



Efficient Top-k Queries for XML Information Retrieval

Gerhard Weikum (weikum@mpi-inf.mpg.de)

Joint work with
Martin Theobald and Ralf Schenkel

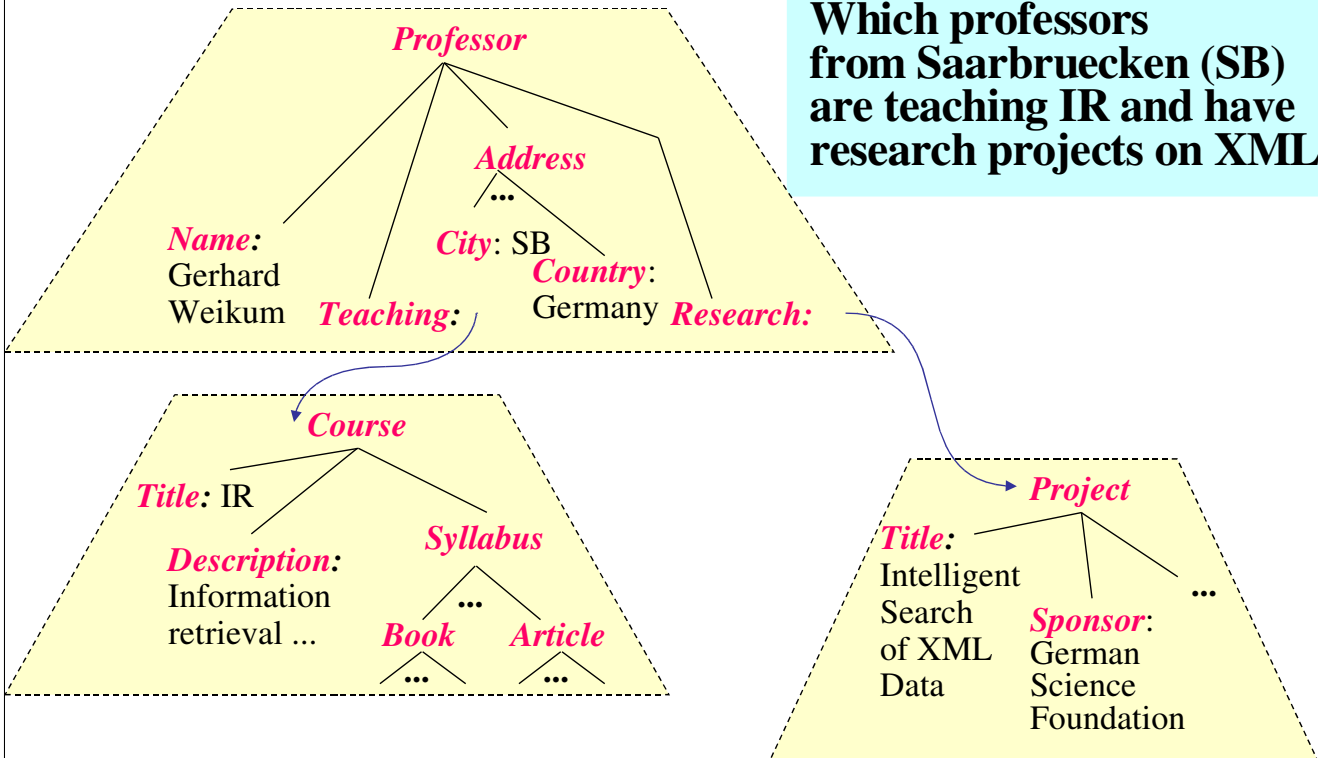
Queries Beyond Google

- *professors from Saarbruecken who teach DB or IR and have projects on XML*
 - *drama with three women making a prophecy to a British nobleman that he will become king*
 - *the woman from Paris whom I met at the PC meeting chaired by Raghu Ramakrishnan*
- “Semantic Search”:
- exploit structure and annotations in the data
 - exploit background knowledge (ontologies/thesauri + statistics)
 - connect/join/fuse information fragments



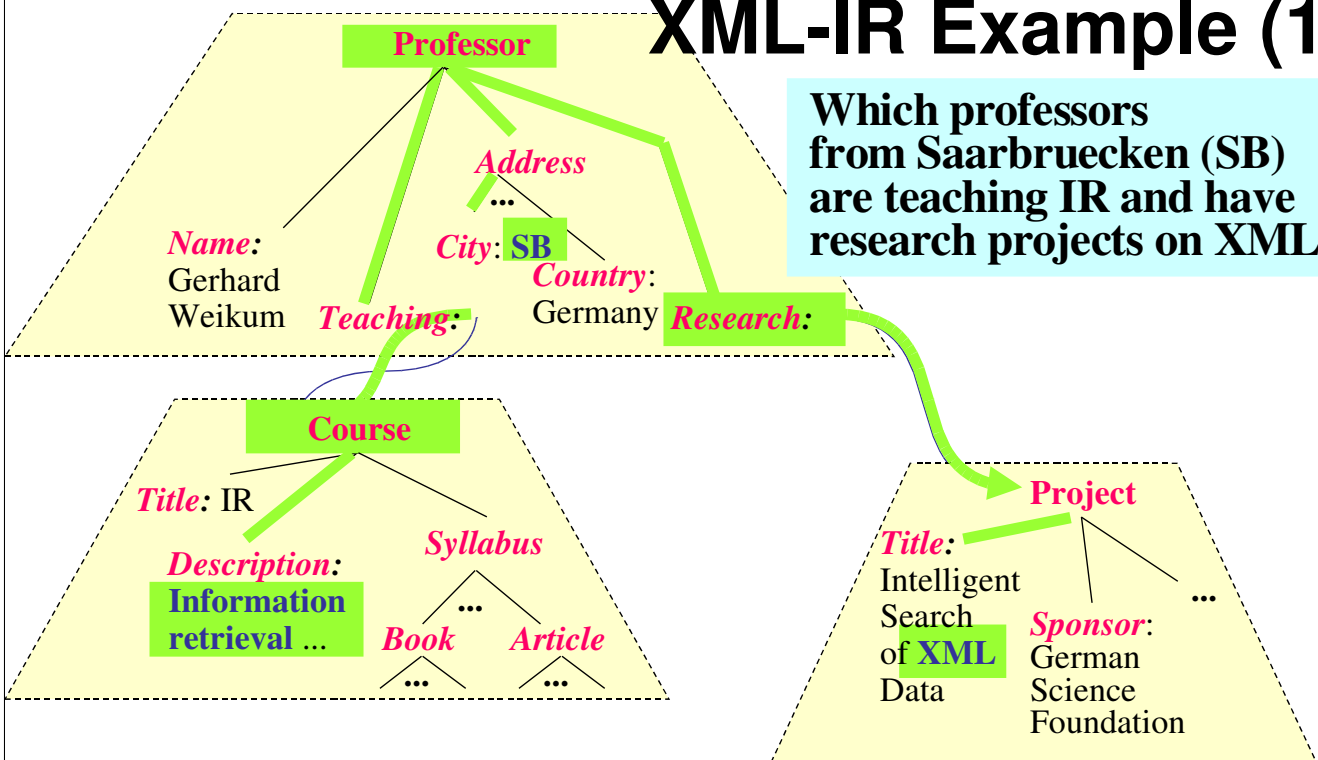
What If The Semantic Web Existed And All Information Were in XML?

Which professors from Saarbruecken (SB) are teaching IR and have research projects on XML?



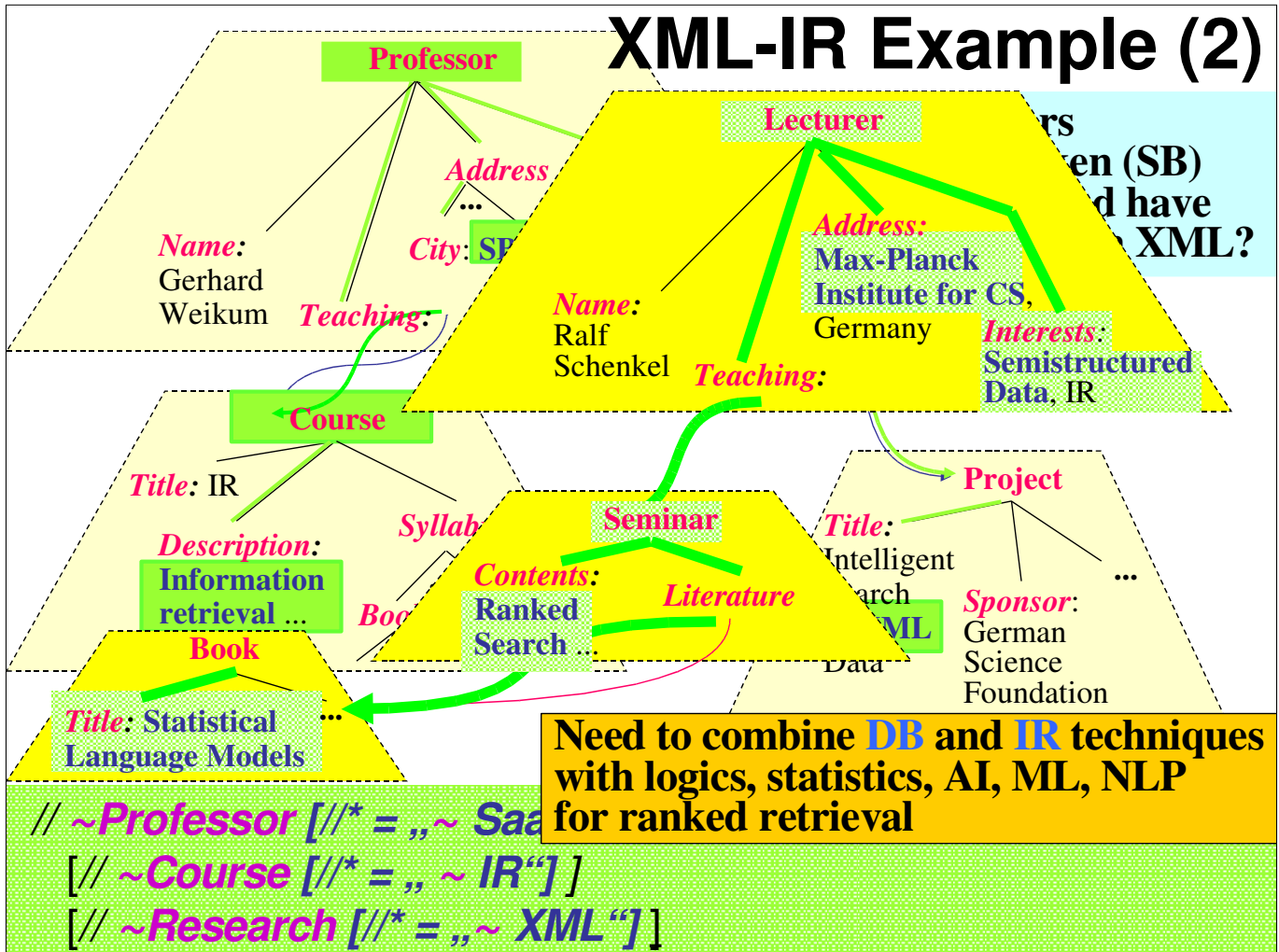
XML-IR Example (1)

