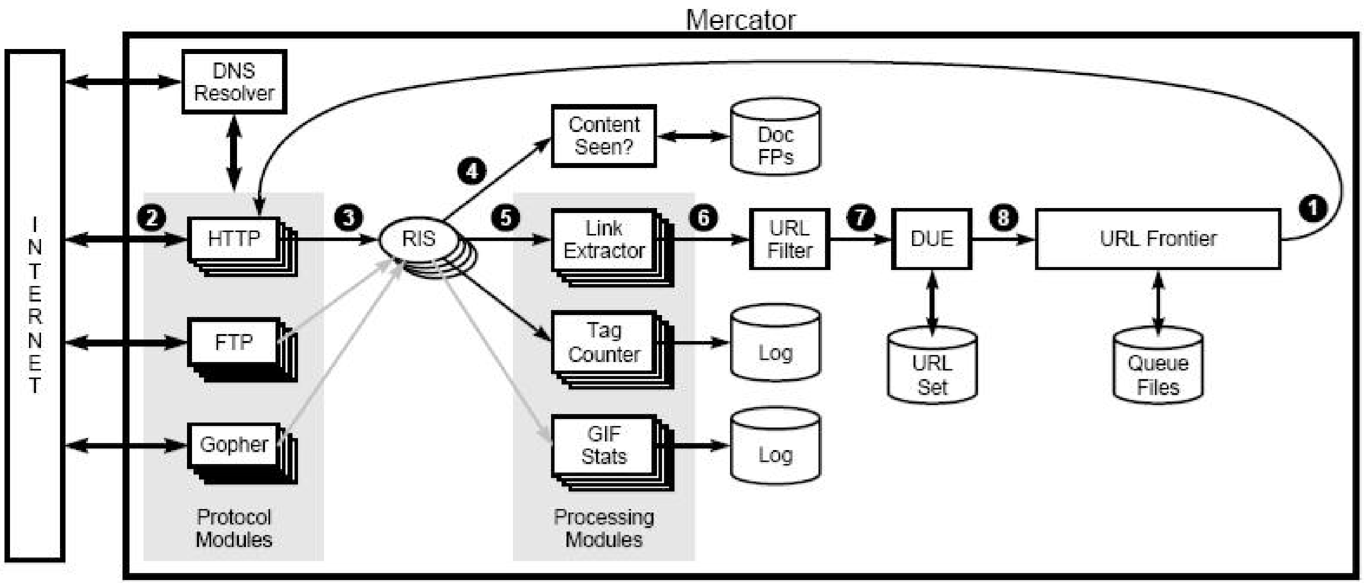
Poor man's crawler

- A home-made small-scale crawler



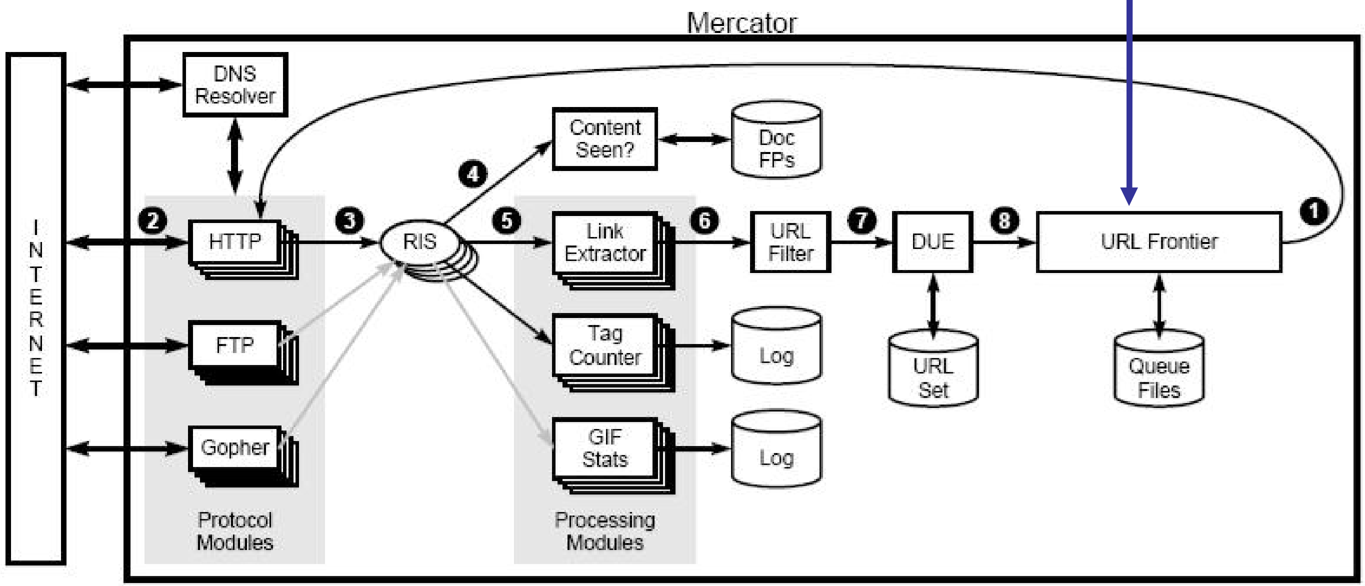
Mercator Crawler [NH01]

