Top-k Query Evaluation with Probabilistic Guarantees

Martin Theobald
Gerhard Weikum
Ralf Schenkel

Max-Planck Institute for Computer Science
Saarbrücken
Germany
Outline

- Computational Model
- Fagin’s NRA as baseline
- Probabilistic threshold test & score predictions
- Queuing strategies for prob. candidate pruning
- Probabilistic guarantees for top-k results
- Experiments
Related Work

- Thresholding algorithms for top-k queries
  - Fagin [99, 01, 03]: TA, NRA, CA
  - Balke, Güntzer & Kießling [00, 01]

- Variants for multimedia search & structured databases
  - Natsev et al. [01], Agrawal & Chaudhuri [03], Chaudhuri & Gravano [04]

- R-tree-like multidimensional indexes & range queries
  - Hjaltason & Samet [99,03], Böhm et al. [01], Bruno & Gravano [02], Ciaccia & Patella [02], Amato et al. [03]

- Multidimensional nearest-neighbor queries
  - Donjerkovic & Ramakrishnan [99], Ciaccia & Patella [00], Tao & Faloutsos & Papadias [03], Singitham et al. [04]
Computational Model for Top-k Queries over an m-Dimensional Data Space

- Cartesian product space $D_1 \times \ldots \times D_m$ and a data set $D \subseteq D_1 \times \ldots \times D_m$
  - of $m$-dimensional data points with $s_i \in D \subseteq \mathbb{R}^m$ or $s_i \in D \subseteq [0,1]^m$
- Monotonous score aggregation function
  - $\text{aggr}: (D_1 \times \ldots \times D_m) \times (D_1 \times \ldots \times D_m) \rightarrow [0,1]$ or $\mathbb{R}_0^+$
- Partial match queries
  - Weak dimensions can be compensated by stronger ones
- Possible score aggregation functions
  - $\text{min, max, sum, product}$ (sum over $\log s_i$), $\text{cosine}$ (normalized vectors)
- Application examples
  - Multimedia data, text documents, web documents, structured records (e.g., product catalogs)
- Access model
  - Inverted index over long index lists
  - $\Rightarrow$ expensive random IO’s & mainly sorted accesses
Fagin’s NRA \textsuperscript{PODS ’01} at a Glance

- \textit{NRA}(q,L):
  \begin{align*}
  \text{top-k} & := \emptyset; \text{candidates} := \emptyset; \text{min-k} := 0; \\
  \text{scan all lists } L_i \ (i = 1..m) \text{ in parallel:} & \\
  & \text{consider item } d \text{ at position } \text{pos}_i \text{ in } L_i; \\
  & \text{E}(d) := \text{E}(d) \cup \{i\}; \\
  & \text{high}_i := s_i(q_i,d); \\
  \text{worstscore}(d) & := \text{aggr}\{s_{\nu}(q_{\nu},d) | \nu \in \text{E}(d)\}; \\
  \text{bestscore}(d) & := \text{aggr}\{\text{aggr}\{s_{\nu}(q_{\nu},d) | \nu \in \text{E}(d)\}, \text{aggr}\{\text{high}_{\nu} | \nu \notin \text{E}(d)\}\}; \\
  \text{if worstscore}(d) & > \text{min-k} \text{ then} \\
  & \text{remove argmin}_d\{\text{worstscore}(d') | d' \in \text{top-k}\} \text{ from top-k;}
  \text{add } d \text{ to top-k} \\
  & \text{min-k} := \min\{\text{worstscore}(d') | d' \in \text{top-k}\}; \\
  \text{else if bestscore}(d) & > \text{min-k} \text{ then} \\
  & \text{candidates} := \text{candidates} \cup \{d\}; \\
  \text{threshold} & := \max \{\text{bestscore}(d') | d' \in \text{candidates}\}; \\
  \text{if threshold} & \leq \text{min-k} \text{ then exit;}
\end{align*}
Evolution of a Candidate’s Score

- **Approximate top-k**
  
  “What is the probability that $d$ qualifies for the top-k?”
Safe Thresholding vs. Probabilistic Guarantees

- NRA based on invariant
  \[ \sum_{i \in E(d)} s_i(d) \leq s(d) \leq \sum_{i \in E(d)} s_i(d) + \sum_{i \notin E(d)} high_i \]

- 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

- **Assuming feature independence**
  - Uniform
  - Zipf
  - Poisson
  - Histograms

- **With feature correlations**
  - Multi-dimensional histograms
  - Moment-Generating Functions & Chernoff-Hoeffding bounds \([Siegel \ '95]\)
Guarantees with Uniform Distributions

