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***

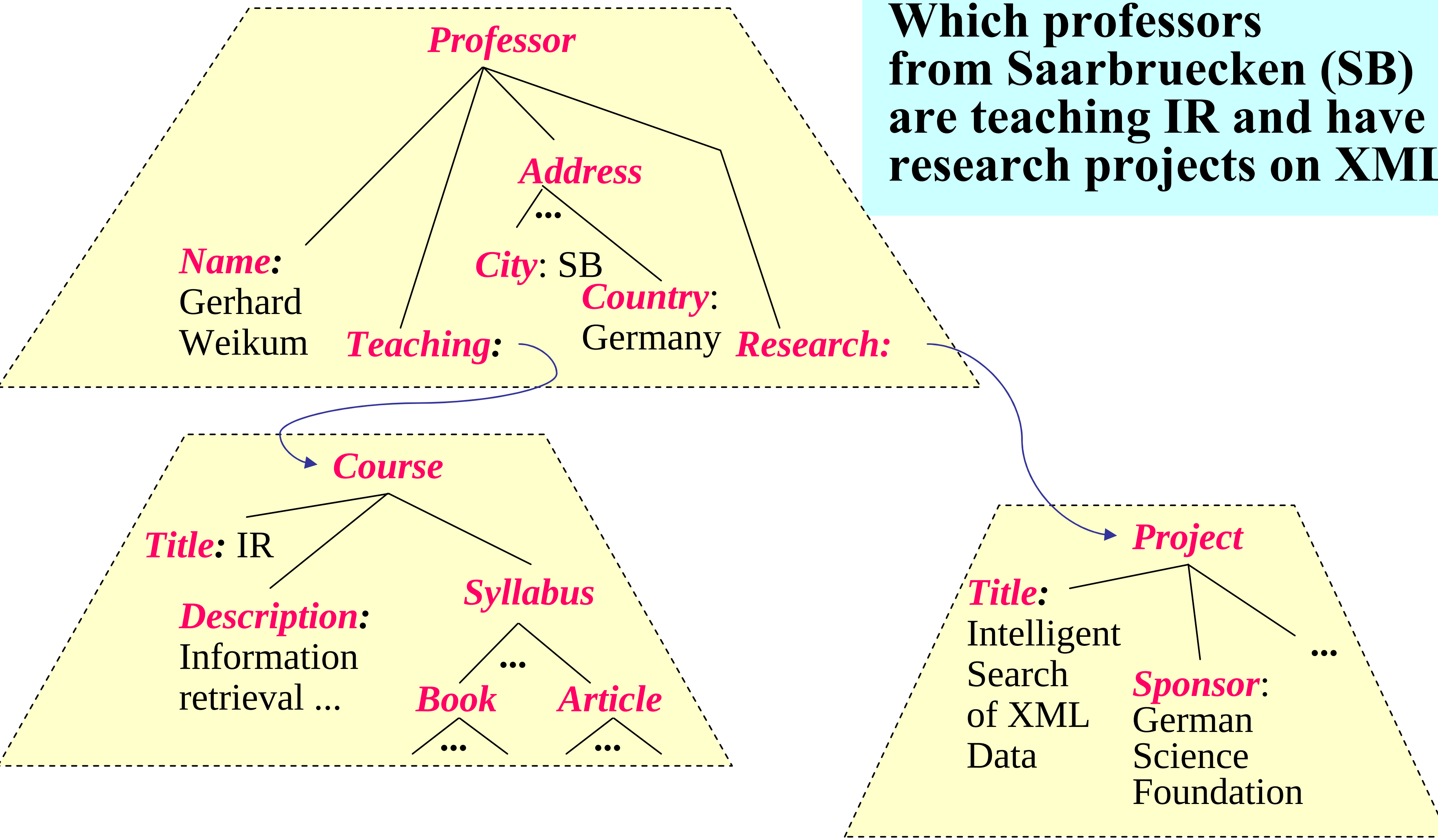
- ***drama with three women making a prophecy to a British nobleman that he will become king***