Which professors from Saarbruecken (SB) are teaching IR and have research projects on XML?



```
// Professor [//* = „ Saarbruecken“]
[// Course [//* = „ IR“ ]
[// Research [//* = „ XML“ ]
```

XML-IR Example (2)

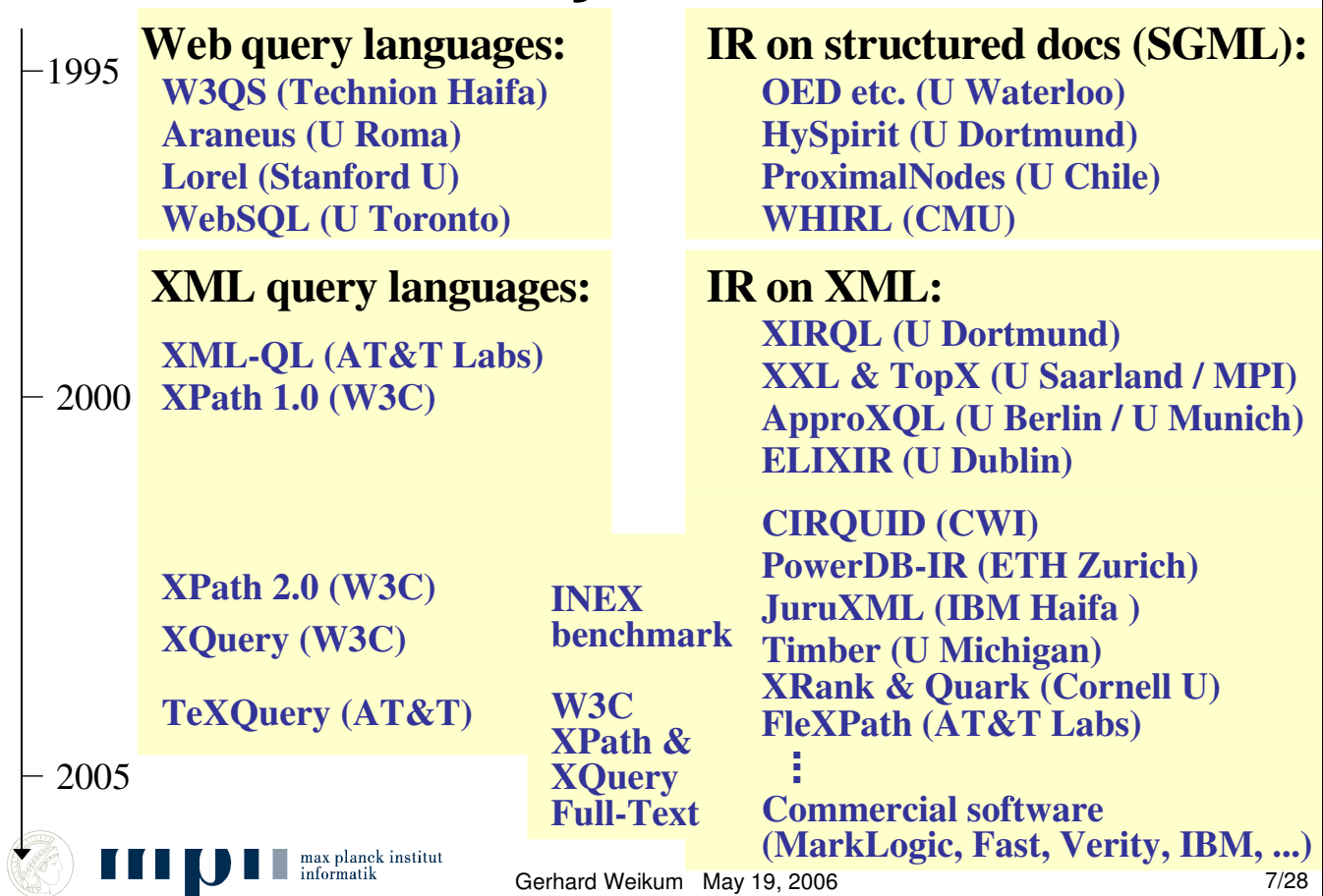


Outline

➔ Motivation and Strategic Direction

- XML IR & Ontologies
- Efficient Top-k QP
- TopX: Efficient XML IR
- Conclusion

XML-IR: History and Related Work



XML-IR Concepts [MPII XXL 00 & TopX 05, U Duisburg XIRQL 00, U Dublin Elixir 00, Cornell XRank & Quark 03, U Michigan 02, U Wisconsin 04, CWI Cirquid 03, AT&T FleXPath 04, W3C XPath Full-Text 05, IBM, Verity, Fast, etc.]

applicable to XML, HTML, Blogs, relational data graphs, etc.

search condition: conjunction of restricted *path expressions*

Elementary conditions on names and

```

// Professor [//* = „Saarbruecken“]
  [// Course [//* = „Information Retrieval“]]
  [// Research [//* = „XML“]]
    
```

„Semantic“ *similarity conditions* on names and contents
~Research [// „~XML“]*

Query result:

- query is a pattern with relaxable conditions
- results are approximate matches to query with similarity scores

Relevance scoring based on

- term-frequency statistics for content similarity (BM25 variant with tf per tag-term pair and tag-specific idf)
- ontological similarity of concept names,
- aggregation of local scores into global scores

Query Expansion and Execution

User query: $\sim c = \sim t1 \dots \sim tm$

Example:

$\sim professor$ and ($\sim course = ,, \sim IR''$)
 //professor[//place = ,,SB''//course = ,,IR''

Term2Concept with WSD

Query expansion

$$\text{exp}(ti) = \{w \mid \text{sim}(ti, w) \geq \theta\}$$

Weighted expanded query

Example:

(*professor lecturer (0.749) scholar (0.71) ...*)
 and ((*course class (1.0) seminar (0.84) ...*)
 = (,,IR'' ,,Web search'' (0.653) ...))

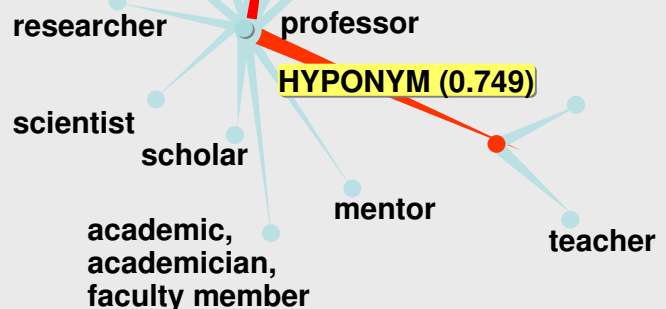
Efficient top-k search
with dynamic expansion

better recall, better mean
precision for hard queries

Thesaurus/Ontology:

concepts, relationships, glosses
from WordNet, Gazetteers,
Web forms & tables, Wikipedia

Problem:
tuning the
threshold θ
→ top-k with
incr. merge



relationships quantified by
statistical correlation measures



Towards a Statistically Semantic Web

Isaac Newton

From Wikipedia, the free encyclopedia.

<Person>

Sir Isaac Newton (25 December 1642 – 20 March 1727 by the Julian calendar in use in England at the time; or 4 January 1643 – 31 March 1727 by the Gregorian calendar) was an English physicist, mathematician, astronomer, philosopher, and alchemist; who wrote the *Philosophiæ Naturalis Principia Mathematica* (published 5 July 1687)¹, where he described universal gravitation and, via his laws of motion laid the groundwork for classical mechanics. Newton also shares credit with Gottfried Wilhelm Leibniz for the development of differential calculus. However, their work was not a collaboration; they developed calculus separately but nearly contemporaneously.

<Person>

Information extraction:

(NLP, reg.exp., lexicon, HMM, CRF, etc.)

Person	TimePeriod	...
Sir Isaac Newton	4 Jan 1643 - ...	
... Leibniz		
... Kneller		

Publication	Topic
Philosophiæ Naturalis	... gravitation

Author	Publication
... Newton	Philosophia ...

Database
(tables or XML)

Scientist
Sir Isaac Newton
... Leibniz

but with confidence < 1

→ Semantic-Web database
with uncertainty!
→ ranked retrieval!



