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?

Name: Gerhard Weikum
City: SB
Country: Germany

Teaching:

Course
Title: IR
Description: Information retrieval ...

Syllabus
Book
Article

Research:

Project
Title: Intelligent Search of XML Data
Sponsor: German Science Foundation

Gerhard Weikum May 19, 2006
XML-IR Example (1)

Professor Name: Gerhard Weikum
Address
City: SB
Country: Germany

Teaching: Course Title: IR
Description: Information retrieval ...
Syllabus

Research: Project Title: Intelligent Search of XML Data
Sponsor: German Science Foundation

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

// Professor [//* = „Saarbruecken“]
[// Course [//* = „IR“]]
[// Research [//* = „XML“]]
Which professors from Saarbruecken (SB) are teaching IR and have research projects on XML?

// ~Professor [//* = "~ Saarbruecken"]
[// ~Course [//* = "~ IR"]]
[// ~Research [//* = "~ XML"]]

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

- XML IR & Ontologies
  - Efficient Top-k QP
  - TopX: Efficient XML IR
- Conclusion
# XML-IR: History and Related Work

<table>
<thead>
<tr>
<th>Year</th>
<th>Web query languages</th>
<th>XML query languages</th>
<th>IR on structured docs (SGML)</th>
<th>IR on XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>1995</td>
<td>W3QS (Technion Haifa) Araneus (U Roma) Lorel (Stanford U) WebSQL (U Toronto)</td>
<td>XML-QL (AT&amp;T Labs) XPath 1.0 (W3C)</td>
<td>OED etc. (U Waterloo) HySpirit (U Dortmund) ProximalNodes (U Chile) WHIRL (CMU)</td>
<td>XIRQL (U Dortmund) XXL &amp; TopX (U Saarland / MPI) APPROXQL (U Berlin / U Munich) ELIXIR (U Dublin)</td>
</tr>
<tr>
<td>2000</td>
<td>XPath 2.0 (W3C) XQuery (W3C)TeXQuery (AT&amp;T)</td>
<td></td>
<td>CIRQUID (CWI) PowerDB-IR (ETH Zurich) JuruXML (IBM Haifa) Timber (U Michigan) XRANK &amp; Quark (Cornell U) FlexXPath (AT&amp;T Labs)</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td>INEX benchmark W3C XPath &amp; XQuery Full-Text</td>
<td>INEX benchmark W3C XPath &amp; XQuery Full-Text</td>
<td>Commercial software (MarkLogic, Fast, Verity, IBM, ...</td>
</tr>
</tbody>
</table>
XML-IR Concepts

