Top-k Query Processing with Probabilistic Guarantees

Martin Theobald

mtb@mpi-sb.mpg.de

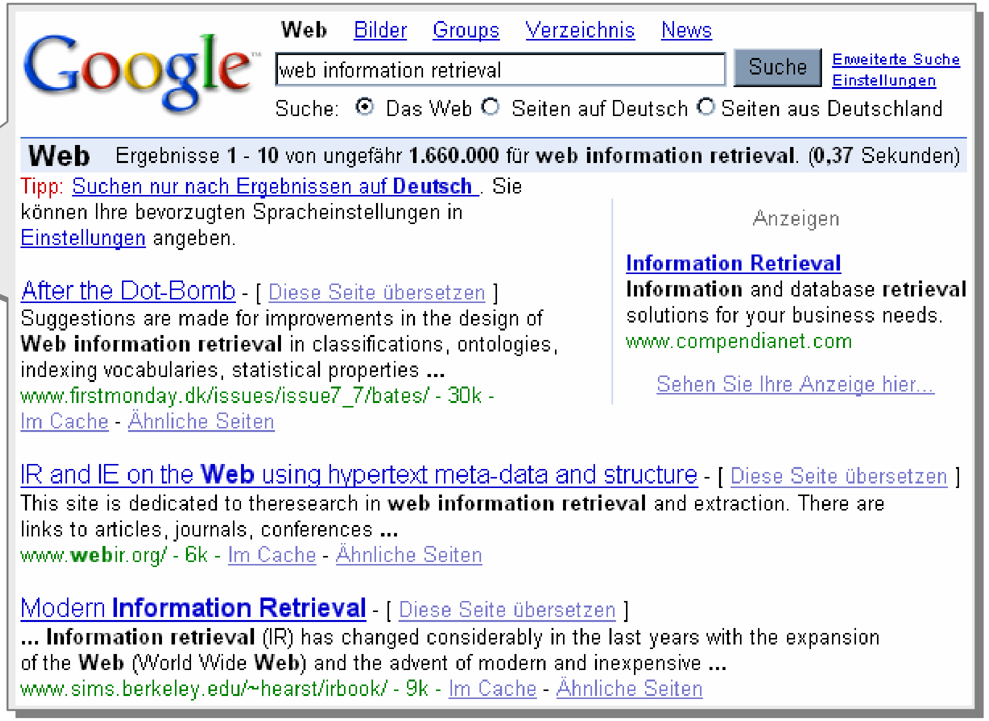
International Max Planck Research School for Computer Science, Saarbrücken

Motivation

- Focus of this work:

- Efficient top-k query evaluation in the style of current web search engines

- **Probabilistic score predictions for early candidate pruning**
- **Efficient queue management for candidate documents**



- *M. Theobald, G. Weikum, and R. Schenkel:*
  “Top-k Query Evaluation with Probabilistic Guarantees."
  VLDB Conference 2004, Toronto, Canada

Moore's Law and the Web

"At our rate of technological development and advances in the semiconductor industry, the **complexity of integrated circuits doubles every 18 months**."
[Moore ´65]

Hardware	**Annual Growth**
CPU Performance	**60%**
Disk Storage Space	**110%**
Random Access Time of Dynamic RAM's	**10%**
Disk Random IO's (Latency)	**8%**
Data	
# Web Pages	**~120%**
Multimedia Data on the Web	**~300%**

- **Google**'s index today contains 4.285.199.774 documents in >> 1 TB
  - *Less than 1s average response time*
  - *Server farm with 10,000+ units*
    → ***massive redundancy***

→ ***Efficient top-k retrieval using mainly sequential IO's to save hardware resources***

Fagin's NRA *[PODS '01]* at a Glance

Corpus: $d_1, \ldots, d_n$

d_1

$s(t_1,d_1) = 0.7$
…
$s(t_m,d_1) = 0.2$

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

```
NRA(q,L):
scan all lists Li (i = 1..m) in parallel & consider doc d at posi
    E(d) := E(d) ∪ {i};
    highi = s(ti,d);
    worstscore(d) := Sum{s(tν,d) | ν∈E(d)};
    bestscore(d):= worstscore(d) + Sum{highν | ν∉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
        candidates := candidates ∪ {d};
    if max {bestscore(d') | d'∈ candidates} ≤ min-k then exit;
```