Outline

- Motivation and Strategic Direction
- XML IR & Ontologies
- Efficient Top-k QP
- TopX: Efficient XML IR
- Conclusion

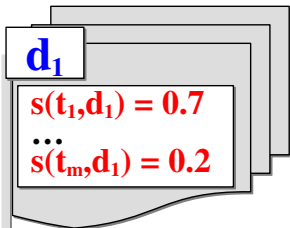


Efficient Top-k Search [Buckley85, Güntzer et al. 00, Fagin01]

TA: efficient & principled top-k query processing with monotonic score aggr.

TA with sorted access only (NRA):
 can 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
 $min-k := \min\{worstscore(d') \mid d' \in top-k\}$;
 else if $bestscore(d) > min-k$ then
 $cand := cand \cup \{d\}$; s
 $threshold := \max\{bestscore(d') \mid d' \in cand\}$;
 if $threshold \leq min-k$ then exit;

Data items: d_1, \dots, d_n



fetch lists
join&sort

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

	Index lists				
t_1	d78	d23	d10	d1	d88
	0.9	0.8	0.8	0.7	0.2
t_2	d64	d23	d10	d10	d78
	0.8	0.6	0.6	0.2	0.1
t_3	d10	d78	d64	d99	d34
	0.7	0.5	0.4	0.2	0.1

$k = 1$

Scan
depth 3

Rank	Doc	Worst-score	Best-score
1	d10	2.1	2.1
2	d78	1.4	2.0
3			1.8
4		1.2	2.0

STOP!

Ex. Google: > 10 mio. terms, > 8 bio. docs, > 4 TB index



Probabilistic Pruning of Top-k Candidates [VLDB 04]

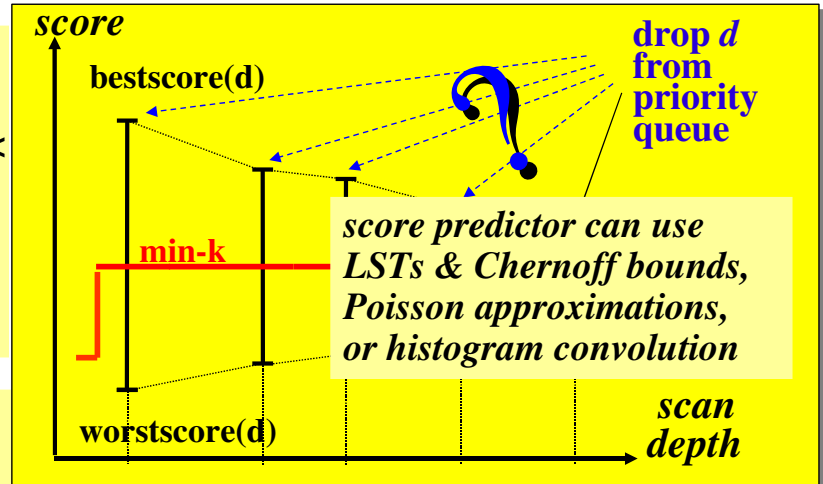
TA family of algorithms based on invariant (with sum as aggr):

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

- Add d to top-k result, if $\text{worstscore}(d) > \text{min-k}$
- Drop d only if $\text{bestscore}(d) < \text{min-k}$, otherwise keep in PQ

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

→ Approximate top-k with probabilistic guarantees:

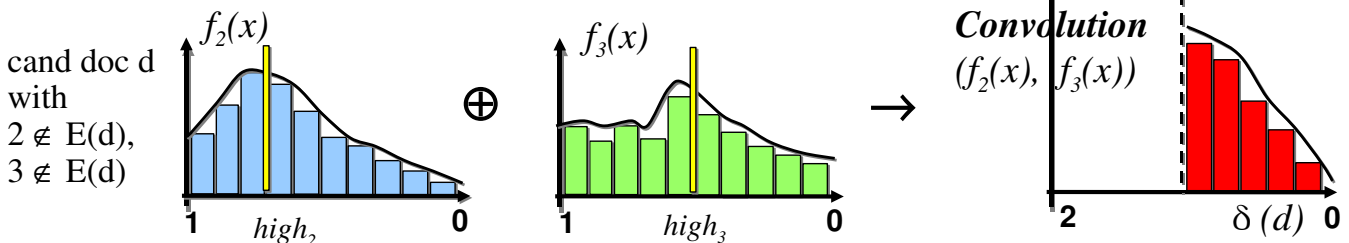


$$p(d) := P\left[\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} S_i > d\right]$$

discard candidates d from queue if $p(d) \leq \epsilon \Rightarrow E[\text{rel. precision}@k] = 1 - \epsilon$



Probabilistic Threshold Test



- postulating **uniform or Pareto** score distribution in $[0, \text{high}_i]$

- compute convolution using LSTs
- use Chernoff-Hoeffding tail bounds or generalized bounds for correlated dimensions (Siegel 1995)

- fitting **Poisson** distribution or **Poisson mixture**

- over equidistant values: $P[d = v_j] = e^{-\alpha_i} \frac{\alpha_i^{j-1}}{(j-1)!}$
- easy and exact convolution

- distribution approximated by **histograms**

- precomputed for each dimension
- dynamic convolution at query-execution time

engineering-wise histograms work best!



Performance Results for .Gov Queries

on .GOV corpus from TREC-12 Web track:
1.25 Mio. docs (html, pdf, etc.)

50 keyword queries, e.g.:

- „Lewis Clark expedition“,
- „juvenile delinquency“,
- „legalization Marihuana“,
- „air bag safety reducing injuries death facts“

*speedup by factor 10
at high precision/recall
(relative to TA-sorted);
aggressive queue mgt.
even yields factor 100
at 30-50 % prec./recall*

	TA-sorted	Prob-sorted (smart)
#sorted accesses	2,263,652	527,980
elapsed time [s]	148.7	15.9
max queue size	10849	400
relative recall	1	0.69
rank distance	0	39.5
score error	0	0.031



.Gov Expanded Queries

on .GOV corpus with query expansion based on WordNet synonyms:

50 keyword queries, e.g.:

- „juvenile delinquency youth minor crime law jurisdiction offense prevention“,
- „legalization marijuana cannabis drug soft leaves plant smoked chewed euphoric abuse substance possession control pot grass dope weed smoke“

	TA-sorted	Prob-sorted (smart)
#sorted accesses	22,403,490	18,287,636
elapsed time [s]	7908	1066
max queue size	70896	400
relative recall	1	0.88
rank distance	0	14.5
score error	0	0.035

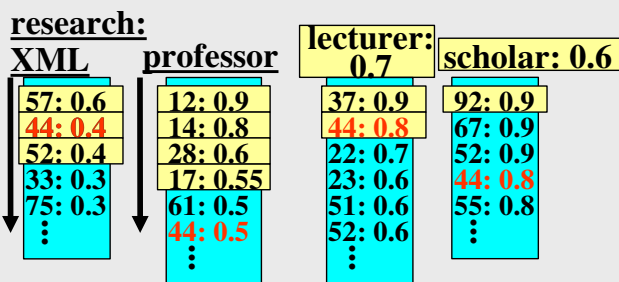


Top-k Queries with Query Expansion [SIGIR 05]

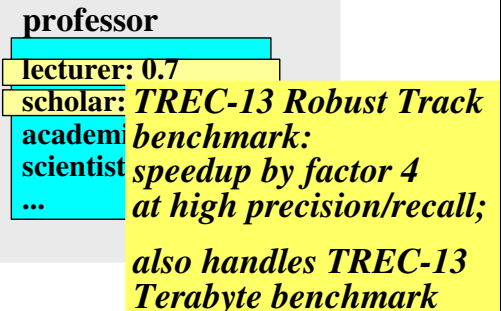
consider expandable query „~*professor and research = XML*“
 with score $\sum_{i \in q} \{ \max_{j \in \text{exp}(i)} \{ \text{sim}(i,j) * s_j(d) \} \}$

dynamic query expansion with
incremental on-demand merging of additional index lists

B+ tree index on tag-term pairs and terms



thesaurus / meta-index



- + much more efficient than threshold-based expansion
- + no threshold tuning
- + no topic drift



Combined Algorithm (CA) for Balanced SA/RA Scheduling [Fagin 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 $\sum_i \text{high}_i \leq \min\{\text{bestscore}(d) \mid d \in \text{final top-}k\}$,

then perform RAs for all d' with $\text{bestscore}(d') > \text{min-}k$: **$4m + k$**)



Index Access Optimization

[joint work with Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Martin Theobald]

For **SA scheduling** plan next b_1, \dots, b_m index scan steps

for **batch of b steps** overall s.t. $\sum_{i=1..m} b_i = b$

and $\text{benefit}(b_1, \dots, b_m)$ 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 $d \in \text{top-k}$) 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

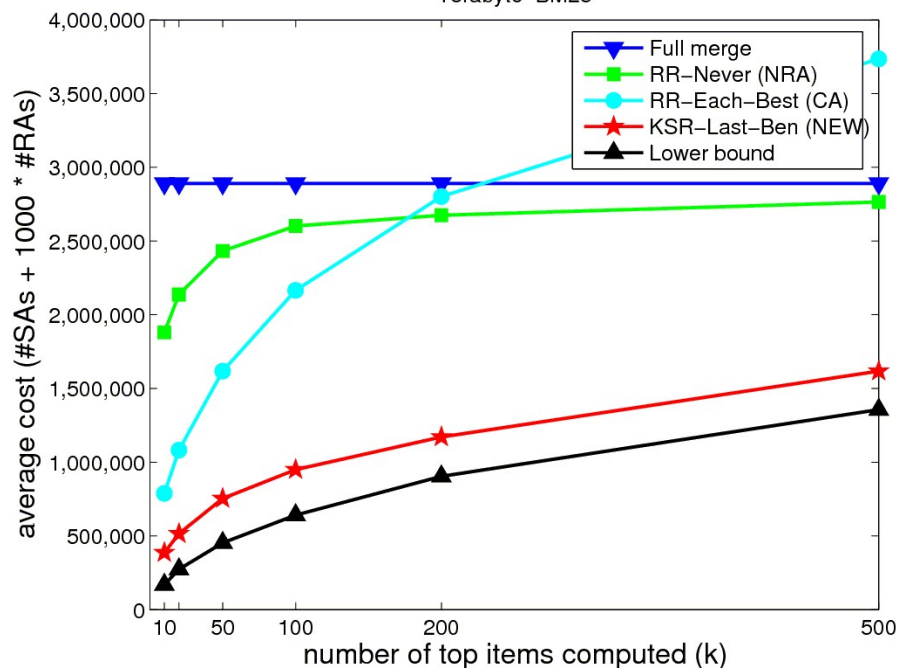
total cost of remaining RAs = total cost of SAs up to t
with **score-prediction & cost model** for deciding RA order