- ***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**

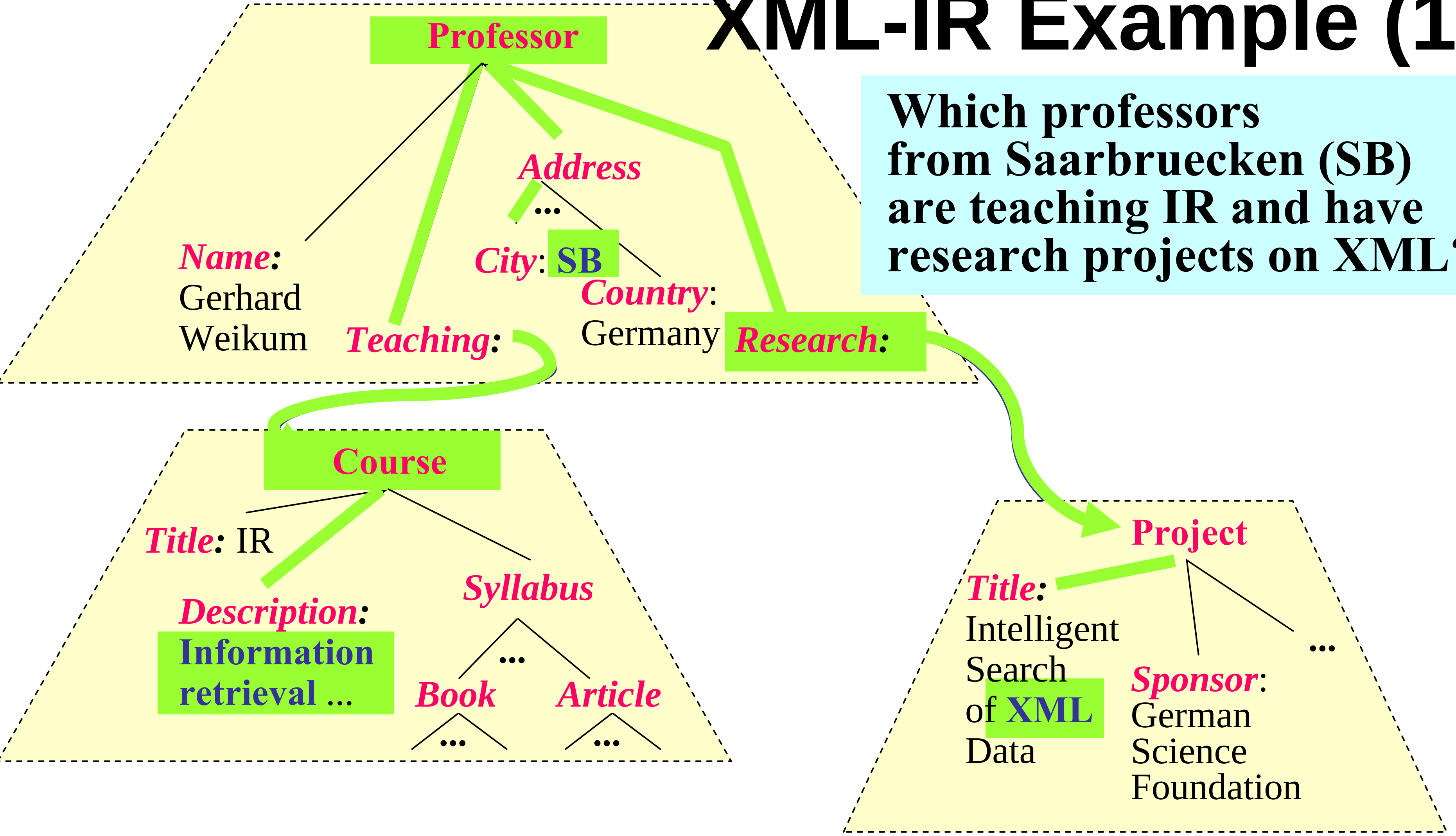
What If The Semantic Web Existed And All Information Were in XML?

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



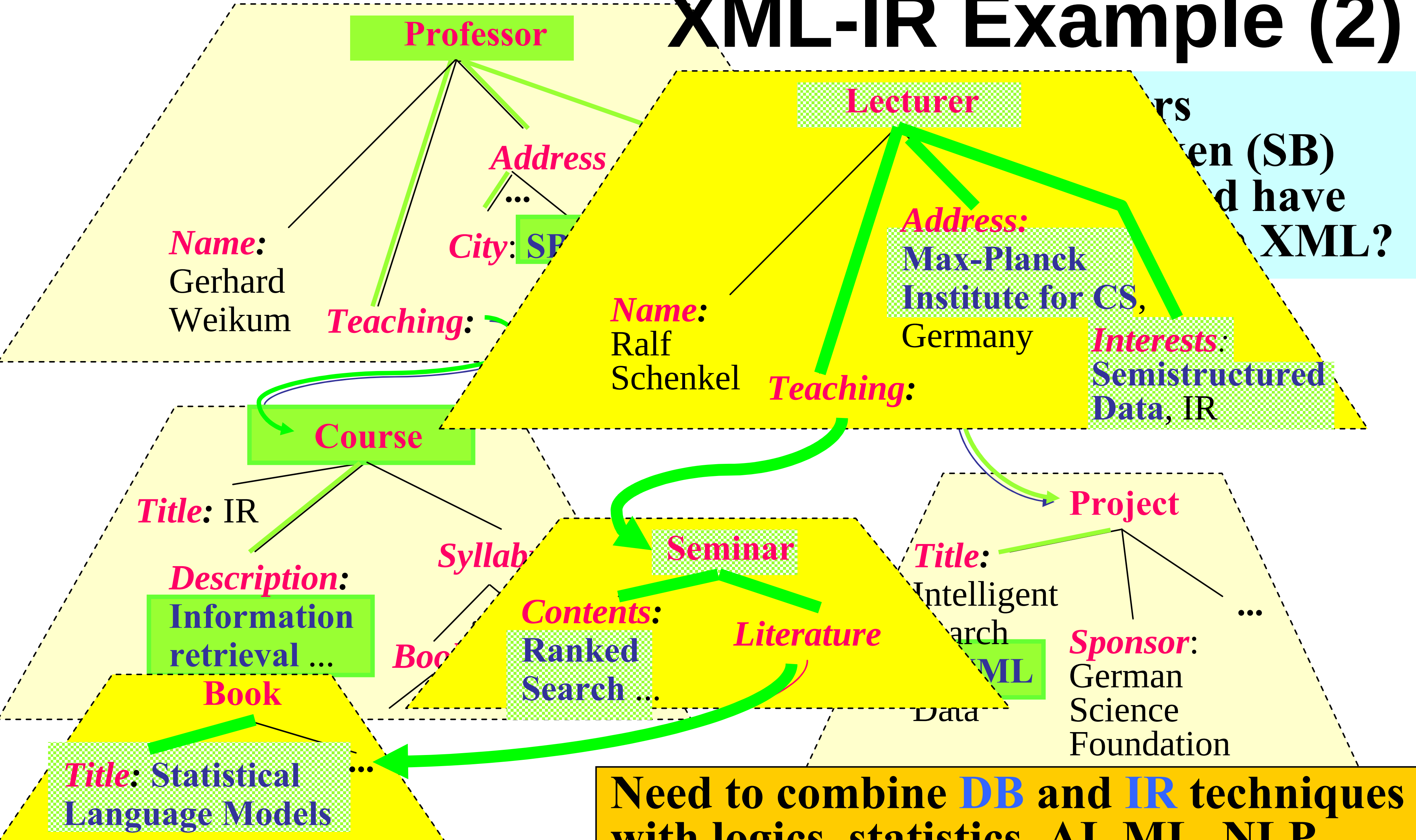
XML-IR Example (1)

