

Discovering and Exploiting Keyword and Attribute-Value Co-occurrences to Improve P2P Routing Indices

Sebastian Michel, Matthias Bender, Nikos Ntarmos,
Peter Triantafyllou, Gerhard Weikum, Christian Zimmer

Max-Planck-Institut für Informatik
Saarbrücken, Germany

RACTI and University of Patras
Patras, Greece

January 25, 2007



Outline

- 1 Motivation
- 2 Minerva
- 3 Problem Statement
- 4 Our Approaches
- 5 Experiments
- 6 Conclusion



P2P Systems

Became famous through file-sharing applications, like Gnutella, KaZAA, Napster.

Applications like:

- Internet telephony (e.g. Skype)
- Filesharing
- Pub/Sub

Question:

Is there an interesting and legal P2P application?



Motivation

Why P2P Web Search?

- Benefit from intellectual input from a large user community. (Bookmarks, click-streams, ...)
- Break information monopolies
- Coverage of the web
- Exploit mostly idle resources

Related to distributed IR, but some additional aspects

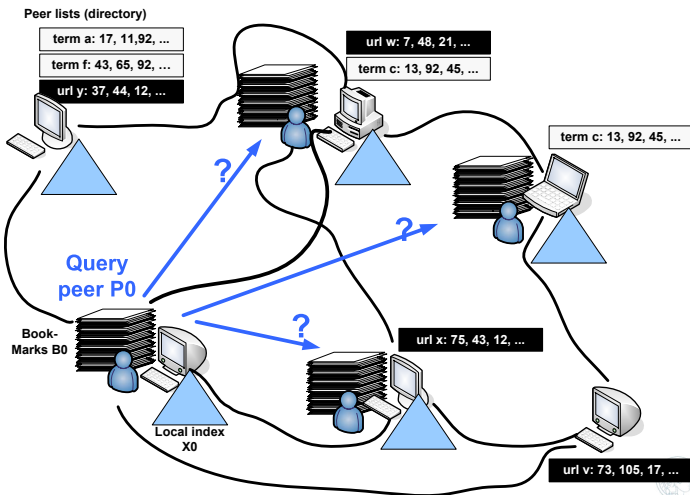
- high dynamics
- each peer has its own collection
- peers are independently crawling the web

Minerva Design Fundamentals

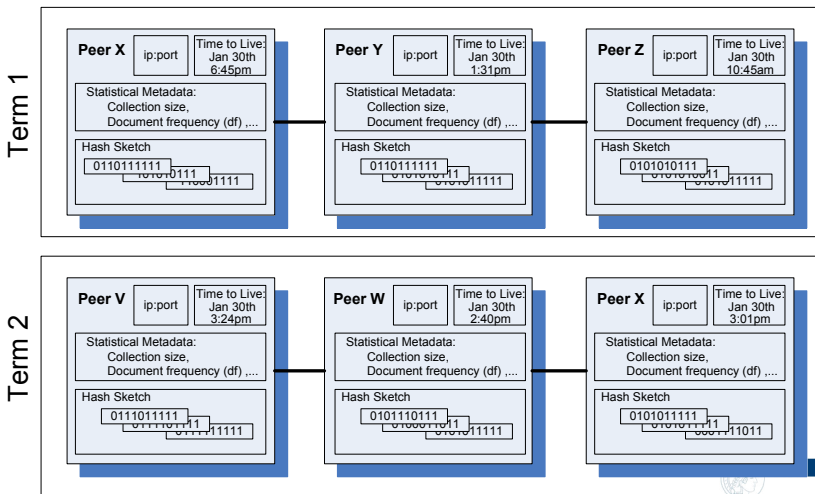
- Peers with local collections, e.g., built by focused crawler. Tailored to the users' specific interest profiles.
- Peers share metadata about local indexes
- Form physically distributed *term* → *peer* directory
- Layered on top of DHT
- Peers use directory to discover promising peers for query



Minerva System Architecture



Inside Peerlists ...



Problem Statement

Consider (conjunctive) query $a \wedge b$

- State-of-the-art: Identify promising peers on *per-key* statistics
 \Rightarrow good peers for a , b **separately** - but how about $a \wedge b$?

Can go arbitrarily wrong:

A peer good for *American politics* and *classical music* might not have a single relevant data item about (American \wedge music).

\Rightarrow **Multi-key necessary for meaningful peer selection**

Contribution: Two detailed approaches:

- estimate statistics $s(a \wedge b)$ from single-key statistics $s(a)$, $s(b)$
- provide explicit **multi-key** statistics $s(a \wedge b)$



Problem Statement

Consider (conjunctive) query $a \wedge b$

- State-of-the-art: Identify promising peers on *per-key* statistics
⇒ good peers for a , b **separately** - but how about $a \wedge b$?