- Consider score intervals \([0, \text{high}_i]\)
- Treat each \(s_i\) as random variable \(S_i\) and predict \(P\left[\sum_i S_i > \delta\right]\)
- For two random variables \(S_1\) and \(S_2\)
  - Density functions \(f_1(x) = 1/\text{high}_1\) and \(f_2(x) = 1/\text{high}_2\)
  - Consider convolution \(f(x) = \int_0^x f_1(z) f_2(x-z) \, dz\)
  - ..but each factor is non-zero in \(0 \leq z \leq \text{high}_1\) and \(0 \leq x-z \leq \text{high}_2\)
    \(\rightarrow\) awkward amount of case differentiations
- Instead, consider moment-generation functions \((i > 1)\)
  - Of the form \(M_i(s) = \int_0^S e^{sx} f_i(x) \, dx = E\left[e^{sS_i}\right]\)
  - Consider convolution \(M(s) = \prod_i M_i(s)\)
- Apply Chernoff-Hoeffding bounds on the tail probabilities
  \(P\left[\sum_i S_i > \delta\right] \leq \inf_{s \geq 0} \{e^{-s\delta} M(s)\}\)
- Ability to capture feature correlations or heterogeneous distributions
Guarantees with Poisson Estimators

- Approximate tf·idf scores by a Poisson distribution truncated over \([0, \text{high}_i]\)
- Reasonably fits large web corpora, e.g., “.Gov”
- Discretize all \(s_i\) in index list \(L_i\) with \(n_i\) items:
  - Let \(S_i\) be a discrete RV with \(n_i\) equi-distant values \(v_j = 1 - j \cdot \text{high}_i / n_i\)
  - then \(P[S_i = v_k] = e^{-\lambda_i} \frac{\lambda_i^k}{k!}\) and \(\sum_{j=1}^{..l} P[S_i = v_j]\) approximated by Incomplete Gamma Fct.
- Consider convolution \((i > 1)\)
  - where \(P[S = v_{k'}] = e^{-\lambda_r} \frac{\lambda_r^k}{k!}\) with \(\lambda_r = \sum_i \lambda_i\)
- Consider conditional probabilities
  \[
  P\left[ \sum_{i \notin E(d)} S_i > v \mid S_i \leq \text{high}_i \text{ for } i \notin E(d) \right] \\
  = 1 - P\left[ \sum_{i \notin E(d)} S_i \leq v \mid S_i \leq \text{high}_i \text{ for } i \notin E(d) \right] \\
  = 1 - \frac{P\left[ \sum_{i \notin E(d)} S_i \leq v \land (S_i \leq \text{high}_i \text{ for } i \notin E(d)) \right]}{\prod_{i \notin E(d)} P[S_i \leq \text{high}_i]}\
  \]
Guarantees with Histograms

- Capture arbitrary distributions in a compact histogram
  - Consider \( n \) buckets \((lb, ub]\) with \( lb[j] = j/n \) and \( ub[j] = (j+1)/n \)
  - Store the frequency \( freq[i] \) and the cumulative frequency \( cum[i] \) of scores for each cell

- Consider convolution

\[
H \cdot freq[j] = \sum_{(j_1, \ldots, j_r) \text{ with } j_1 + \ldots + j_r = i} H_1 \cdot freq[j_1] \cdot \ldots \cdot H_r \cdot freq[j_r]
\]

\[
H \cdot cum[j] = \sum_{l=0..j} H \cdot freq[l]
\]

\( \rightarrow \) in \( O(mn^2) \) time using a binary convolution operation

- Consider conditional probabilities

\[
P\left[ \sum_{i \notin E(d)} S_i > \delta \mid S_i \leq \text{high}_i \text{ for } i \notin E(d) \right]
\]

\( \rightarrow \) by truncating the basic histograms
Queuing Strategies for Probabilistic Candidate Pruning

- Non-negligible prediction overhead
  - $2^{m-1}$ possible convolutions of remainder dimensions per query
  - Frequent predictor updates due to decreasing $high_i$

- Prob. threshold test *embedded* into NRA
  - Periodic candidate pruning (after $r$ sorted accesses)
  - “Reusable” convolutions are temporarily cached

- Queuing 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 $P[\text{top}(q) \text{ can qualify for top-k}] < \varepsilon$

- **Prob-progressive**
  - 1 queue per query
  - Merge all candidates by their bestscores
  - No dominating candidate in terms of prob. prediction
  - Test all candidates periodically
  - For all candidates $d$ in $q$:
    - Drop candidate $d$, if $P[\text{d can qualify for top-k}] < \varepsilon$

- 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 bestscores
  - No dominating candidate in terms of prob. prediction
  - Update & rebuild entire queue
  - Bound size by $b$ top candidates after each rebuild

- **Prob-aggressive**
  - No queue
  - Consider virtual candidate $d_v$ with $E(d_v) = \emptyset$
  - $d_v$ dominates all yet unseen candidates

- Stop heuristically, if
  - $P[\text{top}(q) \text{ can qualify for top-k}] < \varepsilon$

- Stop heuristically, if
  - $P[d_v \text{ can qualify for top-k}] < \varepsilon$
Guarantees for Top-k Results