Inverted Index

t_1	d78 0.9	d23 0.8	d10 0.8	d1 0.7	d88 0.2	…
t_2	d64 0.8	d23 0.6	d10 0.6	d10 0.2	d78 0.1	…
t_3	d10 0.7	d78 0.5	d64 0.4	d99 0.2	d34 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	[illegible]	[illegible]	1.8
4	[illegible]	1.2	2.0

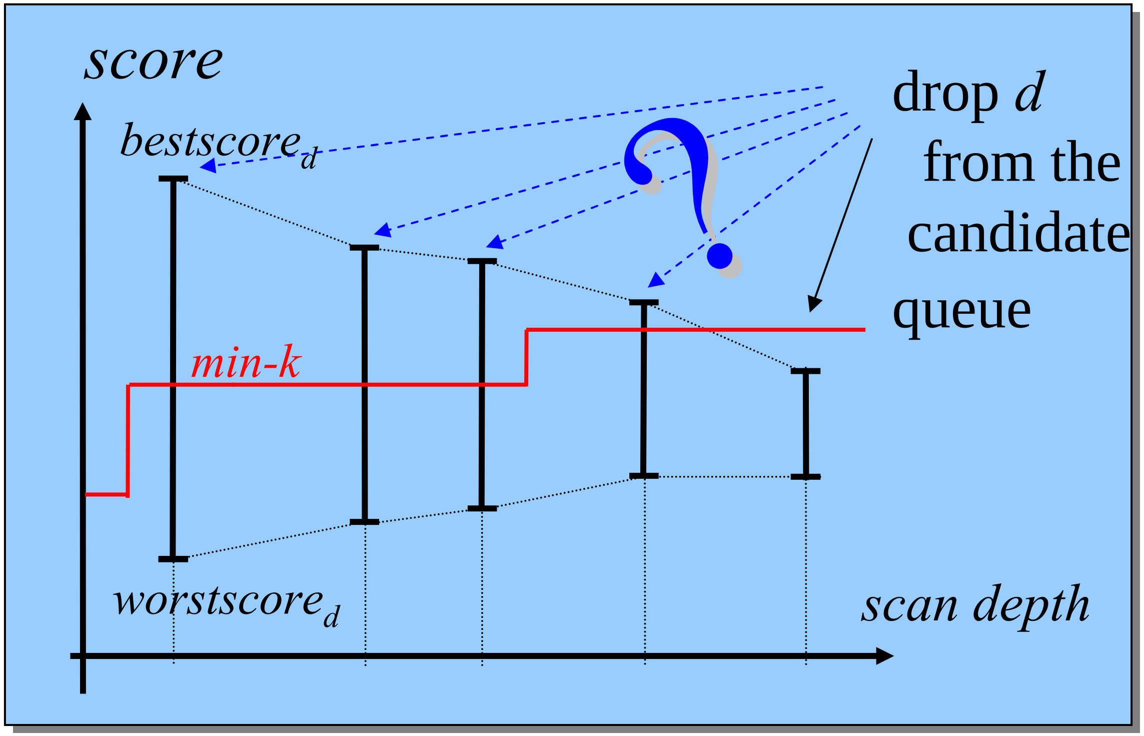
STOP!

How can we be even faster?

Evolution of a Candidate's Score

- Worst- and best-scores slowly converge to final score
- Add *d* to top-k result, if $worstscore_d > min\text{-}k$
- Drop *d* only if $bestscore_d < min\text{-}k$, otherwise keep it in candidate queue

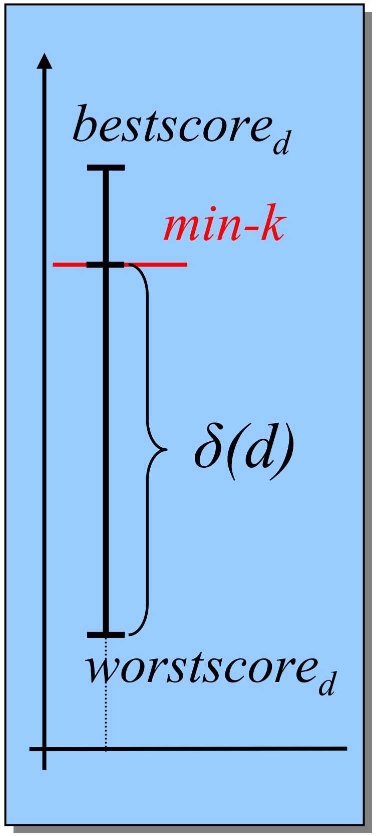
→ **Overly conservative threshold & long sequential index scans**



- **Approximate top-k**

"What is the **probability** that *d* qualifies for the top-k ?"

