The 8th International Conference on e-Business (iNCEB2009)
October 28th-30th, 2009

VALUE-BASED PREDICATE FILTERING TECHNIQUE OF XML DOCUMENT USING ONTOLOGY

Yi Yi Hlaing
University of Computer Studies, Yangon, Myanmar
yeyehlaing@gmail.com

ABSTRACT

Nowadays, information dissemination applications are very popular and much of the data exchange over the Internet. XML plays an important role as a de-facto standard information exchange and is recommended by W3C. The various XML data filtering methods have been developed in most web applications. In this paper, we describe some existing filtering methods and comparison of the existing methods. In addition, we developed a new XML Document filtering system for multiple XPath queries which is based on ontology. The proposed system contains three components: path generation, queries transformation and matching. By using the path generation component, the user profiles are generated from DTD and all generated queries are represented in XPath format. The multiple queries which include value-based predicates are transformed into a single Nondeterministic Finite Automaton (NFA) based model. On the other hand, incoming XML document is parsed by SAX parser. After that, the user profiles of XPath queries are matched with the incoming XML documents. In addition, OWL ontology is added to that filtering process. Therefore, proposed method intends to provide not only exact matched information but also semantic matched information for multiple queries.

Index Terms— NFA, Value-based predicates, Ontology

1. INTRODUCTION

XML filtering mechanisms are very useful for most internet applications. In such systems, user profiles are expressed in XML query languages such as XML-QL, XPath, XQuery and ApproXQL. In this paper, we focus on user profiles represented in XPath. We can also classify XML data filtering methods into Automata-based filtering, Trie-based filtering, Index-based filtering, Sequence-based filtering, Twig pattern based filtering, Stack based filtering, predicate-based filtering and other approaches. The proposed system uses Nondeterministic Finite Automaton (NFA)-based filtering for sharing XPath queries. There are a lot of ontology-based XML querying methods. Here, querying and filtering are too much different. Querying searched in the whole database including history. Filtering is only matched the current new data with user requests. We are interested in filtering.

There are two main contributions in our proposed system. First, the proposed system provides the efficient predicate based filtering for XPath queries to get exact matched information. Second, the usage of ontology in our proposed system allows users to extract semantic information from various forms of XML documents and to be effective to a large domain application.

The rest of the paper is organized as follows: Section 2 describes some existing filtering methods and comparison of the existing methods. In section 3, we present the architecture of our proposed filtering system and the main tasks of our system. In section 4, we explain NFA based model of shared value-based predicates processing. We explain how to perform query transformation processes using ontology reasoning with xml filtering system and some performance analysis are depicted in section 5. Finally, section 6 summarizes our work and present ongoing works.

2. RELATED WORKS AND PROBLEM ISSUES

We now introduce some existing XML filtering methods. The earliest method called XFilter was proposed by M. Altinel in [3]. It is a FSM based approach in which each query is converted into FSM. YFilter which is presented in [1] is a NFA-based approach for shared XPath queries. XTrie proposed in [6] indexes on sequences of elements organized in a trie the redundant matching. AFilter [4] exploits prefix and suffix commonalities the set of XPath queries. XPush [7] proposed the use of a modified deterministic pushdown automaton to simulate the execution of XPath filters and can handle predicates. XSQ [8] exploits the pushdown transducer to share the atomic predicates. This technique enables the sharing of numeric and string constants. Now, Really Simple Syndicate (RSS), an XML application, semantic information is important to deliver the user needs. To the best of our knowledge, the main motivated factors of our works are the most existing methods cannot enhance the processing efficiency of value-based predicate and cannot filter the documents for semantic matched information. So, our system will solve these problems and reduce the filtering time as much as possible. We describe some existing XML filtering methods and compare the existing methods are shown in Table 1 and Table 2.

