GuessXQ, an inference Web-engine for querying XML documents

Daniela da Cruz[1,3], Flávio Xavier Ferreira[1,2,4], Pedro Rangel Henriques[1,5], Alda Lopes Gancarski[2,6], and Bruno Defude[2,7]

[1] University of Minho, Department of Computer Science, CCTC, Campus de Gualtar, Braga, Portugal
[2] Institut TELECOM, TELECOM & Manangement SudParis, CNRS SAMOVAR 9 rue Charles Fourier, 91011 Évry, France
[3] danieladacruz@di.uminho.pt
[4] flavioxavier@di.uminho.pt
[5] prh@di.uminho.pt
[6] Alda.Gancarski@it-sudparis.eu
[7] Bruno.Defude@it-sudparis.eu

Abstract. To search for specific elements in a marked up document we have, at least, two options: XPath and XQuery. However, the learning curve of these two dialects is high, requiring a considerable level of knowledge. In this context, the traditional *Query-by-example* methodology (for Relational Databases) can be an important contribute to make easier this learning process, freeing the user from knowing the specific query languages details or even the document structure.
In this paper, we describe how we implement *Query-by-example* in a Web-application for information retrieval from a collection of structured documents, the GuessXQ system. In essence, we built an engine capable of deduce, from a specific example, the respective XQuery statement. After inferring the generic statement, the engine applies it to all documents in the collection to perform the desired retrieval.
A suitable interface allows the end-user to mark over a sample document, picked up from the collection, the path he wants to select.

1 Introduction

In Structural Document Retrieval [3] the creation of a query that yields valid results strongly depends on the user-friendliness of the search engine interface. As structured queries are powerful but complex to write (the user must have a deep knowledge of the query language as well as the document schema), some specialised editors have been developed to ease this task (XMLSpy[2], EditiX[8],oXygen[9]).

"Example is always more efficacious than precept." This statement, by Samuel Johnson, led HCI[10] researchers to suggest a new interaction paradigm called

[8] http//www.editix.com
[9] http//www.oxygenxml.com
[10] Human-Computer Interaction.

Query-by-example (QBE). Born in the context of database querying [5], typical QBE systems are based on the "fill in the blanks" approach. QBE is based on the concept that the user formulates his query by filling in the appropriate skeleton tables the fields and/or restrictions on fields (the relational selection concept) he intends to search for. We adapt this approach to XML search by allowing to select and restrict entire paths (XML elements) directly on a sample document. There are some other works [1,4,6] which adapt the relational QBE model by showing the XML Schema Definition (XSD) tree instead of the table skeleton. Our system, not only displays the XML Schema tree representation to the user, but also an sample document from the collection. Elements selection and restriction is, then, directly done in the sample document, giving the user a complete indication about the information he is searching for.

From the selected paths, our engine, called GuessXQ, infers the complete query. After building the query, it is applied to all documents of the same type (schema) in the specified collection. The results are shown in a user-friendly Web interface used to select the referred path. The next section presents the GuessXQ system, followed by our conclusions.

2 GuessXQ system

The architecture of our QBE system, the GuessXQ engine, is presented in Figure 1. As depicted, the system is composed by the several modules.

The **Repository** is a collection of XML files grouped by their schema (XSD). This Repository, in a simple way, is composed by three tables: `XMLdocs`, `XSDfamilies` and `Relations`. The `XMLdocs` table stores the name of the XML document and its location; the `XSDfamilies` table stores the name of each XSD and its location; and at last, the `Relations` table relates each XML document with the XSD family where it belongs.

The **Repository access interface** allows the other modules to access the repository in a systematic and simple way. It allows the other modules to select a Schema, a collection of documents, a single document or a path in the document. This module is composed by a set of *fetch* methods for retrieving documents from the repository. This module works as an abstraction layer over the repository, allowing the system to change the repository paradigm in the future without a major architecture modification.

