Inference Document Type (DTD) from XML Document: Web Structure Mining

Nanhay Singh
Asstt. Prof.
Department of Computer Science & Engineering
HBTI, Kanpur (India)

Dr. R. K. Chauhan
Professor & Chairman
Department of Computer Science & Applications
Kurukshetra University (India)

Dr. Raghuraj Singh
Professor & Head
Department of Computer Science & Engineering
HBTI ,Kanpur (India)

ABSTRACT
XML is becoming a prevalent format and de- facto standard for data exchange in many applications. While traditionally, lots of data are stored and managed in relational databases. There is an urgent need to research some efficient methods to convert these data stored in relational databases to XML format when integrating and exchanging these data in XML format.

The semantics of XML schemas are crucial to design, query, and store XML documents and functional dependencies are very important representations of semantic information of XML schemas. As DTDs are one of the most frequently used schemas for XML documents in these days, we will use DTDs as schemas of XML documents here.

This paper studies the problem of schema conversion from relational schemas to XML DTDs. As functional dependencies play an important role in the schema conversion process, the concept of functional dependency for XML DTDs is used to preserve the semantics implied by functional dependencies and keys of relational schemas. A conversion method is proposed to convert relational schemas to XML DTDs in the presence of functional dependencies, keys and foreign keys. The methods presented here can preserve the semantics implied by functional dependencies, keys and foreign keys of relational schemas and can convert multiple relational tables to XML DTDs at the same time.

1. INTRODUCTION
XML (eXtensible Markup Language) has become one of the primary standards for data exchange and representation on the World Wide Web and is widely used in many fields. Historically, lots of data and information are still stored in and managed by relational database management systems, such as Oracle, Sybase, SQL Server, etc. So it is necessary and urgent to develop some efficient methods to convert relational data to XML data in order to take advantage of all the merits of XML. As DTDs (Document Type Definitions) are still the most frequently used schemas for XML documents in these days, we will use DTDs as schemas of XML documents. It is widely accepted that schema semantics plays a very important role in the schema conversion, and functional dependencies, keys, and foreign keys of relational schemas are very important representations of semantic information. So it is significant that the conversion method from relational schemas to XML DTDs must consider the semantics implied by functional dependencies, keys, and foreign keys, and the obtained XML DTDs can represent such semantics in some way. Data Mining is the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World Wide Web. Regression is a data mining function that predicts a number. Profit, sales, mortgage rates, house values, square footage, temperature or distance could all be predicted using regression techniques. For example, a regression model could be used to predict the value of a data warehouse based on web-marketing, number of data entries, size, and other factors.

2. OBJECTIVES
Suppose in any organization, if we want to transfer some database(s) to another organization then we have to give entire XML files to another organization. This process of transfer of XML files is very effective only for the technical persons but for the managing persons, understanding the typical XML files is very tedious process. Hence we have to develop a software that can convert the XML files in such a form so that it can be easily understood by the managing persons also. And therefore the concepts of the DTD i.e. document type descriptor came into existence which shows the hierarchical representation of the XML files. In this paper we used the concept of the web-mining which can be used to fetch some data from the databases easily and effectively. The approach that we have taken to overcome the difficulty in deriving DTDs is to build a tool that will automatically suggest the DTD for a collection of XML documents provided by the user. These XML documents are well formed but do not come with a DTD. The derived DTD will provide an overall schema for the document collection. The assumption that we have made here is that the collection of documents are similarly structured and follows a single naming convention for the tags. This software has been developed based on our proposed algorithms for structural discovery of XML documents. It is further equipped with a frame-based user interface that allows XML documents to be uploaded from the client computers and presents the derived DTD in the Web browser. It further allows users to refine and simplify the derived DTDs by adjusting an input parameter known as maximum repetition factor.

3. PAPER OUTLINE
In this paper, we describe the design and implementation of DTD Converter. We will cover its system architecture in section 4 and gives a brief overview used to represent the instance structure of an XML document and the overall structure of a collection of these documents respectively. Section 5 briefly discusses the problem formulation, that how the conversion of the DTD should take place with the help of automata and regular expression. The overview of the system is explained in section 6 which comprises of the Generalization Subsystem and the Factoring Subsystem. A first version of the DTD Converter has been implemented and described in Section 7 with the help of the User Interface. Section 8 describes some related research and concludes the paper.