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



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

XML-IR Example (2)



Need to combine DB and IR techniques with logics, statistics, AI, ML, NLP for ranked retrieval

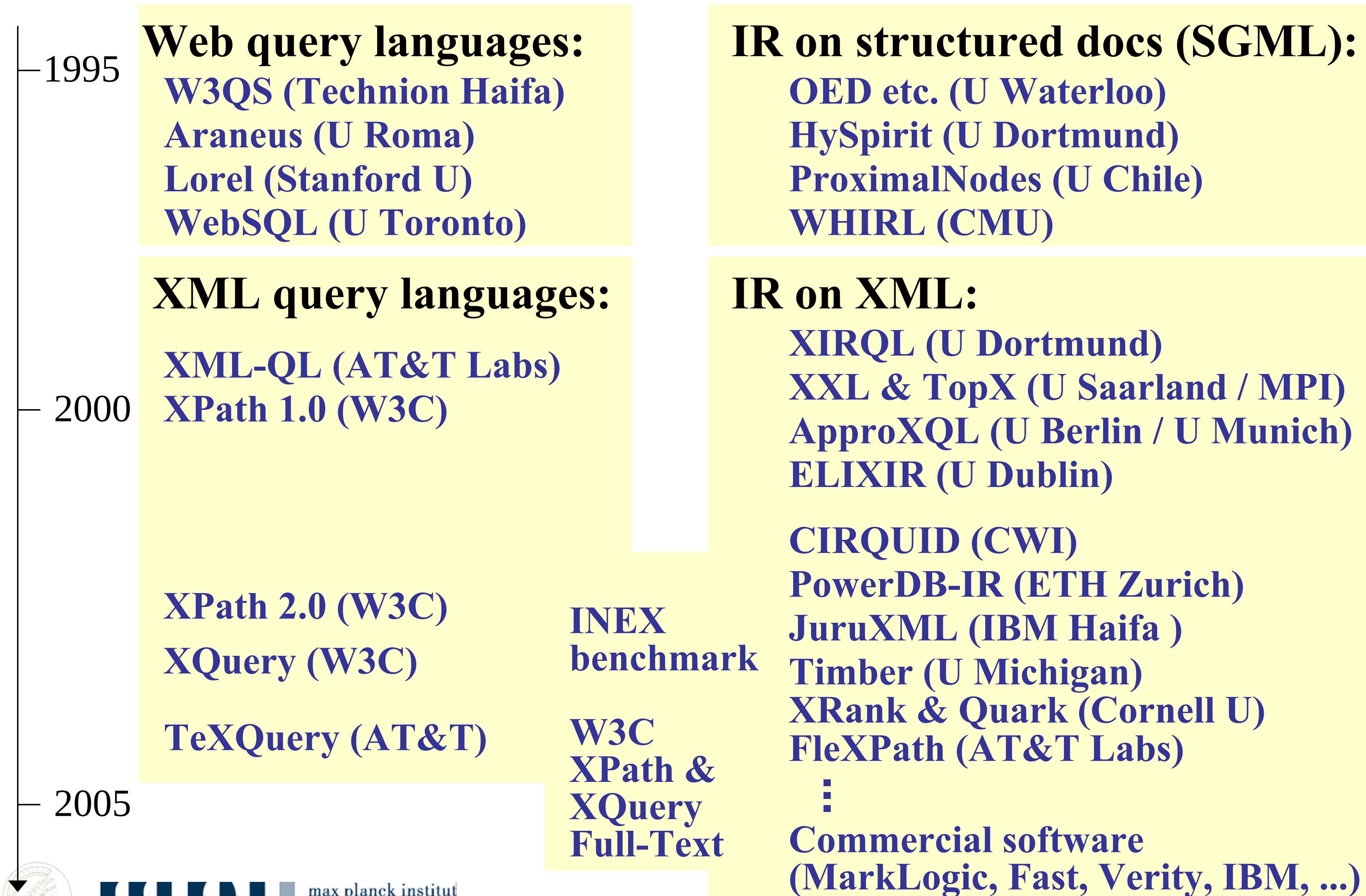
// *~Professor [//* = „~ Saa*
[// *~Course [//* = „ ~ IR“]]
[// *~Research [//* = „~ XML“]]

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 an[d contents]

Query result:
- **query is a pattern with relaxable conditions**
- **results are approximate matches to query with similarity scores**

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

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

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: ~c = ~t1 … ~tm

Example:

~professor and (~course = „~IR“)