§ Not much different from what we described



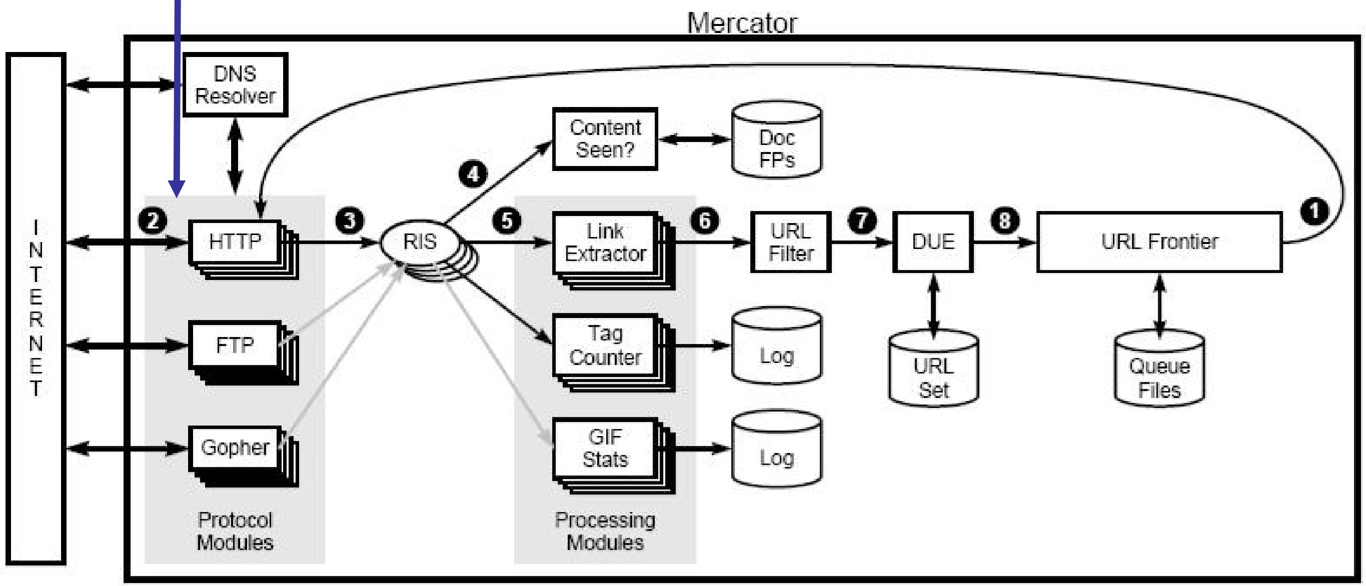
Mercator Crawler [NH01]

the next page to be crawled is obtained from the URL frontier



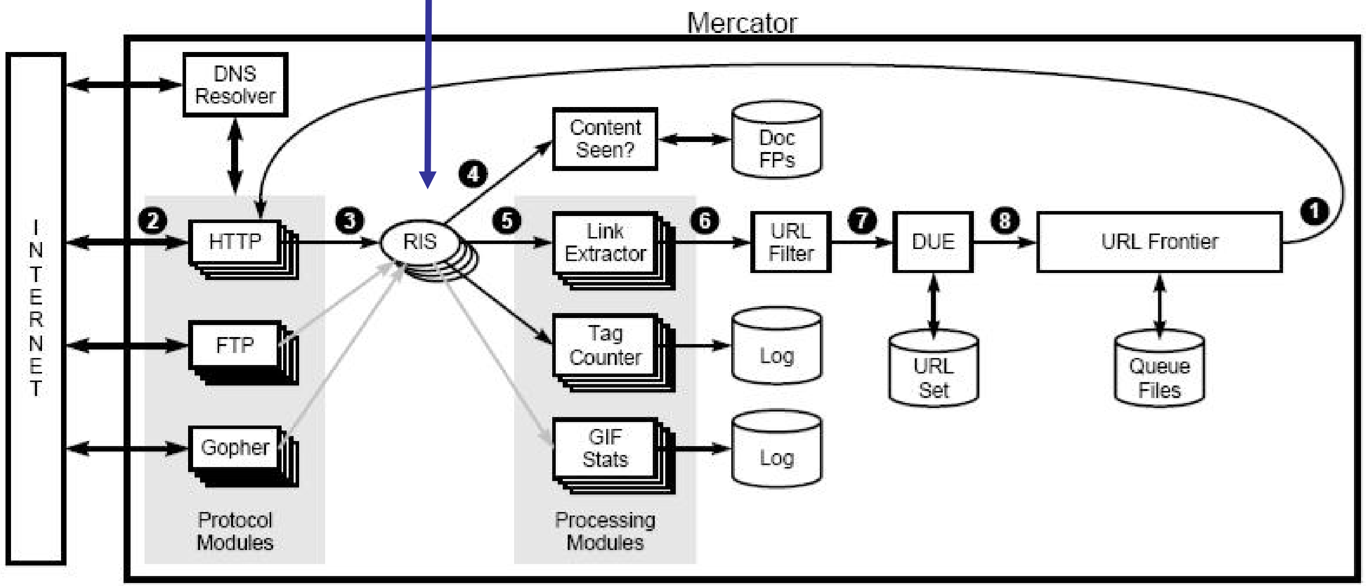
Mercator Crawler [NH01]

the page is fetched using the appropriate protocol



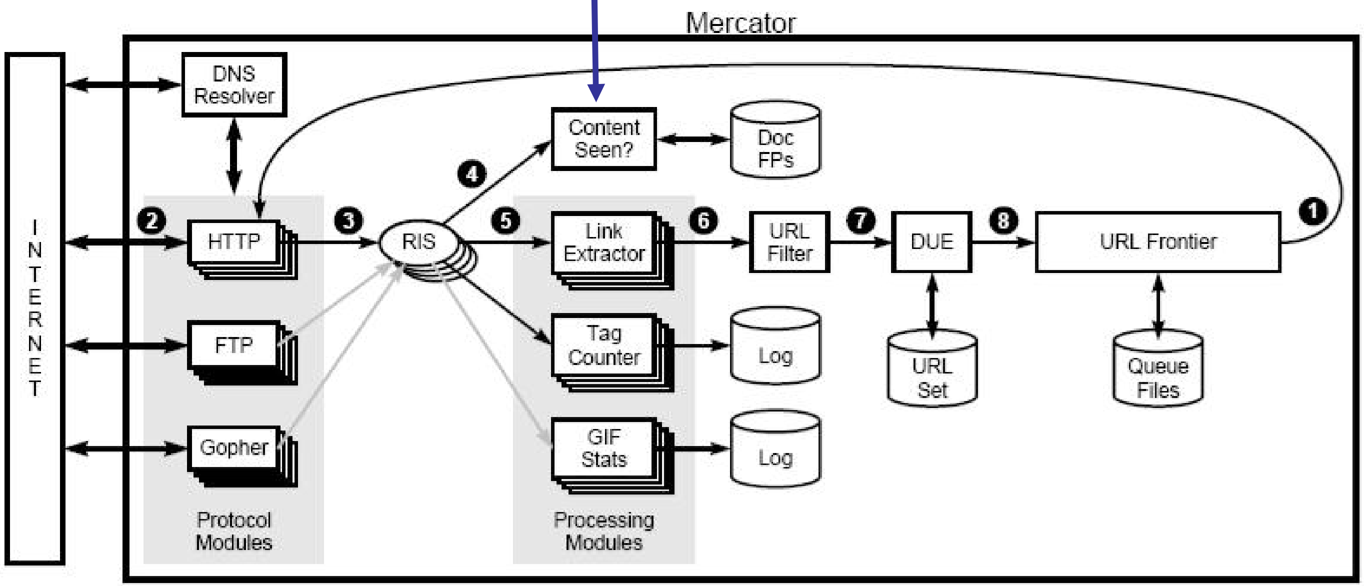
Mercator Crawler [NH01]