The **Interface module** is a GUI responsible for the interaction between the user and the system, allowing the user to set the "example" for the QBE engine. The user starts by choosing a document type (XSD) from the repository. Then, a document belonging to this family is suggested to him. By now, the sample document is selected to be of average size, and to contain the major number of elements/attributes. However, the user can change it, choosing a more significant one from the query perspective (i.e., an XML document that allows to perform more complex or complete queries).

Query specification includes the selection of components (elements or attributes) and the possibility to restrict them to the respective value in the sample

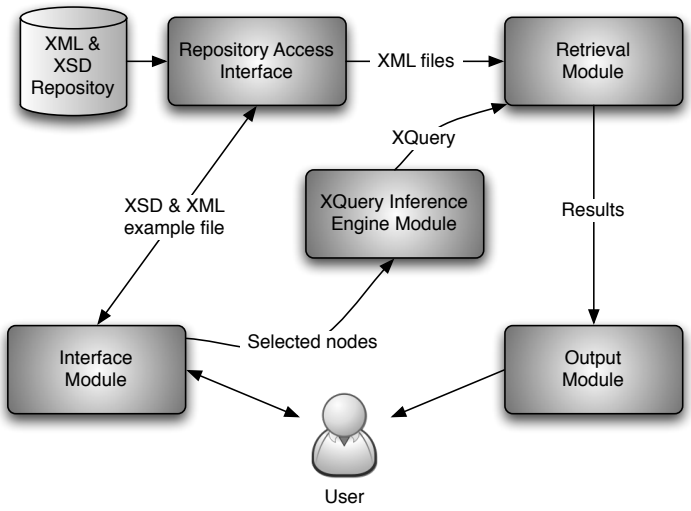

Fig. 1. GuessXQ architecture

document. For example, in Figure 2, the user specifies a query selecting elements (in yellow) and content values (in blue). This selection mechanism is aided by a table that maps each component or value with its corresponding XPath expression. After query specification, the interface module sends to the XQuery Inference Engine module the list of selected nodes and also the referred table, for query generation.

The **XQuery Inference Engine** module has the task of inferring the XQuery query, using the information sent by the Interface module.

The **Retrieval module** is responsible for the query execution. The query is executed in each document belonging to the selected schema, and the results are passed into the next module.

The **Output module** shows the query results, obtained in the previous module.

3 Conclusion and future work

The system we present is dedicated to XML structural retrieval without the need for the user to know XQuery and the XML documents precise structure. The system is based in a user-friendly QBE interface for specifying information needs. In a first step, the user visualises the set of existing XML Schema in the repository in the form of trees (alternatively graphs) and chooses the desired documents collection. After choosing the schema, a sample XML document from the corresponding collection is displayed. Having the schema tree view together with the sample document helps the user to better identify the document parts

Fig. 2. Selection mechanism

he needs. The user specifies the interesting elements or textual parts by just selecting them in the sample document view in an easy way.

As future work, we intend to perform some improvements, like a more powerful/intelligent selection algorithm to choose a better sample document.

References

1. D. Braga and A. Campi. Xqbe: A graphical environment to query xml data. *World Wide Web*, 8(3):287–316, 2005.
2. L. Kim. *The XMLSPY Handbook*. John Wiley & Sons, Inc., New York, NY, USA, 2002.
3. X. Lu. Document retrieval: A structural approach. *Inf. Process. Manage.*, 26(2):209–218, 1990.
4. S. Newman and Z. M. Ozsoyoglu. A tree-structured query interface for querying semi-structured data. *Scientific and Statistical Database Management, International Conference on*, 0:127, 2004.
5. R. Ramakrishnan and J. Gehrke. *Database Management Systems*, chapter 6. 2007.
6. J. H. G. Xiang Li and J. F. Brinkley. XGI: A Graphical Interface for XQuery Creation. In *American Medical Informatics Association Anual Symposium proceedings*, volume 2007, pages 453–457, November 2007.