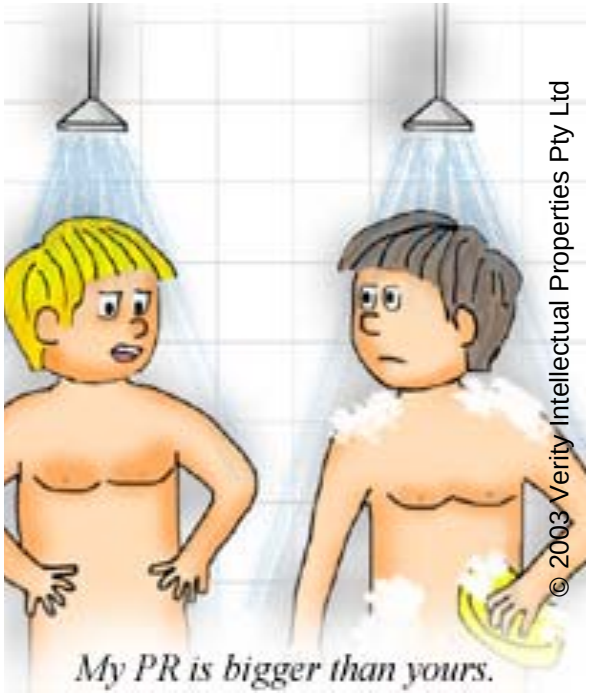
max planck institut informatik

Web, Semantic, and Social Information Retrieval

Gerhard Weikum

weikum@mpi-inf.mpg.de
http://www.mpi-inf.mpg.de/~weikum/

The Google Revolution



but there are also major limitations !

- ☆ great for e-shopping, school kids, scientists, doctors, etc.
- ☆ high-precision results for simple queries
- ☆ superb scalability (now 20 Bio. docs, >1000 queries/sec)
- ☆ continuously enhanced: GoogleScholar, GoogleEarth, alerts, multilingual for >100 languages, query auto-completion, etc.

Why Google Is Not Enough

Difficult queries:

- recent conference papers by computer scientists on percolation theory with application of phase transition models to the analysis of Web graph dynamics
- proteins that inhibit both proteases and some other enzyme
- professors from Germany who teach IR and have EU projects
- differences in Rembetiko music from Greece and from Turkey
- connection between Thomas Mann and Goethe
- market impact of Web2.0 technology in December 2006
- sympathy or antipathy for Germany from May to August 2006
- Nobel laureate who survived both world wars and his children
- drama with three women making a prophecy to a British nobleman that he will become king

What is Beyond Google?

for Advanced Information Requests by „Power Users“
(librarians, market analysts, scientists, students, etc.)

- **background knowledge**
  → ontologies & thesauri, statistics, learning

- **(semi-)structured and „semantic“ data**
  → XML, info extraction, annotation & classification

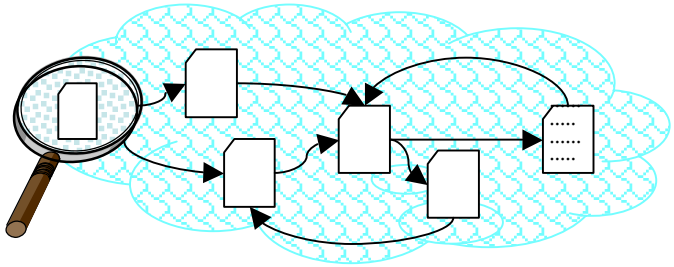
- **humans in the loop, wisdom of crowds**
  → collaboration, recommendation, social networks, P2P

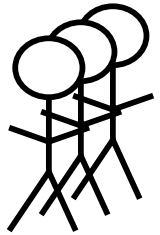
- **context awareness**
  → personalization, geo & time, user behavior, reality mining

Overview

- **Part 1: Web IR**
  - State of the Art
  - Scalability Challenge
  - Quality Challenge
  - Personalization
  - Research Opportunities
- **Part 2: Semantic & Social IR**
  - Ontologies in XML IR
  - Entity Search and Ranking
  - Graph IR
  - Web 2.0 Search and Mining
  - Research Opportunities

System Architecture of a Web Search Engine





crawl | extract & clean | index | search | rank | present

handle dynamic pages, detect duplicates, detect spam

fast top-k queries, query logging, auto-completion

GUI, user guidance, personalization

strategies for crawl schedule and priority queue for crawl frontier

build and analyze Web graph, index all tokens or word stems

scoring function over many data and context criteria

server farm with 1000‘s of computers, distributed/replicated data in high-performance file system, massive parallelism for query processing

Web IR Scoring & Ranking Model

s(d,q) = (linear) combination of
query-specific **relevance** score $s_{rel}(d,q)$ and
query-independent **authority** score $s_{auth}(d)$ and …

simplest form of s_{rel}: **tf*idf** (Salton's vector space model)

$$s(d,q) = \sum_{i \in q} s_i(d)$$

with precomputed term weights
$s_i(d) \sim$ **tf** (term i in d) $*$ **idf** (term i)

term frequency tf, inverse document frequency idf
plus dampening & normalization, e.g.:

$$s_i(d) \sim \log(1 + tf(i,d)) \cdot \log \frac{\#docs}{df(i)}$$

simplest form of s_{auth}: indegree of page (see link analysis)

quality of ranking function empirically evaluated by measures like:
precision, recall, F1, MAP, NDCG, etc.

Further Ranking Criteria for Web IR

Google's US patent 20050071741 (see http://appft1.uspto.gov)
on ranking criteria:

- page inception date (e.g. domain registration or first crawl)
- change frequency and amount of page content change
- appearance and disappearance of links to a page
- change frequency of anchor texts
- freshness and churn of links and trust in links
- click-through rate of query-result pages (incl. purchased results)
- shift in keyword interpretation (e.g. 9-11)
- user behavior (e.g. via toolbar)

etc. etc.

many speculations about use of these criteria,
likely to be considered for combatting spam (and for personalization?)

Principled Ranking by Probabilistic IR

binary features, conditional independence of features [Robertson & Sparck-Jones 1976]

related to but different from
statistical language models

„God does not play dice.“ (Einstein)
IR does.

$$s(d,q) = \frac{P[d \in R(q) \mid \textit{contents of } d]}{P[d \notin R(q) \mid \textit{contents of } d]} = \frac{P[R|d]}{P[\overline{R}|d]}$$

odds for item d with terms d_i being relevant for query $q = \{q_1, \ldots, q_m\}$

$$\sim \prod_{i=1}^{m} \frac{P[d_i|R]}{P[d_i|\overline{R}]} \quad \sim \sum_{i \in q \cap d} \log \frac{p_i}{1-p_i} + \log \frac{1-q_i}{q_i}$$

with $p_i = P[d_i \mid R]$
$q_i = P[d_i \mid \overline{R}]$

Now estimate p_i and q_i values from
- relevance feedback,
- pseudo-relevance feedback,
- corpus statistics

by **MLE** (with statistical **smoothing**)
and store precomputed p_i, q_i in index

Principled Ranking by Probabilistic IR

binary features, conditional independence of features [Robertson & Sparck-Jones 1976]

related to but different from
statistical language models

„God does not play dice.“ (Einstein)
IR does.

$$s(d,q) = \frac{P[d \in R(q) \mid contents\ of\ d]}{P[d \notin R(q) \mid contents\ of\ d]} = \frac{P[R|d]}{P[\overline{R}|d]}$$

odds for item d with terms d_i being relevant for query $q = \{q_1, \ldots, q_m\}$

Relationship to tf*idf

$$\approx \sum_i \log \frac{tf(i,d)}{\sum_k (k,d)} \cdot \frac{\sum_k df(k)}{df(i)}$$

with $p_i = P[d_i \mid R]$
$q_i = P[d_i \mid \overline{R}]$

Now estimate p_i and q_i values from
- relevance feedback,
- pseudo-relevance feedback,
- corpus statistics

by **MLE** (with statistical **smoothing**)
and store precomputed p_i, q_i in index

$$\hat{p}_i = (\#rel.\ docs)/\#docs$$

$$\hat{p}_i = \frac{tf(i,d)}{\sum_k tf(k,d)}$$

$$q_i \approx P[d_i \mid corpus]$$

$$\hat{q}_i = \frac{df(i)}{\sum_k df(k)}$$

max planck institut informatik

Principled Ranking by Probabilistic IR

Generalize term weight $\log \frac{p_i(1-q_i)}{q_i(1-p_i)}$ into $\log \frac{p_{tf(i)} \cdot q_0}{q_{tf(i)} \cdot p_0}$

finally leads to **Okapi BM25** [Robertson/Walker 1994]:

$$w_i(d) := \frac{(k_1+1)tf(i,d)}{k_1((1-b)+b\frac{len(d)}{avglen})+tf(i,d)} \cdot \log \frac{N-df(i)+0.5}{df(i)+0.5}$$

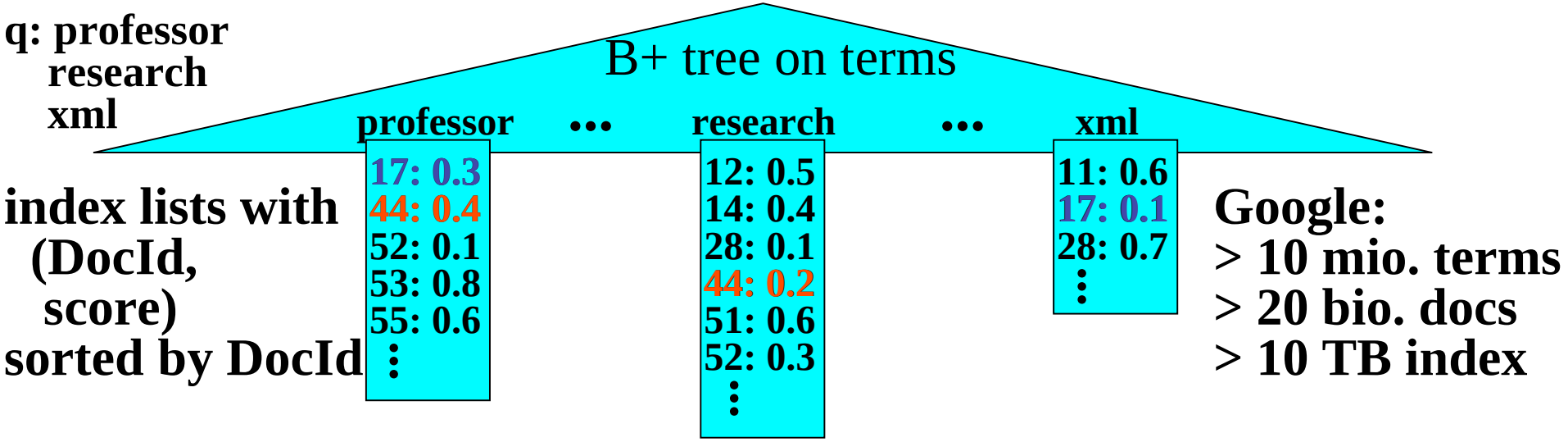
(cf. also pivoted weighting model by A. Singhal et al.)

- adds better length normalization, reduces bias towards long docs
- dampens tf and df, balances score masses from multiple terms
- often wins benchmark tasks (TREC)
- applicable to XML IR and performing well:
  - element-type-specific „df values“ couple tag & term in element
  - score aggregation over multiple elements per document

Examples: **… //Person [Max Planck]**
… //Abstract [Transaction SOAP]
… //Journal//Title [ACM Transaction]

Indexing with Inverted Lists

Vector space model suggests **term-document matrix**,
but data is sparse and queries are even very sparse
→ better use **inverted index lists** with terms as keys for B+ tree

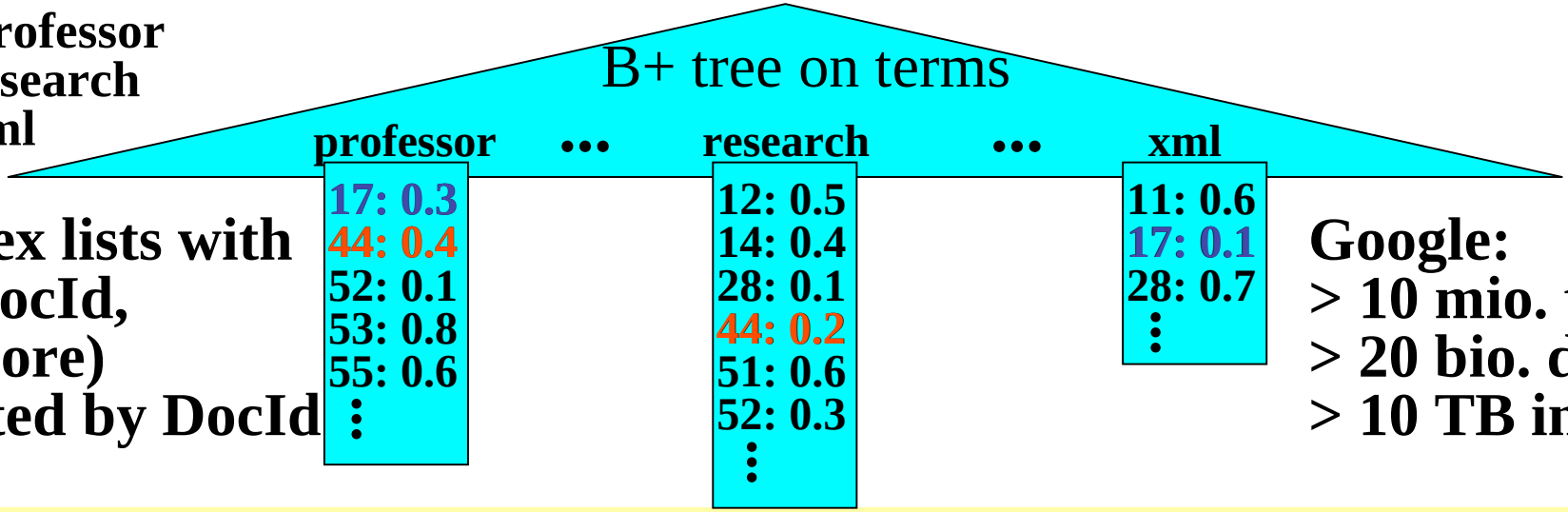


terms can be full words, word stems, word pairs, substrings, N-grams, etc.
(whatever „dictionary terms“ we prefer for the application)

- index-list entries in **DocId order** for fast Boolean operations
- many techniques for excellent **compression** of index lists
- additional **position index** needed for phrases, proximity, etc.
  (or other precomputed data structures)