Rewind Input Stream: an IO abstraction



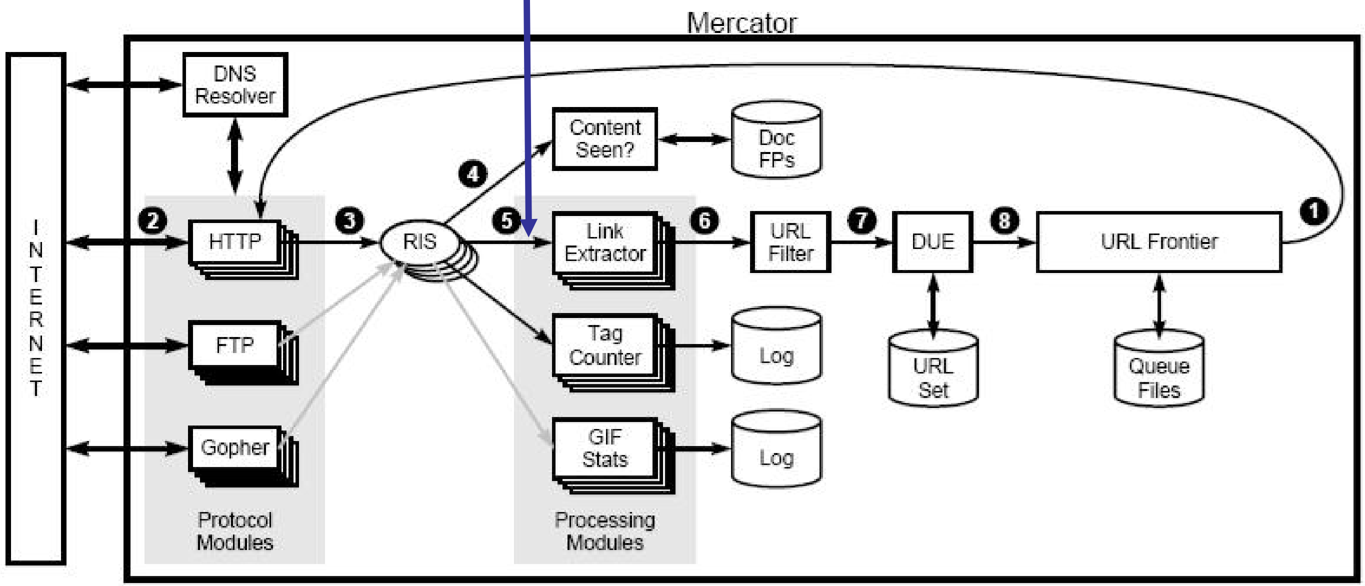
Mercator Crawler [NH01]

check if the content of the page has been seen before (duplicate, or near duplicate elimination)



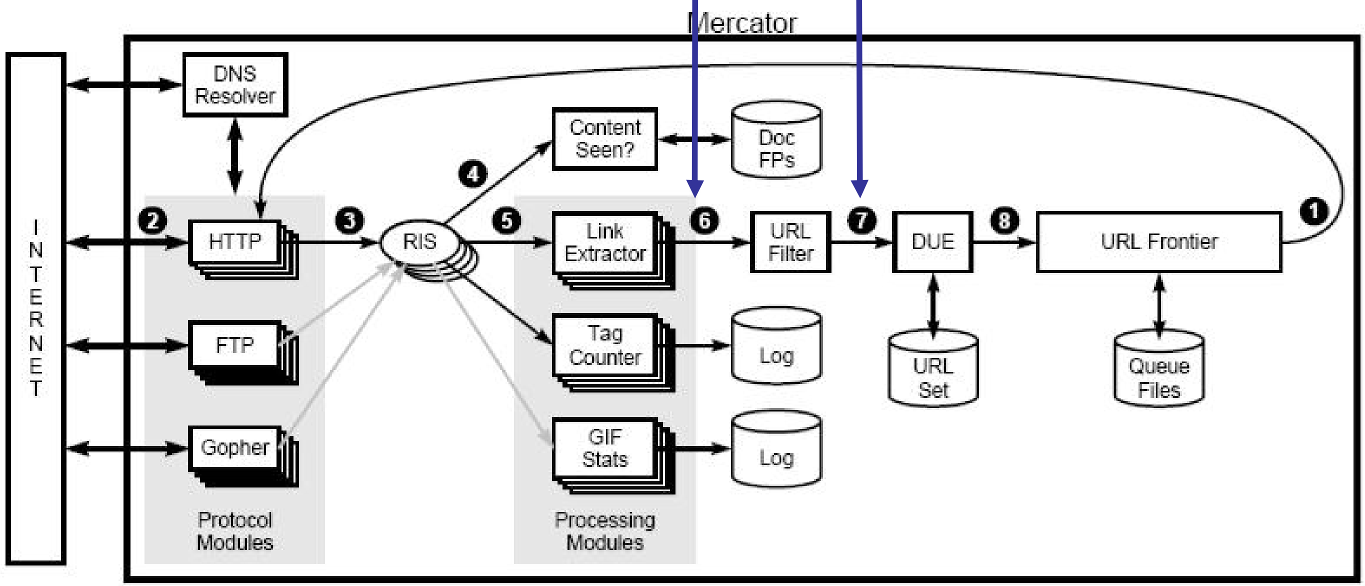
Mercator Crawler [NH01]

process the page (e.g. extract links)



