Suggestion of Promising Result Types for XML Keyword Search

Wei Wang
University of New South Wales, Australia

Joint work with Jianxin Li, Chengfei Liu and Rui Zhou (Swinburne University of Technology, Australia)
Outline

- Motivation
- Scoring Result Types
- Query Processing Algorithms
- Experimental Study
- Conclusions
Motivation

- Keyword query is easy to use for casual users
  
  **If it has a syntax, it isn’t user friendly -- /usr/game/fortune**

- No need to know a query language or schema of the data

- Keyword query is inherently *imprecise*. How to find relevant results?
  
  - Browse all relevant results – Impossible or Unusable!
  
  - Restrict the results
    1. XSEarch, SLCA/ELCA, and their variants
    2. Return result instances from the most likely *query result type* (XReal and our work)
Query Result Types

- A label path such that at least one of its corresponding instances contains all the search keywords

- Intuition: users want to fetch instances of certain entity type with keyword predicates
Ranking

- Score each result type, and select the most promising return type (i.e., the one with the highest score).
- Subtleties: 1 return type $\Rightarrow$ n query templates $\Rightarrow$ n*m result instances
Scoring Individual Results /1

- \( \text{score(result type)} = \text{aggregate( score(instance1), score(instance2), ...)} \)

- Need to score individual result instance \( R \)
  1. Not all matches are equal in terms of content
     - Inverse element frequency (\( \text{ief(x)} \)) = \( N / \# \) nodes containing the token \( x \)
     - E.g., \( \text{Weight(n_i contains a)} = \log(\text{ief(a)}) \)
Scoring Individual Results /2

- $\text{score(result type)} = \text{aggregate( score(instance1), score(instance2), ...)}$

- Need to score individual result instance $R$
  2. Not all matches are equal in terms of structure
     - distance between the match and the root of the subtree
     - also considers avg-depth of the XML tree to attenuate the impact of long paths
Scoring Individual Results /3

• \[ \text{score(result type)} = \text{aggregate} (\text{score(instance1)}, \text{score(instance2)}, ...) \]

• Need to score individual result instance \( R \)
  3. Favor tightly-coupled results
    • When calculating \( \text{dist()} \), discount the shared path segments

Loosely coupled

Tightly coupled
Scoring Individual Results

- score(result type) = aggregate(score(instance1), score(instance2), ...)

- Need to score individual result instance R
  - The final formula

\[
Score(R, Q) = \begin{cases} 
\sum_{i=1}^{n} weight(k_i) & \text{, if } dist'(N, n_i) = 0; \\
\sum_{i=1}^{n} weight(k_i) / \left( \sum_{i=1}^{n} dist'(N, n_i) - \mu_1 \right)^{\mu_2} & \text{, Otherwise.}
\end{cases}
\]
Scoring Return Types

- \( \text{score(result type)} = \text{aggregate}( \text{score(instance1)}, \text{score(instance2)}, \ldots) \)

- **aggregate**: sum up the top-k instance scores
  - \( k = \text{average instance numbers of all query result types} \)
  - Pad 0.0 if necessary
Query Processing Algorithms

- INL (Inverted Node List-based Algorithm)
  - Merge all relevant nodes and group the merged results by different result types
    - Using inverted index + Dewey encoding
  - Calculate the score for each result type by using ranking function
    - Only needs to keep the top-k best scores for each result type
- Slow because of no pruning or skipping
SDI Algorithm

• How to be more efficient?
  • Approximately compute the score of each return type
  • Prune some of the less likely return types
• SDI (Statistic Distribution Information-based Algorithm)
  • Based on several additional indexes:
    • Keyword-path index,
    • Enhanced F&B index (with distributional info).
  • Generate query templates by merging distinct paths
  • Estimate the scores of each query templates
  • Aggregate the scores for each result type
Keyword-Path Index & Query Templates

- Maps each keyword to the set of label paths that characterizes all its occurrences

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Label Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>{ root/students/student/interest, root/books/book/title }</td>
</tr>
</tbody>
</table>

- Merge the label paths to obtain query templates
  - root/students/student[born ~ 1980][interest ~ art]
  - root/books/book/title
  - root/books/book[year ~ 1980][title ~ art]
- Iteratively ascend to its parent label path if a query template has no estimated result
- More refined XSketch synopsis
- Estimate size of certain simple queries, e.g.,
  - size(root/books/book[title])
  - size(root/books/book[name])
- Hardly handles correlation
  - size(root/books/book[title][name])
Structural Distribution

size(root/books/book[title][name]) = 0
Value Distribution

![Diagram of a tree structure with nodes labeled as "interest", "street art", "1980", "visual art", "after 1980", "advertising", "decorative street art", "dramatic art & interest".]