Query Processing on Inverted Lists

q: professor
research
xml



index lists with (DocId, score) sorted by DocId

Google:
> 10 mio. terms
> 20 bio. docs
> 10 TB index

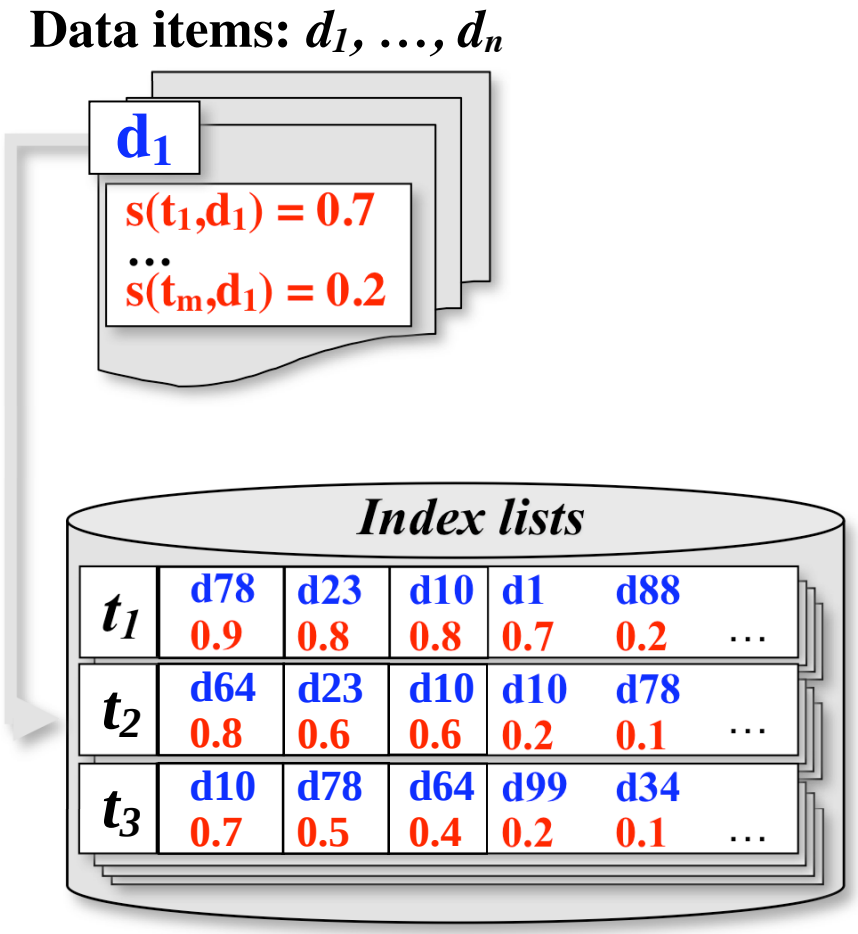
Given: query $q = t_1\ t_2\ ...\ t_z$ with z (conjunctive) keywords
similarity scoring function score(q,d) for docs $d \in D$, e.g.: $\vec{q} \cdot \vec{d}$
with precomputed scores (index weights) $s_i(d)$ for which $q_i \neq 0$

Find: top k results w.r.t. score(q,d) $= aggr\{s_i(d)\}$ (e.g.: $\Sigma_{i \in q}\ s_i(d)$)

join&sort algorithm:

top-k (
$\sigma[term=t_1]$ (index) $\times_{DocId}$
$\sigma[term=t_2]$ (index) $\times_{DocId}$
...
$\sigma[term=t_z]$ (index) $\times_{DocId}$
order by s desc)

Indexing with Score-ordered Lists



index-list entries stored in descending order of per-term score (impact)

aims to avoid having to read entire lists
rather scan only (short) prefixes of lists

Query Processing on Score-ordered Index Lists

Top-k aggregation query over ***R (Item, A1, ..., Am)*** partitions:

*Select **Item, s(R1.A1, ..., Rm.Am)** As **Aggr***
*From Outer Join **R1, …, Rm***
*Order By **Aggr** Limit k*

with **monotone s**: $(\forall\ x_i \geq x_i') \Rightarrow s(x_1 \ldots x_m) \geq s(x_1' \ldots x_m')$

- Precompute (index) **lists sorted in desc attr-value order** (score-ordered, impact-ordered)
- Scan lists by **sorted access (SA)** in round-robin manner
- Perform **random accesses (RA) by Item** when convenient
- Compute aggregation s **incrementally** in **accumulators**
- Stop when **threshold test** guarantees correct top-k (or when heuristics indicate „good enough“ approximation)

Simple & elegant, DB-oriented, theory underpinnings
Can also be adapted & extended to distributed system

Overview

- **Part 1: Web IR**
  - ✓ State of the Art
  - **Scalability Challenge**
  - Quality Challenge
  - Personalization
  - Research Opportunities
- **Part 2: Semantic & Social IR**
  - Ontologies in XML IR
  - Entity Search and Ranking
  - Graph IR
  - Web 2.0 Search and Mining
  - Research Opportunities

Scalability Challenge [Baeza-Yates et al. 2007]

Web index is huge and still growing:
> 10 Mio. terms, > 20 Bio. pages, > 10 TB

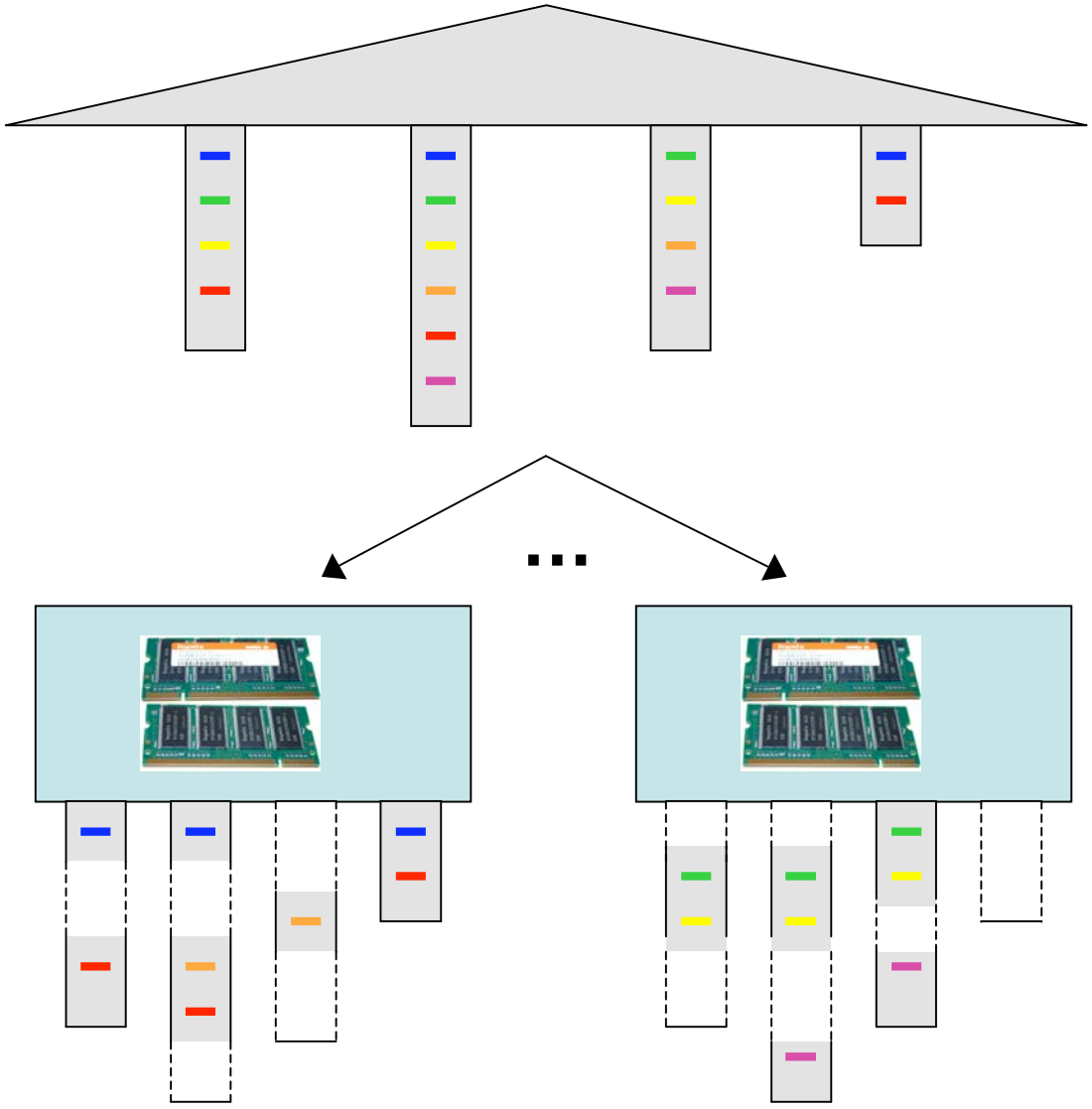
Deep Web, Web 2.0, Web Archive are even bigger!
(complete archive may be > 10 times larger than Web)

Web indexing needs to meet challenging constraints:

- index size
- query throughput
- response time guarantee
- reliability and availability
- freshness guarantee (index maintenance)

→ **Distributed Systems** (server farms, P2P networks ?, …)

Document-Partitioned Index

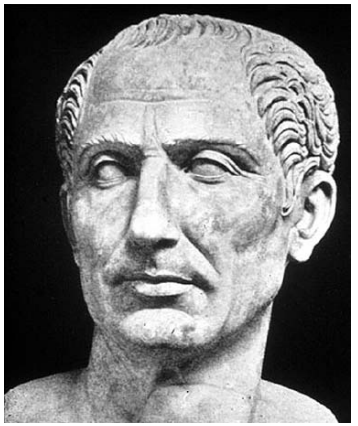


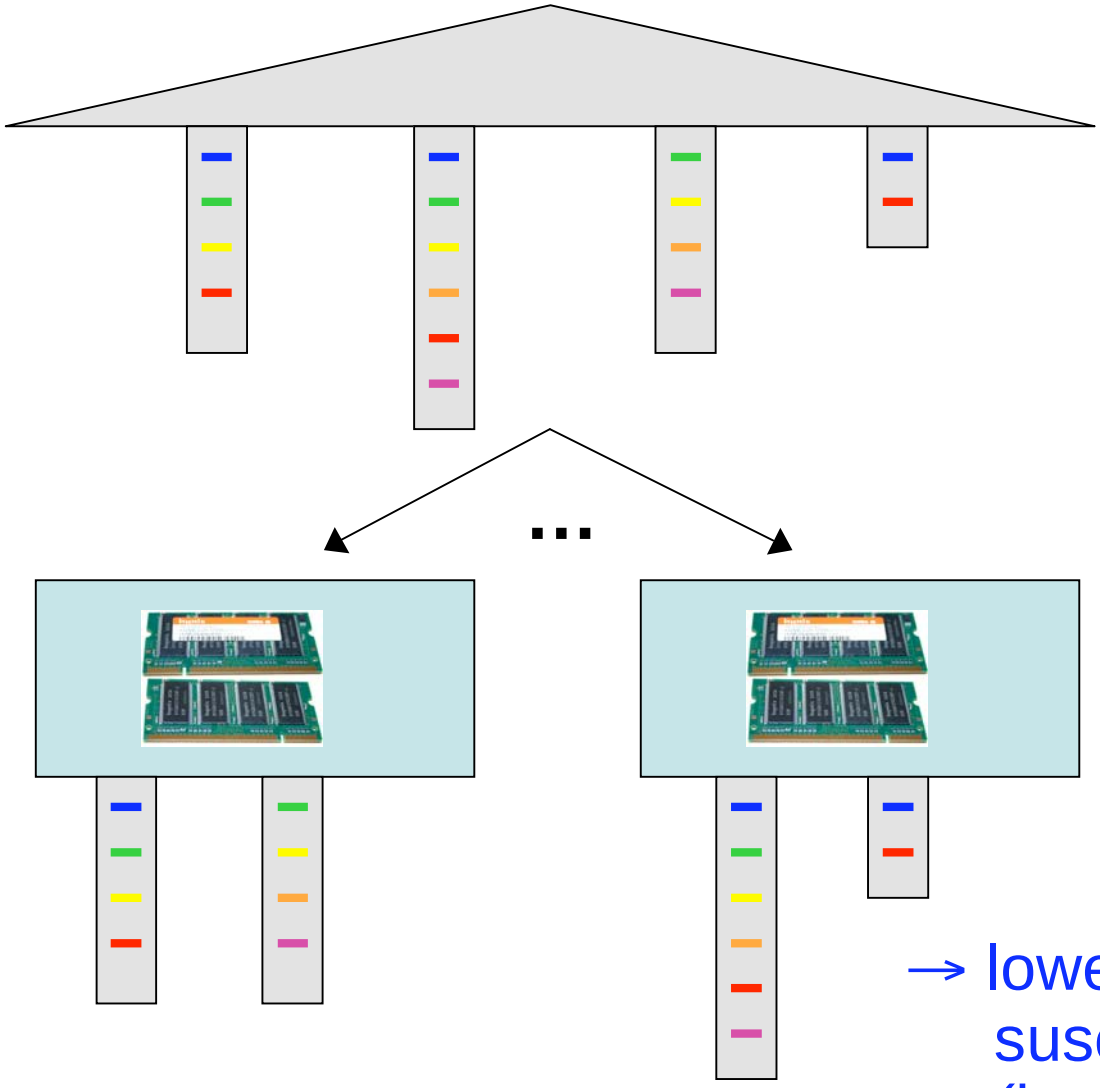
index-list entries are hashed onto nodes by DocId

each complete query is run on each node; results are merged

→ perfect load balance, embarrasingly scalable, easy maintenance

Term-Partitioned Index



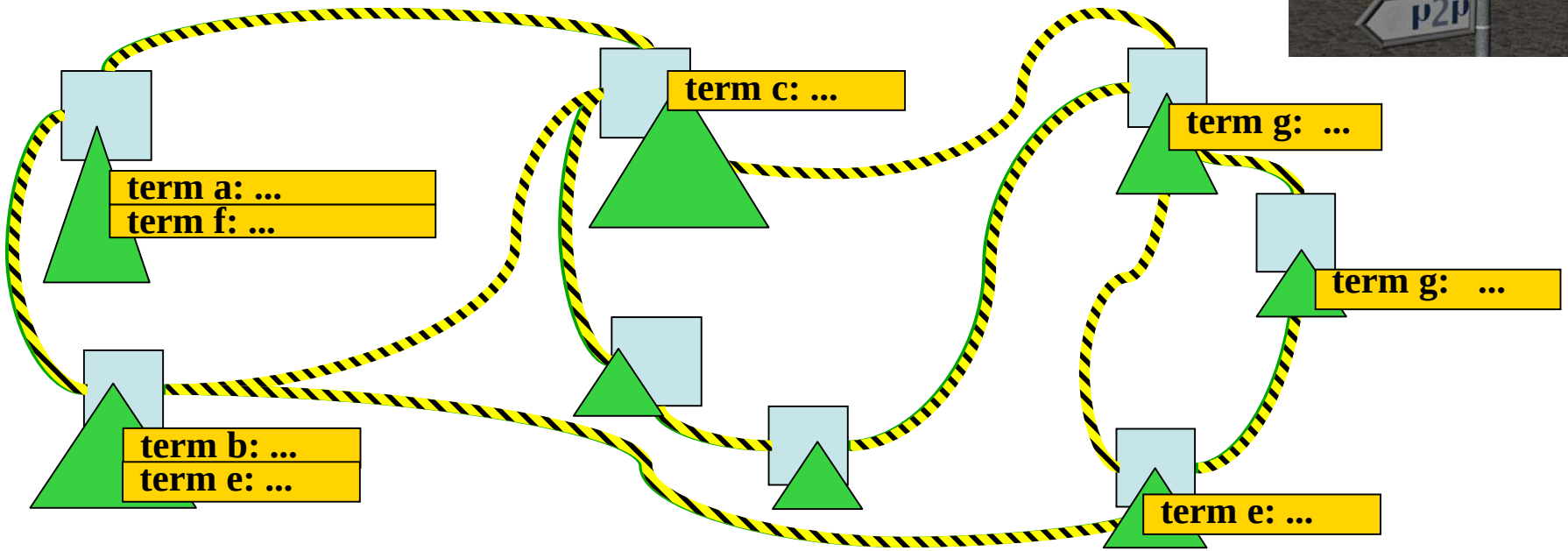


entire index lists are hashed onto nodes by TermId

queries are routed to nodes with relevant terms

→ lower resource consumption, susceptible to imbalance (because of data or load skew), index maintenance non-trivial

Peer-to-Peer Networks for Distributed Indexing



- index entries hashed onto nodes by TermId
- overlay network based on distributed hash table (DHT) with O(log n) key lookup, failure-resilience, replication
- queries are routed to nodes with relevant terms

similar to term partitioning,
but additional issues of latency, dynamics, system mgt.