4. SYSTEM ARCHITECTURE

The DTD Converter is made up of various modules as shown:

- **DTD-Miner User Interface**: The Web Interface allows the user to submit XML files. The Web Interface is also responsible for displaying the generated DTD. In addition, it also allows the user to refine the DTD so as to reduce its complexity using a parameter Maximum Repetition Factor, which represents the maximum number of times a child element may appear in the parent element's definition in the DTD.
- **DTD Generation Module:** This is the main module responsible for the generation of the DTD for a set of structurally similar XML documents supplied by the user.

5. PROBLEM FORMULATION

Our goal is to infer a DTD for a collection of XML documents. Thus, for each element that appears in the XML documents, we aim to derive a regular expression that sub element sequences for the element (in the XML documents) conform to. Note that an element's DTD is completely independent of the DTD for other elements, and only restricts the sequence of sub elements nested within the element. Therefore, for simplicity of exposition, we concentrate on the problem of extracting a DTD for a single element. Let e be an element that appears in the XML documents for which we want to infer the DTD. It is straightforward to compute the sequence of sub elements nested within each <e></e>pair in the XML documents. Let I denote the set of N such sequences, one sequence for every occurrence of element e in the data. The problem we address in this paper can be stated as follows: Given a set of N input sequences nested within element e, compute a DTD for e such that every sequence in conforms to the DTD. As stated, an obvious solution to the problem is to find the most concise regular expression R whose language is I .One mechanism to find such a regular expression is to factor as much as possible, the expression corresponding to the OR of sequences in I. Factoring a regular expression makes it concise without changing the language of the expression. For example, ab|ac can be factored into (a|b)c . An alternate method for computing the most concise regular expression is to first find the automaton with the smallest number of states that accepts I and then derive the regular expression from the automaton. Such concise regular expressions, whose language is exactly I, are unfortunately not good. DTDs, in the sense that they tend to be voluminous and unintuitive. Suppose we have a collection of XML documents that conform to this DTD. Abbreviating the title tag by t and the author tag by a, it is reasonable to expect the following sequences to be the sub element sequences of the article element in the collection of XML documents: t,ta,taa,taa,taaa,taaaa Clearly, the most concise regular expression for the above language is t(a|(aa|(aaa|(aaaa)))) which is definitely much more voluminous and lot less intuitive than a DTD such as ta*. In other words, the obvious solution above never generalizes and would therefore never contain meta-characters like (*) in the inferred DTD. Clearly, a human being would at most times want to use such meta-characters in a DTD to succinctly convey the constraints he/she wishes to impose on the structure of XML documents. Thus, the challenge is to infer, for the set of input sequences I, a general DTD which is similar to what a human would come up with.

6. OVERVIEW OF THE SYSTEM

• The Generalization Subsystem

For each input sequence, the generalization module generates zero or more candidate DTDs that are derived by replacing patterns in the input sequence with regular expressions containing meta-characters like * and | (e.g. (ab)*,(a|b)*). Note that the initial input sequences do not contain meta-characters and so the candidate DTDs introduced by the generalization module are more general. For instance sequences abab and bbbe result in the more general candidate DTDs (ab)*,(a|b)* and b*e to be output by the generalization subsystem. Also, observe that each candidate DTD produced by the generalization module may cover only a subset of the input sequences.

Ideally, in the generalization phase, we should consider all DTDs that cover one or more input sequences as candidates. However, the number of such DTDs can be enormous. For example, the sequence ababaabb is covered by the following DTDs in addition to many more.

(a|b)*, (a|b)*a*b*, (ab)*, (ab)*c*b* Therefore, system employs several novel heuristics, inspired by real-life DTDs, for limiting the set of candidate DTDs.