Can go arbitrarily wrong:

A peer good for *American politics* and *classical music* might not have a single relevant data item about $(\text{American} \wedge \text{music})$.

⇒ **Multi-key necessary for meaningful peer selection**

Contribution: Two detailed approaches:

- estimate statistics $s(a \wedge b)$ from single-key statistics $s(a)$, $s(b)$
- provide explicit multi-key statistics $s(a \wedge b)$



Problem Statement

Consider (conjunctive) query $a \wedge b$

- State-of-the-art: Identify promising peers on *per-key* statistics
⇒ good peers for a , b **separately** - but how about $a \wedge b$?

Can go arbitrarily wrong:

A peer good for *American politics* and *classical music* might not have a single relevant data item about $(\text{American} \wedge \text{music})$.

⇒ **Multi-key necessary for meaningful peer selection**

Contribution: Two detailed approaches:

- estimate statistics $s(a \wedge b)$ from **single-key** statistics $s(a)$, $s(b)$
- provide explicit **multi-key** statistics $s(a \wedge b)$



sk-STAT

Goal: estimate contribution of peer p_i from single-key statistics $s_i(a)$, $s_i(b)$

Idea:

- enrich $s_i(a)$, $s_i(b)$ with suitable statistical summaries
- use statistical summaries to estimate $s_i(a \wedge b)$

Suitable Synopses?

- need to describe p_i 's items containing a
- need to support conceptual intersection
 - which items are contained in both $s(a)$ **and** $s(b)$?
- needs to be compact (limited bandwidth)

Hash Sketches

Estimate #distinct elements in multiset $D = \{d_1, \dots, d_n\}$, $|D|_{dist}$

General Idea:

- apply hash function h to every d_i
- record positions of least-significant 1-bits in a bitmap vector V

Intuition:

- $V[0]$ is set by appr. $\frac{1}{2}$ of the items, $V[1]$ by $\frac{1}{4}$, ...

⇒ Position of most-significant 1-bit estimates $\log(|D|_{dist})$



Using Hash Sketches

Distributivity Theorem

Let $V(A)$, $V(B)$ be bitmap vectors representing sets A , B . Then $V(A \cup B) = V(A) \vee V(B)$ (bit-wise *OR*).

\Rightarrow Estimator for $|A \cup B|_{dist}$

But how to estimate $|A \cap B|_{dist}$?

- No obvious way to *directly* intersect hash sketches.

Instead:

- use estimator for $|A \cup B|_{dist}$
- $|A \cap B|_{dist} = |A|_{dist} + |B|_{dist} - |A \cup B|_{dist}$

Note:

Estimates $|A \cap B|_{dist}$, but not $V(A \cap B)$

sk-STAT

Algorithm

- Enrich $s_i(a)$ with $HS_i(a)$
- For query $a \wedge b$: retrieve $s_i(a)$, $s_i(b)$, estimate p_i 's contribution for $a \wedge b$
- [combine with other indicators of expected quality / novelty]

- + readily perform estimation for *all* key sets
- + no additional statistics required
 - high network traffic (transfer HS for large number of peers)
 - limited accuracy
 - unavailability of $HS(a \wedge b)$



mk-STAT

Goal: Use explicit multi-key statistics for carefully selected key sets

Necessary Steps:

- *Discover* candidate key sets
- *Assess* degree of correlation for candidate key sets
- *Notify* peers to *generate and disseminate* explicit statistics
- *Leverage* additional statistics for peer selection process



Measure of Key Correlation

Our approach: Consider **Conditional Probability** $P(a|b)$

- $$P(a|b) = \frac{df(a \wedge b) / |D|_{dist}}{df(b) / |D|_{dist}} = \frac{df(a \wedge b)}{df(b)}$$

⇒ Estimation without knowledge of $|D|$

- Captures existing asymmetry:

Key a	Key b	$P(a b)$	$P(b a)$
andy	roddick	0.5106	0.0216
anna	kournikova	0.9613	0.0655
berlin	marathon	0.0611	0.0126



Measure of Key Correlation

Our approach: Consider **Conditional Probability** $P(a|b)$

- $$P(a|b) = \frac{df(a \wedge b) / |D|_{dist}}{df(b) / |D|_{dist}} = \frac{df(a \wedge b)}{df(b)}$$

⇒ Estimation without knowledge of $|D|$

- Captures existing asymmetry:

