Personalization of XML text search via search histories

© George Chernishev

Saint-Petersburg State University
Chernishev@gmail.com

Abstract

Full-Text XML search is a difficult problem due to its structural and textual constrains. The latter being individually well explored topics, lack mechanisms of integration with each other. Different approaches were proposed to cope with this problem. A technique of personalization usage to enhance one of these methods is given. The usage of direct or indirect results assessment is presented.

1 Introduction and related work

During the recent years XML has been widely applied and accepted as a standard for data integration and exchanging. As the popularity was growing, so did the range of applications of this format and formidable collections of text documents, represented in XML appeared. This, in turn, led to the actualization of a textual search in XML documents.

XML documents can be divided into following categories:

- Data-Centric
- Document-Centric

The former are data in a classical interpretation: results of mathematical calculations, experimental measurements, etc. They are processed by standard XML query languages like XPath and XQuery [23]. The latter are composed of the mixture of a structure and a text. The well-knows examples of such a collections are IEEE INEX [19], ACM SIGMOD record collection [20], DBLP [18], Shakespeare's plays in XML [22], United States Library of Congress documents in XML [21], etc. Text searching in document-centric XML requires combining two classes of methods. The first one is a classical interaction with hierarchical data, and the second class consists of Information Retrieval methods. Simple reuse of data-oriented XML query languages resulted in poor performance. The main drawbacks of this were the absence of ranking mechanisms, a poor granularity of results, the problem of weight manipulation etc.

New approaches [1, 4, 8, 10, 11] helped to solve or alleviate these problems. These can be generally classified into the following groups [7]:

- Keyword search and IR methods reuse. That group is composed of the attempts to implement traditional Informational Retrieval methods to XML search.
- Tagged search, where term receives additional information about a tag, where it should be found.
- Navigation approach. This approach attempts to enrich navigational languages with some means to allow textual search.
- A mixture of XQuery-like language and keyword search. The goal of these systems is to add a feature of textual search to XML query language, while leaving untouched its initial capabilities to search non-textual data.

A full overview of this classification, and an extensive list of example systems can be found in [7]. A few systems deal with user's awareness of document structure directly. The main portion of these is the representatives of the third and the fourth groups, and a large effort was put into this issue recently. These efforts aim to lower the impact of the scarcity of user's awareness of a document structure. There exist two main approaches to this problem.

- The first method [12, 13, 14, 15] is based upon the idea of differentiation of users by the level of structure knowledge, and division into a few groups. In the mentioned papers it was suggested to divide users into two groups: the first one is composed of those who have no knowledge about document schema at all, and those who have partial knowledge about hierarchical relationship on elements. Then, the modification of the language was proposed. The main idea of this modification is "Less power is better", and this is achieved through restrictive modifications. These modifications restrict an expressive power of the language for the purposes of safety. Here, safety means protection from errors introduced by a query designer. For example, the following restrictions could be used: forbid using parent-child axis and using instead ancestor-descendant or restrict usage of predicates to a narrower class.
- Second method employs the idea of relaxations [2, 3]. The papers present the following idea: having a query, with a tree pattern representation, a query evaluator constructs additional trees using special modification technique - relaxations. The goal of these relaxations is to enrich the variety of the expected results with possible relevant information. The trees after such transformations could range from the base tree to the tree containing only one node with all

predicates attached. Then, these queries are evaluated, and their results are combined and presented to user.

There are a lot of exterior to XML evaluation and information retrieval techniques dedicated to improving quality and speed of search using: onthologies, statistics etc [6, 16]. These are concentrating on different aspects of the problem. One of these methods refines the quality of results via methods of per-user personalization [5]. These methods utilize some external information during or earlier the phase of a query evaluation. Personalization could be implemented in a variety of means, ranging from a different order of results presentation due to different ranking schemas, to a completely different set of results, due to additional constraints.