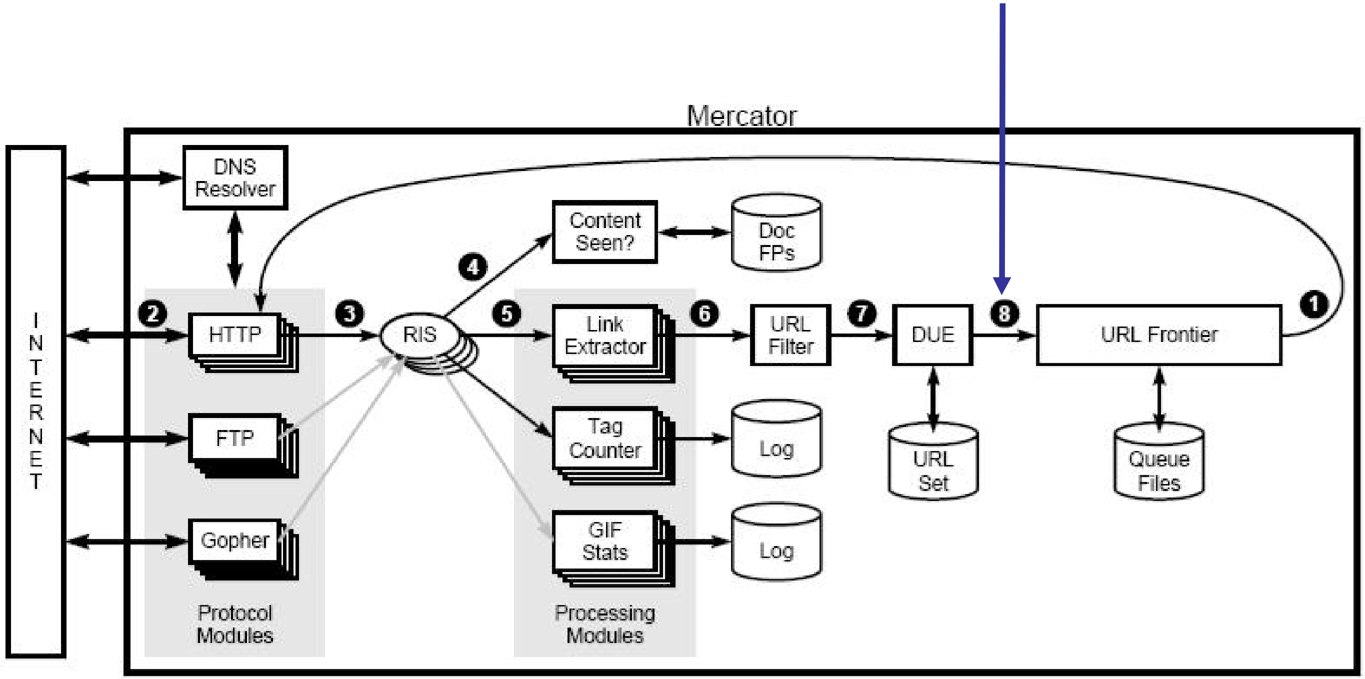
Mercator Crawler [NH01]

check if the links should be filtered out (e.g. spam) or if they are already in the URL set

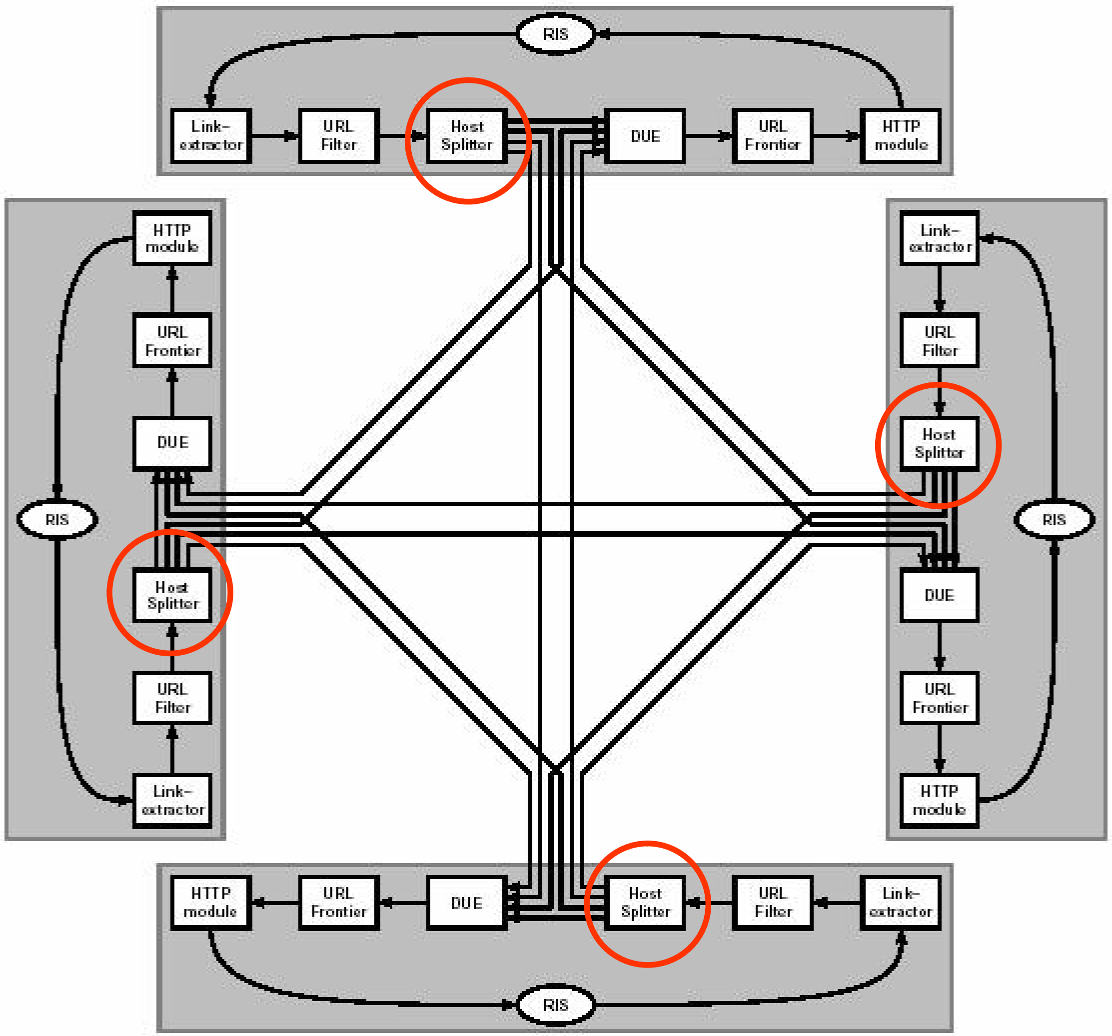


Mercator Crawler [NH01]

if not visited, add to the URL frontier, prioritized (in the case of continuous crawling, you may add also the source page, back to the URL frontier)



Distributed Crawling



- Each process is responsible for a partition of URLs
- The Host Splitter assigns the URLs to the correct process
- Most links are local so traffic is small
- UbiCrawler: Use of consistent hashing to achieve load balancing and fault tolerance.