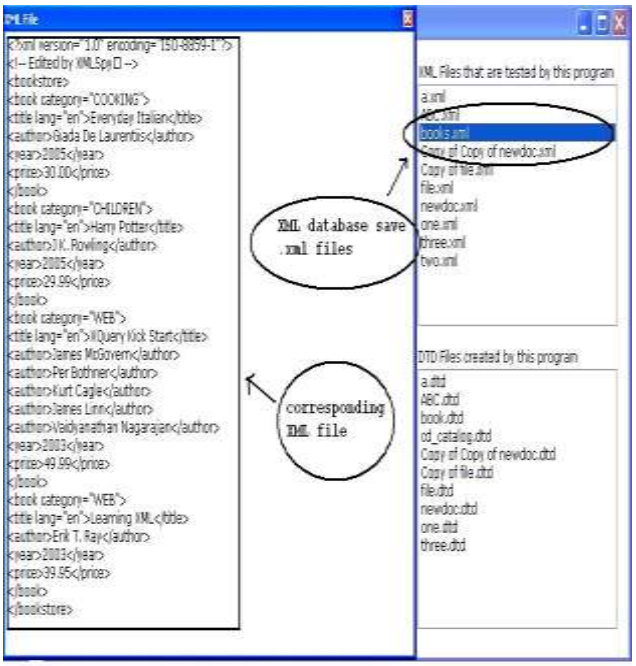
• The Factoring Subsystem

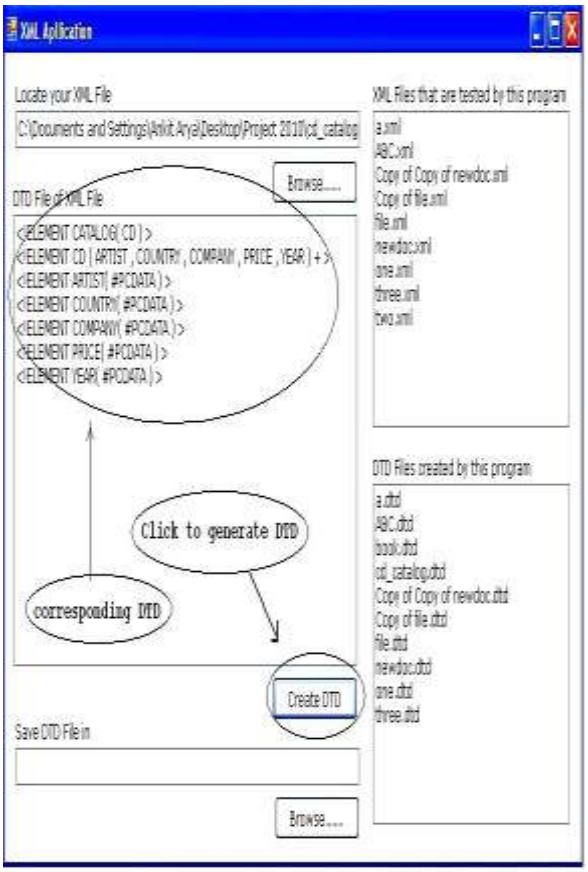
The factoring component factors two or more candidate DTDs into a new candidate DTD. The length of the new DTD is smaller than the sum of the sizes of the DTDs factored. For example candidate DTDs b*d and b*e representing the expression b*d|b*e, when factored, result in the DTD b*(d|e).Although factoring leaves the semantics of candidate DTDs unchanged, it is nevertheless an important step. The reason being that factoring reduces the size of the DTD and thus the cost of encoding the DTD, without seriously impacting the cost of encoding input sequences using the DTD. Thus, since the DTD encoding cost is a component of the MDL cost for a DTD, factoring can result in certain DTDs being chosen by the MDL module that may not have been considered earlier. We appropriately modify factoring algorithms for boolean functions in the logic optimization area to meet our needs. However, even though every subset of candidate DTDs can, in principle, be factored, the number of these subsets can be large and only a few of them result in good factorizations. We propose novel heuristics to restrict our attention to subsets that can be factored effectively.

7. USER INTERFACE OF THE SOFTWARE

In this section, we give an illustration of an interaction with the prototype system in order to provide a feel of how the DTD-Converter system works. There are three main screens of the DTD-Converter system: the initial document input screen, the file directory screen and the DTD display screen.

The system begins at the initial document input screen. The user can then submit his own XML documents by first selecting the XML document in his local machine using the "Browse..." button and then upload the file to the system using the "Open" button and the added file should be reflected in the file directory screen. The Maximum Repetition Factor can also be optionally specified here by entering the required value in the text box provided.





8. DESCRIPTION

This phase of graphical user interface shows the functionality of browse option which provide the functionality of loading the XML file to end user. The XML file may be located to any position in the database or in a specific location if we talk about any particular system. On clicking the browse option the window will appear which will show the position of various files stored at different locations. Locate your XML file field shows the current location of the XML file which will be used to create the DTD file. After browsing the XML file, the preferred location will be pointed to the execution server which will go for the current location of the file and load it in the current database of our software and also load for creation of the DTD. In the next step, we will go for creating the DTD by hitting create DTD button. After hitting this button, the uploaded location of XML file will be preferred and corresponding DTD will be generated and it will be displayed on the screen. It will save the corresponding DTD to its own database for the future use. The software saves the current DTD for further use in its own database. It also provides the functionality of saving the DTD file manually. Save option does this task significantly. On hitting this option the preferred location will appear which can be changed as per the user's choice. While the preferred location is chosen by the user it will hit the "OK" option on the software. After doing this, the corresponding file will be saved on the desired location. File created option will be shown to the user to specify this task. This figure shows the overall architecture of the XML file which can be accessed from the software's database. It also provides the display window which shows the entire saved file in the XML database. We can access any of its database file for further use. It is some kind of monitoring system for XML database which periodically monitors the current position of the database. At last it also shows the database monitoring system for DTDs stored in the database as same as the XML database monitoring system. We can go for any of the DTDs stored in the database for further use. It may be used as the regulating rules.