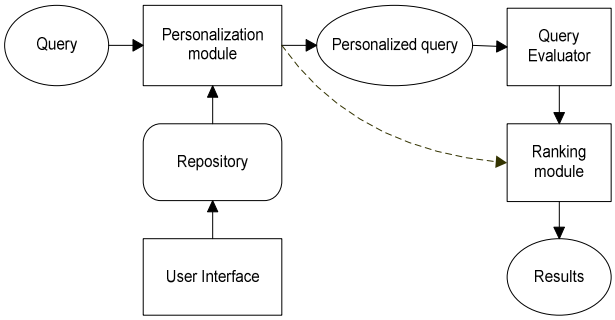

Figure 1: Architecture of query engine with personalization

In [5] a method of rule-based personalization was proposed. Its architecture is shown on Fig. 1. Personalization occurs in two modules: the personalization module, which performs query rewriting and the ranking module, which employs different mechanisms of ranking corresponding to different user profiles. The system uses a repository of user profiles, which contain rules specific to a given user. When the query is formed, the system rewrites it according to these rules. Contextual hints, concerning the information field, represented in a form of rules, are fed into the aforementioned system. This additional information could be, for example user's geographical location, his age, interests. These data could change the nature of search and supply more relevant answers.

In this paper I present the idea of application of these mechanisms to user's awareness of a document structure. The proposal is to extract (from query history) and keep some facts about user's knowledge of document. Query evaluator would utilize these facts, performing the rewriting.

2 Model

2.1 Overview

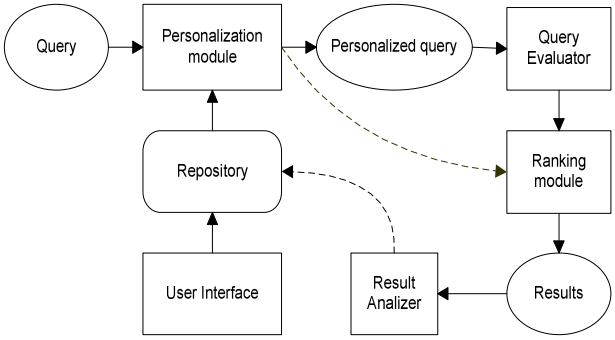

Figure 2: Architecture of the proposed query engine with personalization and feedback

The architecture of the system partially reuses the design of the personalized query engine from the previous chapter; it is shown in Figure 2. Let's see how query flows through the stages mentioned in the diagram. When the query reaches into personalization module it is copied and rewritten according to applicable transformations of its tree pattern. These applicable transformations are acquired from the repository. Then, these queries are evaluated; their results are mixed according to the personal ranking scheme, and presented to user. By choosing the relevant documents user can notify the system about effectiveness of existing transformations, or of the necessity of new ones. The result analyzer is responsible for dealing with user feedback from delivered results.

Our mechanism of the query rewriting will be based on a modified well-known technique of query rewriting called relaxations [2]. In a general case, relaxation is a query, derived from the source query, which is complying with some subset of rules. The following different approaches to relaxations exist [2, 9, 15]. The core can be briefly described as:

- Axis generalization. This relaxation is a modified source query, where parent-child condition is substituted with ancestor-descendant condition. Consider such an example query (in XPath notation): /a/b. The relaxed query is the /a//b.
- Leaf deletion. An idea of constraint dropping is the basis of this relaxation. Its result is a set of queries, each one of them has some leaves clipped out.
- Subtree promotion. It is possible to get relevant results from a modified query, which has some of its nodes lifted up the hierarchy. An extreme case of this structural promotion is the leaf promotion. The example of application this relaxation to /a/b[./c] is /a[./c]/b.
- Contains promotion. Another case of promotion is the textual constrains promotion. This operation takes the constraint of some node and then moves it to its parent, resulting in a new relaxed query. Given a query like this a/b/c["textual constraint"] a possible relaxation would be a a/b["textual constraint"]/c.

We would change the sense and the notion of relaxations. To distinguish the sense, we will call our

modification of relaxations – transformations. The above-listed relaxations are designed by their in-system meaning, we would aim to redesign them guided by more explicit user's specificity.

Possible transformations could be divided into two classes:

- Structural transformations

The aim of these transformations is to determine user awareness regardless of textual content.