//professor[//place = „SB“]//course = „IR“

Term2Concept with WSD

Query expansion

exp(ti)={w | sim(ti,w)≥ θ}

Weighted expanded query

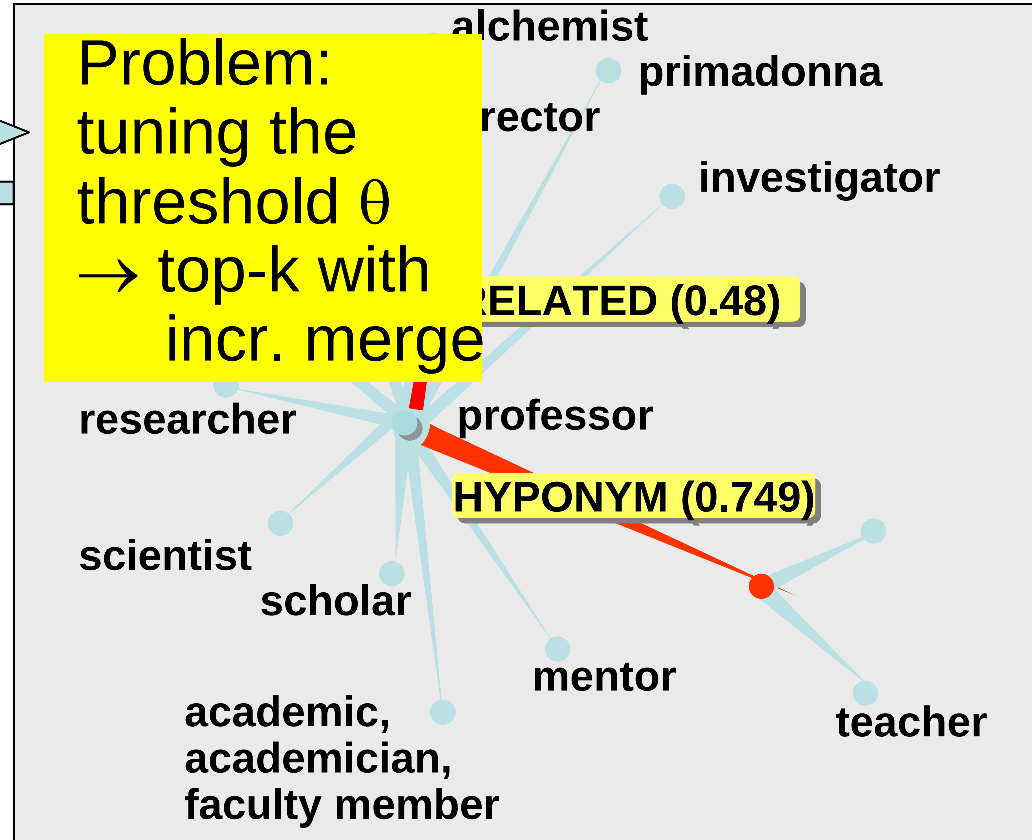
Example:

(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

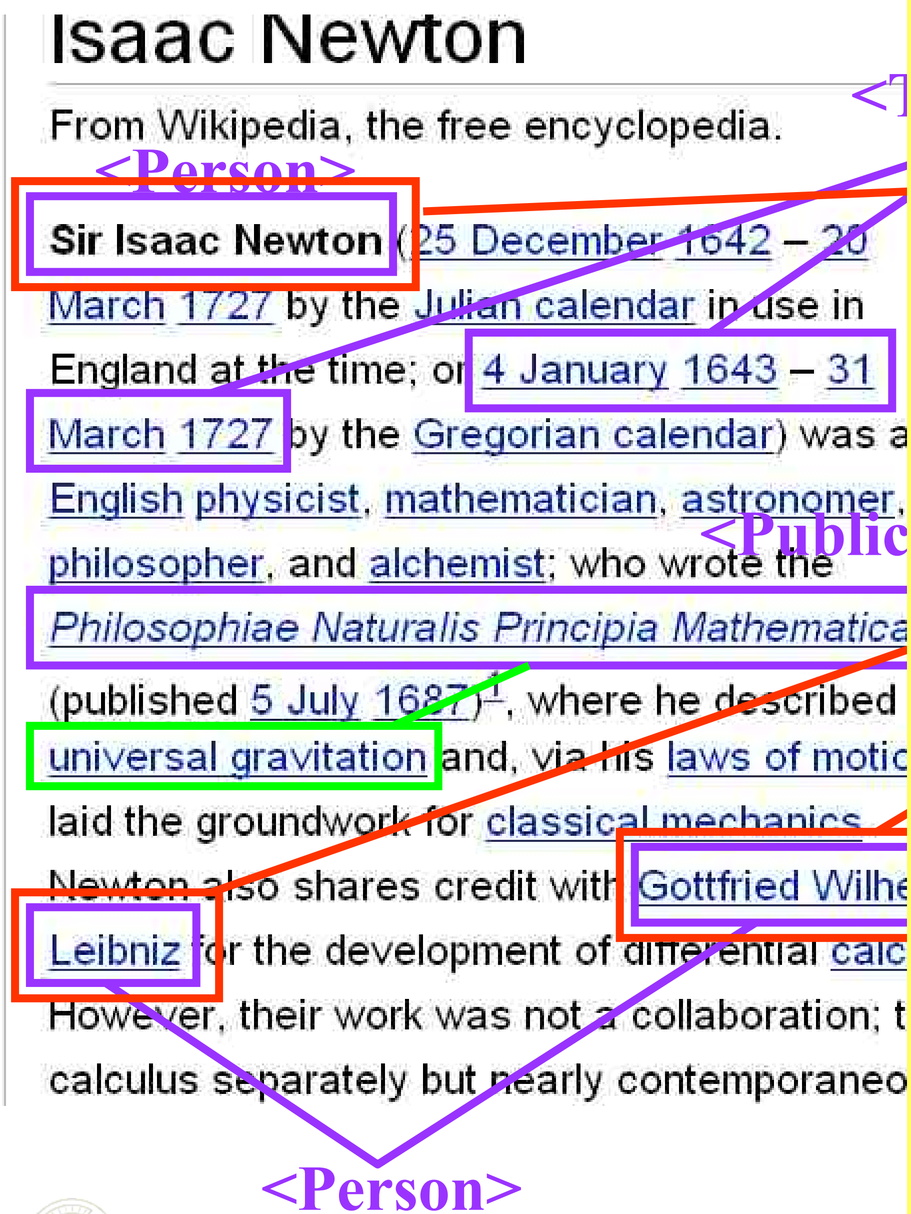
better recall, better mean precision for hard queries