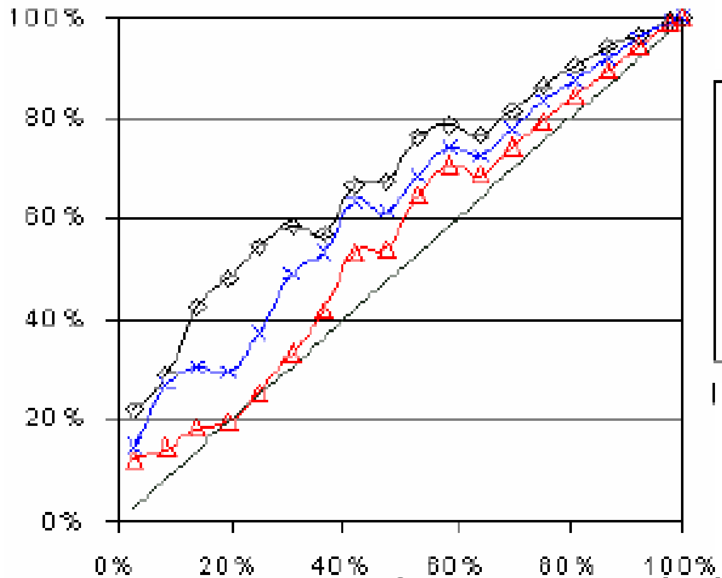
Crawling order

- Best pages first
  - possible quality measures
    - in-degree
    - PageRank
  - possible orderings
    - Breadth First Search (FIFO)
    - in-degree (so far)
    - PageRank (so far)
    - random

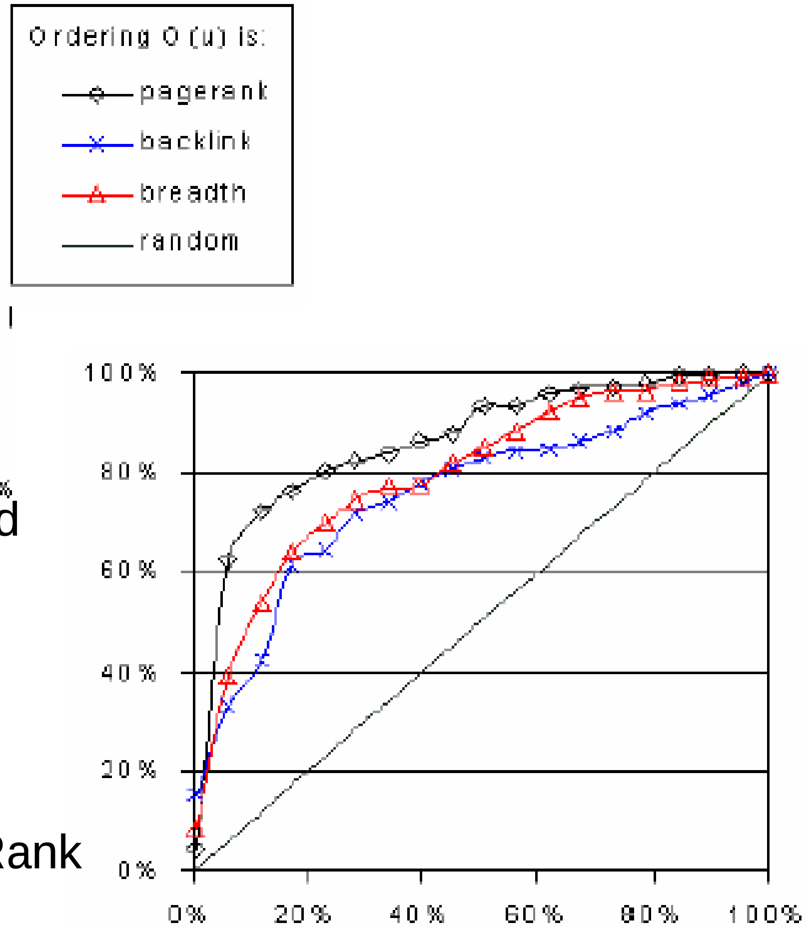
Crawling order [CGP98]