Key a	Key b	$P(a b)$	$P(b a)$
andy	roddick	0.5106	0.0216
anna	kournikova	0.9613	0.0655
berlin	marathon	0.0611	0.0126



mk-STAT: Query-driven Key Set Discovery

Limit Discovery of Interesting Key Sets to Query Logs

Key set is beneficial only if it is frequently queried (*support*) and if its keys coincide in a certain fraction of occurrences (*confidence*)

⇒ **Highly correlated** key sets are of interest

Efficient piggybacking onto existing communication:

- include *complete query* when retrieving $s(a)$ from $p(a)$
⇒ $p(a)$ sees all queries containing a
- $p(a)$ performs query log analysis to discover candidate key sets



mk-STAT: Data-driven Assessment

Assess relatedness based on data

Flawed intuition: Pick key pairs with correlation *above* threshold

- For queries with highly correlated keys, e.g., Anna Kournikova, it is sufficient to find good peers for Kournikova alone

⇒ **Hardly correlated** key sets benefit from extra information

Efficient assessment:

- $p(a)$ retrieves $HS(b)$ from $p(b)$ (only *one* combined HS)
- estimates $P(a|b)$ and $P(b|a)$
- identifies key sets with *both* probabilities below threshold



mk-STAT: Notification and Statistics Dissemination

Notification of Peers

Use statistics refreshments to disseminate assessed key sets:

- When p_i publishes $s_i(a)$ to $p(a)$, $p(a)$ returns key set $(a \wedge b)$
- p_i can start to produce $s(a \wedge b)$

Where to store $s(a \wedge b)$?

1st thought: identify $p(a \wedge b)$, typically: $p(a \wedge b) \neq p(a) \neq p(b)$

⇒ Load-balancing peers - but what about communication?!?

2nd thought: Use $p(a)$, $p(b)$

⇒ contacted anyway, piggybacking to avoid extra messages



mk-STAT: Notification and Statistics Dissemination

Notification of Peers

Use statistics refreshments to disseminate assessed key sets:

- When p_i publishes $s_i(a)$ to $p(a)$, $p(a)$ returns key set $(a \wedge b)$
- p_i can start to produce $s(a \wedge b)$

Where to store $s(a \wedge b)$?

1st thought: identify $p(a \wedge b)$, typically: $p(a \wedge b) \neq p(a) \neq p(b)$

⇒ Load-balancing peers - but what about communication?!?

2nd thought: Use $p(a)$, $p(b)$

⇒ contacted anyway, piggybacking to avoid extra messages



mk-STAT: Enhanced Peer Selection

Remember: $s(a \wedge b)$ stored at $p(a)$, $p(b)$

- makes peer selection easy
- provides simple fall-back strategy

Enhanced Peer Selection Algorithm

For query $(a \wedge b)$, query initiator p_i :

- contact $p(a)$, $p(b)$ as usual
- if available, retrieve $s(a \wedge b)$, otherwise $s(a)$, $s(b)$
 - No need for p_i to know about existence of key set $(a \wedge b)$!
- [combine with other indicators of expected quality / novelty]



Experimental Setup

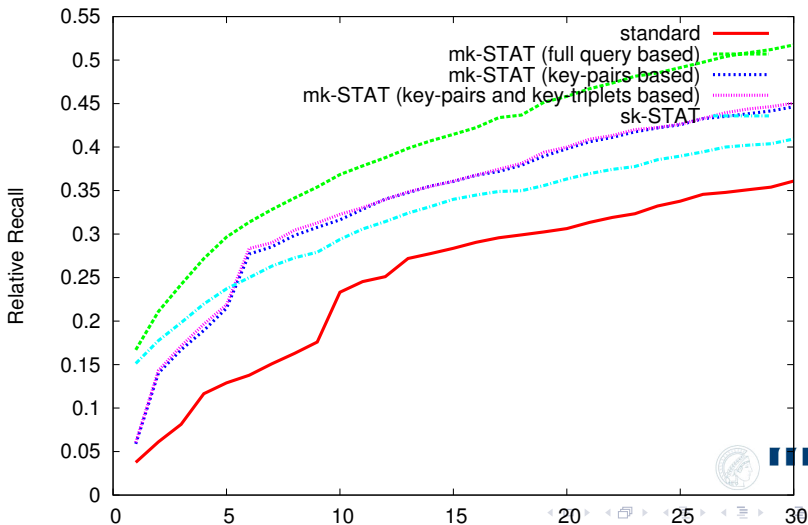
Gnutella Dataset:

- April 2003 crawl of Gnutella network
- ~ 850,000 file descriptors, ~ 4,000 user ids, 11,000+ queries
- Key set discovery: US top-40 single charts (April 12, 2003)
- Queries: All music-related queries from crawl
- Report *relative recall* averaged over *distinct* queries

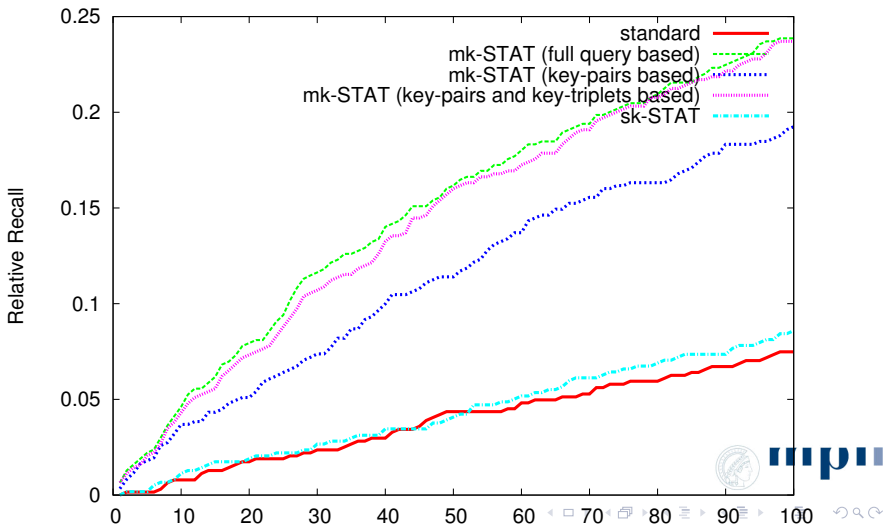
Web Dataset:

- .GOV collection (TREC benchmark), ~ 1.2m documents
- randomly assign documents to 750 peers (*stress test!*)
- Queries: multi-term queries of TREC 2003 distillation track
- Report *relative recall*

Experimental Results: Gnutella Data



Experimental Results: Web Data



Conclusion

- Key co-occurrences beneficial to peer selection
 - P2P Web Search
 - any other kind of attribute-value-style data
- sk-STAT: using existing single-key statistics
 - simple
 - versatile
 - limited accuracy
- mk-STAT: using explicit multi-key statistics
 - manageable extra effort
 - pays off



Thank You

Thank you for your attention.



Hash Sketches Illustrated

	lsb		msb		
$h(d_1)$	0	0	1	1	$p(h(d_1)) = 2$
$h(d_2)$	1	0	1	1	$p(h(d_2)) = 0$
$h(d_3)$	0	1	1	0	$p(h(d_3)) = 1$
$h(d_4)$	1	0	1	0	$p(h(d_4)) = 0$

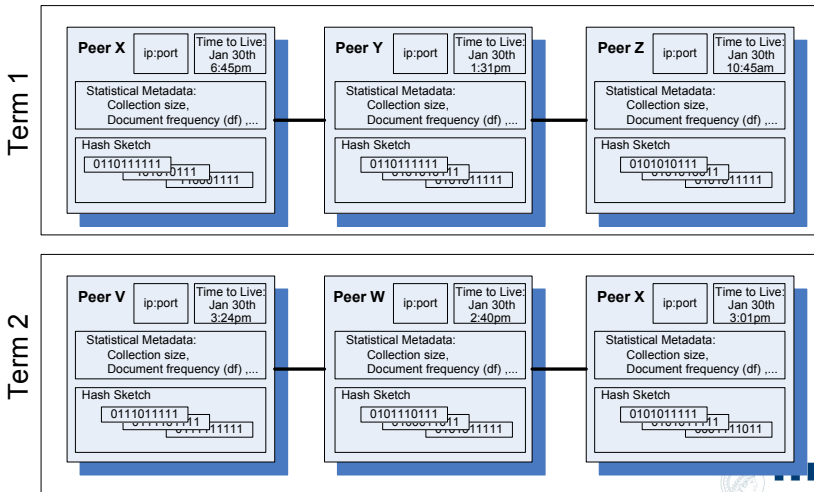
	B[0]	...	B[3]	
B	1	1	1	0

Position of most-significant 1-bit of B: B[2]

Estimation for n: $\log(n) \sim 2 \rightarrow n \sim 2^2 = 4$



PeerLists



Distributed Hash Tables

Example: Chord

- Structured P2P overlay
- Map peers and keys to the same cyclic id space
- *lookup* method to find peer currently responsible for a key
- Naive Routing: Traverse successor links linearly
- Enhanced Routing: *Finger Tables* pointing to distant peers