Thesaurus/Ontology:
concepts, relationships, glosses from WordNet, Gazetteers, Web forms & tables, Wikipedia



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 a
English physicist, mathematician, astronomer,
philosopher, and alchemist; who wrote the
Philosophiae Naturalis Principia Mathematica
(published 5 July 1687)[1], where he described
universal gravitation and, via his laws of motio
laid the groundwork for classical mechanics.
Newton also shares credit with Gottfried Wilhe
Leibniz for the development of differential calc
However, their work was not a collaboration; t
calculus separately but nearly contemporaneo

Information extraction:
(NLP, reg.exp., lexicon, HMM, CRF, etc.)

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

Publication	Topic
Philosophiae 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 !

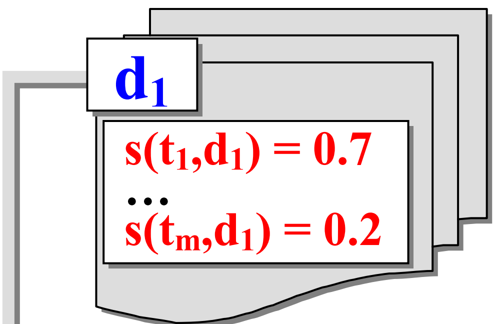
Outline

- ✓ 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]

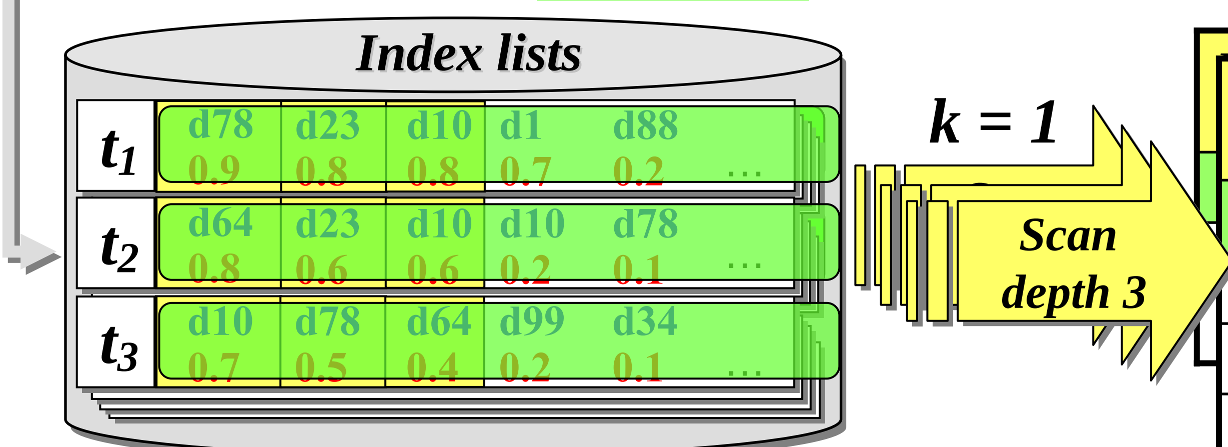
TA: efficient & principled top-k query processing with monotonic score aggr.

Data items: $d_1, \dots, d_n$



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

fetch lists join&sort



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

<u>TA with sorted access only (NRA):</u>
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{$s(t_\nu,d) \mid \nu \in E(d)$};
bestscore(d) := aggr{worstscore(d), aggr{$high_\nu \mid \nu \notin E(d)$}};
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}; s
threshold := max {bestscore(d') | d'∈ cand};
if threshold ≤ min-k then exit;