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

search condition: conjunction of restricted *path expressions*

**Elementary conditions** on names and contents

```
// 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

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.

Sir Isaac Newton (25 December 1642 – 31 March 1727) was an English physicist, mathematician, astronomer, philosopher, and alchemist; who wrote the Philosophiae Naturalis Principia Mathematica (published 5 July 1687), 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 both worked calculus separately but nearly contemporaneously.

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

| Person       | TimePeriod      | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sir Isaac Newton</td>
<td>4 Jan 1643 - ...</td>
</tr>
<tr>
<td>... Leibniz</td>
<td>... Kneller</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publication</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philosophiae Naturalis</td>
<td>... gravitation</td>
</tr>
</tbody>
</table>

Author Publication

<table>
<thead>
<tr>
<th>Author</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>... Newton</td>
<td>Philosophia ...</td>
</tr>
</tbody>
</table>

Database (tables or XML)

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: \(d_1, \ldots, d_n\)

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

Index lists

<table>
<thead>
<tr>
<th>(t_1)</th>
<th>(d_{78})</th>
<th>(d_{23})</th>
<th>(d_{10})</th>
<th>(d_1)</th>
<th>(d_{88})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>0.7</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(t_2)</th>
<th>(d_{64})</th>
<th>(d_{23})</th>
<th>(d_{10})</th>
<th>(d_{10})</th>
<th>(d_{78})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8</td>
<td>0.6</td>
<td>0.6</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(t_3)</th>
<th>(d_{10})</th>
<th>(d_{78})</th>
<th>(d_{64})</th>
<th>(d_{99})</th>
<th>(d_{34})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.7</td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**fetch lists join&sort**

**TA with sorted access only (NRA):**

- can index lists; consider \(d\) at pos \(i\) in \(L_i\);
- \(E(d) := E(d) \cup \{i\}\); \(\text{high}_i := s(t_i, d)\);
- \(\text{worstscore}(d) := \text{aggr}\{s(t_\nu, d) \mid \nu \in E(d)\}\);
- \(\text{bestscore}(d) := \text{aggr}\{\text{worstscore}(d), \text{aggr}\{\text{high}_\nu \mid \nu \notin E(d)\}\}\};

If \(\text{worstscore}(d) > \text{min-k}\) then add \(d\) to top-k

- \(\text{min-k} := \min\{\text{worstscore}(d') \mid d' \in \text{top-k}\}\);

Else if \(\text{bestscore}(d) > \text{min-k}\) then

- \(\text{cand} := \text{cand} \cup \{d\}\); \(s\)

- \(\text{threshold} := \max \{\text{bestscore}(d') \mid d' \in \text{cand}\}\);

If \(\text{threshold} \leq \text{min-k}\) then exit;

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

- **Scan depth 1**
- **Scan depth 2**
- **Scan depth 3**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Doc</th>
<th>Worst-score</th>
<th>Best-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d10</td>
<td>2.1</td>
<td>2.1</td>
</tr>
<tr>
<td>2</td>
<td>d78</td>
<td>1.4</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1.2</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**STOP!**
Probabilistic Pruning of Top-k Candidates [VLDB 04]

TA family of algorithms based on invariant (with sum as aggr):
\[ \sum_{i \in E(d)} s_i(d) \leq s(d) \leq \sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} \text{high}_i \]

- 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:
\[
p(d) := P\left[ \sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} S_i > \delta \right]
\]
discard candidates \( d \) from queue if \( p(d) \leq \varepsilon \) ⇒ \( E[\text{rel. precision}@k] = 1 - \varepsilon \)

Score predictor can use LSTs & Chernoff bounds, Poisson approximations, or histogram convolution

\( \text{drop } d \) from priority queue

\( \text{scan depth} \)
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“

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

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

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

Gerhard Weikum  May 19, 2006
Top-k Queries with Query Expansion [SIGIR 05]

Consider expandable query "~professor and research = XML" with score
\[ \sum_{i \in q} \{ \max_{j \in \text{exp}(i)} \{ \text{sim}(i,j) \ast s_j(d) \} \} \]

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

B+ tree index on tag-term pairs and terms

<table>
<thead>
<tr>
<th>Tag</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>0.6</td>
</tr>
<tr>
<td>professor</td>
<td>0.4</td>
</tr>
<tr>
<td>57</td>
<td>0.6</td>
</tr>
<tr>
<td>44</td>
<td>0.4</td>
</tr>
<tr>
<td>52</td>
<td>0.4</td>
</tr>
<tr>
<td>33</td>
<td>0.3</td>
</tr>
<tr>
<td>75</td>
<td>0.3</td>
</tr>
<tr>
<td>12</td>
<td>0.9</td>
</tr>
<tr>
<td>14</td>
<td>0.8</td>
</tr>
<tr>
<td>28</td>
<td>0.6</td>
</tr>
<tr>
<td>17</td>
<td>0.55</td>
</tr>
<tr>
<td>61</td>
<td>0.5</td>
</tr>
<tr>
<td>44</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer:</td>
<td>0.7</td>
</tr>
<tr>
<td>scholar:</td>
<td>0.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>professor</td>
<td>0.9</td>
</tr>
<tr>
<td>92</td>
<td>0.9</td>
</tr>
<tr>
<td>67</td>
<td>0.9</td>
</tr>
<tr>
<td>52</td>
<td>0.9</td>
</tr>
<tr>
<td>44</td>
<td>0.8</td>
</tr>
<tr>
<td>55</td>
<td>0.8</td>
</tr>
<tr>
<td>52</td>
<td>0.9</td>
</tr>
<tr>
<td>44</td>
<td>0.8</td>
</tr>
<tr>
<td>55</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>scholar:</td>
<td>0.7</td>
</tr>
<tr>
<td>professor</td>
<td>0.3</td>
</tr>
<tr>
<td>academ</td>
<td>0.53</td>
</tr>
<tr>
<td>scientist</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Thesaurus / meta-index

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 $\frac{C_{RA}}{C_{SA}} = r$

perform NRA (TA-sorted)
...
after every $r$ rounds of SA ($m \times r$ 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 $\text{bestscore}(d) > \min-k$): $4m + k$
For **SA scheduling** plan next $b_1, \ldots, b_m$ index scan steps for **batch of b steps** overall s.t. $\sum_{i=1..m} b_i = b$ 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 $d \in \text{top-k}$) 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]

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 Li (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

Top-k results (k=2):

min-k = 0.0
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“)] ]

<table>
<thead>
<tr>
<th></th>
<th>Join &amp;Sort</th>
<th>Struct Index</th>
<th>TopX (ε=0.0)</th>
<th>TopX (ε=0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#sorted accesses @10</td>
<td>9,122,318</td>
<td>761,970</td>
<td>635,507</td>
<td>426,986</td>
</tr>
<tr>
<td>#random accesses @10</td>
<td>0</td>
<td>3,245,068</td>
<td>64,807</td>
<td>59,414</td>
</tr>
<tr>
<td>CPU sec</td>
<td>12.01</td>
<td>17.02</td>
<td>1.38</td>
<td>1.27</td>
</tr>
<tr>
<td>relative recall @10</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>precision@10</td>
<td>0.34</td>
<td>0.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAP@1000</td>
<td>0.17</td>
<td>0.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
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*