Performance of SA/RA Scheduling Methods

[joint work with Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Martin Theobald]

Terabyte-BM25



**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 $k=50$
- **KSR-Last-Ben (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



Outline

- Motivation and Strategic Direction
- XML IR & Ontologies
- Efficient Top-k QP
- TopX: Efficient XML IR
- Conclusion



TopX Search on XML Data [VLDB 05]

Example query (NEXI, XPath Full-Text):

```
//book[about(//„Information Retrieval“ „XML“) and  
//affiliation[about(// „Stanford“)] and  
//reference[about(// „Page Rank“) ]//publisher//country
```

apply & adapt (prob.) TA-style top-k query processing

Problems → Solutions:

- 0) disk space is cheap, disk I/O is not:
 - precompute and store scores for entire subtrees
- 4) content conditions (CC) on both **tags and terms**
 - build index lists for each tag-term pair
- 2) scores for elements or **subtrees**, docs as results
 - block-fetch all elements for the same doc
- 3) test **path conditions (PC)**, but avoid RAs
 - test PCs on candidates in memory via Pre&Post coding and carefully schedule RAs
- 4) PCs may be **relaxable**
 - unsatisfied PCs result in score penalty



TopX Algorithm

based on index table (with several B+ tree indexes):

L (Tag, Term, MaxScorePerDoc, DocId, Score, ElemId, Pre, Post)

decompose query: content conditions (CC) & path conditions (PC);

//conditions may be optional or mandatory

for each index list L_i (extracted from L by tag and/or term) do:

block-scan next elements from same doc d;

test evaluable PCs of all elements of d;

drop elements and docs that do not satisfy mandatory PCs or CCs;

update score bookkeeping for d;

consider random accesses for d by cost-based scheduler;

drop d if (prob.) score threshold is not reached;



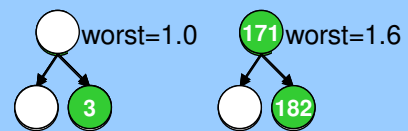
TopX Query Processing By Example

//sec[clustering]

//title[xml]

//par[evaluation]

Top-k results (k=2):



min-k = 0.6

1.0 //sec[clustering]

eid	docid	score	pre	post
46	2	0.9	2	15
9	2	0.5	10	8
171	5	0.85	1	20
84	3	0.1	1	12

0.85
0.1

1.0 //title[xml]

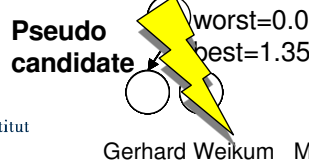
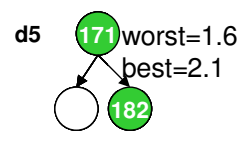
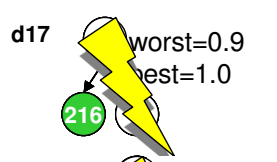
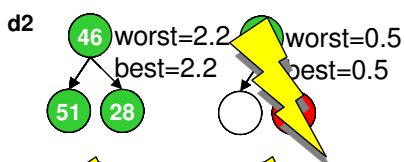
eid	docid	score	pre	post
216	17	0.9	2	15
72	3	0.8	10	17
51	2	0.5	4	12
671	31	0.4	12	23

0.8
0.5

1.0 //par[evaluation]

eid	docid	score	pre	post
3	1	1.0	1	21
28	2	0.8	8	14
182	5	0.75	3	7
96	4	0.75	6	4

0.8
0.75



Candidate queue



Experimental Results: INEX Benchmark

on *IEEE-CS journal and conference articles*:

12,000 XML docs with 12 Mio. elements, 7.9 GB for all indexes

20 CO queries, e.g.: „XML editors or parsers“

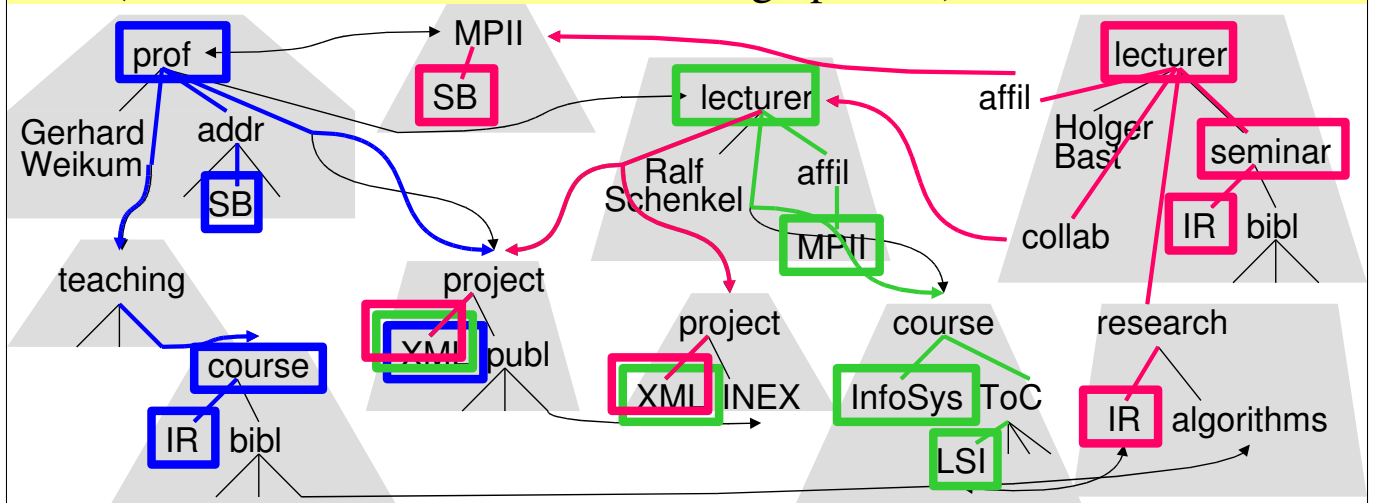
20 CAS queries, e.g.: `//article[//bibl[about(//„QBIC“)] and //p[about(//„image retrieval“)]]`

	Join & Sort	Struct Index	TopX ($\epsilon=0.0$)	TopX ($\epsilon=0.1$)
#sorted accesses @10	9,122,318	761,970	635,507	426,986
#random accesses @10	0	3,245,068	64,807	59,414
CPU sec	12.01	17.02	1.38	1.27
relative recall @10	1	1	<i>TopX outperforms</i>	
precision@10	0.34	0.34	<i>Join&Sort by factor > 10</i>	
MAP@1000	0.17	0.17	<i>and</i>	
			<i>beats StructIndex by factor > 20 on INEX, factor 2-3 on IMDB</i>	



Challenge: XML IR on Graphs

Q: professor from SB teaching IR and research on XML on graph with scores as node weights and weighted edges (XML with XLinks, Web-to-XML graph, etc.)



Combine content-score aggregation with **result-subgraph compactness**
 → compute top-k Steiner trees (or top-k MSTs as approximation)
 initial work: BANKS (IIT Bombay), SphereSearch (MPII)



Outline

- Motivation and Strategic Direction
- XML IR & Ontologies
- Efficient Top-k QP
- TopX: Efficient XML IR
- Conclusion



Conclusion: Ongoing and Future Work

Observations:

- *XML IR*: enterprise search, DLs, data integr., *Web + InfoExtraction*
- Approximations with *statistical guarantees* are key to obtaining *Web-scale efficiency*

(TREC'04 TB: 25 Mio. docs, 700 000 terms, 5-50 terms per query;
Wikipedia for INEX'06: 880 000 docs, 130 Mio. elements)

Challenges:

- *Scheduling* of index-scan steps and random accesses and efficient consideration of *correlated dimensions*
- Integrate *info-extraction confidence values* into XML similarity search (content & ontology & structure)
- Generalize TopX to arbitrary *graphs*
- Integration of top-k operator into *physical algebra* and *query optimizer* of XML engine
- *Re-invent SQL* and XQuery with *probabilistic ranking*



Backup Slides



Ontologies & Thesauri: Example WordNet

IR&NLP Approach
e.g. WordNet Thesaurus (Princeton)
(> 100 000 concepts
with lexical & linguistic relations)

woman, adult female – (an adult female person)
=> amazon, virago – (a large strong and aggressive woman)
=> donna -- (an Italian woman of rank)
=> geisha, geisha girl -- (...)
=> lady (a polite name for any woman)
..
=> wife – (a married woman, a man's partner in marriage)

=> witch – (a being, usually female, imagined to
have special powers derived from the devil)

1 of 4 senses of woman
Sense 1
woman, adult female -- (an adult female person (as opposed to a man);
=> Eve -- ((Old Testament) Adam's wife in Judeo-Christian mytho
=> black woman -- (a woman who is Black)
=> white woman -- (a woman who is White)

=> maenad -- (an unnaturally frenzied or distraught woman)
=> matron, head nurse -- (a woman in charge of nursing in a medical institution)

"Hyponyms (...is a kind of this), brief" search for noun "woman"

Query Expansion Example

From TREC 2004 Robust Track Benchmark:

Title: International Organized Crime

Description: Identify organizations that participate in international criminal activity, the activity, and, if possible, collaborating organizations and the countries involved.