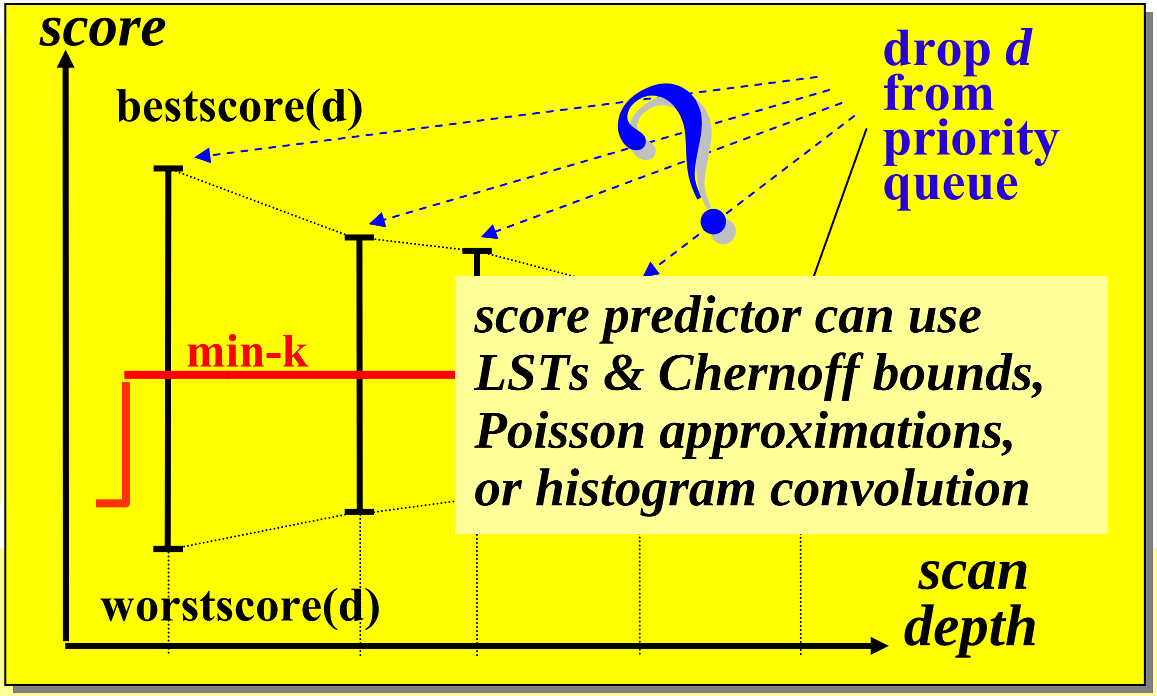
Probabilistic Pruning of Top-k Candidates [VLDB 04]

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

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

- **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)



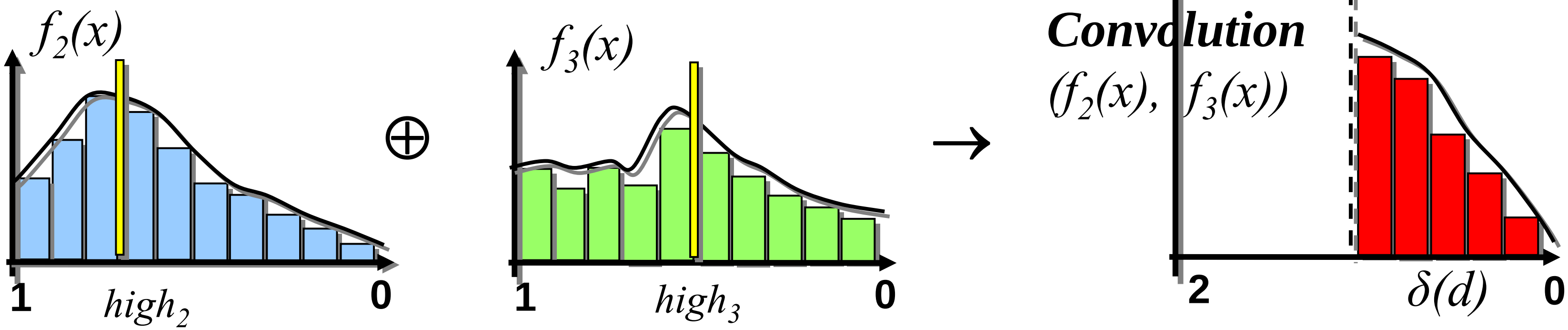
→ Approximate top-k with probabilistic guarantees:

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

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

Probabilistic Threshold Test

cand doc d
with
2 ∉ E(d),
3 ∉ E(d)



- **postulating uniform or Pareto score distribution in [0, 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 tim**

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.:

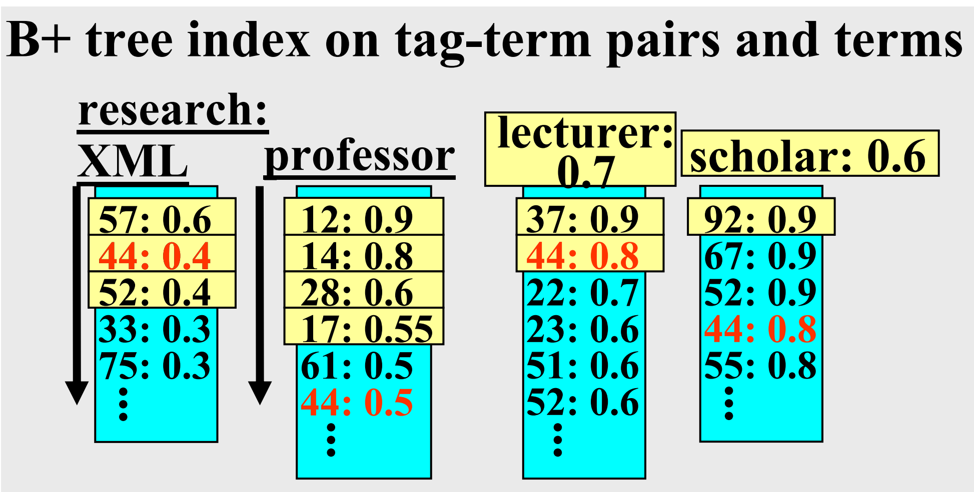
- **„juvenile delinquency *youth minor crime law jurisdiction offense prevention*“,**
- **„legalization marijuana *cannabis drug soft leaves plant smoked chewed euphoric abuse substance possession control pot grass dope weed smoke*“**