Peer-to-Peer Networks for Distributed Indexing

Variations and generalizations:

- employ P2P network in **data center** vs.
  use volunteers' **home computers** over WAN
- can use many enhancements from P2P systems
  for low latency, load balancing, reliability & availability
- index partitioning could be derived from **document clustering**
  - based on document-content terms or
  - on document appearance in query results & result clicks

  → queries are routed to **most similar clusters**
- each peer could autonomously index its **own local content**

  → query routing finds **best peers** („metasearch")

But: scalable P2P for Web search remains open problem

Caching

What is cached?

- index lists for individual terms
- entire query results
- postings for multi-term intersections

Where is an item cached?

- in RAM of responsible server-farm node
- in front-end accelerators or proxy servers
- as replicas in RAM of all (many) server-farm or P2P nodes

When are cached items dropped?

- estimate for each item: temperature = access-rate / size
- when space is needed, drop item with lowest temperature
  Landlord algorithm [Cao/Irani 1997, Young 1998], generalizes LRU-k [O'Neil 1993]
- prefetch item if its predicted temperature is higher than the temperature of the corresponding replacement victims

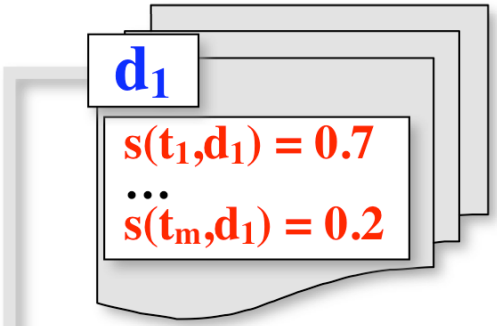
Threshold Algorithm (TA) for QP [Fagin 01, Güntzer 00, Nepal 99, Buckley 85]

simple & DB-style;
needs only O(k) memory

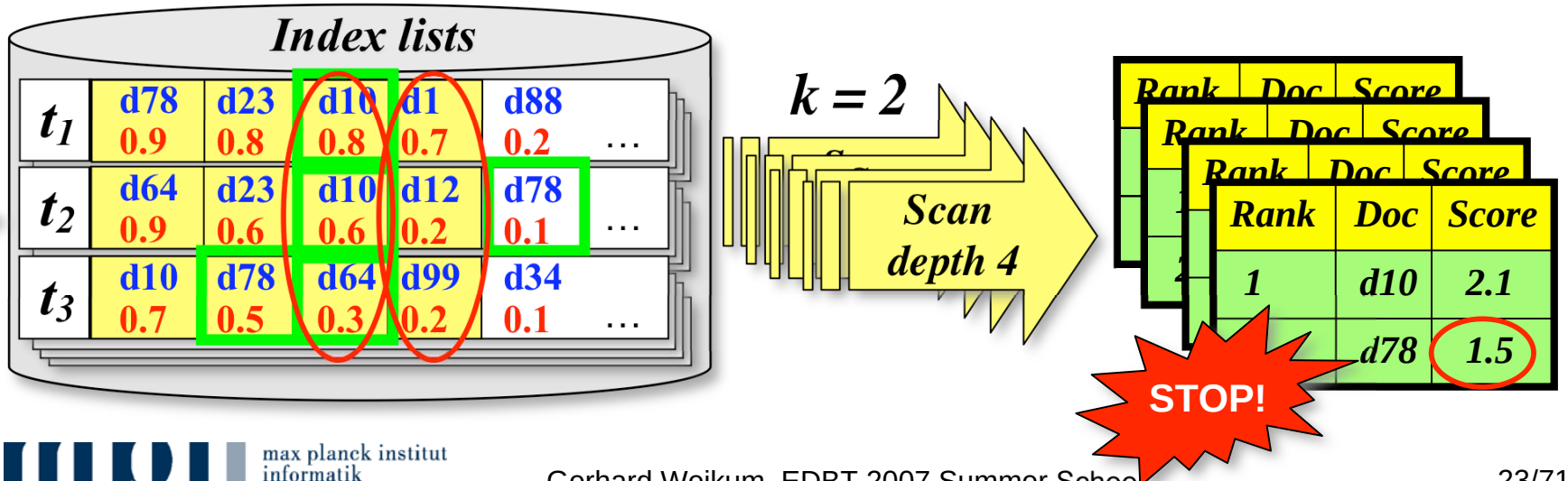
Threshold algorithm (TA):

scan index lists; consider d at pos_i in L_i;
$high_i := s(t_i,d)$;
if d ∉ top-k then {
look up $s_v(d)$ in all lists L_v with $v \neq i$;
score(d) := aggr $\{s_v(d) \mid v=1..m\}$;
if score(d) > min-k then
add d to top-k and remove min-score d';
min-k := min{score(d') | d' ∈ top-k};
threshold := aggr $\{high_v \mid v=1..m\}$;
if threshold ≤ min-k then exit;

Data items: $d_1, \ldots, d_n$



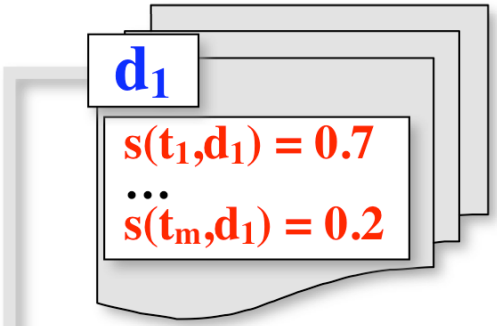
Query: $q = (t_1, t_2, t_3)$



TA with Sorted Access Only (NRA) [Fagin 01, Güntzer et al. 01]

sequential access (SA) faster
than random access (RA)
by factor of 20-1000

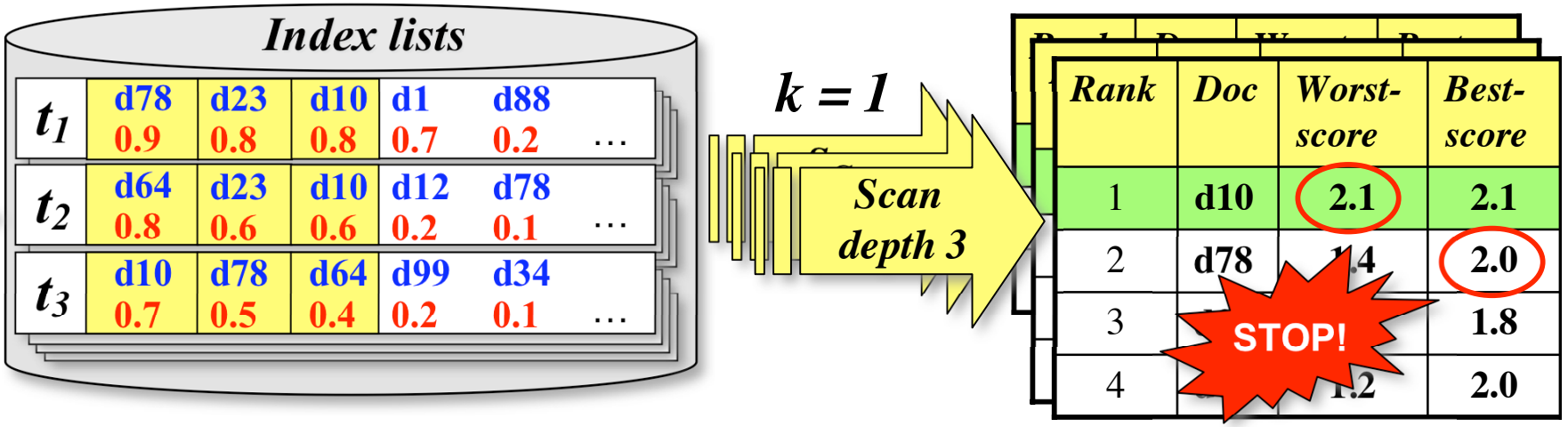
Data items: $d_1, \ldots, d_n$



Query: $q = (t_1, t_2, t_3)$

No-random-access algorithm (NRA):

scan index lists; consider d at pos_i in L_i;
$E(d) := E(d) \cup \{i\}$; $high_i := s(t_i,d)$;
worstscore(d) := aggr{$s(t_v,d) \mid v \in E(d)$};
bestscore(d) := aggr{worstscore(d),
aggr{$high_v \mid v \notin E(d)$}};
if worstscore(d) > min-k then add d to top-k
min-k := min{worstscore(d') | d' ∈ top-k}
else if bestscore(d) > min-k then
cand := cand ∪ {d};
threshold := max {bestscore(d') | d'∈ cand};
if threshold ≤ min-k then exit;



Implementation Reality

- **Limitation of asymptotic complexity:**
  - **m** (#lists) and **k** (#results) are important **parameters**
- **Priority queues:**
  - straightforward use of Fibonacci heap has **high overhead**
  - better: **periodic rebuild** of bounded-size PQs
- **Memory management:**
  - **peak memory use** as important for performance as scan depth
  - aim for **early candidate pruning** even if scan depth stays the same
- **Hybrid block index:**
  - pack index entries into big blocks in desc score order
  - keep **blocks in score order**
  - keep **entries within a block in item id order**
  - after each block read: merge-join first, then PQ update

Approximate Top-k Queries

- ***IR heuristics*** for impact-ordered lists [Anh/Moffat: SIGIR'01]:
  Accumulator Limiting, Accumulator Thresholding

- **Approximation TA** [Fagin et al.: JCSS'03]:
  θ-approximation T' for q with $\theta > 1$ is a set T' of items with:
  - $|T'|=k$ and
  - for each $d' \in T'$ and each $d'' \notin T'$: $\theta * score(q,d') \geq score(q,d'')$

  Modified TA:
  ... stop when **min-k ≥ aggr ($high_1$, ..., $high_m$) / θ**

- **Probabilistic Top-k** [Theobald et al.: VLDB'04]:
  guarantee small deviation from exact top-k result
  with high probability

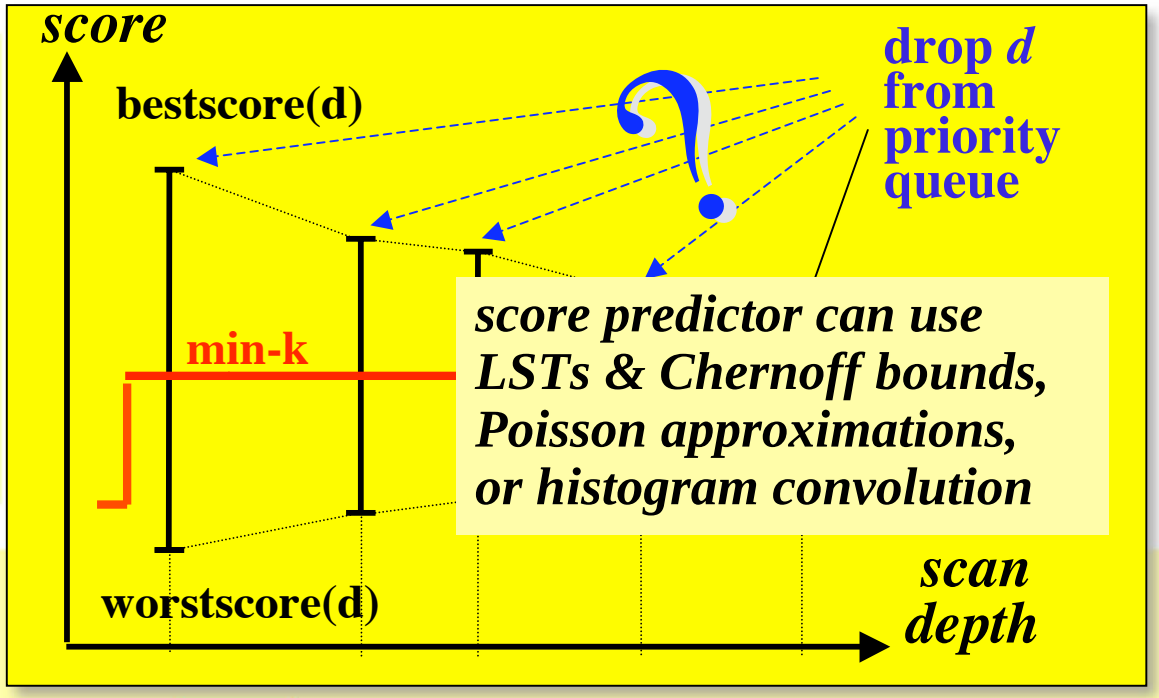
Probabilistic Top-k [Theobald et al.: VLDB'04]

TA family of algorithms based on invariant (with sum as aggr):

$$\underbrace{\sum_{i \in E(d)} s_i(d)}_{\text{worstscore(d)}} \leq s(d) \leq \underbrace{\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} high_i}_{\text{bestscore(d)}}$$

- Add *d* to top-k result, if *worstscore(d)* > *min-k*
- Drop *d* only if *bestscore(d)* < *min-k*, otherwise keep in PQ

→ Often overly conservative (deep scans, high memory for PQ)



→ Approximate top-k with probabilistic guarantees:

$$p(d) := P\left[\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} S_i > \delta\right]$$

discard candidates d from queue if p(d) ≤ ε ⇒ E[rel. precision@k] = 1–ε

Combined Algorithm (CA) for Balanced SA/RA Scheduling [Fagin et al. 03]

cost ratio $C_{RA}/C_{SA} = r$

perform **NRA** (TA-sorted)
...
after every r rounds of SA (m*r scan steps)
perform **RA** to look up all missing scores of „**best candidate**“ in Q

cost **competitiveness** w.r.t. „optimal schedule“
(scan until $\Sigma_i \text{high}_i \leq \min\{\text{bestscore}(d) \mid d \in$ **final top-k**$\}$,
then perform RAs for all d‘ with bestscore(d‘) > min-k): **4m + k**

IO-Top-k Scheduling [Bast et al.: VLDB'06]