Query: Redisplay Overview

~META[1.00|1.00]{{gangdom[1.00|1.00] gangland[0.742|1.00] organ[0.213|1.00] & crime[mafia[0.154|1.00], "sicilian[0.066|1.00] & hand[0.213|1.00], crime[0.31 columbian[0.686|0.20], cartel ...

1 sense of organized crime

Searches for organized crime: 135550 sorted accesses in 41.07s

Results:

- Interpol Chief on Fight Against Mafia, Mafia, Sicilian Mafia
- Economic Counterintelligence Tyranny but evolved into a ch
- Dresden Conference Views Black Hand -- (a secret terr
- Report on Drug Weapons S
- SWITZERLAND CALLED ...

... for organizing the illicit export of metals and import of arms. It is extremely difficult for the law-enforcement organs to investigate and stamp out corruption among leading officials.

A parliamentary commission accused Swiss prosecutors today of doing little to stop drug and money-laundering international networks from pumping billions of dollars through Swiss companies.

... organized criminal activities)

Performance Results for IMDB Queries

on IMDB corpus (Web site: Internet Movie Database):

375 000 movies, 1.2 Mio. persons (html/xml)

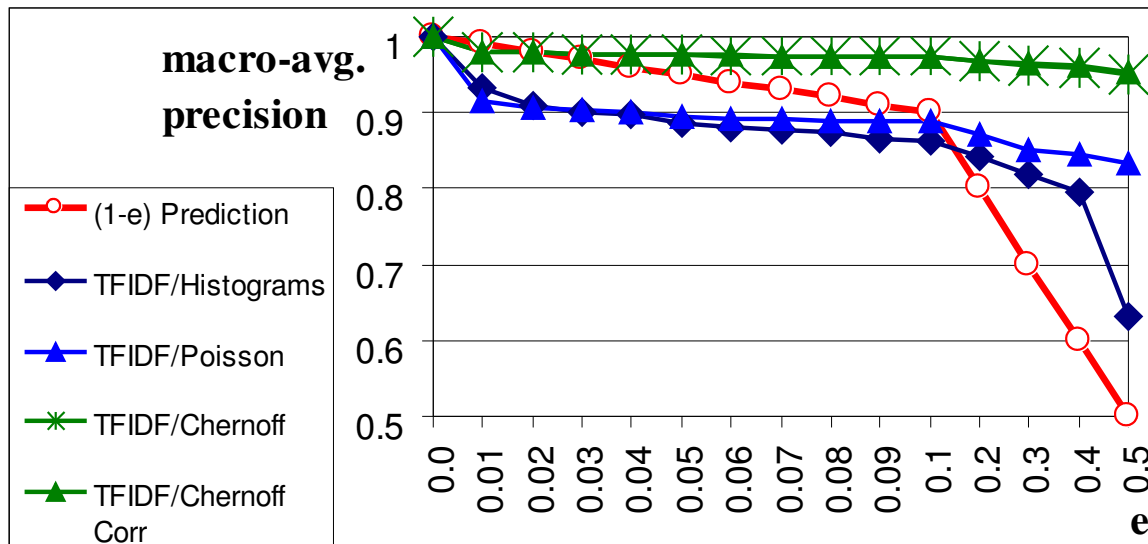
20 structured/text queries with Dice-coefficient-based similarities of categorical attributes Genre and Actor, e.g.:

- $Genre \supseteq \{Western\} \wedge Actor \supseteq \{John Wayne, Katherine Hepburn\} \wedge Description \supseteq \{sheriff, marshall\}$
- $Genre \supseteq \{Thriller\} \wedge Actor \supseteq \{Arnold Schwarzenegger\} \wedge Description \supseteq \{robot\}$

	TA-sorted	Prob-sorted (smart)
#sorted accesses	1,003,650	403,981
elapsed time [s]	201.9	12.7
max queue size	12628	400
relative recall	1	0.75
rank distance	0	126.7
score error	0	0.25



Comparison of Probabilistic Predictors



Experiments with TREC-13 Robust Track

on Acquaint corpus (news articles):

528 000 docs, 2 GB raw data, 8 GB for all indexes

50 most difficult queries, e.g.:

„transportation tunnel disasters“

„Hubble telescope achievements“

potentially expanded into:

„earthquake, flood, wind, seismology, accident, car, auto, train, ...“

„astronomical, electromagnetic radiation, cosmic source, nebulae, ...“

*speedup by factor 4
at high precision/recall;*

*no topic drift, no need
for threshold tuning;*

*also handles TREC-13
Terabyte benchmark*

	no exp. ($\epsilon=0.1$)	static exp. ($\theta=0.3,$ $\epsilon=0.0$)	static exp. ($\theta=0.3,$ $\epsilon=0.1$)	incr. merge ($\epsilon=0.1$)
#sorted acc.	1,333,756	10,586,175	3,622,686	5,671,493
#random acc.	0	555,176	49,783	34,895
elapsed time [s]	9.3	156.6	79.6	43.8
max #terms	4	59	59	59
relative recall	0.934	1.0	0.541	0.786
precision@10	0.248	0.286	0.238	0.298
MAP@1000	0.091	0.111	0.086	0.110

with Okapi BM25 scoring model



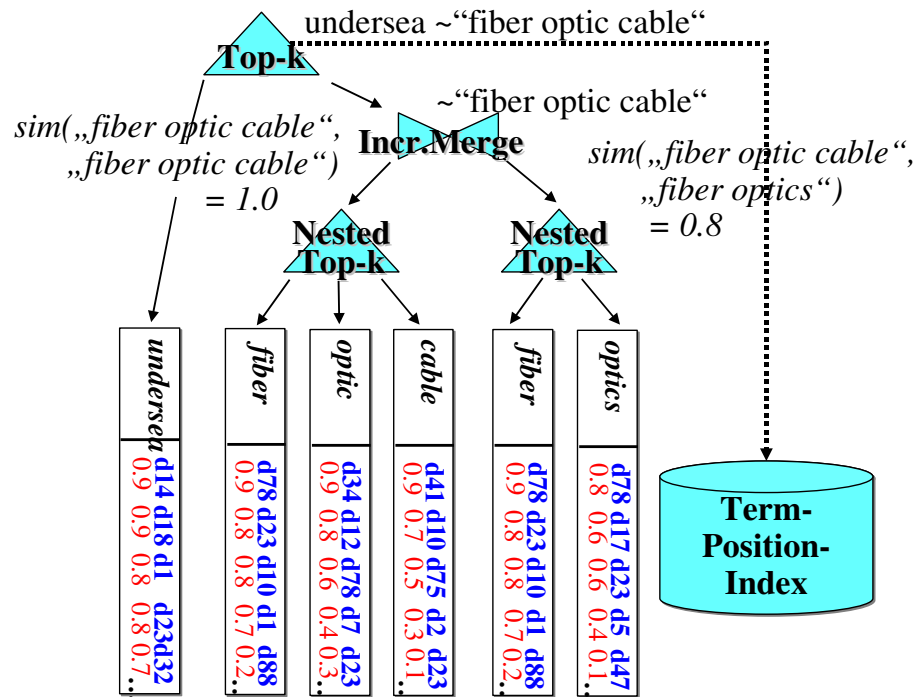
Nested Top-k Operators for Phrase Matching

phrase matching requires lookups in term-position index

→ expensive random accesses or

extra handicap for effective candidate pruning

→ embed phrase matching in operator tree as nested top-k



35/28

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

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

perform **NRA** (TA-sorted)

with [worstscore, bestscore] bookkeeping in priority queue Q
and round-robin SA to m index lists

...

after every r rounds of SA (i.e. $m \cdot r$ scan steps)

perform **RA** to look up all missing scores of „best candidate“ in Q
(where „best“ is in terms of
bestscore, worstscore, or $E[\text{score}]$, or $P[\text{score} > \text{min-k}]$)

cost **competitiveness** w.r.t. „optimal schedule“

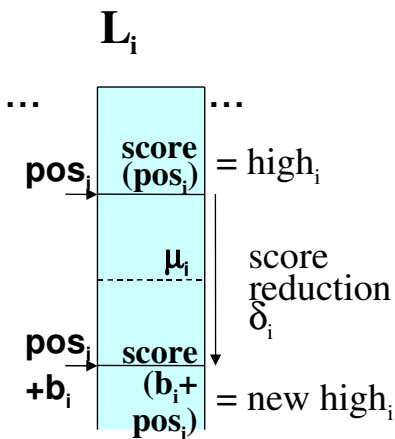
(scan until $\sum_i \text{high}_i \leq \min\{\text{bestscore}(d) \mid d \in \text{final top-k}\}$,

then perform RAs for all d' with $\text{bestscore}(d') > \text{min-k}$: **$4m + k$**)



Scheduling Example

available info:



+ histograms
+ convolutions

L_1	L_2	L_3
→ A: 0.8	→ G: 0.7	→ Y: 0.9
→ B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	→ P: 0.3
→ F: 0.17	→ Y: 0.5	→ F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	→ W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