### Value Distribution for Year

<table>
<thead>
<tr>
<th>$E_{1980}$</th>
<th>$CS_{year}$</th>
<th>$f_{1980}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3/5</td>
</tr>
</tbody>
</table>

### Value Distribution for Title

<table>
<thead>
<tr>
<th>$E_{art}$</th>
<th>$CS_{title}$</th>
<th>$f_{art}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4/5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E_{dramatic}$</th>
<th>$E_{art}$</th>
<th>$CS_{title}$</th>
<th>$f_{dramatic';art}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1/5</td>
</tr>
</tbody>
</table>

Estimation

- /root/books/book ➔ 6
- /root/books/book/year ➔ 1
- /root/books/book/title ➔ 1
- \( h_{\text{book}[\text{year}][\text{title}]} = \frac{4}{6} \)
- \( f_{1980|\text{year}} = \frac{3}{5} \)
- \( f_{\text{art}|\text{title}} = \frac{4}{5} \)
- Final estimation = 1.92
Recap

- SDI
  - Retrieve the relevant label paths by the keyword-path index
  - Generate query templates by merging distinct paths
  - Estimate the scores of each query templates
  - Aggregate the scores for each result type
Experiment Setup /1

- Three real datasets used:
  - NASA: astronomical data
  - UWM: course data derived from university websites.
  - DBLP: computer science journals and proceedings
Experiment Setup /2

- 18 Keyword queries:

<table>
<thead>
<tr>
<th>Queries</th>
<th>NASA</th>
<th>UWM</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1i</td>
<td>{magnitude}</td>
<td>{level}</td>
<td>{evaluation}</td>
</tr>
<tr>
<td>Q2i</td>
<td>{photographic}</td>
<td>{archeology}</td>
<td>{object oriented}</td>
</tr>
<tr>
<td>Q3i</td>
<td>{photographic magnitude}</td>
<td>{Najoom}</td>
<td>{Frank Manola}</td>
</tr>
<tr>
<td>Q4i</td>
<td>{rotation dipersion}</td>
<td>{individual supervision}</td>
<td>{concepts applications}</td>
</tr>
<tr>
<td>Q5i</td>
<td>{cape photographic}</td>
<td>{building technologies}</td>
<td>{multimedia data type}</td>
</tr>
<tr>
<td>Q6i</td>
<td>{Optically proper motion}</td>
<td>{approved performance organization}</td>
<td>{Frank database 1983}</td>
</tr>
</tbody>
</table>
**NASA:** XReal only focuses on one node at the higher level while INL and SDI can reach to the more detailed nodes.

**UWM:** For Q12 and Q32, INL and SDI can predict more meaningful results than XReal does. For Q42, XReal can do better.

**DBLP:** All methods produce the same results because the structure of DBLP is so flat.

### Table 4: Promising Result Types for Each Query

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Queries</th>
<th>INL</th>
<th>XReal</th>
<th>SDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>$Q_{11}$</td>
<td>{para, title, definition}</td>
<td>{tableHead}</td>
<td>{para, title, definition}</td>
</tr>
<tr>
<td></td>
<td>$Q_{21}$</td>
<td>{para, title, definition}</td>
<td>{tableHead}</td>
<td>{title, para, definition}</td>
</tr>
<tr>
<td></td>
<td>$Q_{31}$</td>
<td>{fields}</td>
<td>{tableHead}</td>
<td>{para, descriptions}</td>
</tr>
<tr>
<td></td>
<td>$Q_{41}$</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td></td>
<td>$Q_{51}$</td>
<td>{fields}</td>
<td>{tableHead}</td>
<td>{fields, source}</td>
</tr>
<tr>
<td></td>
<td>$Q_{61}$</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>UWM</td>
<td>$Q_{12}$</td>
<td>{restrictions, title, comments}</td>
<td>{level}</td>
<td>{restrictions, title, comments}</td>
</tr>
<tr>
<td></td>
<td>$Q_{22}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{32}$</td>
<td>{instructor}</td>
<td>{section_listing}</td>
<td>{instructor}</td>
</tr>
<tr>
<td></td>
<td>$Q_{42}$</td>
<td>{root}</td>
<td>{title}</td>
<td>{root}</td>
</tr>
<tr>
<td></td>
<td>$Q_{52}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{62}$</td>
<td>{restrictions}</td>
<td>{restrictions}</td>
<td>{restrictions}</td>
</tr>
<tr>
<td>DBLP</td>
<td>$Q_{13}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{23}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{33}$</td>
<td>{author}</td>
<td>{author}</td>
<td>{author}</td>
</tr>
<tr>
<td></td>
<td>$Q_{43}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{53}$</td>
<td>{title}</td>
<td>{title}</td>
<td>{title}</td>
</tr>
<tr>
<td></td>
<td>$Q_{63}$</td>
<td>{proceedings}</td>
<td>{proceedings}</td>
<td>{proceedings}</td>
</tr>
</tbody>
</table>
Efficiency

- SDI’s speedup against XReal: $3x \sim 10x$
- Speedup is even more significant on other two datasets
Conclusions

- Alleviates the inherent imprecision of keyword queries by scoring their result types
  - Can only return instances from the most promising one
  - Or take such score into consideration in the final ranking function
- Efficient estimation-based method to find most promising return types
- Experimental results demonstrates both the effectiveness and efficiency of the proposed approach
Q & A

Our Keyword Search Project Homepage:
http://www.cse.unsw.edu.au/~weiw/project/SPARK.html
Related Work

- [Liu & Chen, SIGMOD07]
  - Classifies XML nodes into one of three node types
  - However, it only identifies a specific return node type for each result.

- XReal [Bao et al, ICDE09]
  - Summarizing the statistic information between element nodes and all tokens in the leaf nodes
  - IR style method is used to infer the result type based on the statistics.
  - However, it does not model the correlation among the XML elements and values.