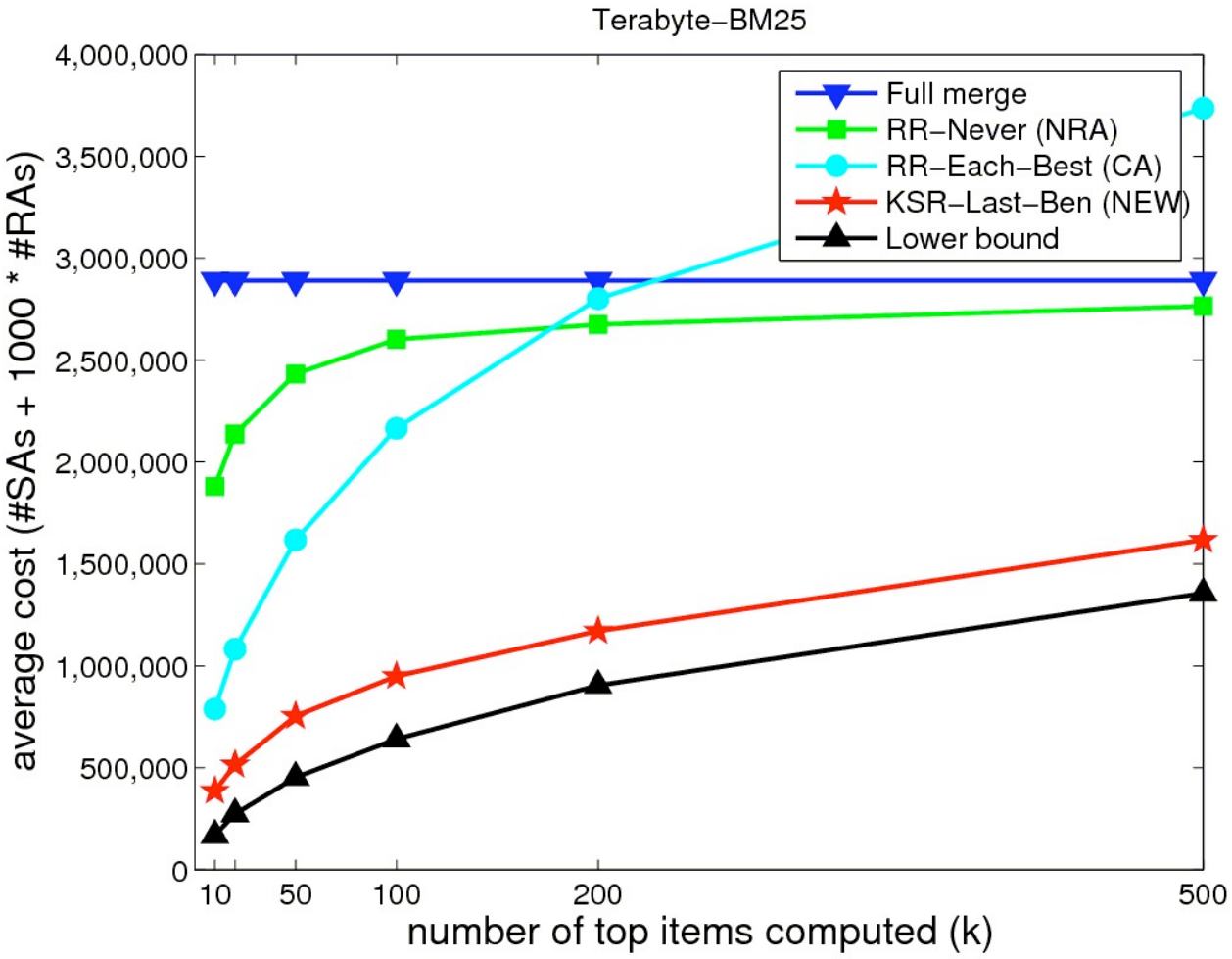
For **SA scheduling** plan next $b_1, ..., b_m$ index scan steps
for **batch of b steps** overall s.t. $\Sigma_{i=1..m} \; b_i = b$
and benefit($b_1, ..., b_m$) is max!

solve **knapsack-style** NP-hard problem for batched scans,
or use greedy heuristics

Perform **additional RAs** when helpful
1) to increase min-k (increase worstscore of $d \in$ top-k) or
2) to prune candidates (decrease bestscore of $d \in Q$)

Last Probing (2-Phase Schedule):
perform RAs for all candidates at point t when
total cost of remaining RAs = total cost of SAs up to t
with **score-prediction & cost model** for deciding **RA order**

Performance of SA/RA Scheduling Methods



absolute run-times for TREC'04 Robust queries on Terabyte .Gov data: (C++, Berkeley DB, Opteron, 8 GB):

- **full merge:**
  170 ms per query
- **RR-Never (NRA):**
  155 ms for k=10
  195 ms for k=50
- **RR-Last-Best (NEW):**
  30 ms for k=10
  55 ms for k=50

Example query: *kyrgyzstan united states relation*
15 mio. list entries, NEW scans 2% and performs 300 RAs for 10 ms response time

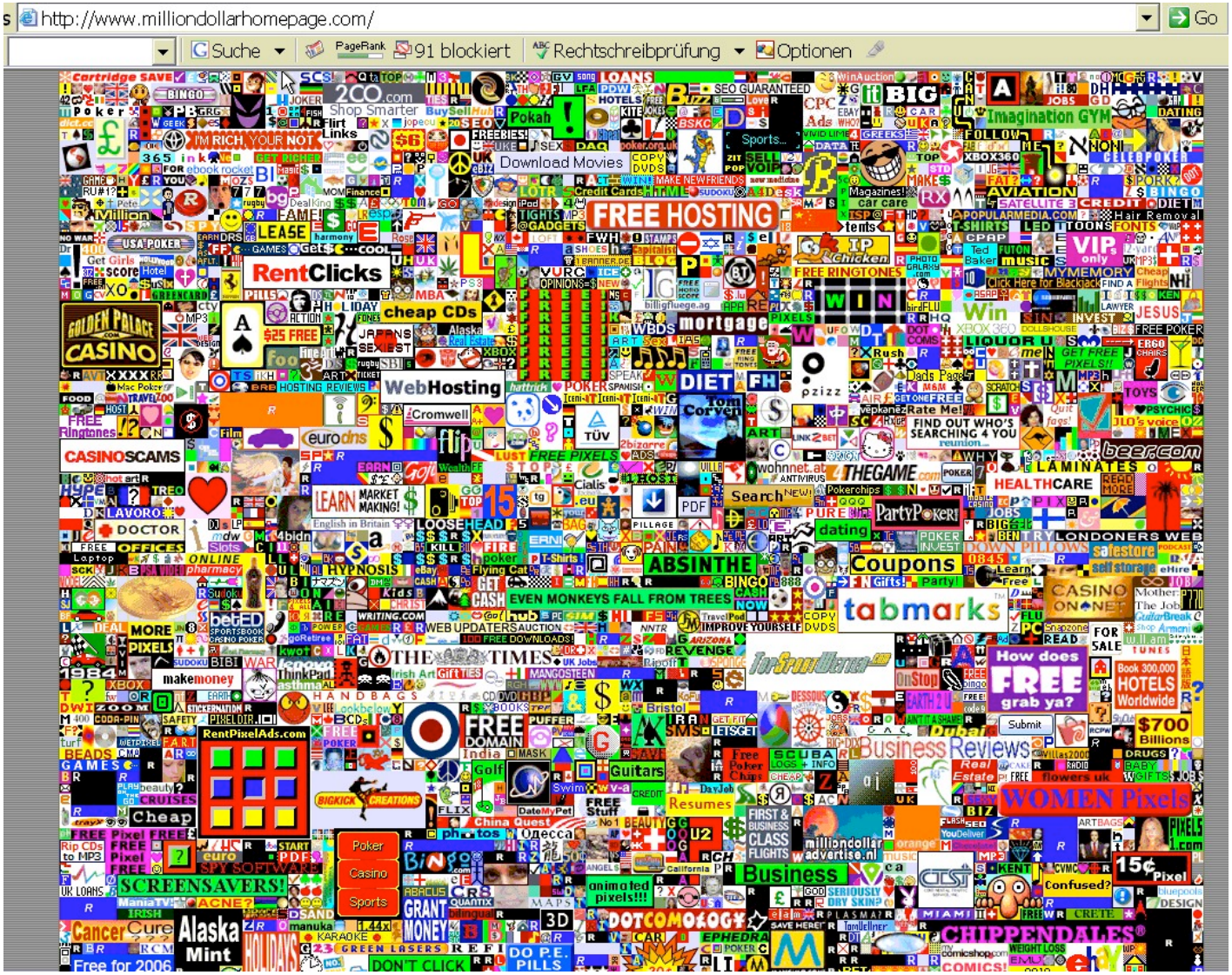
Summary: Scalability Challenge

- Scalable solution needed for **data size & load**
  and future **growth** (and enhancing functionality)
- Today's commercial solution – **index partitioning by DocId**
  in **RAM** of large server farm – works very well, but is expensive
- Alternative **distributed & P2P architectures** should be studied,
  need to solve **load balancing** and **index maintenance**
- **Caching** and efficient **top-k query processing**
  have mature algorithmics, could be strong assets

Overview

- **Part 1: Web IR**
  - ✓ State of the Art
  - ✓ Scalability Challenge
  - **Quality Challenge**
  - Personalization
  - Research Opportunities
- **Part 2: Semantic & Social IR**
  - Ontologies in XML IR
  - Entity Search and Ranking
  - Graph IR
  - Web 2.0 Search and Mining
  - Research Opportunities

Quality Challenge



online casinos,
free movies,
cheap software,
buy MBA diploma,
V!-4-gra,
get rich now now now,
...

Source: www.milliondollarhomepage.com

Wisdom of Crowds: PageRank [Page/Brin 1998]

PageRank (PR): links are endorsements & increase page authority
authority is higher if links come from high-authority pages

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(p) \cdot t(p,q)$$

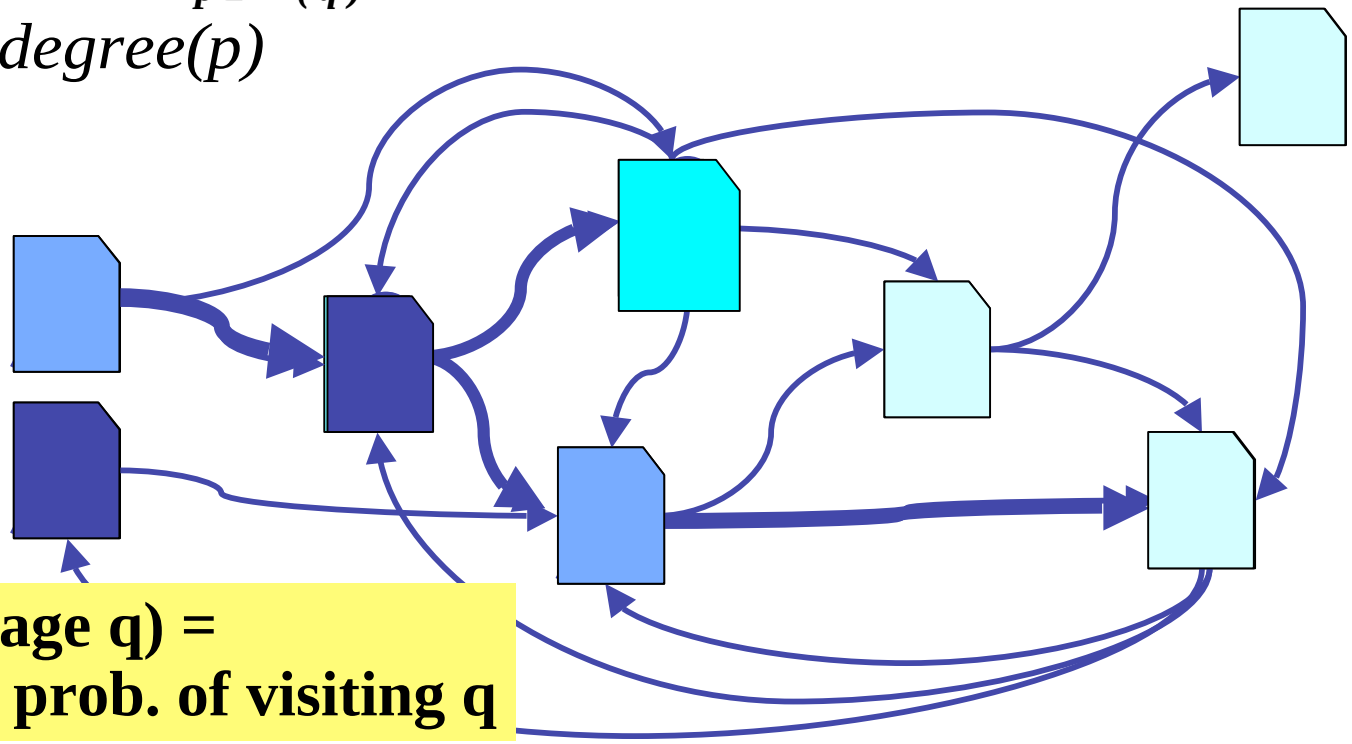
Social Ranking

with $t(p,q) = 1/outdegree(p)$

and $j(q) = 1/N$

equivalent to
principal eigenvector:

$$p_{n\times 1} = A_{n\times n} \times p_{n\times 1}$$



**Authority (page q) =
stationary prob. of visiting q**

random walk: uniformly random choice of links + random jumps;
add **bias** to transitions and jumps for personal PR, TrustRank, etc.

More Formally: Page Rank r(q)

given: directed Web graph G=(V,E) with |V|=n and
adjacency matrix A: A_{ij} = 1 if (i,j)∈E, 0 otherwise

Def.: $$r(q) = \varepsilon / n + (1-\varepsilon) \sum_{(p,q)\in G} r(p) / outdegree(p)$$
with $0 < \varepsilon \leq 0.25$

Theorem: With A'_{ij} = 1/outdegree(j) if (j,i)∈E, 0 otherwise:

$$\vec{r} = \frac{\vec{\varepsilon}}{n} + (1-\varepsilon) A' \vec{r} \quad \Leftrightarrow \quad \vec{r} = \left(\frac{\vec{\varepsilon}}{n} \vec{1}^T + (1-\varepsilon) A' \right) \vec{r}$$

$\vec{r}$ is **Eigenvector** of a modified adjacency matrix M

Iterative computation of r(q) (Power Iteration, Jacobi method):

- Initialization: $r^{(0)}(q) := 1/n$
- Improvement by: $\vec{r}^{(i+1)} = M \vec{r}^{(i)}$

typically converges after about 100 iterations

Personalized PageRank [Haveliwala et al. 2002]

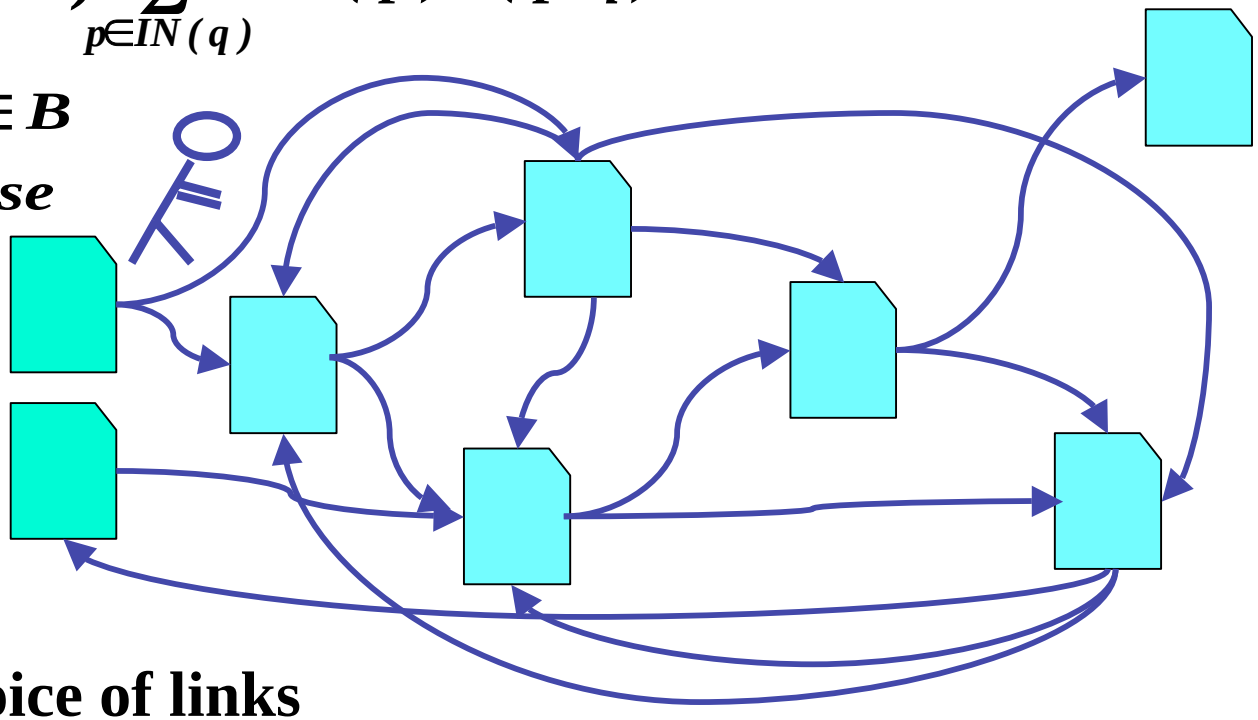
Idea: random jumps favor designated high-quality pages such as personal bookmarks, frequently visited pages, etc.