- For \textit{Prob-con} and \textit{Prob-pro}
  - Probability $p_{\text{miss}}$ of missing a true top-k object equals the probability of erroneously dropping a candidate from the queue
  - For each candidate $p_{\text{miss}} \leq \varepsilon$
  - $P[\text{recall} = r/k] = P[\text{precision} = r/k] = \binom{k}{r} (1 - p_{\text{miss}})^r p_{\text{miss}}^{(k-r)} \leq \binom{k}{r} (1 - \varepsilon)^r \varepsilon^{(k-r)}$
  - $E[\text{precision}] = E[\text{recall}] = \sum_{r=0..k} P[\text{precision} = r/k] \cdot r/k = (1 - \varepsilon)$
  - Lower expected values for \textit{Prob-smart} and \textit{Pro-aggr}
Experimental Setup

Data collections

- **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
  - 20 queries of the type:
    - “Genre \( \supseteq \{ \text{Western} \} \land Actor \supseteq \{ \text{John Wayne, Katherine Hepburn} \} \land Description \supseteq \{ \text{Sheriff, Marshall} \}””
Evaluation Metrics

- **Efficiency**
  - \#sorted accesses
  - \textit{time}: wall-clock elapsed time
  - \textit{memory}: peak level of working memory for all priority queues

- **Quality**
  - \textit{Precision & recall} (at macro-average)
  - \textit{Rank distance} := \( \frac{1}{k} \sum_{i=1..k} |i - \text{truerank}(i)| \)
  - \textit{Score error} := \( \frac{1}{k} \sum_{i=1..k} |\text{score}_i^{(\text{approx})} - \text{score}_i^{(\text{exact})}| \)
### 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>
<th>rank distance</th>
<th>score distance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gov</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRA</td>
<td>2,263,652</td>
<td>148.7</td>
<td>10,849</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Prob-con</td>
<td>993,414</td>
<td>25.6</td>
<td>29,207</td>
<td>0.87</td>
<td>16.9</td>
<td>0.007</td>
</tr>
<tr>
<td>Prob-pro</td>
<td>1,659,706</td>
<td>44.2</td>
<td>6,551</td>
<td>0.87</td>
<td>16.8</td>
<td>0.006</td>
</tr>
<tr>
<td>Prob-smart</td>
<td>527,980</td>
<td>15.9</td>
<td>400</td>
<td>0.69</td>
<td>39.5</td>
<td>0.031</td>
</tr>
<tr>
<td>Prob-agg</td>
<td>20,435</td>
<td>0.6</td>
<td>0</td>
<td>0.42</td>
<td>75.1</td>
<td>0.089</td>
</tr>
<tr>
<td><strong>XGov</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRA</td>
<td>22,403,490</td>
<td>7,908</td>
<td>70,896</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Prob-con</td>
<td>10,165,677</td>
<td>6,448</td>
<td>51,893</td>
<td>0.90</td>
<td>10.9</td>
<td>0.038</td>
</tr>
<tr>
<td>Prob-pro</td>
<td>20,006,283</td>
<td>1,791</td>
<td>12,435</td>
<td>0.95</td>
<td>9.3</td>
<td>0.031</td>
</tr>
<tr>
<td>Prob-smart</td>
<td>18,287,636</td>
<td>1,066</td>
<td>400</td>
<td>0.88</td>
<td>14.5</td>
<td>0.035</td>
</tr>
<tr>
<td>Prob-agg</td>
<td>133,745</td>
<td>2</td>
<td>0</td>
<td>0.35</td>
<td>80.7</td>
<td>0.182</td>
</tr>
<tr>
<td><strong>IMDB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRA</td>
<td>1,003,650</td>
<td>201.9</td>
<td>12,628</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Prob-con</td>
<td>463,562</td>
<td>17.8</td>
<td>14,990</td>
<td>0.71</td>
<td>119.9</td>
<td>0.18</td>
</tr>
<tr>
<td>Prob-pro</td>
<td>490,041</td>
<td>69.0</td>
<td>9,173</td>
<td>0.75</td>
<td>122.5</td>
<td>0.14</td>
</tr>
<tr>
<td>Prob-smart</td>
<td>403,981</td>
<td>12.7</td>
<td>400</td>
<td>0.54</td>
<td>126.7</td>
<td>0.25</td>
</tr>
<tr>
<td>Prob-agg</td>
<td>41,821</td>
<td>0.7</td>
<td>0</td>
<td>0.18</td>
<td>171.5</td>
<td>0.39</td>
</tr>
</tbody>
</table>
Sensitivity Studies for Gov:

precision vs. sorted accesses for $k = 20$

macro-avg. precision

- Prob-con
- Prob-pro
- Prob-smart
- Prob-agg

#sorted accesses

- 2,500,000
- 2,000,000
- 1,500,000
- 1,000,000
- 500,000
- 0
Impact of Probabilistic Predictions for Gov

Original scores
- \(tf \cdot idf\) weights

Artificially generated score distributions
- Uniform
- Zipf
Conclusions & Ongoing Work

- “Top-k is a heuristic anyway, so why not do it probabilistically?”
- Major performance gains of factor of up to 8 at ~80% precision
- **Semantic extensions** for query-specific weights and efficient query expansion
- Applied at this year’s TREC benchmark
  - Robust Track – hard queries
  - Web Track – $tf \cdot idf$ and global *PageRank* scores
  - Terabyte Track – 480 GB web documents
- Further extensions for **XML support** incl. path similarities