"hot" page = high in-degree

% of
"hot"
pages

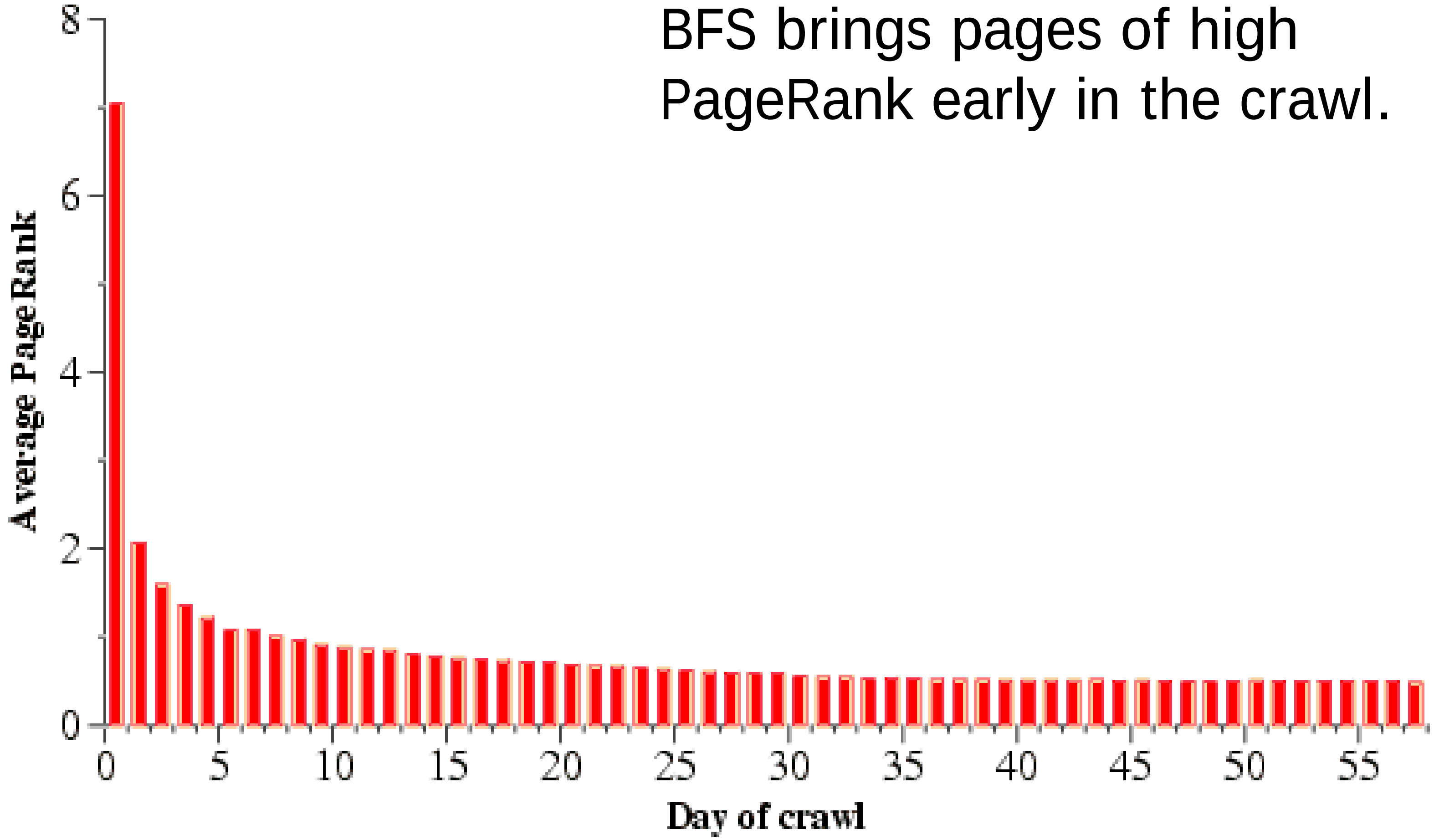


percentage of pages crawled



"hot page = high PageRank

Crawling order [NW01]



BFS brings pages of high PageRank early in the crawl.

Duplication

- Approximately 30% of the Web pages are duplicates or near duplicates
- Sources of duplication
  - Legitimate: mirrors, aliases, updates
  - Malicious: spamming, crawler traps
  - Crawler mistakes
- Costs:
  - wasted resources
  - unhappy users

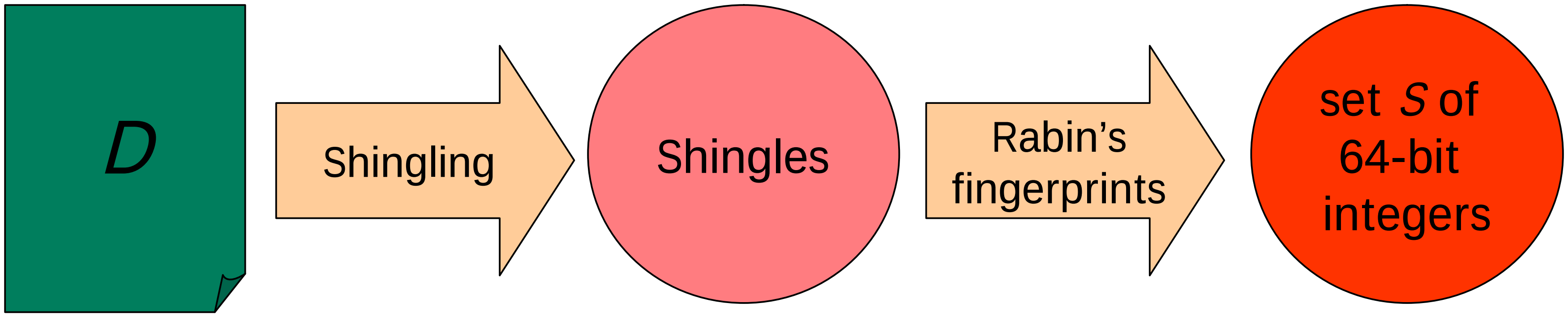
Observations

- Eliminate both duplicates and near duplicates
- Computing pairwise edit distance is too expensive
- Solution
  - reduce the problem to set intersection
  - sample documents to produce small sketches
  - estimate the intersection using the sketches

Shingling

- Shingle: a sequence of w contiguous words

```
a rose is a rose is a rose
a rose is a
  rose is a rose
       is a rose is
          a rose is a
            rose is a rose
```



Rabin's fingerprinting technique

- Comparing two strings of size n

a = 10110 **a=b?**

b = 11010 O(n) too expensive!

f(a)=f(b)?