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(p) \cdot t(p,q)$$

with

$$j(q) = \begin{cases} 1/|B| & \text{for } q \in B \\ 0 & \text{otherwise} \end{cases}$$

Authority (page q) = stationary prob. of visiting q



random walk:
uniformly random choice of links
+ biased jumps to personal favorites (or trusted pages or ...)

see also: Jeh 2003, Benczur 2004, Gyöngyi 2004, Guha 2004

Efficient Computation of Personalized PR

PageRank equation: $\vec{r}_k = \varepsilon\ \vec{p}_k + (1-\varepsilon) A' \vec{r}_k$

Theorem:

Let $\mathbf{u_1}$ and $\mathbf{u_2}$ be **personal preference vectors for jump targets**, and let $\mathbf{r_1}$ and $\mathbf{r_2}$ denote the corresponding **PPR vectors**.
Then for all $\alpha_1, \alpha_2 \geq 0$ with $\alpha_1 + \alpha_2 = 1$ the following holds:

$$\boldsymbol{\alpha_1 r_1 + \alpha_2 r_2 = (1-\varepsilon) A' (\alpha_1 r_1 + \alpha_2 r_2) + \varepsilon (\alpha_1 u_1 + \alpha_2 u_2)}$$

Corollary:

For preference vector u with m non-zero components and **singleton vectors $\mathbf{e_p}$** with $(e_p)_i = 1$ for i=p, 0 for i≠p, the following holds:

$$u = \sum_{p=1}^{m} \alpha_p \cdot e_p \qquad \text{with constants } \alpha_1 \ldots \alpha_m$$

and $$r = \sum_{p=1}^{m} \alpha_p \cdot r_p \qquad \text{for PPR vector r}$$

Hubs and Authorities: HITS [Kleinberg 1999]

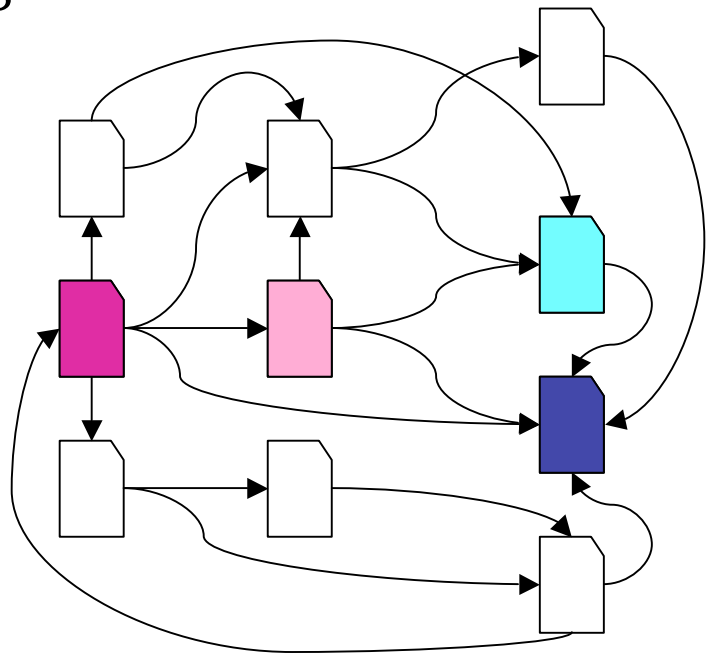
For web graph G=(V,E) and query-specific base set B ⊂ V find

good **authorities** with **authority score** $x_q = \sum_{(p,q)\in E} y_p$

and good **hubs** with **hub score** $y_p = \sum_{(p,q)\in E} x_q$

Iterative approximation of principal Eigenvectors

$$\left.\begin{aligned}\vec{x} &= A^T \vec{y} \\ \vec{y} &= A\vec{x}\end{aligned}\right\} \Rightarrow \begin{aligned}\vec{x} &:= A^T \vec{y} := A^T A \vec{x} \\ \vec{y} &:= A\vec{x} := A A^T \vec{y}\end{aligned}$$



Link Analysis: State of the Art

- Many extensions to PR and HITS:
  sink handling, edge weighting, normalization, dampening, etc.
- Many additional algorithms with subtle differences
- Many applications:
  sim-based implicit links, clicks as links, temporal authority, etc.

Mature engineering, but theory still not satisfying:

Algorithms A and B are **similar on the class $\mathcal{G}$** of graphs with ***n nodes*** under ***authority distance measure d*** if for $n \to \infty$:

$$\max \{d(A(G),B(G)) \mid G \in \mathcal{G}\} = o(\text{maxdist}(x,y \mid d, L_q, \|x\|_q=\|y\|_q=1))$$

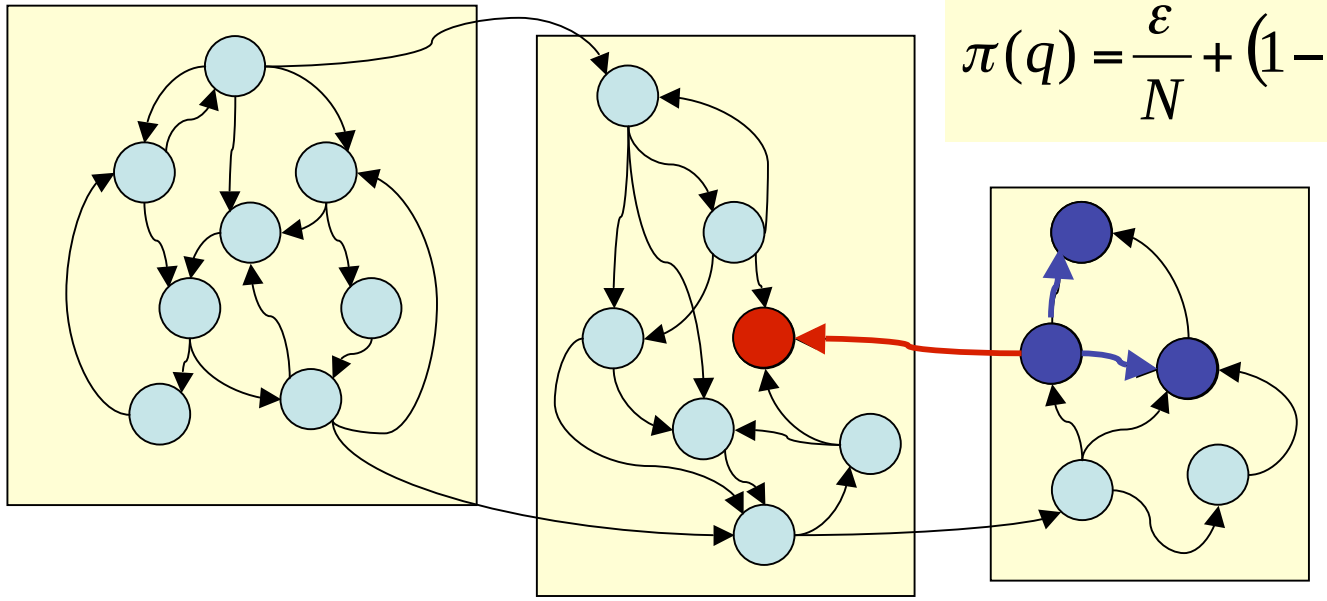
For G, G‘ the ***link distance*** d_{link} is: $d_{link}(G,G') = |(E \cup E') - (E \cap E')|$
For G let $C_k(G) = \{G' \in \mathcal{G} \mid d_{link}(G,G') \leq k\}$.
Algorithm A is **stable on the class $\mathcal{G}$** of graphs with ***n nodes*** under ***authority distance measure d*** if for every $k > 0$ for $n \to \infty$:

$$\max \{d(A(G),A(G')) \mid G \in \mathcal{G},\ G' \in C_k(G)\} = o(\text{maxdist}(x,y \mid d, L_q, \|x\|_q=\|y\|_q=1))$$

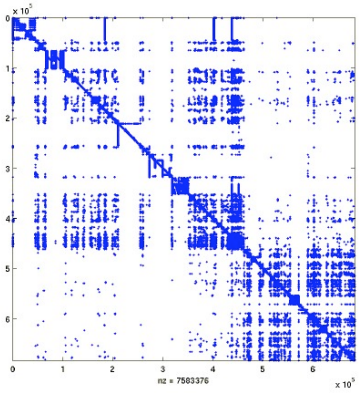
Distributed PageRank (PR)

Page authority important for final result scoring

Exploit locality in Web link graph: construct block structure (disjoint graph partitioning) based on sites or domains



$$\pi(q) = \frac{\varepsilon}{N} + (1-\varepsilon)\sum_{p \in IN(q)} \frac{\pi(p)}{out(p)}$$

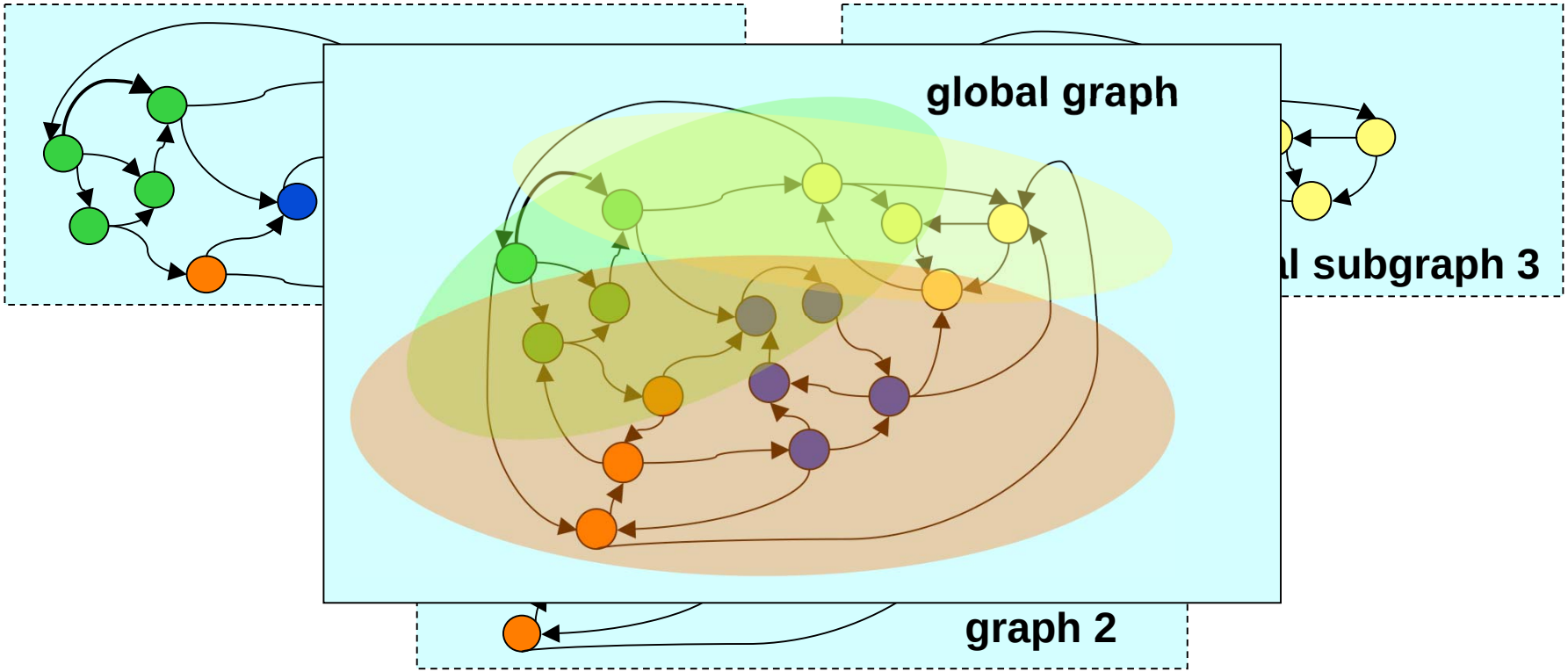


Compute page PR within site/domain & site/domain weights,

- combine page scores with site/domain scores
  [Kamvar03, Lee03, Broder04, Wang04, Wu05] or
- communicate PR mass propagation across sites
  [Abiteboul00, Sankaralingam03, Shi03, Kempe04, Jelasity05]

Decentralized PageRank (PR)

Decentralized computation in peer-to-peer network
with arbitrary, a-priori unknown **overlaps of graph fragments**



generalizable to graph spectral analysis applied to:

- pages, sites, tags, users, groups, queries, clicks, opinions, etc. as nodes
- assessment and interaction relations as weighted edges
- can compute various notions of authority, reputation, trust, quality

JXP (Juxtaposed Approximate PageRank)

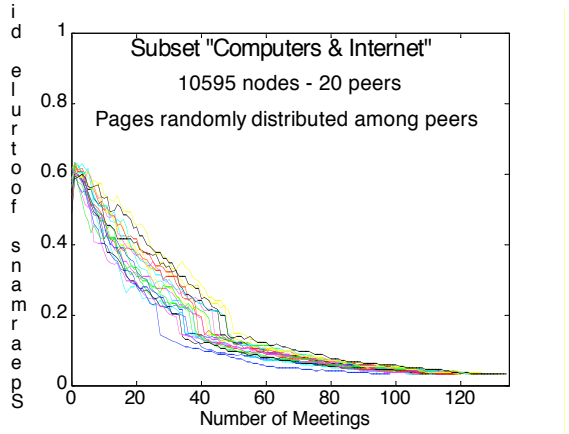
[J.X. Parreira et al.: WebDB 05, VLDB 06, VLDB Journal]

scalable, decentralized P2P algorithm based on
Markov-chain aggregation (state lumping) [Courtois 1977, Meyer 1988]