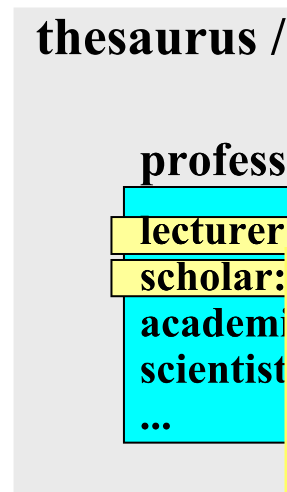
	TA-sorted	Prob-sorted (smart)
#sorted accesses	22,403,490	18,287,636
elapsed time [s]	7908	1066
max queue size	70896	400
relative recall	1	0.88
rank distance	0	14.5
score error	0	0.035

Top-k Queries with Query Expansion [SIGIR 05]

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

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





TREC-13 Robust Track benchmark: speedup by factor 4 at high precision/recall;

also handles TREC-13 Terabyte benchmark

+ 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 Σ_i high$_i$ $\leq$ min{bestscore(d) | d $\in$ *final top-k*},
then perform RAs for all d‘ with bestscore(d‘) > min-k): $4m + k$

Index Access Optimization [VLDB 06]

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

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

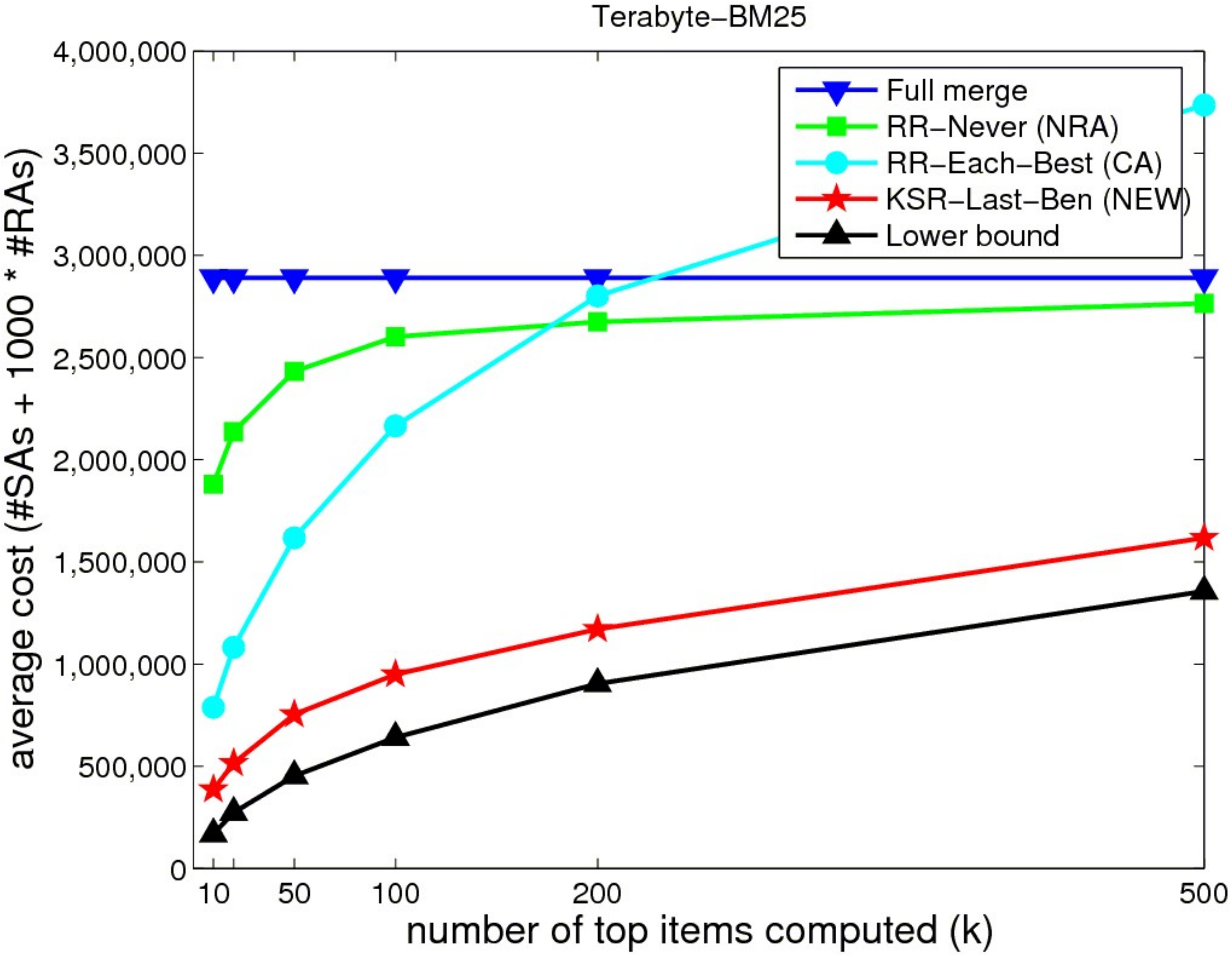
perform RAs for all candidates at point t when

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

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



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
1) 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

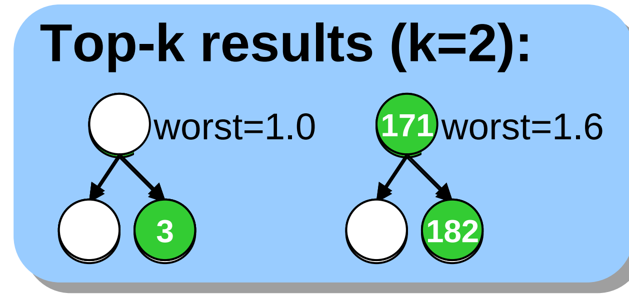
based on index table (with several B+ tree indexes):
L (Tag, Term, MaxScorePerDoc, DocId, Score, ElemId, Pre, Post)