Table 1. Some Existing XML Filtering Systems

XML filtering system	**Filtering mechanism**	**Query Language and nature**	**Data Structure Used and Additional characteristics**
XFilter	FSM	XPath simple	Query Index
YFilter	NFA/DFA	XPath share	Stack, Detection of common prefixes
XTrie	Subsequence matching	XPath complex	Substring indexing, substring sharing, ordered matching
AFilter	Stack	XPath	Exploitation of prefix and suffix commonalities, lazy techniques
ApproXFilter	Tree, Thesaurus	ApproXQL	Tree DAG (fully query or extended query)
XPush	Push-down automaton	XPath	High scalability, lazy
Ontology based Filter	Twig-pattern with Prefix Path Streaming, Ontology	XPath	Tree, Stack, Ontology

Table 2. Comparison of Some Existing Systems

XML filtering system	**Filtering mechanism**	**Accuracy**	**Filtering Time**
XFilter	Uses FSM	Only exact match result	Slowest of all because of FSM concept and no group queries
YFilter	Transform queries into a single NFA by sharing the prefix. Top-down Approach	Only exact match result	Increase as the number of branches and queries distinct
XTrie	Supports tree shaped XPath expressions containing predicate, decompose path expressions into several	Low probability of false positives due to aggregation sub-strings with a sequence of labels	Good the response time for large number of queries involving predicates but it is limited due to the heavy computation of the aggregation processes
AFilter	Prefix Caching and Suffix Clustering , Bottom-up Approach	Only exact matched result	Slower than GFilter if the depth of XML document recursively varied
ApproX Filter	Transform all queries into normalized form, extend them using synonyms, build a DAG, parse document and traverse DAG	Low probability of false negatives may occur due to the normalization of subscriptions	Relatively efficient

3. ARCHITECTURE OF PROPOSED XML FILTERING ENGINE

The overview of proposed system is shown in Figure1 and the main tasks of proposed XML filtering engine are shown in Figure 2.

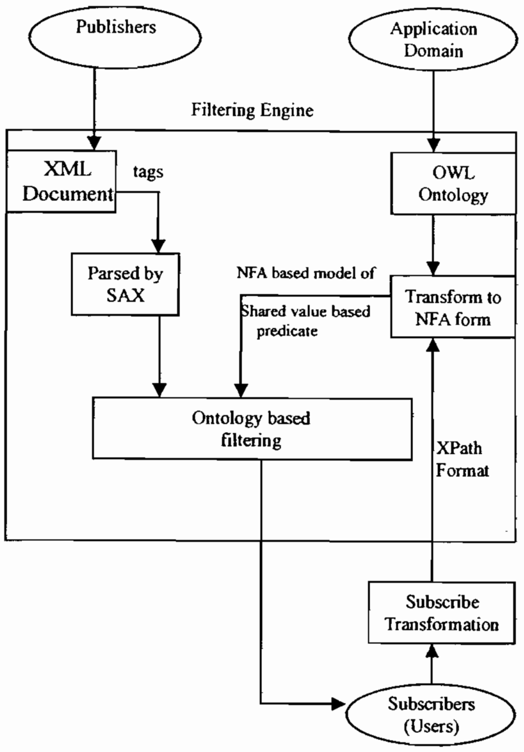

Figure 1. Architecture of Proposed XML Filtering Engine

In the proposed system, DTD is one main input to generate the XML documents and user profiles. For XML documents generation, IBM's XML Generator is used. In the filtering context, many queries representing the interests of the user community are stored and must be checked upon the arrival of a new document. The basic components of the proposed system are path generator, XPath parser, parsing XML document, queries transformation and matching. The query generator generates random query strings according to the input DTD. XPath parser takes queries written in XPath, parses them and sends the parsed profiles to the filtering engine. New profiles can be added to a running filtering engine only when the engine is not actively engaged in processing a document. The user profiles of XPath queries are transformed into Nondeterministic Finite Automaton (NFA) based model for sharing all path expressions and to enhance the processing efficiency of vale-based predicate. On the other hand, XML document is parsed by SAX parser. For building specific application domain, we applied web ontology language (OWL) which enables to express much richer relationships, thus yielding a much enhanced inference capability. And it also provides an XML vocabulary to define classes, properties and their relationships. Finally, matching XML document and NFA based model of user profile. Then, the matched results will be produced to particular users.