- each peer represents external, a priori unknown part of the global graph by one superstate, a **„world node“**

peers meet randomly

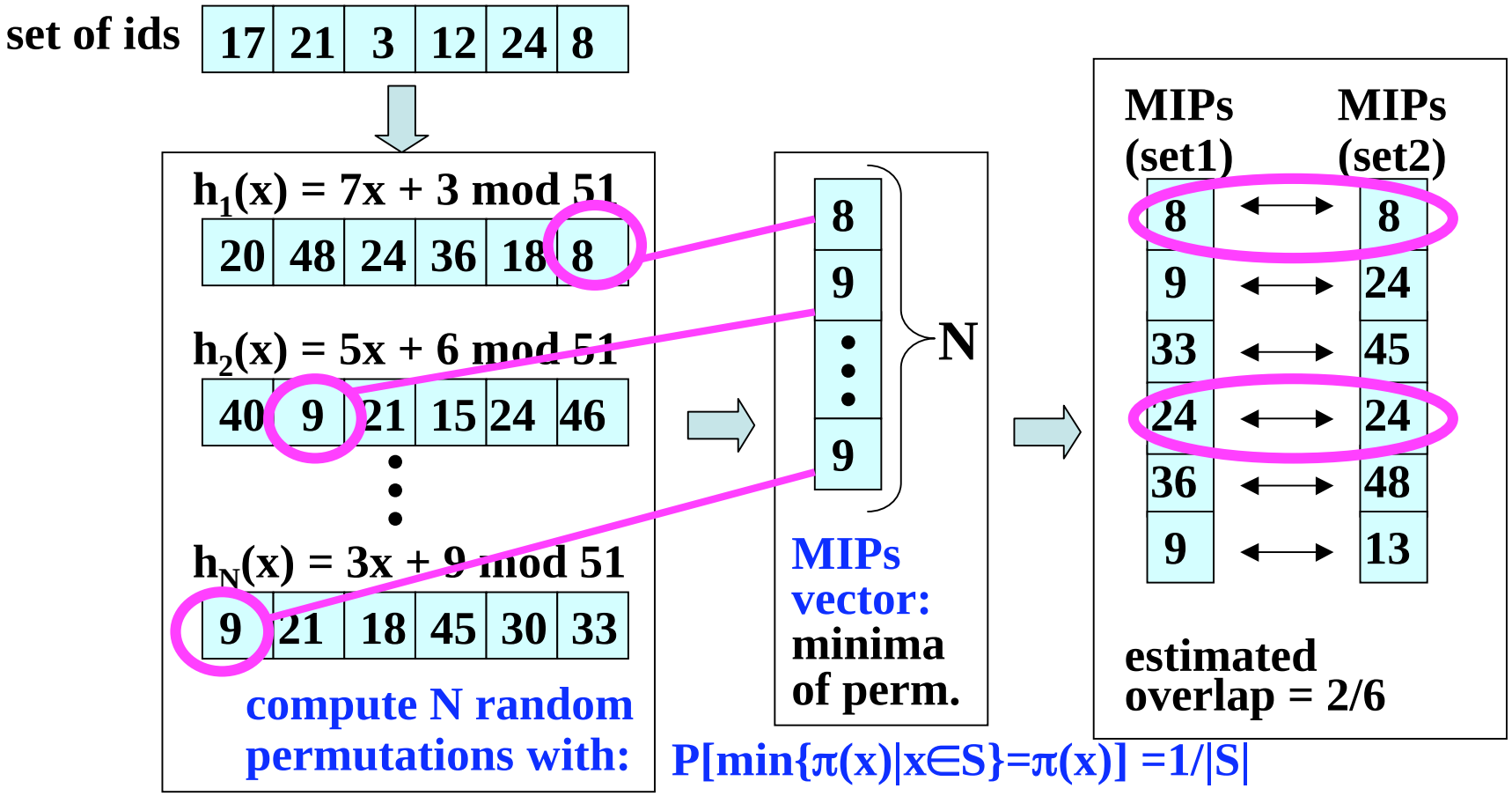
- exchange local graph fragments & PR vectors
- learn incoming edges to nodes of local graph
- compute local PR on enhanced local graph
- keep only improved PR and own local graph
- don‘t keep other peers‘ graph fragments



Theorem: JXP scores converge to global PR scores

convergence sped up by **biased p2pDating** strategy:
prefer peers whose nodeset of outgoing links
has high overlaps with our nodeset (use MIPs as synopses)

Min-Wise Independent Permutations [Broder 97]



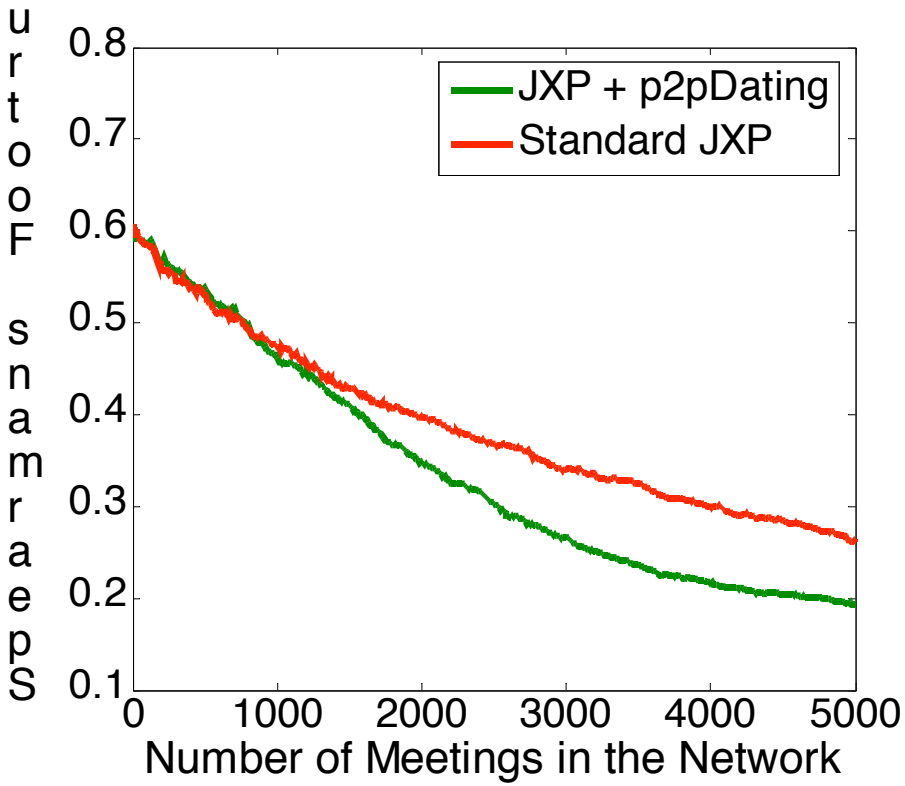
MIPs are **unbiased estimator of overlap**:

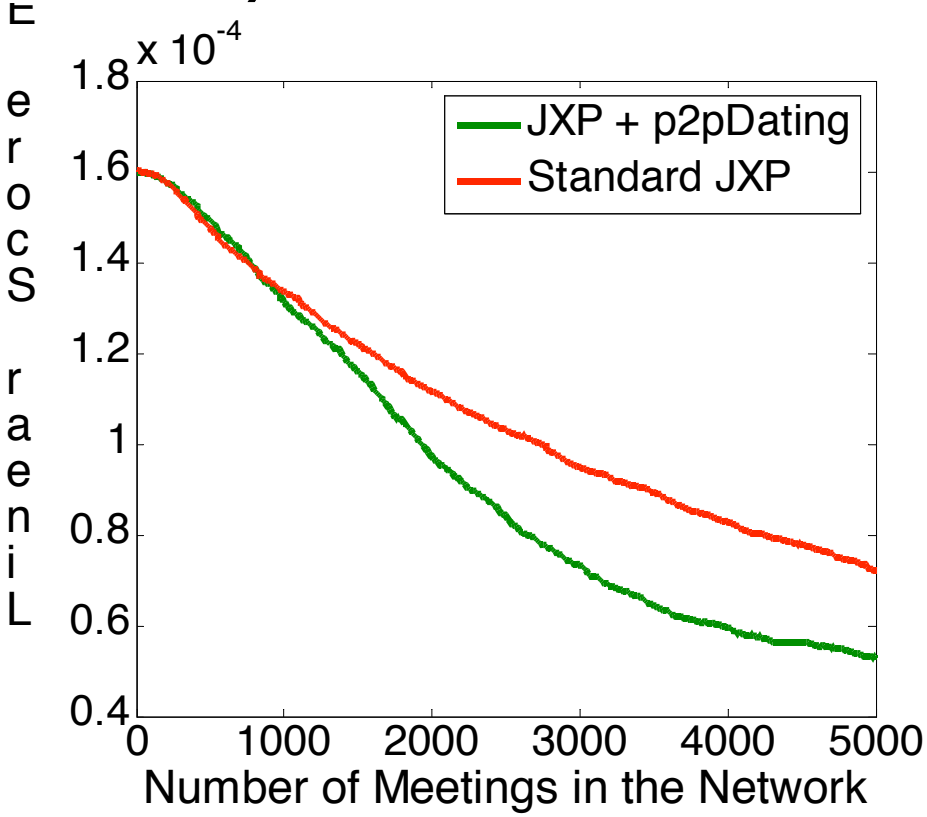
P [min {h(x) | x∈A} = min {h(y) | y∈B}] = **|A∩B| / |A∪B|**

MIPs can be viewed as repeated sampling of x, y from A, B

JXP Small-Scale Experiments

100 peers with simulated crawls of Amazon products categories (with recommended similar products as links)





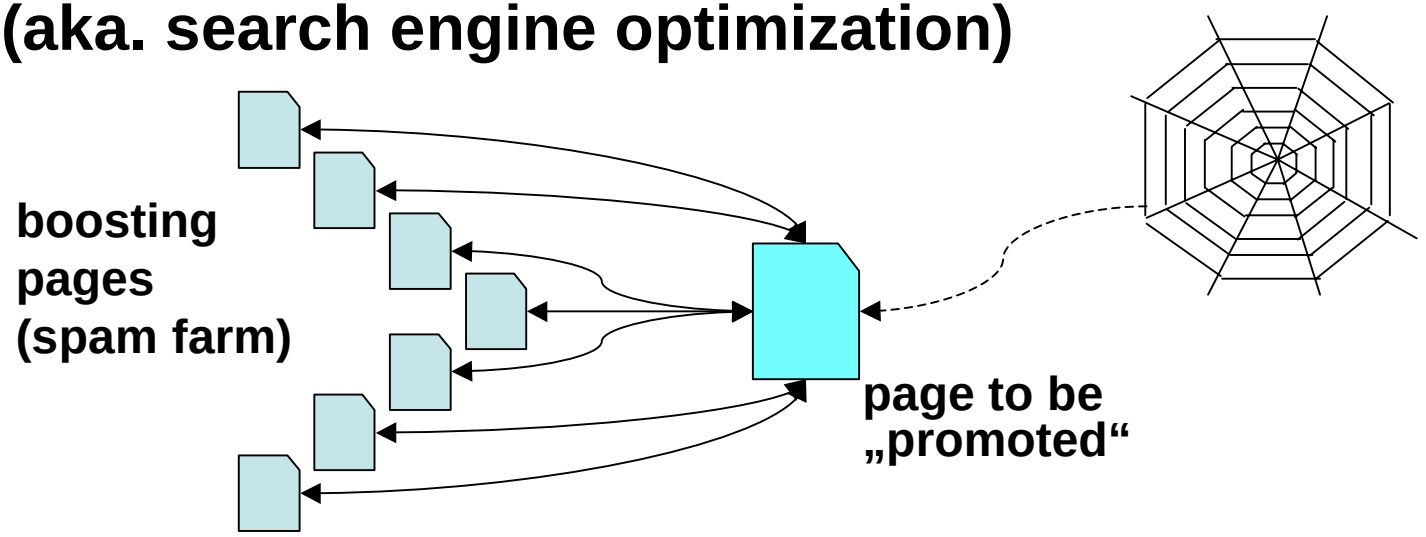
similar and more results for real Web data

also improves precision of query-result ranking, and query routing by combining quality-novelty with JXP mass

What about cheating peers?

Spam: Not Just for E-mail Anymore

Distortion of search results by „spam farms“ (aka. search engine optimization)



Susceptibility to manipulation and lack of trust model is a major problem:

- Successful 2004 DarkBlue SEO Challenge: *„nigritude ultramarine“*
- Pessimists estimate 75 Mio. out of 150 Mio. Web hosts are spam

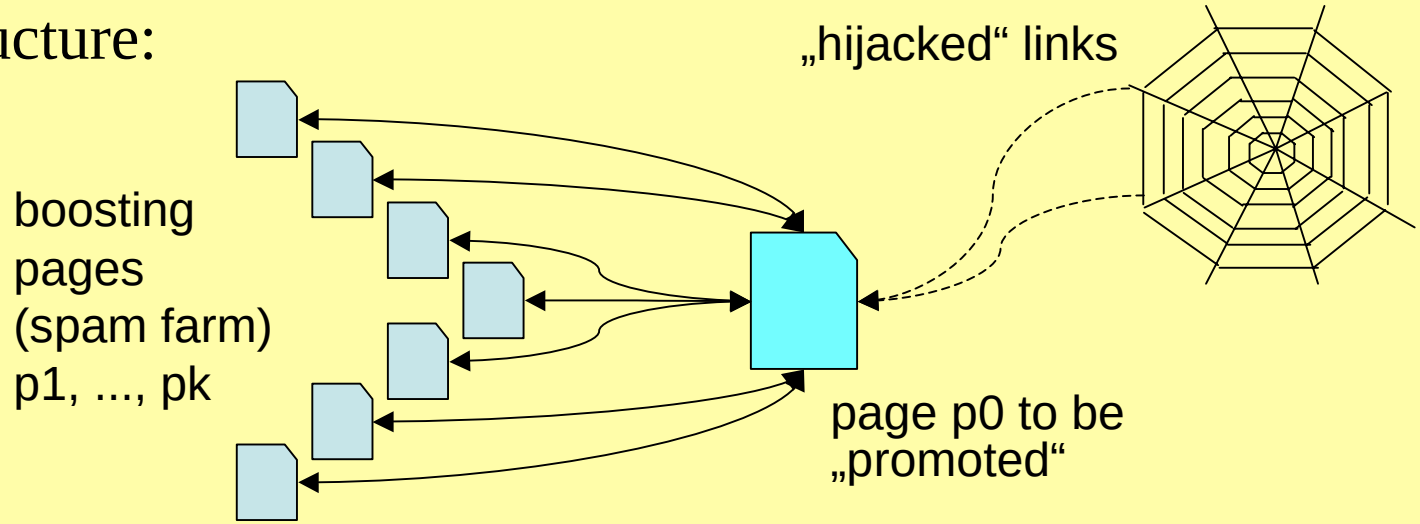
Research challenge:

- Robustness to egoistic and malicious behavior
- **Trust/Distrust** models and mechanisms

unclear borderline between spam and community opinions

Spam Farms and their Effect

Typical structure:



Web transfers to p0 the „hijacked“ score mass („leakage“)
$\lambda = \Sigma_{q \in IN(p0)-\{p1..pk\}}$ PR(q)/outdegree(q)

Theorem: p0 obtains the following PR authority:

$$PR(p0) = \frac{1}{1-(1-\varepsilon)^2}\left((1-\varepsilon)\lambda + \frac{\varepsilon((1-\varepsilon)k+1)}{n}\right)$$

The above spam farm is optimal within some family of spam farms (e.g. letting hijacked links point to boosting pages).