$$A = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$

$$B = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$$

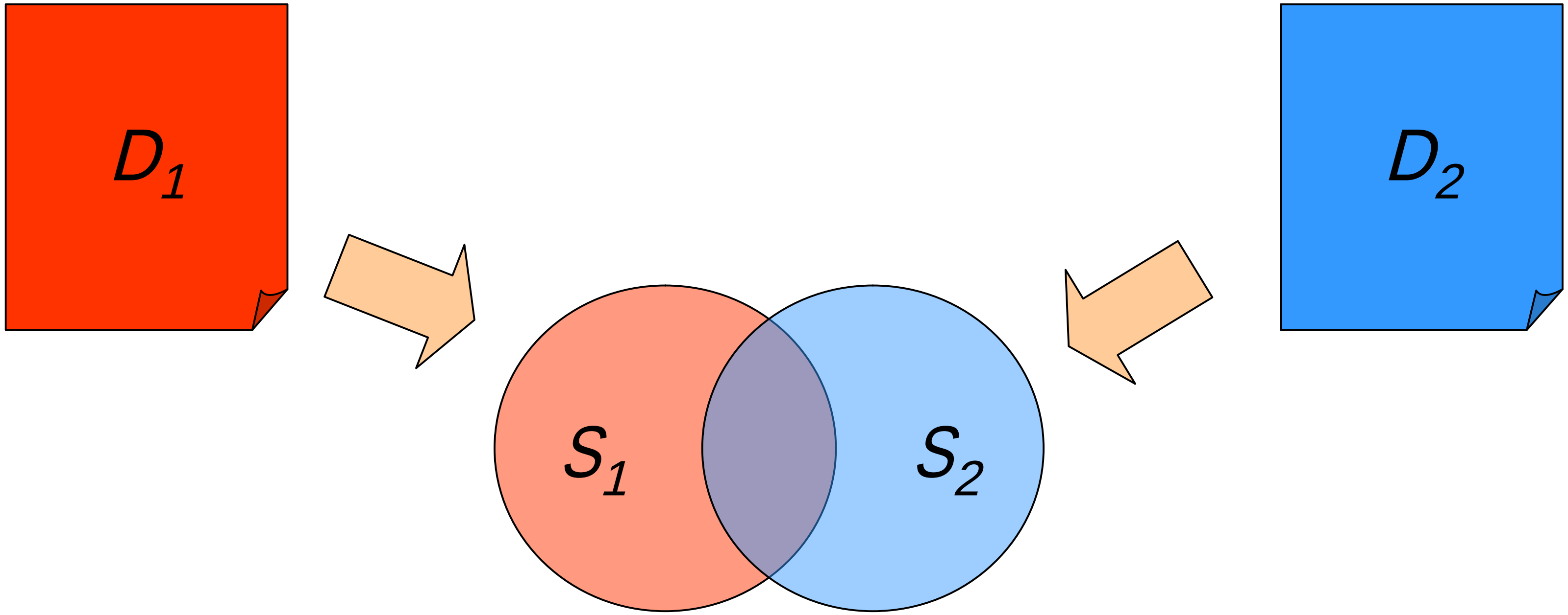
f(a)= A mod p **p = small random prime**

f(b)= B mod p **size O(logn loglogn)**

- if **a=b** then **f(a)=f(b)**

  if **f(a)=f(b)** then **a=b** with high probability

Defining Resemblance



$$\text{reseblance} = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Jaccard coefficient

Sampling from a set

- Assume that $S \subset U$
  - e.g. $U = \{a,b,c,d,e,f\}$, $S=\{a,b,c\}$
- Pick uniformly at random a permutation σ of the universe U
  - e.g $\sigma=\langle d,f,b,e,a,c \rangle$
- Represent S with the element that has the smallest image under σ
  - e.g. $\sigma=\langle d,f,b,e,a,c \rangle$ $b = \sigma\text{-min}(S)$
- Each element in S has equal probability of being $\sigma\text{-min}(S)$

Estimating resemblance

- Apply a permutation σ to the universe of all possible fingerprints $U=[1...2^{64}]$
- Let $\alpha = \sigma\text{-min}(S_1)$ and $\beta = \sigma\text{-min}(S_2)$

$$\Pr(\alpha = \beta) = ?$$

Estimating resemblance

- Apply a permutation σ to the universe of all possible fingerprints $U=[1...2^{64}]$
- Let $\alpha=\sigma\text{-min}(S_1)$ and $\beta= \sigma\text{-min}(S_2)$

$$\Pr(\alpha = \beta) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

- Proof:
  - The elements in $S_1 \cup S_2$ are mapped by the same permutation σ.
  - The two sets have the same σ-min value if $\sigma\text{-min}(S_1 \cup S_2)$ belongs to $S_1 \cap S_2$

Example

Universe U = {a,b,c,d,e,f}

$S_1 = \{a,b,c\}$

$S_2 = \{b,c,d\}$

$S_1 \cap S_2 = \{b,c\}$

$S_1 U\ S_2 = \{a,b,c,d\}$

$\sigma(U) = \langle e,*,*,f,*,*\rangle$

We do not care where the elements e and f are placed in the permutation

$\sigma\text{-min}(S_1) = \sigma\text{-min}(S_2)$ if * is from {b,c}

The element in * can be any of the {a,b,c,d}

$$\Pr(\sigma - \min(S_1) = \sigma - \min(S_2)) = \frac{|\{b,c\}|}{|\{a,b,c,d\}|} = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

Filtering duplicates

- Sample k permutations of the universe $U=[1...2^{64}]$
- Represent fingerprint set S as

  $$S'=\{\sigma_1\text{-min}(S), \sigma_2\text{-min}(S),... \sigma_k\text{-min}(S)\}$$

- For two sets S_1 and S_2 estimate their resemblance as the number of elements S_1' and S_2' have in common
- Discard as duplicates the ones with estimated similarity above some threshold r

min-wise independent permutations

- Problem: There is no practical way to sample from the universe $U=[1...2^{64}]$
- Solution: Sample from the (smaller) set of min-wise independent permutations [BCFM98]
- min-wise independent permutation σ
  for every set X
    for every element x of X
      x has equal probability of being the minimum element of X under σ

Other applications

- This technique has also been applied to other data mining applications
  - for example find words that appear often together in documents

	w1	w2	w3	w4
d1	1	0	1	1
d2	1	0	1	1
d3	0	1	0	1
d4	1	0	0	0
d5	1	1	1	0

w1 = {d1,d2,d4,d5}
w2 = {d3,d5}
w3 = {d1,d2,d3,d5}
w4 = {d1,d2,d3}

Other applications

- This technique has also been applied to other data mining applications
  - for example find words that appear often together in documents

	w1	w2	w3	w4
d1	1	0	1	1
d2	1	0	1	1
d3	0	1	0	1
d4	1	0	0	0
d5	1	1	1	0

w1 = {d1,d2,d4,d5}
w2 = {d3,d5}
w3 = {d1,d2,d3,d5}
w4 = {d1,d2,d3}

w1 = {**d1**,**d2**}
w2 = {**d3**,**d5**}
w3 = {**d1**,**d2**}
w4 = {**d2**,**d3**}