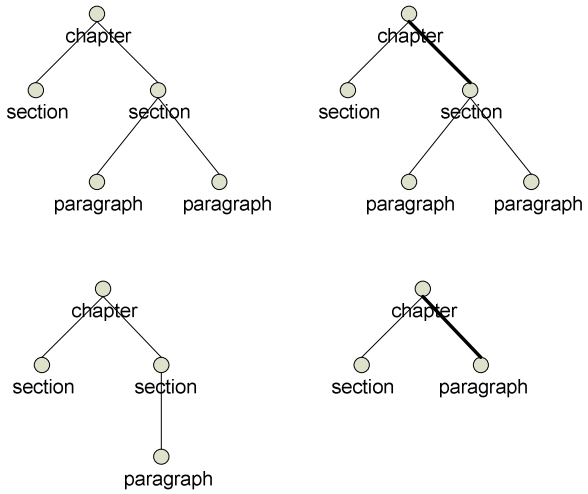

Figure 3: Structural transformations

Here is an example of transformations: the leftmost tree is the initial query, the solid line marks an ancestor-descendant relationship and the thick line is a parent-child relationship. Then, we have a list of some possible transformations, which became less and less strict.

- Content transformations

Their aim is to determine user's awareness of the subject in general. This class of transformations guides the dissemination of textual constraints. For example, having a bush representing a query, a set of acceptable transformations will consist of trees, the content predicates of which are "lifted up" to parent nodes.

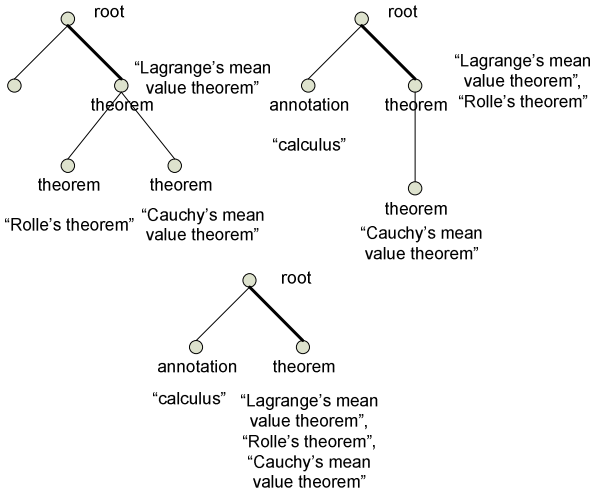

Figure 4: Content transformations

Figure 4 shows an example of possible transformations with textual constraints. The leftmost tree pattern in the upper row is the initial query. The rightmost tree pattern shows the least specific query which could supply user with relevant results, which can't be found by initial query. Thus, it could relieve user from additional work: for example, searching for a few related facts (the example shows a book of calculus theorems, some of which are proved on the basis of others) in a mathematical book, without knowing their hierarchical relations, one can write a dozen of queries with irrelevant results or even, nothing.

The core of this system is the module Result Analyzer and its subsystem called history. The history contains user's queries, system response (query results), and users response (it may consist of the chosen item, a set of chosen items, users time to examine a particular result or some other behavior which can be a hint of success or failure).

2.2 History properties

Now we describe some important history properties:

1) Storage of a query, results and user's judgment.

The main role of the history is to create all necessary conditions for the fact analyzer and supply all possible transformations.

2) History management

History must include control interface which should allow flexible customization of the acquisition and usage of rules, which represent user's knowledge (through transformations)

3) Dynamical nature of history: updates and discards.

User's knowledge is not static. During the work with the document, a user acquires the knowledge about its schema. Also, he could get knowledge about a textual content of some subset of nodes and start using it very effectively. To enhance quality of search, this model should update its facts about user's knowledge, deleting or substituting the former. Or there exist another kind of cases - when a user didn't used the system for a long time, one can assume that his knowledge has reduced, so the system has to mark this by discarding the related data. Also, history should cope with dynamicalness of the source data. That means that adequate responses to updates and in a lesser extend to deletes of the underlying XML source data are required. This situation with the preference of updates upon deletes is generated by the nature the texts: it is not likely that information would be deleted.

4) Hierarchy and compositionality of facts.

The facts of user awareness could be combined with each other, effectively giving new knowledge. Conversely, the system may have more general knowledge of user awareness, but the query involves some specific information, which should be deduced.