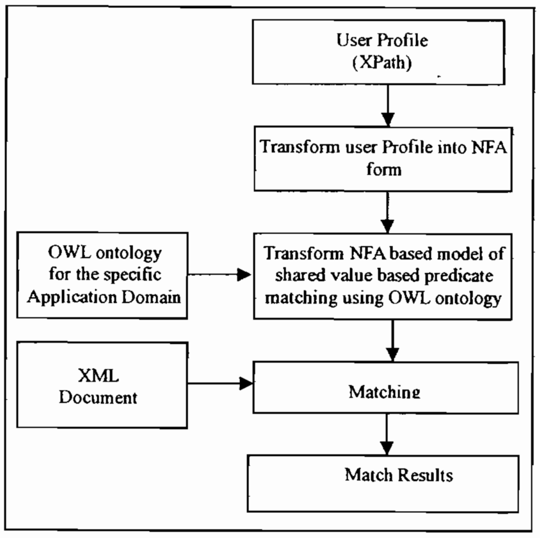
User Profile (XPath)
Transform user Profile into NFA form
OWL ontology for the specific Application Domain
Transform NFA based model of shared value based predicate matching using OWL ontology
XML Document
Matching
Match Results

Figure 2. Main tasks of Proposed XML Filtering Engine

4. NFA-BASED MODEL OF SHARED VALUE-BASED PREDICATES PROCESSING

In our proposed system, the user profiles of XPath queries including value-based predicates are transformed into NFA based model. In order to perform an efficient execution of value-based predicates, proposed system separates NFA into structure matching and value-based predicate matching. In structure matching, NFA combine all path queries into a single NFA and share the common prefixes of the paths. In value-based predicate matching, NFA identify the common prefix characters of the operand in the predicate and share the processing among them.

The execution of the value-based predicate NFA machine implemented using a hash table. The value-based predicate NFA considers a single character as an event and executes the predicate with the event-driven method. When the XML document arrives at the filtering engine, XML document is parsed by SAX parser and generating an event for each character. The generated event is sent to the handler and generates transition in the value-based predicate NFA.

In order to improve the semantic adequacy of the results of XPath queries into NFA form are transformed with the help of ontology. This step must take certain assumptions about the relationships between the terms defined in the ontology and the structure of XML documents. This mechanism in general performs two steps. First, it analyses the original NFA form. Second, it converts the NFA form into a richer NFA form that embodies ontological background knowledge which is to append into structural NFA.

In the matching stage, the value-based predicate matching is performed until after the structure matching has been completed. If the structure matching aspects are not satisfied, the evaluation of the remaining predicate matching can be avoided. And also, when a predicate of a query fails, the evaluation of the remaining predicates of that query can be avoided. If the structure matching and the remaining value-based predicate matching are satisfied, the query matches the current XML document. Figure 3 and Figure 4 show the examples of a structural NFA and a value-based predicate NFA representing five queries in Table 3.

Table 3. XPath Queries Example

Q1=/dblp/thesis/author[text()="June Brown"]
Q2=/dblp/thesis/author[text()="June Smith"]
Q3=/dblp/thesis/title
Q4=/dblp/thesis/year[text()= "2006"]
Q5=/dblp/thesis/year [text()= "2007"]