Safe Thresholding vs. Probabilistic Guarantees



- NRA based on invariant

$$\underbrace{\sum_{i \in E(d)} s_i(d)}_{worstscore_d} \le s(d) \le \underbrace{\sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} high_i}_{bestscore_d}$$

- Relaxed into ***probabilistic threshold test***

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

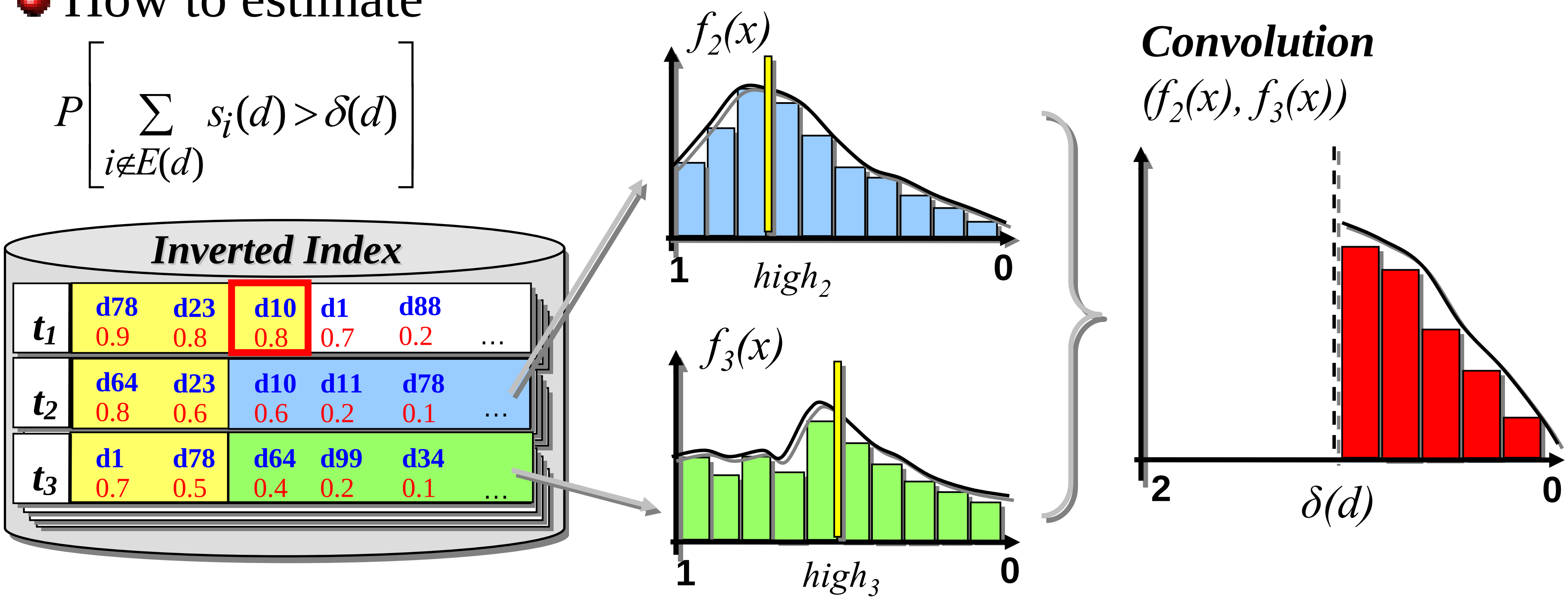
- Or equivalently, with $\delta(d) := \min_k - \sum \{ s_i \mid i \in E(d) \}$

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

Probabilistic Score Predictions

- How to estimate

$$P\left[\sum_{i \notin E(d)} s_i(d) > \delta(d)\right]$$



- Approximate each basic distribution
  - **Poisson estimator** or **pre-computed histograms**
  - Compute convolution over all $f_i(x)$ where $i \notin E(d)$
- Moment-Generating Functions & **Chernoff-Hoeffding bounds** *[Siegel `95]*

Is It Worth all the Effort?