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]



min-k = 1.0

1.0 0.85 0.1

//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

1.0 0.8 0.5

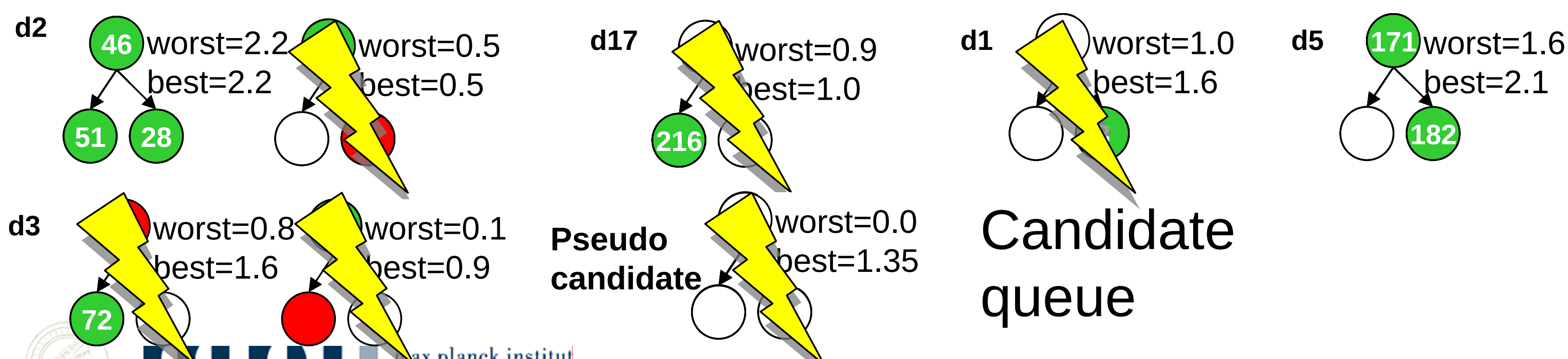
//title[xml]

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

1.0 0.8 0.75

//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



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 (ε=0.0)	TopX (ε=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		
precision@10	0.34	0.34		
MAP@1000	0.17	0.17		

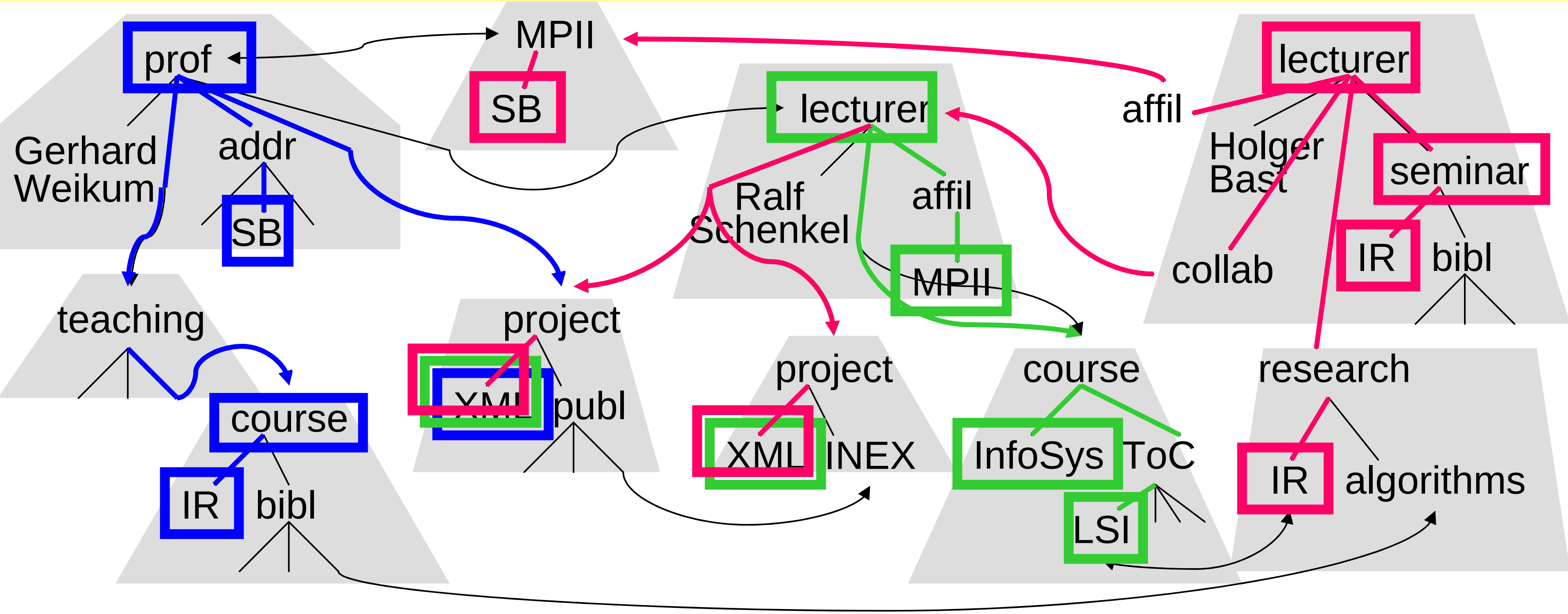
TopX outperforms Join&Sort by factor > 10 and
beats StructIndex by factor > 20 on INEX, factor 2-3 on IMDB

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***