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]

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Annual Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Performance</td>
<td>60%</td>
</tr>
<tr>
<td>Disk Storage Space</td>
<td>110%</td>
</tr>
<tr>
<td>Random Access Time of Dynamic RAM’s</td>
<td>10%</td>
</tr>
<tr>
<td>Disk Random IO’s (Latency)</td>
<td>8%</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td></td>
</tr>
<tr>
<td># Web Pages</td>
<td>~120%</td>
</tr>
<tr>
<td>Multimedia Data on the Web</td>
<td>~300%</td>
</tr>
</tbody>
</table>

- **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: \(d_1, \ldots, d_n\)

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

Inverted Index

\[
\begin{array}{c|c|c|c|c|c|c}
\hline
\text{t1} & d78 & d23 & d10 & d1 & d88 & \ldots \\
\hline
0.9 & 0.8 & 0.8 & 0.7 & 0.2 & \ldots \\
\hline
\text{t2} & d64 & d23 & d10 & d10 & d78 & \ldots \\
\hline
0.8 & 0.6 & 0.6 & 0.2 & 0.1 & \ldots \\
\hline
\text{t3} & d10 & d78 & d64 & d99 & d34 & \ldots \\
\hline
0.7 & 0.5 & 0.4 & 0.2 & 0.1 & \ldots \\
\hline
\end{array}
\]

1. NRA\((q,L)\):
2. scan all lists \(L_i (i = 1..m)\) in parallel & consider doc \(d\) at pos \(i\)
3. \(E(d) := E(d) \cup \{i\}\);
4. \(\text{high}_i = s(t_i,d)\);
5. \(\text{worstscore}(d) := \sum \{s(t_v,d) | v \in E(d)\}\);
6. \(\text{bestscore}(d) := \text{worstscore}(d) + \sum \{\text{high}_v | v \notin E(d)\}\);
7. if \(\text{worstscore}(d) > \text{min-k}\) then
8. add \(d\) to \(\text{top-k}\)
9. \(\text{min-k} := \min \{\text{worstscore}(d') | d' \in \text{top-k}\}\);
10. else if \(\text{bestscore}(d) > \text{min-k}\) then
11. \(\text{candidates} := \text{candidates} \cup \{d\}\);
12. if \(\max \{\text{bestscore}(d') | d' \in \text{candidates}\} \leq \text{min-k}\) then exit;

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 \( \text{worstscore}_d > \text{min-}k \)
- Drop \( d \) only if \( \text{bestscore}_d < \text{min-}k \), 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
  \[
  \sum_{i \in E(d)} s_i(d) \leq s(d) \leq \sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} \text{high}_i
  \]

- Relaxed into *probabilistic threshold test*
  \[
  p(d) := P \left[ \sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} s_i(d) > \min_k \right] \leq \varepsilon
  \]

- Or equivalently, with \( \delta(d) := \min_k - \sum \{s_i \mid i \in E(d)\} \)
  \[
  p(d) := P \left[ \sum_{i \notin E(d)} s_i(d) > \delta(d) \right] \leq \varepsilon
  \]
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
  $$P[\ \text{top}(q) \ \text{can qualify for top-k}] \leq \varepsilon$$

**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
  $$P[\ d \ \text{can qualify for top-k}] \leq \varepsilon$$

- 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 \( P[\text{top}(q) \text{ can qualify for top-k}] \leq \epsilon \)

Stop heuristically, if \( P[\text{d}_v \text{ can qualify for top-k}] \leq \epsilon \)
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 \{\text{Western}\} \land \text{Actor} \supseteq \{\text{John Wayne, Katherine Hepburn}\} \land \text{Description} \supseteq \{\text{Sheriff, Marshall}\}””
### Baseline

**k=20 & ε = 0.1**

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