3 Further work

Further work includes the evaluation of existing fact manipulation models, experiments with them. The aim is to find appropriate model and tailor it for XML retrieval needs, or invent it. Also, this model should allow subsistent level of self-manipulation, to provide sufficient level of freedom of model altering to user needs. This model lacks specific ways of determining user response, and they should be introduced. The model will also undergo some refinement caused by alternation of the relaxation model: that not all kinds of transformations can be derived from existing systems, and not all relaxation will fit into this model due to different purposes of these systems.

References

[1] Shurug Al-Khalifa, Cong Yu, H. V. Jagadish. Querying structured text in an XML Database. *In Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 4-15, 2003.

[2] Sihem Amer-Yahia and SungRan Cho and Divesh Srivastava. Tree Pattern Relaxation. In *Proceedings of the 8th International Conference on Extending Database Technology*, pages 496-513, 2002.

[3] Sihem Amer-Yahia and Laks V. S. Lakshmanan and Shashank Pandit. FleXPath: flexible structure and full-text querying for XML. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 83-94, 2004.

[4] Sihem Amer-Yahia, Chadvar Botev,Jayavel Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *Proceedings of the 13th international conference on World Wide Web*, pages 583-594, 2004.

[5] Amer-Yahia, S.; Fundulaki, I.; Jain, P. & Lakshmanan, L. Personalizing XML text search in PIMENT. In *Proceedings of the 31st international conference on Very large data bases, VLDB Endowment,* pages 1310-1313, 2005.

[6] Sihem Amer-Yahia, Emiran Curtmola, Alin Deutsch. Flexible and efficient XML search with complex full-text predicates. *In Proceedings of the 2006 ACM SIGMOD international conference on Management of data,* pages 575-586, 2006.

[7] Sihem Amer-Yahia and Mounia Lalmas. XML search: languages, INEX and scoring, *SIGMOD Record*, pages 16-23, December 2006.

[8] Sara Cohen and Jonathan Mamou and Yaron Kanza and Yehoshua Sagiv. XSEarch: a semantic search engine for XML. *In Proceedings of the 29th international conference on Very large data bases*, pages 45-56, 2003

[9] C. Delobel and M.C. Rousset. A Uniform Approach for Querying Large Tree-structured Data through a Mediated Schema. International Workshop on Foundations of Models for Information Integration

[10] Norbert Fuhr, Kai Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Research and Development in Information Retrieval, ACM-SIGIR*, New Orleans, pages 172-180 2001.

[11] Lin Guo and Feng Shao and Chavdar Botev and Jayavel Shanmugasundaram. XRANK: ranked keyword search over XML documents. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 16-27, 2003.

[12] Jaap Kamps and Maarten Marx and Maarten de Rijke and Börkur Sigurbjörnsson. Best-match querying from document-centric XML. In *Proceedings of the 7th International Workshop on the Web and Databases*, pages 55-60, 2004.

[13] Jaap Kamps, Maarten Marx, Maarten de Rijke, Börkur Sigurbjörnsson. Structured queries in XML retrieval. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 4-11, 2005.

[14] Jaap Kamps, Maarten Marx, Maarten de Rijke, Börkur Sigurbjörnsson. Articulating information needs in XML query languages. In *ACM Transactions on Information Systems (TOIS),* Volume 24 , Issue 4,pages 407 – 436, 2006.

[15] T. Schlieder. Similarity Search in XML Data using Cost-Based Query Transformations. ACM SIGMOD 2001 Web and Databases Workshop. May, 2001.

[16] Börkur Sigurbjörnsson and Jaap Kamps and Maarten de Rijke. Processing content-oriented XPath queries. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages, 371-380 2004.

[17] Anja Theobald, Gerhard Weikum. An Index Based XXL Search Engine for querying XML data with relevance ranking. In *Proceedings of the EDBT Conference, Prague*, 2002.

[18] DBLP in XML. http://dblp.uni-trier.de/xml

[19] INEX: Initiative for the Evaluation of XML Retrieval. http://inex.is.informatik.uni-duisburg.de

[20] SIGMOD record in XML. http://acm.org/sigmod/record/xml/

[21] The Library of Congress. http://lcweb.loc.gov/crsinfo/xml

[22] The plays of Shakespeare in XML. http//www.oasis-open.org/cover/bosakShakespeare200.html, by J. Bosak.

[23] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. W3C Working Draft. http://www.w3.org/TR/2005/WD-xquery-full-text-20050404/