$\delta = 1.46$

$\delta = 1.7$

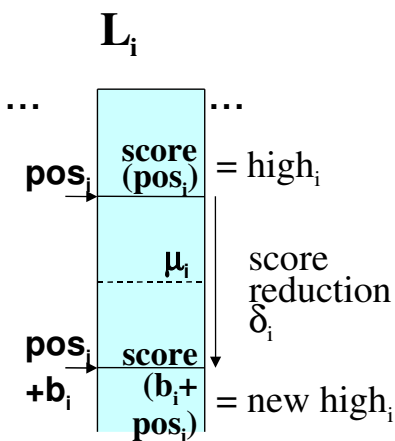
batch of $b = \sum_{i=1..m} b_i$ steps:
choose b_i values so as to
achieve high score reduction δ

+ carefully chosen RAs:
score lookups for
„interesting“ candidates



Scheduling Example

available info:



+ histograms
+ convolutions

L_1	L_2	L_3
→ A: 0.8	→ G: 0.7	→ Y: 0.9
→ B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	→ P: 0.3
→ F: 0.17	→ Y: 0.5	→ F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	→ W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

$\delta = 1.46$

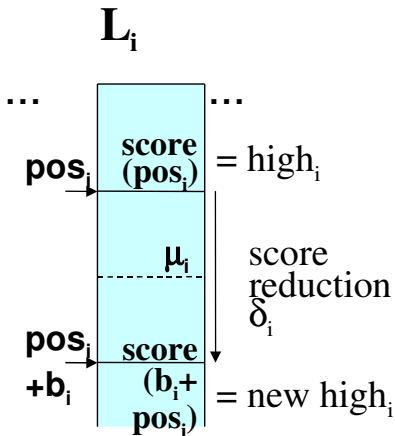
$\delta = 1.7$

choose b_i values so as to
achieve high score reduction δ



Scheduling Example

available info:



+ histograms
+ convolutions

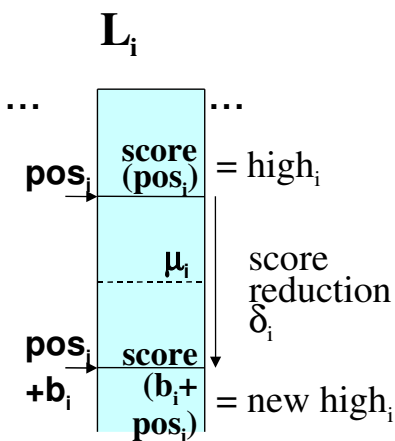
L_1	L_2	L_3
A: 0.8	G: 0.7	Y: 0.9
B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	P: 0.3
F: 0.17	Y: 0.5	F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

compute top-1 result
using flexible SAs and RAs



Scheduling Example

available info:



+ histograms
+ convolutions

L_1	L_2	L_3
A: 0.8	G: 0.7	Y: 0.9
B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	P: 0.3
F: 0.17	Y: 0.5	F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

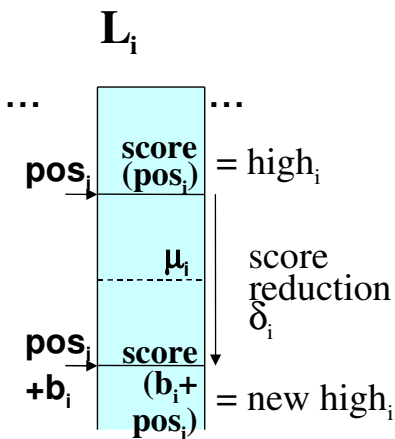
candidates:

A: [0.8, 2.4]	Y: [0.9, 2.4]
G: [0.7, 2.4]	?: [0.0, 2.4]



Scheduling Example

available info:



+ histograms
+ convolutions

L_1	L_2	L_3
A: 0.8	G: 0.7	Y: 0.9
B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	P: 0.3
F: 0.17	Y: 0.5	F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

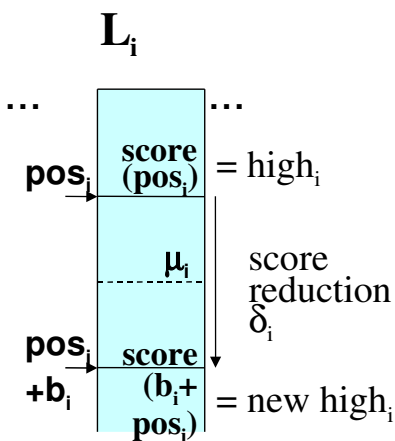
candidates:

A: [1.5, 2.0]	Y: [0.9, 1.6]
G: [0.7, 1.6]	?: [0.0, 1.4]



Scheduling Example

available info:



+ histograms
+ convolutions

L_1	L_2	L_3
A: 0.8	G: 0.7	Y: 0.9
B: 0.2	H: 0.5	A: 0.7
K: 0.19	R: 0.5	P: 0.3
F: 0.17	Y: 0.5	F: 0.25
M: 0.16	W: 0.3	S: 0.25
Z: 0.15	D: 0.25	T: 0.2
W: 0.1	W: 0.2	Q: 0.15
Q: 0.07	A: 0.2	X: 0.1
⋮	⋮	⋮

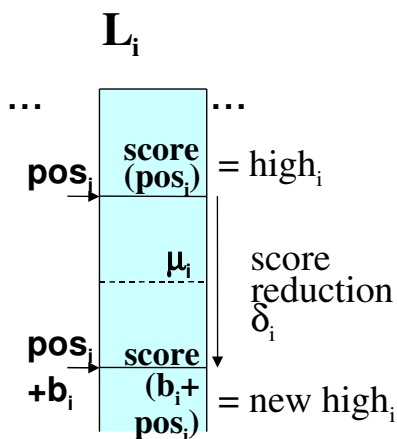
candidates:

A: [1.5, 2.0]	Y: [1.4, 1.6]
G: [0.7, 1.2]	

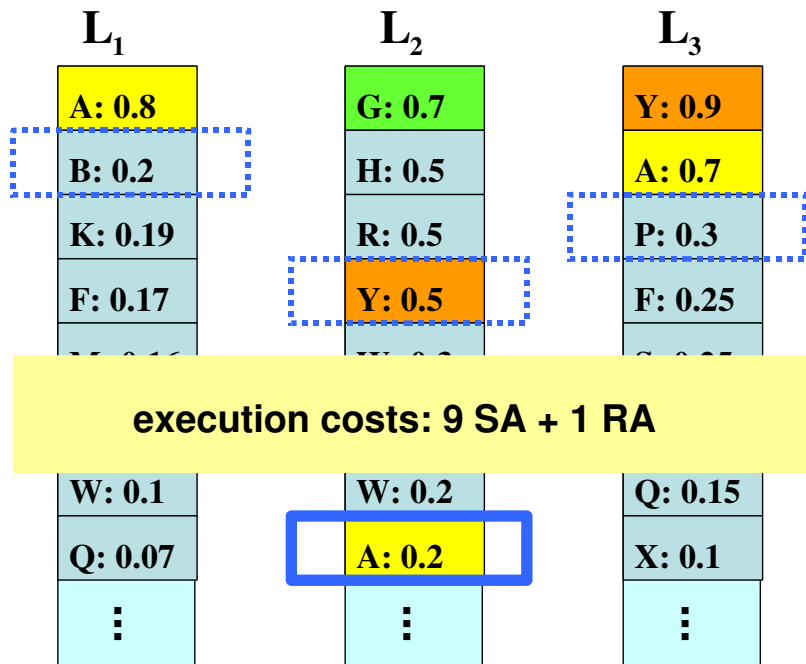


Scheduling Example

available info:



+ histograms
+ convolutions



candidates:

A: [1.7, 2.0]

~~Y: [1.4, 1.6]~~



SA Scheduling: Objective and Optimization

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

possible benefit definitions:

$$benefit(b_1..b_m) = \sum_{i=1..m} \Delta_i \quad \text{with} \quad \Delta_i = (high_i - score_i(pos_i + b_i)) / b_i$$

score gradient

$$benefit(b_1..b_m) = \sum_{i=1..m} \delta_i \quad \text{with} \quad \delta_i = score_i(pos_i) - score_i(pos_i + b_i)$$

score reduction

Solve **knapsack-style** NP-hard optimization problem
(e.g. for batched scans) or use greedy heuristics:

$$b_i := b * benefit(b_i=b) / \sum_{v=1..m} benefit(b_v=b)$$



SA Scheduling: Benefit Aggregation

Consider current top-k T and candidate queue Q;
 for each $d \in Q$ we know $E(d) \subseteq 1..m$, $R(d) = 1..m - E(d)$,
 $bestscore(d)$, $worstscore(d)$, $p(d) = P[score(d) > min-k]$

$$benefit(d, b_1..b_m) =$$

$$surplus(d)^{-1} \cdot \sum_{i \in E(d)} (high_i - score(pos_i + b_i))$$

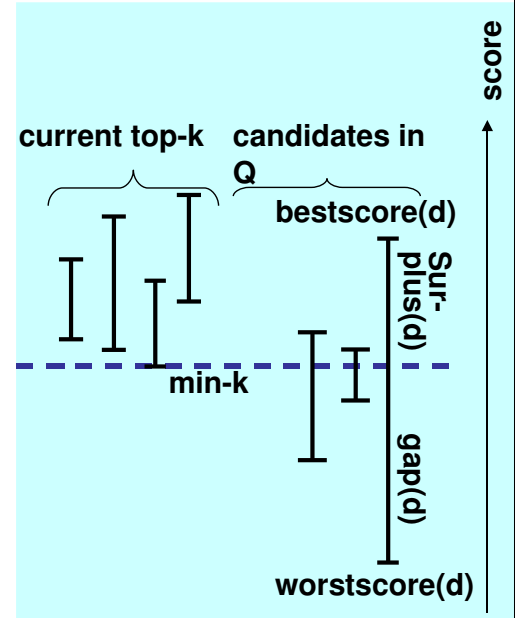
$$+ gap(d)^{-1} \cdot \sum_{i \notin E(d)} \mu_i$$

$$\text{with } surplus(d) = bestscore(d) - min-k$$

$$gap(d) = min-k - worstscore(d)$$

$$\mu_i = E[score(j) \mid j \in [pos_i, pos_i + b_i]]$$

$$benefit(b_1..b_m) = \sum_{d \in Q} benefit(d, b_1..b_m)$$



weighs documents and dimensions in benefit function
 max planck institut
 informatik