- Non-negligible **prediction overhead**
  - 2^m-1 possible convolutions
  - Expensive computation
  - Frequent predictor updates

- **Queue management**

  as a key role for query evaluation
  - Which candidates are tested?
  - How often is a candidate tested?
  - What actions are taken when a candidate fails the test?

Conservative Queuing

- ***Prob-conservative***
  - ***2^m -1 queues*** per query
  - Group candidates by remainder sets $\{1..m\} - E(d)$
  - Top candidate dominates all candidates within each queue
  - Test top candidate only
  - For all queues q :

    Drop queue q, if

    P[***top(q)*** *can qualify for top-k]* ≤ ε

- ***Prob-progressive***
  - ***1 queue*** per query
  - Merge all candidates by their best-scores
  - No dominating candidate in terms of score prediction
  - Test all candidates periodically
  - For all candidates d in q :

    Drop candidate d, if

    P[**d** *can qualify for top-k]* ≤ ε

- Stop by safe min-k threshold test or when all queues are empty

Aggressive Queuing

- ***Prob-smart***
  - ***1 bounded queue*** per query
  - Merge all candidates by their best-scores
  - No dominating candidate in terms of score prediction
  - Update & rebuild entire queue periodically
- ***Prob-aggressive***
  - ***No queue***
  - Consider virtual candidate d_v with $E(d_v) = \emptyset$
  - d_v dominates all yet unseen candidates

- Stop heuristically, if

  P[**top(q)** *can qualify for top-k]* ≤ ε

- Stop heuristically, if

  P[$\mathbf{d_v}$ *can qualify for top-k]* ≤ ε

International Max Planck Research School
MPI for Computer Science

UNIVERSITAS SARAVIENSIS

Experimental Setup

- **Gov**
  - TREC-12 Web Track's .Gov collection
  - 1.250.000 web documents (html, doc, pdf)
  - 50 keyword queries from the Topic Distillation task, $m \leq 5$
  - *e.g. "legalization marijuana"*
- **XGov**
  - Gov with manual query expansion, $m \leq 20$
  - *e.g. "legalization law marijuana cannabis drug abuse pot …"*
- **IMDB**
  - Structured (XML) version of the Internet Movie Data Base
  - 375.000 movie files, 1.200.000 person files
  - Mixed text and categorical attributes: Genre, Actor, Description
  - *e.g. "Genre* $\supseteq$ *{Western}* $\wedge$ *Actor* $\supseteq$ *{John Wayne, Katherine Hepburn}* $\wedge$ *Description* $\supseteq$ *{Sheriff, Marshall}"*

Baseline

k=20 & ε = 0.1

Dataset	Method	# sorted accesses	execution time (sec.)	max. queue size	macro-avg. precision
Gov	*original NRA*	2,263,652	148.7	10,849	1.0
	Prob-con	993,414	25.6	29,207	0.87
	Prob-pro	1,659,706	44.2	6,551	0.87
	Prob-smart	527,980	15.9	400	0.69
	Prob-agg	20,435	0.6	0	0.42
XGov	*original NRA*	22,403,490	7,908	70,896	1.0
	Prob-con	10,165,677	6,448	51,893	0.90
	Prob-pro	20,006,283	1,791	12,435	0.95
	Prob-smart	18,287,636	1,066	400	0.88
	Prob-agg	133,745	2	0	0.35
IMDB	*original NRA*	1,003,650	201.9	12,628	1.0
	Prob-con	463,562	17.8	14,990	0.71
	Prob-pro	490,041	69.0	9,173	0.75
	Prob-smart	403,981	12.7	400	0.54
	Prob-agg	41,821	0.7	0	0.18

Conclusions & Ongoing Work

- **Performance vs. Quality**
  - **Speedup** factor of up to **10** at **~ 80%** prec. *(Prob-smart)*
  - **Speedup** factor of more than **100** at **~ 40%** prec. *(Prob-agg)*
- **Semantic extensions**
  - **Query-specific weights**
  - **Efficient query expansions**
- **TREC (Text REetrieval Conference) - benchmark competition**
  - **Robust Track** – hard queries
  - **Web Track** – $tf \cdot idf$ and global *PageRank* scores
  - **Terabyte Track** – 480 GB web documents
- **XML retrieval (XPath)**
  - **Structural query conditions, path similarities & no predefined retrieval unit**