From PageRank to TrustRank

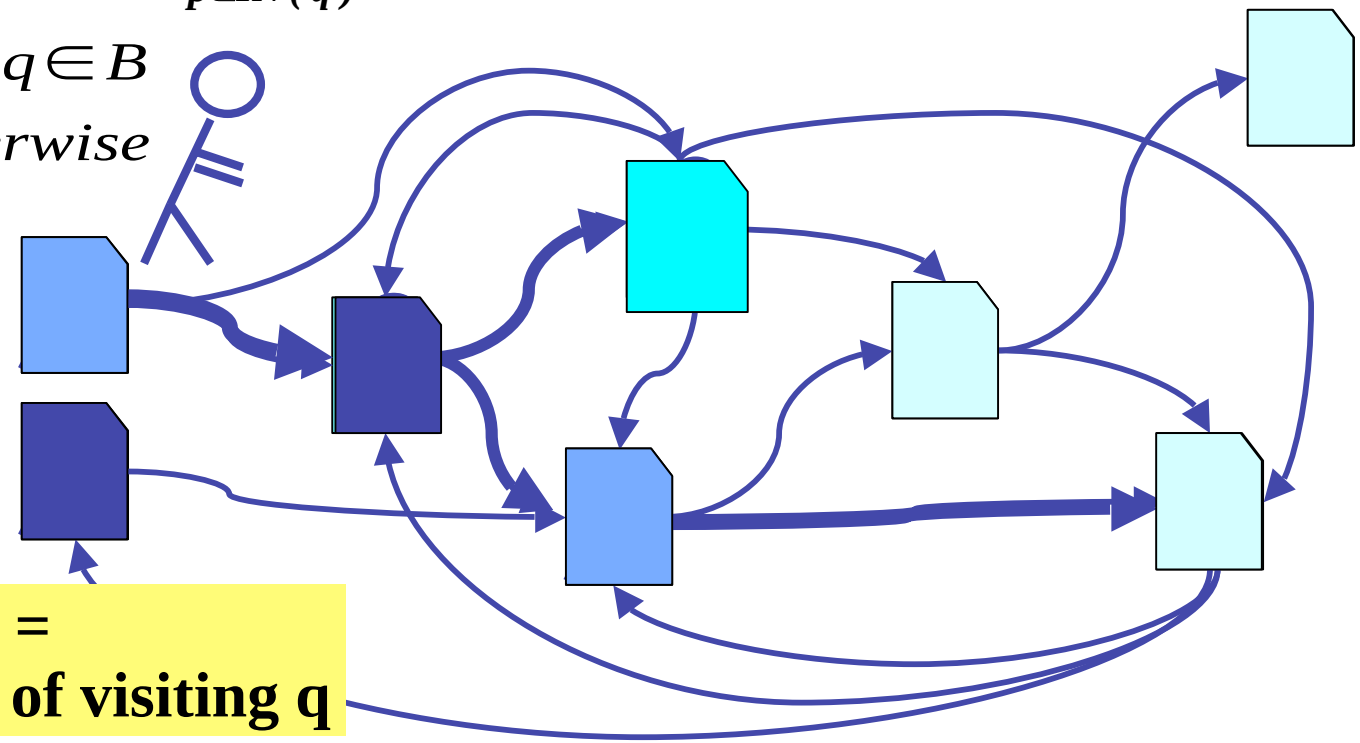
[Kamvar et al.: WWW'03, Gyöngyi et al.: VLDB'04]

Idea: random jumps favor designated high-quality pages such as bookmarks, popular pages, trusted hubs, etc.

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(p) \cdot t(p,q)$$

with

$$j(q) = \begin{cases} 1/|B| & for\ q \in B \\ 0 & otherwise \end{cases}$$



Authority (page q) = stationary prob. of visiting q

random walk: uniformly random choice of links

+ biased jumps to trusted pages

Countermeasures: TrustRank and BadRank

TrustRank:

start with explicit **set T of trusted pages** with trust values t_i

define random-jump vector r by setting $r_i = t_i /$ if i ∈B and 0 else

propagate TrustRank mass to successors

$$TR(q) = \tau\, r_q + (1-\tau) \sum_{p \in IN(p)} TR(p) / \text{outdegree}(p)$$

BadRank:

start with explicit **set B of blacklisted pages**

define random-jump vector r by setting $r_i = 1/|B|$ if i∈B and 0 else

propagate BadRank mass to predecessors

$$BR(p) = \beta\, r_p + (1-\beta) \sum_{q \in OUT(p)} BR(q) / \text{indegree}(q)$$

Problems:

maintenance of explicit lists is difficult

difficult to understand (& guarantee) effects

Spam, Damn Spam, and Statistics

Spam detection based on **statistical deviation**:

- **content spam**:
  compare the word frequency distribution
  to the general distribution in „good sites“
- **link spam:**
  find outliers in outdegree and indegree distributions
  and inspect intersection

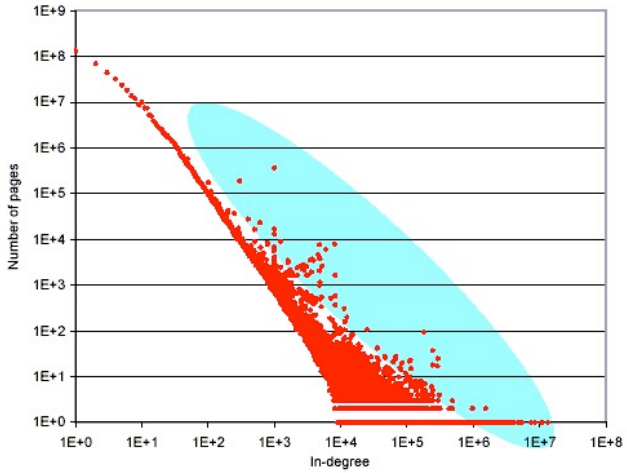


Figure 5: Distribution of in-degrees

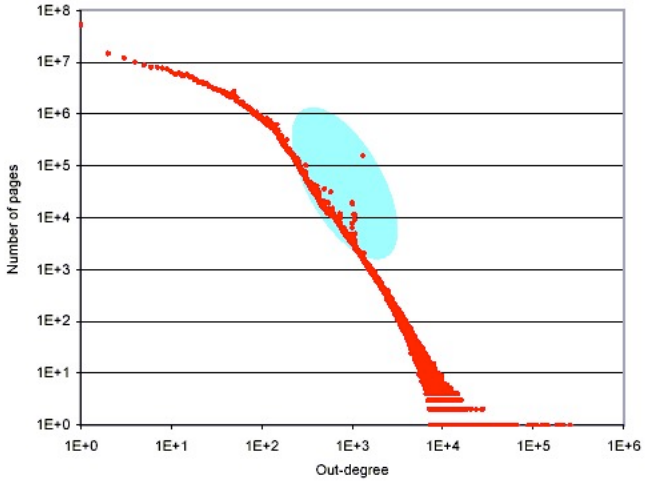


Figure 4: Distribution of out-degrees

typical for Web:
$P[\text{degree}=k] \sim (1/k)^{\alpha}$
$\alpha \approx 2.1$ for indegrees
$\alpha \approx 2.7$ for outdegrees

Source: D. Fetterly, M. Manasse, M. Najork: WebDB 2004

SpamRank [Benczur et al. 2005]

Key idea:
Inspect **PR distribution** among a suspected page's neighborhood in a power-law graph
→ should also be **power-law distributed**, and deviation is suspicious
(e.g. pages that receive their PR from many low-PR pages)

3-phase computation:

1) for each page q and supporter p compute approximate PPR(q) with random-jump vector r_p=1 and 0 otherwise
   $PPR_p(q)$ is interpreted as **support of p for q**
2) for each page p compute a penalty based on PPR vectors
3) define one PPR vector with penalties as random-jump prob's and compute SpamRank as „personalized" BadRank

true authority = PageRank - SpamRank

SpamRank Phase 1 Details

$PPR_p(q)$ with singleton random-jump vector

= probability that a random tour starting at p visits q:

$$PPR_p(q) = \sum_{\substack{\text{tours } t: \\ p \to q}} P[t]\varepsilon(1-\varepsilon)^{length(t)}$$

(geometric distr. tour length)

$$P[t : w_1 w_2 ... w_k \ \ of\ length\ k-1] = \prod_{i=1}^{k-1} 1/\text{outdegree}(w_i)$$

approximate $PPR_p(q)$ vectors by Monte Carlo simulation:

- generate tours of length 1, 2, etc.
- generate random tour starting at p,
- count as „success“ with geometr. weight if q is end point

ratio of „success“ to „trials“ is unbiased estimator of $PPR_p(q)$

Learning Spam Features [Drost/Scheffer 2005]

Use classifier (e.g. Bayesian predictor, SVM) to predict „spam vs. ham“ based on page and page-context features

Most discriminative features are:

- tfidf weights of words in p0 and IN(p0)
- avg, #inlinks of pages in IN(p0)
- avg. #words in title of pages in OUT(p0)
- #pages in IN(p0) that have same length as some other page in IN(p0)
- avg. # inlinks and outlinks of pages in IN(p0)
- avg. #outlinks of pages in IN(p0)
- avg. #words in title of p0
- total #outlinks of pages in OUT(p0)
- total #inlinks of pages in IN(p0)
- clustering coefficient of pages in IN(p0) (#linked pairs / n(n-1) possible pairs)
- total #words in titles of pages in OUT(p0)
- total #outlinks of pages in OUT(p0)
- avg. #characters of URLs in IN(p0)
- #pages in IN(p0) and OUT(p0) with same MD5 hash signature as p0
- #characters in domain name of p0
- #pages in IN(p0) with same IP number as p0

But spammers may learn to adjust to the anti-spam measures. It‘s an arms race!

Summary: Quality Challenge

- **Link analysis** has been primary means for quality/authority
  - strong foundations in linear algebra and stochastics,
  - many variations and extensions, distributed & P2P algorithms
  - mature engineering, no fully convincing theory yet
- **Web spam** is a major battle field
- Link analysis techniques extended for **spam detection**
- **Machine learning** techniques are attractive
- Spam combat will remain an **arm's race**, unless
  there is a breakthrough in **adversarial IR**
- New forms of spam combat needed for
  Web 2.0 ("splog"), **Web archival** (online detection), etc.

Overview

- **Part 1: Web IR**
  - ✓ State of the Art
  - ✓ Scalability Challenge
  - ✓ Quality Challenge
  - **Personalization**
  - Research Opportunities
- **Part 2: Semantic & Social IR**
  - Ontologies in XML IR
  - Entity Search and Ranking
  - Graph IR
  - Web 2.0 Search and Mining
  - Research Opportunities

Personalized Search & Ranking

- query interpretation depends on **personal interests and bias**
- need to learn user-specific weights for
  multi-criteria ranking (relevance, authority, freshness, etc.)
- can exploit **user behavior**
  (feedback, bookmarks, query logs, click streams, etc.)

or

or

User Behavior and Context

- leverage **implicit feedback**:
  - normal user behavior (clicks, queries, etc.),
  - no cognitive burden
  - full control over privacy constraints
- analyze query type and user context:
  - **recurrent query**: re-find or re-evaluate
  - **refinement or rephrasing**:
    unsuccessful or successful
  - **new session** (potential shift of user interest)

Some Approaches to Personalization

simple method: **expand query** by adding terms or adjust weights, e.g. using the Rocchio method:

$$q' = \alpha q + \frac{\beta}{|D^+|}\sum_{d \in D^+} d - \frac{\gamma}{|D^-|}\sum_{d \in D^-} d$$

with $\alpha, \beta, \gamma \in [0,1]$
and typically $\alpha > \beta > \gamma$

classical technique with relevance feedback or pseudo-relevance feedback; here D^+ (and D^-) are based on user's history

re-rank search results for personalization based on

- Rocchio method
- statistical language model
- machine-learning regression models
  with pair-wise preferences as input (Ranking SVM, RankNet):
  clicked result vs. higher-ranked non-clicked results in history

integrate personal (or community) behavior into **link analysis**

Rewriting & Re-Ranking for Query Chains

[Radlinski/Joachims: KDD'05, Elbassuoni et al.: SIGIR'07 Workshop]

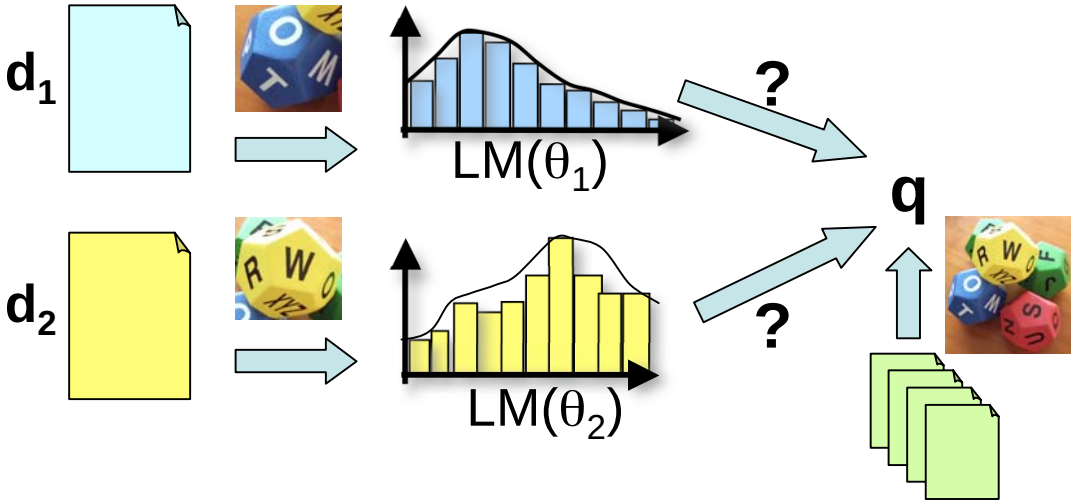
- transparently intercept all http traffic at client:
  monitor queries, clicks, sessions (query chains)
- build client-side index of page-access history
- consider history for query rewriting or result re-ranking

<u>at query time:</u>

- recognize recurrent queries and
  rewrite into successful query of previous query chain
- heuristically distinguish new sessions vs. ongoing sessions
  and use different Rocchio expansions for result re-ranking:
  - long-term history of queries and clicks for new session
  - clicked vs. non-clicked pages for current session

Statistical Language Models (LM's)

[Maron/Kuhns 1960, Ponte/Croft 1998, Lafferty/Zhai 2001]



- each doc has LM: generative prob. distr. with parameters θ
- query q viewed as sample
- estimate likelihood that q is sample of LM of doc d
- rank by descending likelihoods (best „explanation" of q)

$$s(d,q) = P[q|d] = P[q|\theta] = P[q_1 \ldots q_m|\theta] \sim \sum_i \log \frac{tf(i,d)}{\sum_k tf(k,d)}$$

MLE:sparseness & overfitting

$$s(d,q) = P[q|\theta] = \lambda P[q|d] + (1-\lambda)P[q]$$

mixture model

MAP with prior (**background**) for **smoothing**

$$\sim \sum_i \log\left(\lambda \frac{tf(i,d)}{\sum_k tf(k,d)} + (1-\lambda)\frac{df(i)}{\sum_k df(k)}\right)$$

$$\sim \sum_i \log\left(1 + \frac{tf(i,d)}{\sum_k tf(k,d)} \cdot \frac{1-\lambda}{\lambda}\frac{\sum_k df(k)}{df(i)}\right)$$

prior could be personalized

Applied to cross-lingual IR, entity search (MSR Beijing), etc.

