

Ranking Query Results for XML IR

Debora Donato

Yahoo! Research – Barcelona, Spain

Seminars of Computer Networks

- 1 Introduction
- 2 XML ranking
- 3 Query Processing

- 1 Introduction
- 2 XML ranking
- 3 Query Processing

In a nutshell

The extensible markup language (XML) is essentially a textual representation of the hierarchical (tree-like) data where a meaningful piece of data is bounded by matching starting and ending tags such as

```
<text>This is an XML document</text>
```

The nested structure of tags capture additional semantics of the information

Advantages of the XML tagged structure

- XML structure exploited for better searching (compared to text or html search)
- Context information used for semantic search.

XML Tree Representation

XML document modeled as a tree

$$G = (V_E, V_T, E, root, oid, lab, ord)$$

- V_E : set of element nodes;
- V_T : set of term nodes (leaf nodes);
- E : set of edges (spanning tree over $V_E \cup V_T$);
- $root \in V_E$;
- lab is function that assigns a label to each node:
 - tag name for $node \in V_E$;
 - keyword for $node \in V_T$
- oid is a function that assigns unique id to each node
- ord is a function that assigns a unique ordinal number (sibling position)

Indexing XML Data

Given the previous XML tree representation

- XML Inverted list
- Dewey Inverted Lists
- Structure Indexes
- Vector-Space Model

XML Query (1)

- Query issued to the system.

XML Query (1)

- Query issued to the system.
- Query processed over the index structure.

XML Query (1)

- Query issued to the system.
- Query processed over the index structure.
- List of results returned to the user

XML Query (1)

- Query issued to the system.
- Query processed over the index structure.
- List of results returned to the user

XML Query (1)

- Query issued to the system.
- Query processed over the index structure.
- List of results returned to the user

How relevant are these results to the user query?

XML Query (2)

- Users are not interested in all the results.

XML Query (2)

- Users are not interested in all the results.
- top k ($k < 10$)

XML Query (2)

- Users are not interested in all the results.
- top k ($k < 10$)
- Need for ranking functions.

- 1 Introduction
- 2 XML ranking
- 3 Query Processing

Link Based Ranking

- Based on calculating the objective importance of documents determined by the hyperlinked structure of documents.
- **ElemRank**[Guo et al., 2003] is used for ranking XML documents
- Derived by making appropriate modifications to PageRank, similar function used for ranking HTML documents. Hierarchy of elements (XML element tree)

Web Information Retrieval

- Link analysis (structure analysis):
 - weighting documents
 - improve result ranking

Web Information Retrieval

- Link analysis (structure analysis):
 - weighting documents
 - improve result ranking

Page rank approach

- web as directed graph G
- “random walk” of a web surfer
- follow hyperlinks with probability $(1 - \epsilon)$
- “random jump” with probability ϵ

XRANK - Keyword Search over XML documents

- results:
 - XML elements that contain all searched keywords
- ranking:
 - at granularity of XML elements
 - based on hyperlink structure
- advantages:
 - user does not have to learn a query language
 - no knowledge about the structure of XML documents is needed
- generalized keyword search engine
 - both HTML and XML are possible

Data Model

$G = (V, CE, HE)$: collection of XML documents

V : set of XML elements (tags and attributes)

CE : set of containment edges

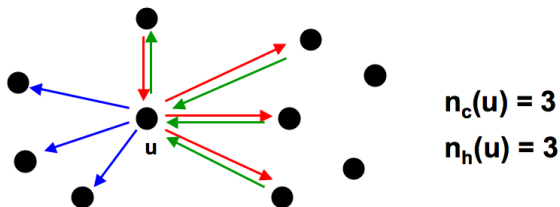
HE : set of hyperlinked edges

$(u, v) \in CE \Leftrightarrow v$ is a sub-element of u

$(u, v) \in HE \Leftrightarrow u$ contains a hyperlink to v

$contains(v, k) \Leftrightarrow v$ (in)directly contains the keyword k

ElemRank Computation(1)



→ containment edge → reverse containment edge → hyperlink edge

ElemRank Computation(2)

Intuition behind PageRank

For *HE* links, if an important page p_1 points to a page p_2 , then p_2 is likely to be important

PageRank

$$p(v) = \frac{1-d}{N} + d \sum_{(u,v) \in HE} \frac{p(u)}{n_h(u)}$$

where n is the total number of documents and $n_h(u)$ is the number of out-going links of u

ElemRank Computation (3)

Intuition behind elemRank

For *CE* links, if a node has a high rank also its children should have high rank. So reverse containment edges are added to the PageRank formula.

Forward ElemRank propagation

$$e(v) = \frac{1-d}{n} + d \sum_{(u,v) \in E} \frac{e(u)}{n_h(u) + n_c(u) + 1}$$

where n is the total number of XML document, $n_h(u)$ is the number of out-going links of u , $n_c(u)$ is the number of sub-elements of u and $E = HE \cup CE \cup CE^{-1}$

ElemRank Computation (4)

Last observation

The ElemRank of the parent elements must be directly proportional to the aggregate ElemRank of its children

Backward ElemRank propagation

$$e(v) = \frac{1 - d_1 - d_2 - d_3}{n_d n_{de}(v)} + d_1 \sum_{(u,v) \in HE} \frac{e(u)}{n_h(u)} + d_2 \sum_{(u,v) \in CE} \frac{e(u)}{n_c(u)} + d_3 \sum_{(u,v) \in C} \dots$$

Ranking Functions (1)

ranking functions should take into account:

- result specificity
- hyperlinks
- keyword proximity

$Q = (k_1, k_2, \dots, k_n)$: search query keyword

$R = \text{Result}(Q)$: result list of Q

$v \in Q$: result element

$$r(v, k_i) = \text{ElemRank}(v_t) \times \text{decay}^{t-1} \quad (0 \leq \text{decay} \leq 1)$$

where v_t is the node that contains the keyword k_i

Ranking Functions (2)

m occurrences of keyword k

computation of r_1, \dots, r_m

$$\hat{r}(v, k_i) = f(r_1, \dots, r_m)$$

with accumulation function f - e.g. max or sum

query q consists of keywords k_1, \dots, k_n

$$R(v, q) = \left(\sum_{1 \leq i \leq n} \hat{r}(v, k_i) \right) \times p(v, k_1, \dots, k_n)$$

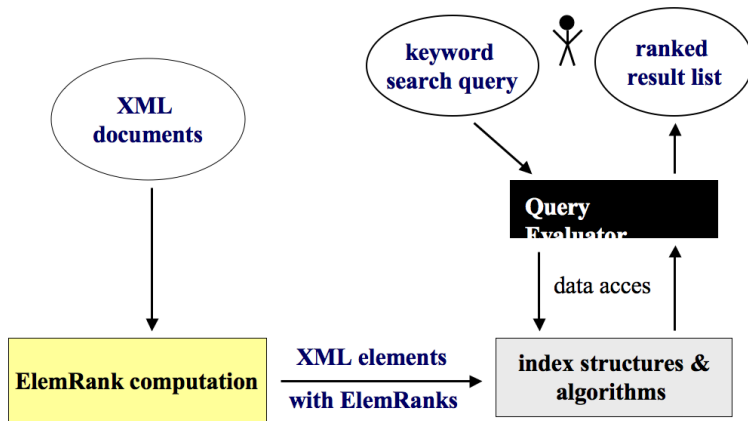
where $p(v, k_1, \dots, k_n)$ is a keyword proximity function

- 1 Introduction
- 2 XML ranking
- 3 Query Processing**

Query Processing

- Scanning through the index structures.
- Getting nodes corresponding to all the keywords.
- Joining these lists to get nodes containing all the keywords.
- Ranking the results using the ranking function.
- Finding and returning the best k results to the user.
- All the steps must be performed very efficiently.

XRank System



Dewey Inverted Lists - DIL

Dewey Inverted Lists capture the ancestor-descendant information.

- The joins of Dewey inverted lists are not simple since the result ids have to be inferred from the ids of the descendants.
- The processing involves merging the inverted lists and simultaneously computing the longest common prefix of the Dewey ids of different lists.
- A result heap keeps track of top k results seen so far.
- A Dewey stack stores the id, rank and position list of the current Dewey id and also keep tracks of the longest common prefixes.

DIL - Example

Table: Dewey inverted List sorted by Dewey ID

keyword	Dewey ID	Rank	Position List
K1	5.0.3.0.0	88	32
	6.0.3.4.0	38	89
...
K2	5.0.3.0.1	84	38
	8.2.4.1.2	99	32

- Query: " $k_1 k_2$ "
- Entry with smallest Dewey id: 5.0.3.0.0
- Next entry 5.0.3.0.1
- Longest common subsequence 5.0.3.0 (the deepest ancestor contains the query keywords)

DIL - Query processing

- Two data structures:
 - **Result heap** that keeps track of the top m results seen so far,;
 - **Dewey Stack** that stores the ID, rank and position list of the current Dewey ID and also keeps track of the longest common prefixes.

Integrated Structure Index and Inverted Lists

- Consider a path expression query.
- Extract the structure component and evaluate it over the structure index.
- The result set S of index I_{ds} is the union of the extents of all the index nodes that match the structure component of the query.
- The text component is evaluated over the inverted list.
- Consider only those entries e where $e.indexid \in S$.
- Inverted list joins are skipped wherever possible.

Path Expression Queries

- For specifying structure as well content.
- Arbitrary level of nesting.

Simple Path Expression

A simple path expression is a sequence $s_1 l_1 s_2 l_2 \dots s_k l_k$ where l_i except l_k is a tag name, l_k may be a tag name or a keyword, s_i is either / (parent- child) or // (ancestor-descendant) traversals.

Branching Path Expression

A simple path expression is a sequence $s_1 l_1 [Pred_1] s_2 l_2 [Pred_2] \dots s_k l_k [pred_k]$ where each $Pred_i$ is optional l_i except l_k is a tag name, l_k may be a tag name or a keyword, s_i is either / (parent- child) or // (ancestor-descendant) traversals.

Path Expression Examples

Query:

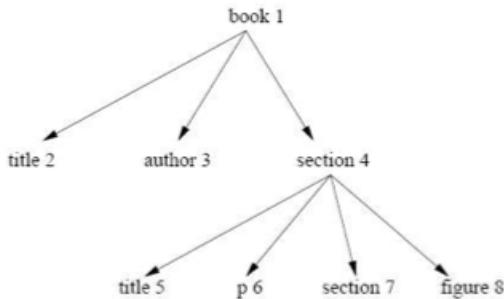
- `//section[/section/title/“web”]/figure/title`

Structure Component:

- `//section[/section/title]/figure/title`
- $S = \langle 4, 9, 12 \rangle$

Text component:

- `//section[/3“web”]/2title`



Computation of top k results

- A relevance list is an additional inverted list.
- Contains entries sorted by relevance.
- Any ranking function can be used to calculate the relevance.
- The inverted lists of all the keywords are scanned simultaneously
- For each entry, the relevance of the node against all the keywords is calculated and the top k results are maintained
- The algorithm terminates when it realizes that none of the future entries can be a part of the top k results.



Guo, L., Shao, F., Botev, C., and Shanmugasundaram, J. (2003).

Xrank: Ranked keyword search over xml documents.

In *Proc. of SIGMOD*.