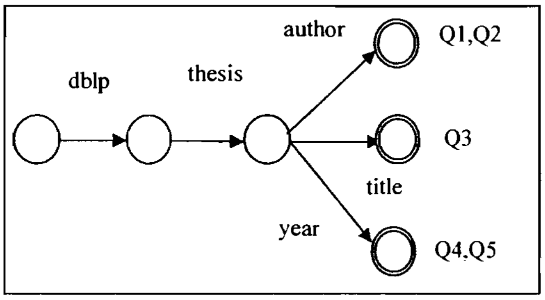

Figure 3. Structural NFA Example

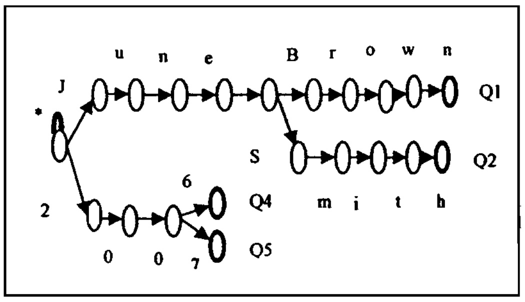

Figure 4. Value-based Predicate NFA Example

5. ONTOLOGY IN PROPOSED MECHANISM

Ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary (Neches and colleagues, 1991). OWL is the de-facto standard for ontology languages and is a W3C recommendation. OWL is an extension of RDF. OWL provides an XML vocabulary to define classes, properties and their relationships [9]. Proposed system applies OWL language for semantic matched information. Proposed system performs the query transformation processes to provide semantic information. The query transformation processes are shown in Table 4.

Table 4. Query Transformation Algorithm For Semantic

```
Input: Xpath queries
Output: Semantic NFA based model
1. Parsing owl ontology classes by OWL API
2. while queries
3. begin
4.   if the query node contains in the ontology
       classes
       then
5.   The original query is transformed into
       Semantic query
6.   Appending Semantic query into
       structural NFA queries
7. endif
8.end
9. return All NFA queries;
```

Before performing the query transformation, OWL is parsed by OWL API parser. This parser is a high level programmatic interface for accessing and manipulation OWL ontology. Main inputs of the query transformation processes are OWL ontology and user profiles of XPath queries. If the query node is in the ontology class, that query node is transformed with the subclasses of ontology. In the matching stage of the system, both the original queries and transformed queries will be processed. Then, transformed queries will be appended into structural NFA. In this way, proposed system can provide the semantic information for the user queries.

For semantic matched results, we will explain with the following example. The examples of xml document and ontology domain are shown in Table 5 and Table 6 respectively. In this example, without using ontology, there is no matched query although queries in Table 3 are the semantic matched information. With ontology, the semantic matched queries will be output. Therefore, the proposed system can provide the semantic matched results by taking the queries transformation. Matched information from XML Document will be produced.

Table 5. XML Document Example

```
<dblp>
  <masterthesis mdate="2002-01-03"
 key= "ms/Brown92">
  <author>June  Brown</author>
  <title>... </title>
  <year>... </year>
  <school>... </school>
 </masterthesis>
 <phdthesis pdate="2002-05-0"
key="ms/John3">
  <author>June  Smith</author>
  <title>... </title>
  <year>... </year>
  <school>... </school>
 </phdthesis>
 <dissertation  mdate="2002-01-03"
key="ms/Smith92">
 <author>Kurt P. Smith</author>
 <title>... </title>
     ...
</dissertation>
</dblp>
```

Table 6. OWL Example

```
<owl:Class rdf:ID="dissertation">
<rdfs:subClassOf>
<owl:Class rdf:ID="thesis"/>
</rdfs:subClassOf>
<rdfs:label>Topic</rdfs:label>
<rdfs:comment>Master Thesis</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="masterthesis">
<rdfs:subClassOf>
<owl:Class rdf:ID="thesis"/>
</rdfs:subClassOf>
<rdfs:label>Topic</rdfs:label>
<rdfs:comment>Master Thesis</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="phdthesis">
<rdfs:subClassOf>
<owl:Class rdf:ID="thesis"/>
</rdfs:subClassOf>
<rdfs:label>Topic</rdfs:label>
<rdfs:comment>PhD Thesis</rdfs:comment>
</owl:Class>
```