LM with User-History Background Model

for current query q_k leverage
prior **query history $H_q = q_1 \ldots q_{k-1}$** and
prior **click stream $H_c = d_1 \ldots d_{k-1}$** as background LMs

Mixture model:

$$s(d, q_k) = P[q_k \mid \theta] = \lambda P[q_k \mid d] + (1 - \lambda) P[q_k]$$

with $P[q_k] = \sum_{j=1..k-1} \alpha_j P[q_j] + \beta_j P[d_j]$ such that $\sum_j \alpha_j + \beta_j = 1$

- details of LM parameter estimation more sophisticated
- can also use user's desktop data (files, mails, browser cache) as background model (component)

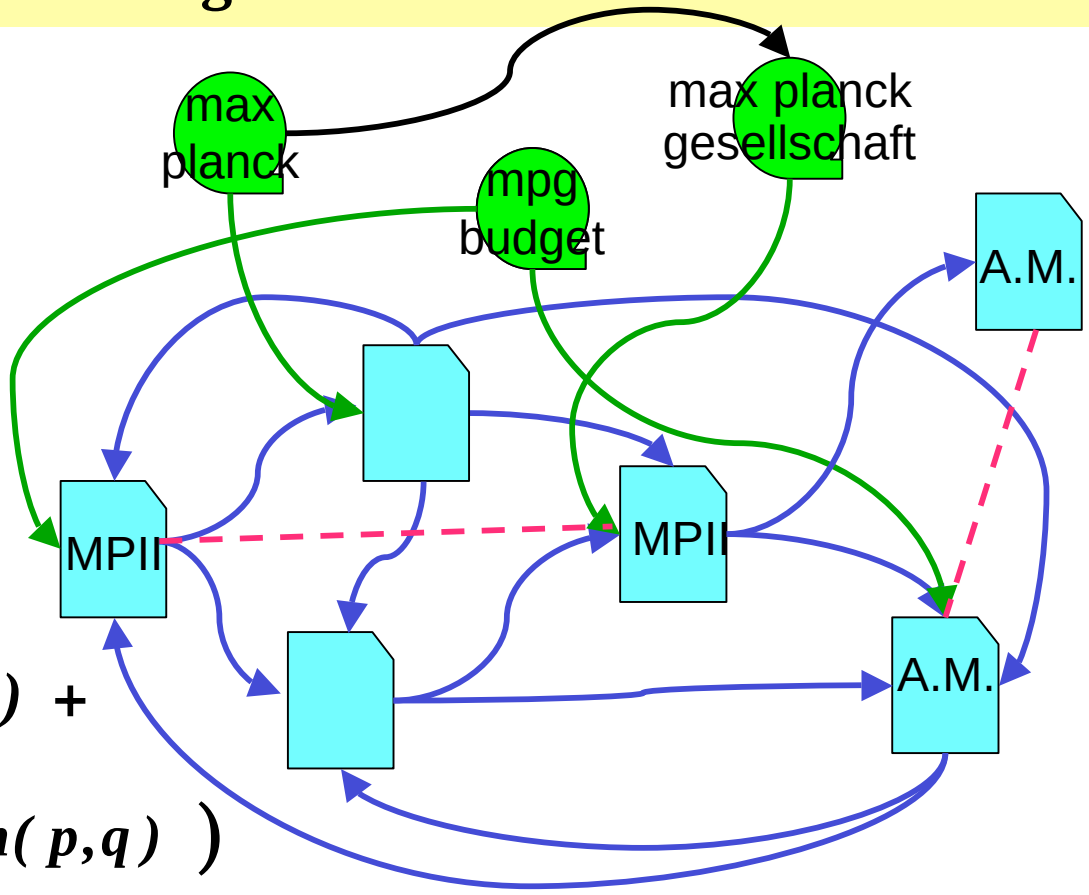
Exploiting Query Logs and Click Streams

[J. Luxenburger et al.: WISE 2004, WebDB 2006]

from PageRank: uniformly random choice of links + random jumps

to QRank: + query-doc transitions + query-query transitions
+ doc-doc transitions on implicit links (w/ thesaurus)
with probabilities estimated from log statistics

$$PR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \sum_{p \in IN(q)} PR(p) \cdot t(p,q)$$



$$QR(q) = \varepsilon \cdot j(q) + (1-\varepsilon) \cdot \Big(\alpha \sum_{p \in explicitIN(q)} PR(p) \cdot t(p,q) + (1-\alpha) \sum_{p \in implicitIN(q)} PR(p) \cdot sim(p,q) \Big)$$

Small-Scale Experiments

Setup:

70 000 Wikipedia docs, 18 volunteers posing Trivial-Pursuit queries
ca. 500 queries, ca. 300 refinements, ca. 1000 positive clicks
ca. 15 000 implicit links based on doc-doc similarity

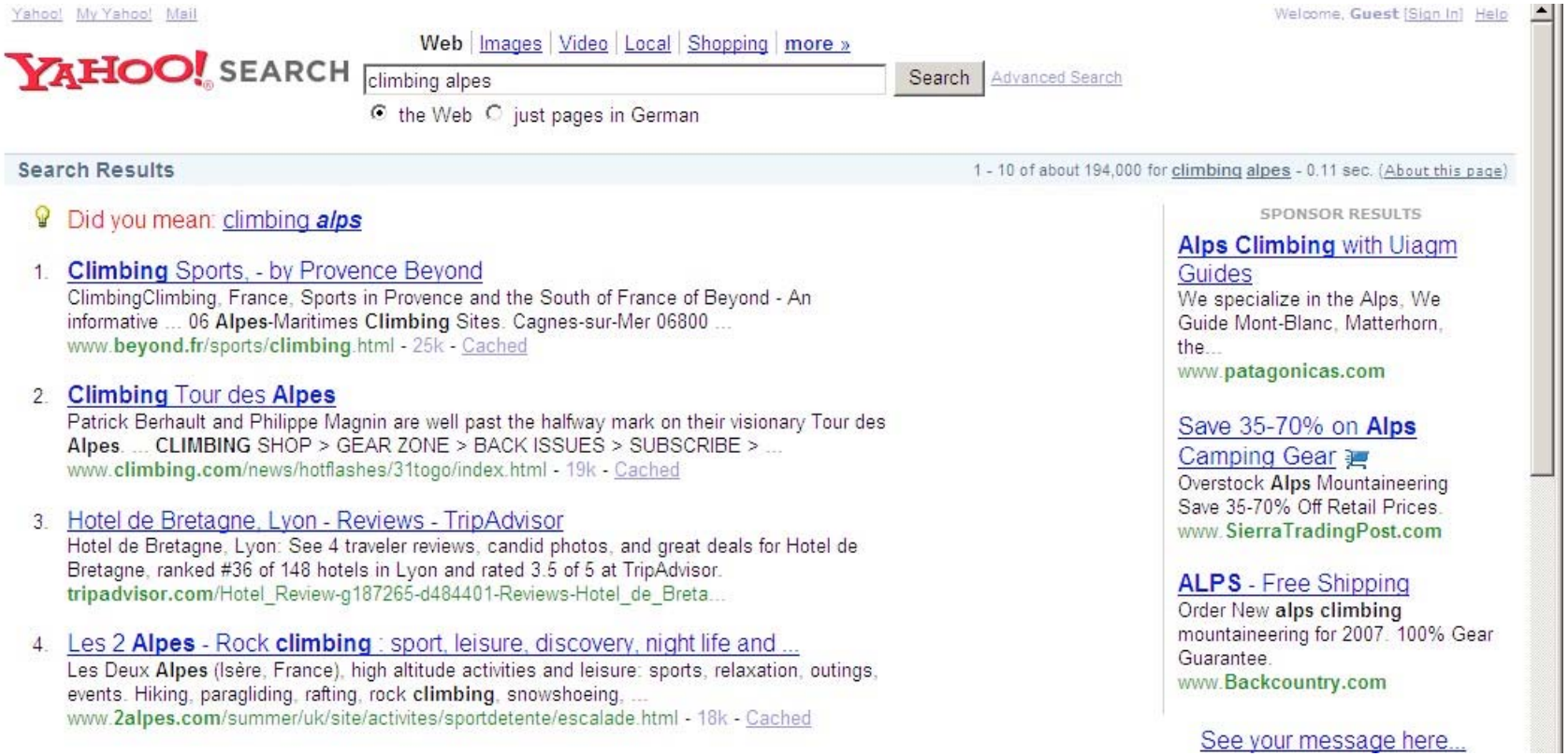
Results (assessment by blind-test users):

- QRank top-10 result preferred over PageRank in 81% of all cases
- QRank has 50.3% precision@10, PageRank has 33.9%

Untrained example query „philosophy“:

	PageRank	QRank
1.	Philosophy	Philosophy
2.	GNU free doc. license	GNU free doc. license
3.	Free software foundation	Early modern philosophy
4.	Richard Stallman	Mysticism
5.	Debian	Aristotle

Digression: Smart Ads (Information Supply)



- personalized & contextual **generation of advertisements** for query words, query topics, page/news/blog topics, etc.
- with consideration of **pricing & auctions**
- **3-way matchings**: users, publishers/keywords/topics, advertisers

see WWW'07 and SIGIR'07 tutorials by Baeza-Yates, Broder, Raghavan

Summary: Personalization

- Leveraging **implicit user behavior** is key
- Personalization can take place at either **client or server** side
- Statistical learning from query logs and click streams is hot topic
- Sparse human input suggests **transfer learning**, leveraging **community behavior**
- **Query-time efficiency** is crucial
- Recognizing **different roles and contexts** of same user (support entire **tasks** rather than queries)
- Personalization transparently **embedded** in apps & workflows (e.g. mobile phone services, job hunting, e-science work, etc.)
- Integration with **HCI and cognitive models** highly needed

Overview

- **Part 1: Web IR**
  - ✓ State of the Art
  - ✓ Scalability Challenge
  - ✓ Quality Challenge
  - ✓ Personalization
  - **Research Opportunities**
- **Part 2: Semantic & Social IR**
  - Ontologies in XML IR
  - Entity Search and Ranking
  - Graph IR
  - Web 2.0 Search and Mining
  - Research Opportunities

Future Web Search Functionality

- **Deep Web** with partly unified sources
  (in enterprise, across digital libraries, for vertical domains)
- **Entities** instead of pages, relations between entities
- **PubSub** for continuous info demand and alerting
- **Time-travel search** on Web history
- **Embedded search**, search as Web service
  (cell phones, mashup apps, ads)
- **Multimedia search** (photos, video, music, speech)
  boosted by **social networks**
- Natural-language **QA** and **cross-lingual IR**
- Structured/semantic search (XML, RDF, light-weight SQL) ???

Web IR: Research Opportunities

- **Ultra-scalable P2P indexing** with industrial-strength guarantees and 10 times lower cost/performance than commercial engines
- Further **improving TA family**:
  provable run-time guarantees; beyond monotonic aggregation
- Improving the **theory of link analysis** incl. distributed algorithms: comparing methods, stability, convergence rates, etc.
- Making P2P-style link analysis **resilient to cheating**
- **Adversarial spam detection** incl. online detection
- Making **personalization** more **context-aware & task-oriented**, with consideration of HCI and cognitive psychology

Thank You !

Literature on Web IR (1)

search engine architecture:

- R.A. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, F. Silvestri: Challenges on Distributed Web Retrieval. ICDE 2007
- L.A. Barroso, J. Dean, U. Hölzle: Web Search for a Planet: The Google Cluster Architecture. IEEE Micro 23(2), 2003
- S. Brin, L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW 1998
- A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan: Searching the Web. TOIT 2001

indexing and caching:

- J. Zobel, A. Moffat: Inverted files for text search engines. ACM Comput. Surv. 38(2), 2006
- X. Long, T. Suel: Three-level caching for efficient query processing in large Web search engines. WWW 2005
- R. Lempel, S. Moran: Predictive caching and prefetching of query results in search engines. WWW 2003
- R. Baeza-Yates et al.: The Impact of Caching on Search Engines. SIGIR 2007

query processing:

- R. Fagin, A. Lotem, M. Naor: Optimal aggregation algorithms for middleware. JCSS 66(4), 2003
- C. Buckley, A. F. Lewit: Optimization of Inverted Vector Searches. SIGIR 1985
- V.N. Anh, A. Moffat: Pruned query evaluation using pre-computed impacts. SIGIR 2006
- H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum: IO-Top-k: Index-access Optimized Top-k Query Processing. VLDB 2006
- M. Theobald et al.: Top-k Query Evaluation with Probabilistic Guarantees. VLDB 2004

Literature on Web IR (2)

link analysis:

- A. Borodin, G.O. Roberts, J.S. Rosenthal, P. Tsaparas: Link analysis ranking: algorithms, theory, and experiments. ACM Trans. Internet Techn. 5(1): 231-297 (2005)
- Amy N. Langville, Carl D. Meyer: Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 2006.
- S. Abiteboul, M. Preda, G. Cobena: Adaptive on-line page importance computation. WWW 2003
- D. Kempe, F. McSherry: A decentralized algorithm for spectral analysis. STOC 2004
- A.Z. Broder, R. Lempel, F. Maghoul, J.O. Pedersen: Efficient PageRank approximation via graph aggregation. Inf. Retr. 9(2), 2006
- J.X. Parreira, C. Castillo, D. Donato, S. Michel, G. Weikum: The JXP Method for Robust PageRank Approximation in a Peer-to-Peer Web Search Network. VLDB Journal 2007

spam detection:

- Z. Gyöngyi, H. Garcia-Molina: Spam: It's Not Just for Inboxes Anymore, IEEE Computer 2005
- Z. Gyöngyi, H. Garcia-Molina: Link Spam Alliances, VLDB 2005
- I. Drost, T. Scheffer: Thwarting the Nigritude Ultramarine: Learning to Identify Link Spam, ECML 2005
- A.A. Benczur, K. Csalongany, T. Sarlos, M. Uher: SpamRank – Fully Automatic Link Spam Detection, AIRWeb Workshop, 2005
- R. Guha, R. Kumar, P. Raghavan, A. Tomkins: Propagation of Trust and Distrust, WWW 2004
- Workshop on Adversarial Information Retrieval on the Web**,** http://airweb.cse.lehigh.edu/2007/

Literature on Web IR (3)

personalization:

- T. Haveliwala: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Trans. on Knowledge and Data Engineering, 2003
- G. Jeh, J. Widom: Scaling personalized web search. WWW 2003
- T. Joachims, F. Radlinski, Search Engines that Learn from Implicit Feedback, IEEE Computer 40(8), 2007
- F. Radlinski, T. Joachims, Query Chains: Learning to Rank from Implicit Feedback. KDD 2005
- P.-A. Chirita, C.S. Firan, W. Nejdl: Personalized Query Expansion for the Web. WWW 2007
- X. Shen, B. Tan, C. Zhai: Context-Sensitive Information Retrieval Using Implicit Feedback. SIGIR 2005
- B. Tan, X. Shen, C. Zhai: Mining Long-Term Search History to Improve Search Accuracy. KDD 2006
- J. Luxenburger, G. Weikum: Exploiting Community Behavior for Enhanced Link Analysis and Web Search. WebDB 2006