Gerhard Weikum May 19, 2006

45/28

Random-Access Scheduling

Perform additional RAs when helpful

- 1) to increase min-k (increase worstscore of $d \in \text{top-k}$) or
- 2) to prune candidates (decrease bestscore of $d \in Q$)

For 1) **Top Probing:**

- perform RAs for current top-k (whenever min-k changes),
 - and possibly for best d from Q
- (in desc. order of bestscore, worstscore, or $P[score(d) > min-k]$)

For 2) **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

(would be optimal for linear increase of SA-cost(t) and
 hyperbolically decreasing remaining-RA-cost(t))

with **score-prediction & cost model** for deciding RA order

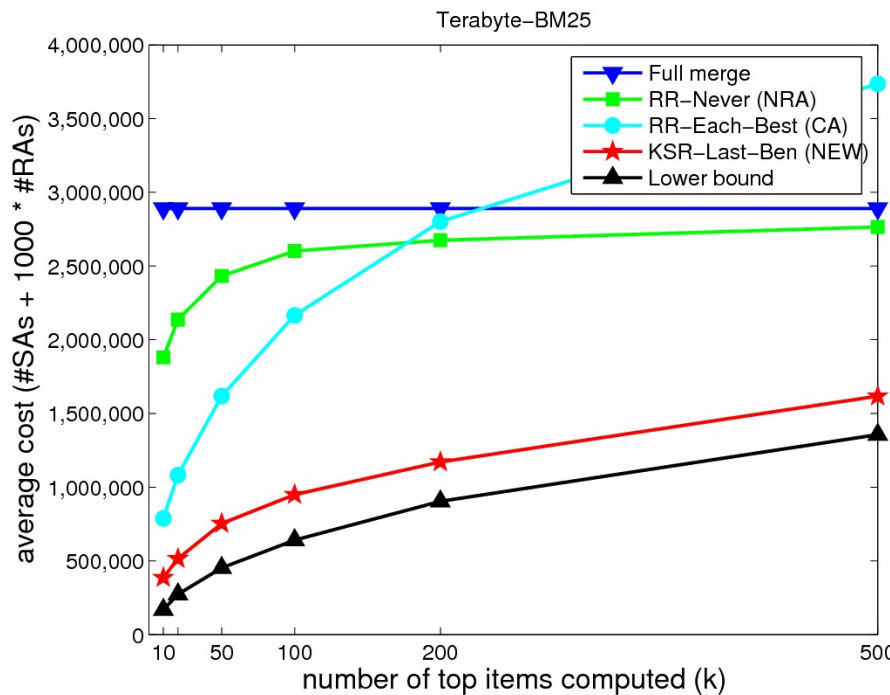


max planck institut
 informatik

Gerhard Weikum May 19, 2006

46/28

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



Query Scores for Content Conditions

- Basic scoring idea within element statistics

$$score(e, T[c_1, \dots, c_n])$$

tag	N	avglength	k ₁	b
article	16,850	2,903	10.5	0.75
sec	96,709	413	10.5	0.75
par	1,024,907	32	10.5	0.75
fig	109,230	13	10.5	0.75

- Content-based scores cast into an **Okapi-BM25 probabilistic model** with **element-specific model parameterization**

$$score(e, T[c_1, \dots, c_n]) = \prod_{i=1}^n \frac{(k_1 + 1)tf(c_i, e)}{K + tf(c_i, e)} \times \log \frac{N_T - ef_T(c_i) + 0.5}{ef_T(c_i) + 0.5}$$

$$with K = k_1 \frac{\sum_e \frac{length(e)}{avg_e\{length(e') | e' \text{ with tag } T\}}{c(1 - b) + b}$$

where
 tf(c_i, e) number of occurrences of c_i in element e
 N_T number of elements with tag T
 ef_T(c_i) number of elements with tag T that contain c_i



Scoring Model for TopX

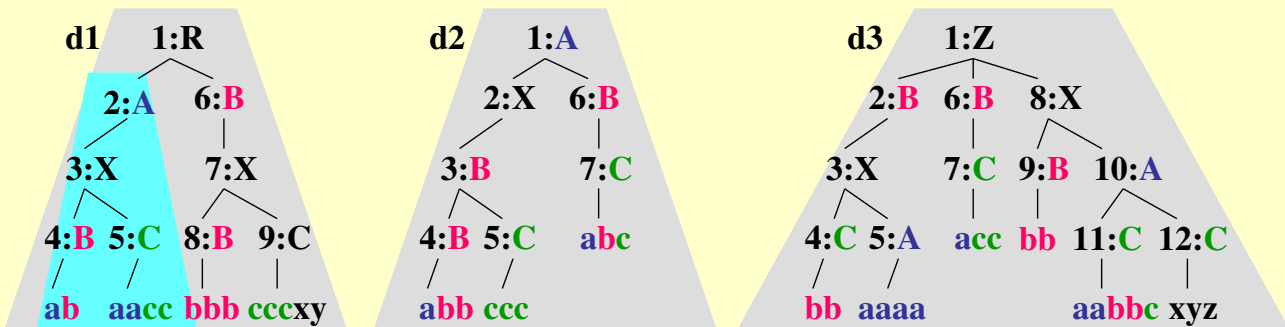
restricted to tree data (disregarding links)
using tag-term conditions for scoring

score (doc d or subtree rooted at $d.n$)
for query q with tag-term conditions $X_1[t_1], \dots, X_m[t_m]$
matched by nodes n_1, \dots, n_m

$$= \sum_{i=1}^m \frac{\text{relevance}(n_i, t_i) \cdot \text{specificity}(X_i, t_i)}{\text{compactness}(n_i)}$$

relevance: tf-based or BM25
specificity: idf per tag type
compactness: subtree size

Example:



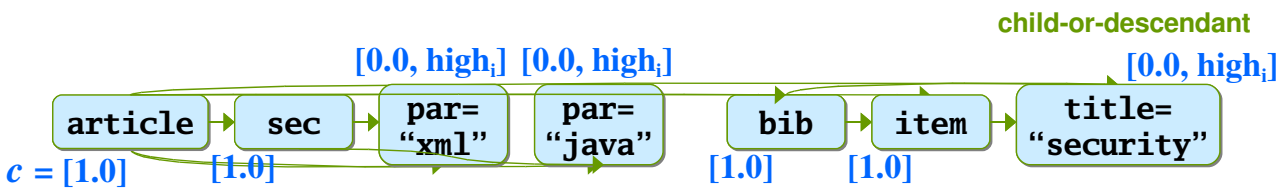
Example query: `//A [//“a“ & //B[//“b“] & //C[//“c“]`



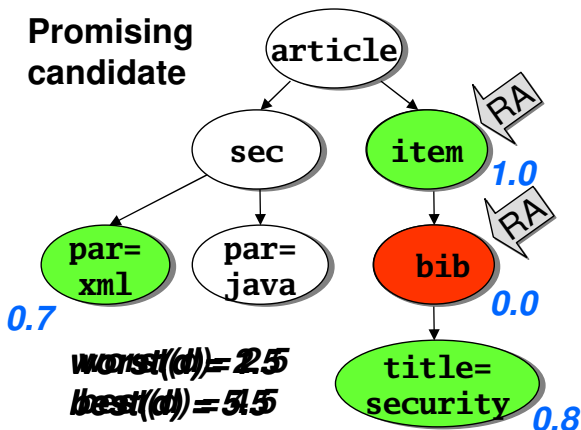
Incremental Testing of Path Conditions

Query:

`//article[//sec//par=xml java]//bib//item//title=security`



Promising candidate



- **Complex query DAGs**
 - Transitive closure of PC's
 - Aggregate add'l **score mass c** for PC i if all edges rooted at i are satisfied
- **Incrementally test PC's**
 - Quickly decrease bestscores for pruning
 - **Schedule RAs** based on cost model; **Ben-Probe: $EWC-RA(d) < EWC-SA(d)$** based on histogram convolutions and selectivity estimation



Scheduling: Cost Model

• BenProbe-Scheduling

- Analytic cost model
- Basic idea
 - Compare expected access costs to the costs of an optimal schedule
 - Access costs on d are wasted, if d does not make it into the final top-k (considering both content & structure)
- Compare different **Expected Wasted Costs (EWC)**
 - $EWC-RA_s(d)$ of looking up d in the **structure**
 - $EWC-RA_c(d)$ of looking up d in the **content**
 - $EWC-SA(d)$ of not seeing d in the next batch of b
- Schedule batch of RAs on d , if

$EWC-SA =$

$$\frac{b \cdot m}{|Q|} \sum_{d \in Q} \left(\prod_{i \notin E(d)} \left(1 - \frac{b}{n} \right) \right)$$

$$EWC-RA(d) [RA] < EWC-SA [SA]$$



Structural Selectivity Estimator

- Split the query into a set of characteristic patterns, e.g., **twigs, descendants & tag-term pairs**

- Consider **structural selectivities**

$P[d \text{ satisfies all structural conditions } Y] =$

$$\prod_{i \in Y} p_i$$

$P[d \text{ satisfies a subset } Y' \text{ of structural conditions } Y] =$

$$\sum_{Y' \subseteq Y} \prod_{i \in Y'} p_i \prod_{i \notin Y'} (1 - p_i)$$

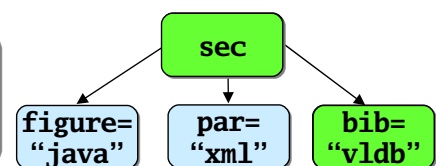
- Consider **binary correlations** between structural patterns and/or tag-term pairs