‹d3,d1,d5,d2,d4›

‹d2,d5,d4,d1,d3›

The indexing module

- Inverted Index
  - for every word store the doc ID in which it appears
- Forward Index
  - for every document store the word ID of each word in the doc.
- Lexicon
  - a hash table with all the words
- Link Structure
  - store the graph structure so that you can retrieve in nodes, out nodes, “sibling” nodes
- Utility Index
  - stores useful information about pages (e.g. PageRank values)

Google’s Indexing module (circa 98)

- For a word w appearing in document D, create a hit entry
  - plain hit: [cap | font | position]
  - fancy hit: [cap | 111 | type | pos]
  - anchor hit: [cap | 111 | type | docID | pos]

Forward Index

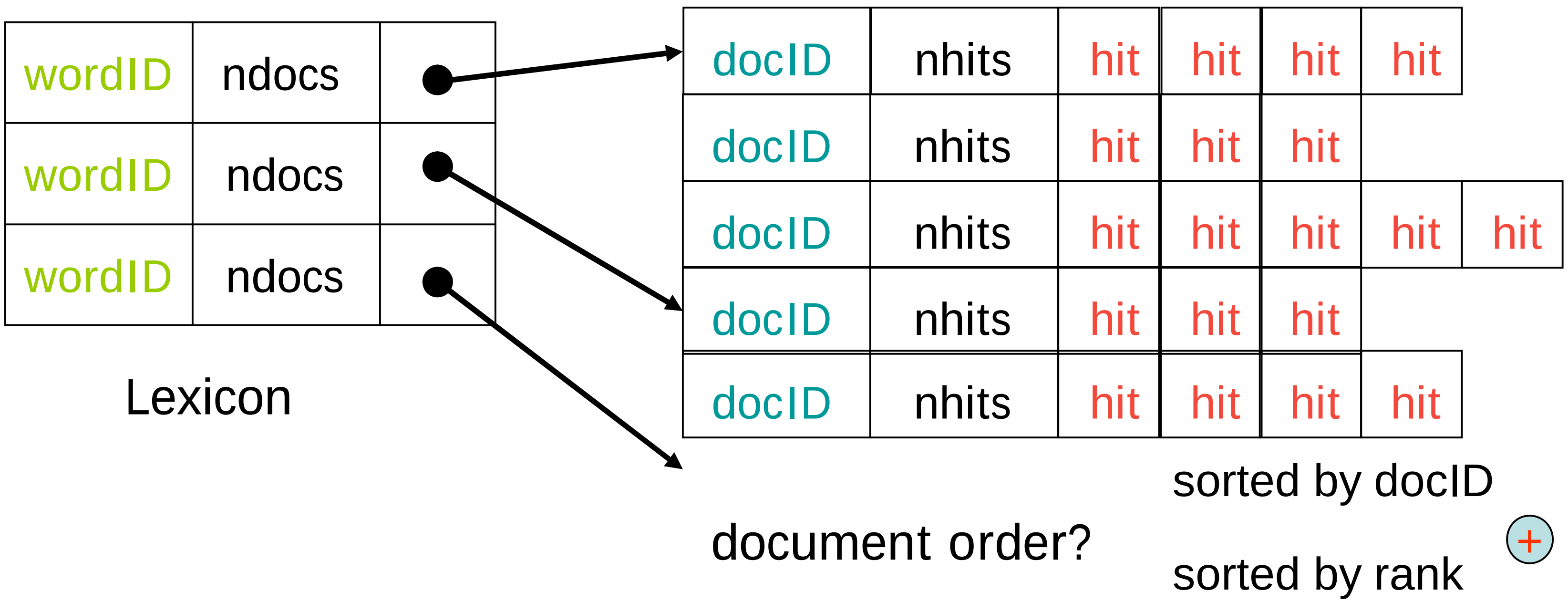
- For each document store the list of words that appear in the document, and for each word the list of hits in the document

docID	wordID	nhits	hit	hit	hit	hit	
	wordID	nhits	hit	hit	hit		
	NULL						
docID	wordID	nhits	hit	hit	hit		
	wordID	nhits	hit	hit	hit	hit	hit
	NULL						

docIDs are replicated in different barrels that store specific range of wordIDs
This allows to delta-encode the wordIDs and save space

Inverted Index

- For each word, the lexicon entry points to a list of document entries in which the word appears



Query Processing

- Convert query terms into wordIDs
- Scan the docID lists to find the common documents.
  - phrase queries are handled using the pos field
- Rank the documents, return top-k
  - PageRank
  - hits of each type × type weight
  - proximity of terms

Disclaimer

No, this talk is not sponsored by Google

Acknowledgements

- Many thanks to Andrei Broder for many of the slides

References

- Ricardo Baeza-Yates, Berthier Ribeirio-Neto, Modern Information Retrieval, Adison-Wesley, 1999
- [NH01] Marc Najork, Allan Heydon High Performance Web Crawling, SRC Research Report, 2001
- A. Broder, On the resemblance and containment of documents
- [BP98] S. Brin, L. Page, The anatomy of a large scale search engine, WWW 1998
- [FMNW03] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. A Large-Scale Study of the Evolution of Web Pages. 12th International World Wide Web Conference (May 2003), pages 669-678
- [NW01] Marc Najork and Janet L. Wiener. Breadth-First Search Crawling Yields High-Quality Pages. 10th International World Wide Web Conference (May 2001), pages 114-118.
- Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, Sriram Raghavan "Searching the Web." *ACM Transactions on Internet Technology, 1(1): August 2001*.
- [CGP98] Junghoo Cho, Hector Garcia-Molina, Lawrence Page "Efficient Crawling Through URL Ordering." *In Proceedings of the 7th World Wide Web conference (WWW7),* Brisbane, Australia, April 1998.
- [CGM00] Junghoo Cho, Hector Garcia-Molina "The Evolution of the Web and Implications for an incremental Crawler." *In Proceedings of 26th International Conference on Very Large Databases (VLDB),* September 2000.
- [BCSV04] P. Boldi, B. Codenotti, M. Santini, S. Vigna, UbiCrawler: a scalable fully distributed Web crawler, Software Practice and Experience, Volume 34(8), pp 711-726