9. CONCLUSION AND FUTURE WORK

With the increasing complexities of Web documents and the emergence of XML, user will have a strong need or a tool to automatically extract structures of Web documents. This project describes the DTD-Mining software that is designed for these purposes. The DTD generating software is a prototype system build from the framework for structural re-engineering XML documents. It has the ability to automatically generate DTD from a set of similarly structured XML documents submitted by the user. However, the software does not support the generation of attribute types and entity references and does not handle hyperlinks and multimedia data. We are currently looking to further extensions to support attribute types and entity reference generation and also the use of hyperlinks to create inter-document structures. Work is also being done in various methods of simplifying the DTDs generated. The DTD generating System is a prototype for the structural re-engineering framework proposed. It enables the automatic generation of a DTD from a large set of XML documents while not compromising the ease of use of the user. The system built by is one research effort that draws some conceptual resemblance to the DTD generation. The system attempts to infer the structure of HTML documents by drawing clues from display and formatting information of the HTML tags and also the indentation to guess the structure of the Web documents. The system however does not deal with XML and DTD. The structural mining component of the NoDoSE by Adelbreg also draws some similarities to DTD generating system. NoDoSE is a semi-automatic system for data extraction. NoDoSE is primarily based on plain text files but the HTML Parser component allows HTML files to be handled. There are two main drawbacks of this system for mining structures from XML documents. Firstly, we feel that the degree for user intervention is too extensive and may prove to be tedious for the user when the documents to be parsed are large. Secondly, the NoDoSE does not support XML. The difference between HTML and XML is that in XML documents, the structure of the document can be enclosed within user defined tags. The DTD Generator is a tool that is able to generate the DTD for a given XML document. The main problem with the DTDs generated from the DTD Generator is that it will generate a DTD for every document i.e., the system cannot handle the generation of an overall DTD for a set of structurally similar XML documents. One of the main features of XML is that it allows the user to define their own grammar rules. However, in our opinion, the user would usually define a set of rules for a collection of Web documents rather than a separate set of rules for a single document. This makes the ability for the system to generate an overall DTD essential. It is however, interesting to note the way the DTD Generator attempts to handle attribute and attribute types. The work, however, does not address how generated DTDs can be simplified by users using some quantitative measure.

REFERENCES

[1] "The XML Handbook" by Charles F. Goldfarb and Paul Prescod under the "Prentice Hall of India" publication.

[2] B. AdelBerg. NoDoSE - A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents. In ACM SIGMOD International Conference on Management of Data, pages 283–294, 1998.

[3] N. Ashish and C. Knoblock. Wrapper Generation for Semi-structured Internet Sources. ACM SIGMOD Record, 26(4):8–15, 1997.

[4] D. Beech, S. Lawrence, and M. Maloney. XML Schema Part 1: Structures. Technical report, World Wide Web Consortium, http://www.w3.org/1999/05/06-xmlschema-1, 1999.

[5] P. Biron and A. Malhotra. XML Schema Part 2: Datatypes. Technical report, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-2/, 1998.

[6] T. Bray, C. Frankston, and A. Malhotra. Document Content Description for XML. Technical report, World Wide Web Consortium, http://www.w3.org/TR/1998/NOTE-dcd-19980731.html, 1998.

[7] T. Bray, J. Paoli, and C. Sperberg. Extensible Markup Language (XML) 1.0. Technical report, World Wide Web Consortium, http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[8] M. Fuchs, M. Maloney, and A. Milowski. Schema for Object-oriented XML. Technical report, World Wide Web Constorium, http://www.w3.org/TR/NOTE-SOX, 1998.

[9] M. Kay. SAXON DTD Generator - A Tool to Generate XML DTDs. At http://home.iclweb.com/icl2/mhkay/dtdgen.html