$P[d \text{ satisfies all structural conditions } Y] =$

$$\prod_{i \in Y} \frac{p_i p_{i+1} + \text{cov}(X_i, X_{i+1})}{p_i}$$

- With $\text{cov}(X_i, X_j)$ estimated from **contingency tables, query logs, etc.**

```
//sec[//figure="java"]
  [//par="xml"]
  [//bib="vldb"]
```



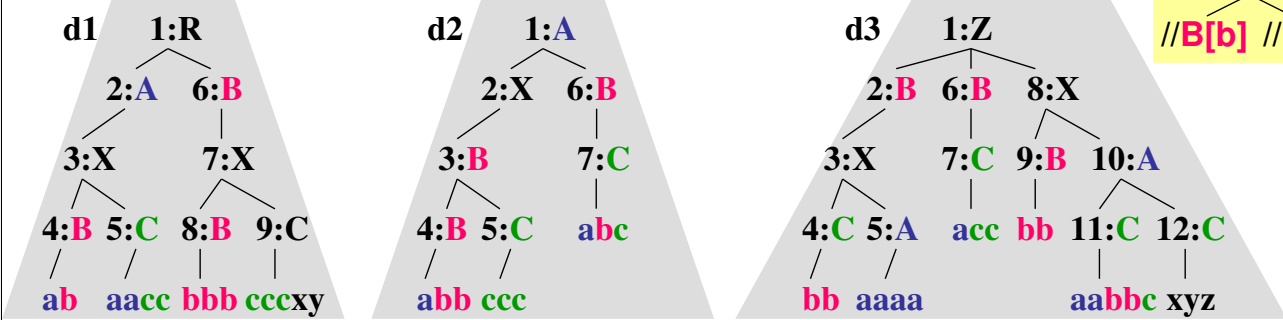
$EWC-RA_s(d)$

//sec[//figure]//par	$p_1 = 0.682$
//sec[//figure]//bib	$p_2 = 0.001$
//sec[//par]//bib	$p_3 = 0.002$
//sec//figure	$p_4 = 0.688$
//sec//par	$p_5 = 0.968$
//sec//bib	$p_6 = 0.002$
//sec	$p_7 = 0.986$
//par="xml"	$p_8 = 0.067$
//figure="java"	$p_9 = 0.011$
//bib="vldb"	$p_{10} = 0.023$



TopX Example Data

Ex. query: //A [//“a“ & //B[//“b“] & //C[//“c“]]



pre-computed index table	Tag	Term	MaxScore	DocId	Score	ElemId	Pre	Post
	A	a	1	d3	1	e5	5	2
	A	a	1	d3	1/4	e10	10	9
	A	a	1/2	d1	1/2	e2	2	4
block-scans:	A	a	2/9	d2	2/9	e1	1	7
(A, a, d3, ...)	B	b	1	d1	1	e8	8	5
(B, b, d1, ...)	B	b	1	d1	1/2	e4	4	1
(B, b, d1, ...)	B	b	1	d1	3/7	e6	6	8
(C, c, d2, ...)	B	b	1	d3	1	e9	9	7
(A, a, d1, ...)	B	b	1	d3	1/3	e2	2	4
(B, b, d3, ...)	B	b	2/3	d2	2/3	e4	4	1
(B, b, d3, ...)	B	b	2/3	d2	1/3	e3	3	3
(C, c, d3, ...)	B	b	2/3	d2	1/3	e6	6	6
(A, a, d2, ...)	C	c	1	d2	1	e5	5	2
(B, b, d2, ...)	C	c	1	d2	1/3	e7	7	5
(C, c, d1, ...)	C	c	2/3	d3	2/3	e7	7	5
(A, a, d2, ...)	C	c	2/3	d3	1/5	e11	11	8
(B, b, d2, ...)	C	c	3/5	d1	3/5	e9	9	6
(C, c, d1, ...)	C	c	3/5	d1	1/2	e5	5	2

with appropriate B+ tree index on (Tag, Term, MaxScore, DocId, Score, ElemId)

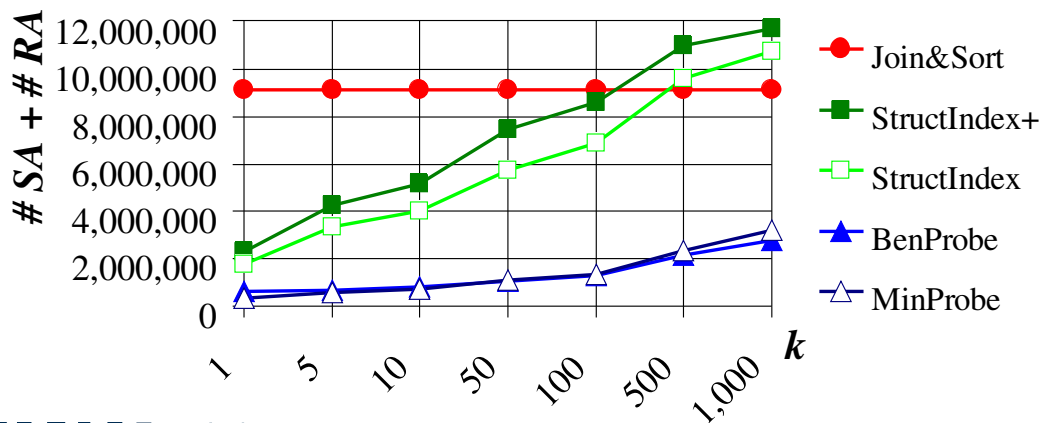
Experiments – Datasets & Competitors

- **INEX '04 benchmark setting**
 - 12,223 docs; 12M elemt's; 119M content features; **534MB**
 - 46 queries, e.g., //article[//bib="QBIC" and //p="image retrieval"]
- **IMDB (Internet Movie Database)**
 - 386,529 docs; 34M elemt's; 130M content features; **1,117 MB**
 - 20 queries, e.g., //movie[//casting[//actor="John Wayne" //role="Sheriff"] // [//year="1959" and //genre="Western"]
- **Competitors**
 - **DBMS-style Join&Sort**
 - Using the TopX schema
 - **StructIndex**
 - Top-k with separate inverted indexes for content & structure
 - DataGuide-like structural index
 - Full evaluations → no uncertainty about final document scores
 - No candidate queue, eager random access
 - **StructIndex+**
 - **Extent chaining** technique for DataGuide-based extent identifiers (skip scans)



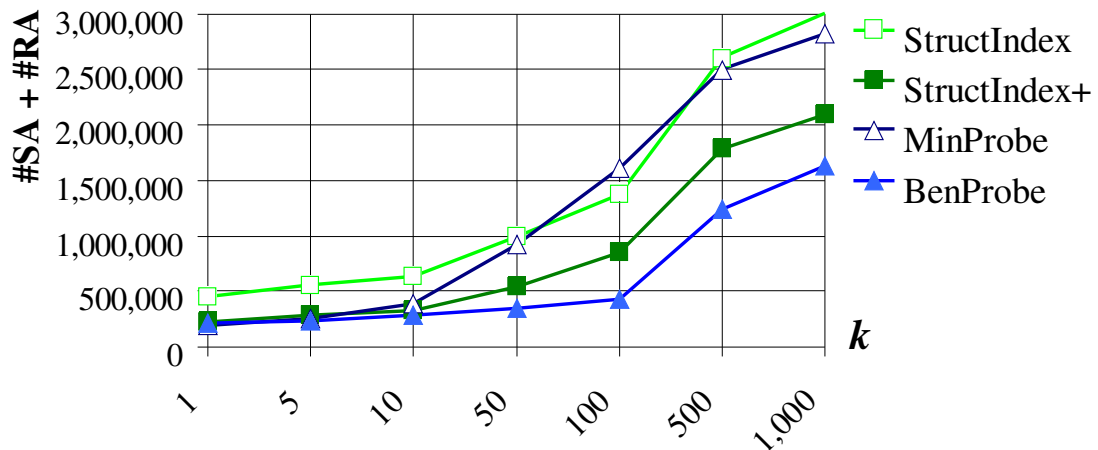
INEX Results

	k	epsilon	# SA	# RA	CPU	P@k	MAP@k	relPrec
Join&Sort	10	n/a	9,122,318	0	0.261	0.34	0.09	1.00
StructIndex	10	n/a	761,970	3,25,068	0.37			
StructIndex+	10	n/a	77,482	5,074,384	1.87			
TopX - MinProbe	10	0.0	635,507	64,807	0.03			
TopX - BenProbe	10	0.0	723,169	84,424	0.07			
TopX - BenProbe	1,000	0.0	882,929	1,902,427	0.35	0.03	0.17	1.00



IMDB Results

	k	epsilon	# SA	# RA	CPU	P@k	MAP@k	relPrec
Join&Sort	10	n/a	14,510,077	0	37.7	n/a	1.00	
StructIndex	10	n/a	346,697	291,655	0.16			
StructIndex+	10	n/a	22,445	301,647	0.17			
TopX - MinProbe	10	0.0	317,380	72,196	0.08			
TopX - BenProbe	10	0.0	241,471	50,016	0.06			



INEX with Probabilistic Pruning

	k	ϵ	# SA	# RA	CPU	P@ k	MAP@ k	relPrec
TopX - MinProbe	10	0.00	635,507	64,807	0.03	0.34	0.09	1.00
	10	0.25	392,395	56,952	0.05	0.34	0.08	0.77
	10	0.50	231,109	48,963	0.02	0.31	0.08	0.65
	10	0.75	102,118	42,174	0.01	0.33	0.08	0.51
	10	1.00	36,936	35,327	0.01	0.30	0.07	0.38