5.1. Dataset Specification

In fact, we are currently developing our proposed system. Therefore, in this paper, we do not present a through analysis. Here, we will present only the results of some experiments. All experiments will perform on a Windows Vista computer with 2GB of memory and Pentium Dual CPU 2.16 GHz. All codes were written in java. Our code was compiled using Eclipse 3.3 (Europa) with JVM memory 512MB. We evaluate the performance of proposed system on both real dataset (DBLP) and synthesized dataset (NITF) setup, which are described in Table 7. DBLP dataset [16] is an XML document, including information about papers, thesis, books and authors. Each paper's information is represented as a fragment of the XML document. DBLP is shallow and wide document. NITF is a DTD for News Industry Text Format. For NITF dataset, XML documents are generated from DTD by using IBM's XML generator tool. Figure 5 shows the query transformation time of XPath queries into NFA form for both dataset's queries. It is measured on varying the number of queries. Real dataset's queries take time more than synthesized dataset's when the number of queries increases.

Table 7. Dataset setup

Dataset	Data size	# of Elements	# of Attributes	Max Depth
NITF.dtd	82KB	123	513	4.17
DBLP.dtd	8KB	37	23	2.87
DBLP.xml	10KB	202	25	3
DBLP.owl	10KB	174	88	

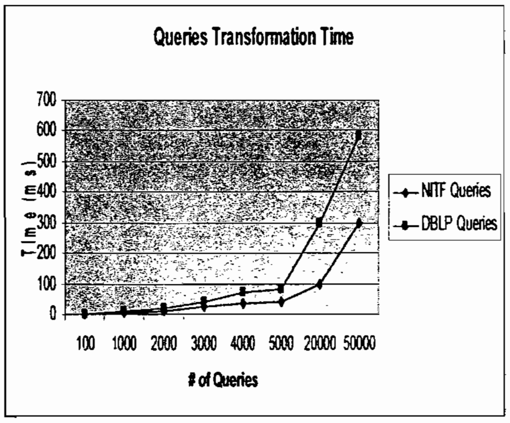

Figure 5. Queries Transformation Time

6. CONCLUSION

In this paper, we introduced a new XML document filtering system for multiple queries, which is based on ontology for getting semantic information. Our system intends to provide the exact matched information and the semantic matched information of the users' queries. Later, our filtering engine's performance analysis will be presented comparing with the existing methods. And also, we will apply the concept of proposed filtering mechanism in web application such as alerting services for digital library. Getting how much approximate, depends on ontology of a specific application domain. Therefore, our system intends to construct improved ontology domain to provide the more semantic information. We intend to use both real datasets and synthesized datasets. As parts of our ongoing work, we are going to show several performance studies such as efficiency, scalability and filtering time, how much approximate values and exact values will be outputs that our proposed method.

References:

[1] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer, "Path Sharing and Predicate Evaluation for High-Performance XML Filtering," ACM Trans. Database Systems, Vol. 28, Issue 4, pp. 467-516,2003.

[2] C. Chan, P. Felber, M. Garofalakis, and R. Rastogi, "Efficient Filtering of XML Documents with XPath Expressions," In Proc. IEEE Int. Conf. Data Engineering, pp. 235, 2002.

[3] Mehmet Altinel and Michael J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In Proceedings of the 26th VLDB Conference, pages 53–64, Cairo, Egypt, September 2000.

[4] K. Selc,uk Candan, Wang-Pin Hsiung, Songting Chen,Jun'ichi Tatemura and Divyakant Agrawal. AFilter: adaptable XML filtering with prefix-caching suffix-clustering. In *Proceedings of the 32nd VLDB Conference*, pages 559–570, Seoul, Korea, Sept. 2006.

[5] J. Kwon, P. Rao, B. Moon, S. Lee, FiST: scalable XML document filtering by sequencing twig patterns, in: Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005, pp. 217–228.

[6] B. Ludäscher, P. Mukhopadhyay, Y. apakonstantinou, A transducer-based XML query processor, in: Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002, pp. 227–238.

[7] A.K. Gupta, D. Suciu, Stream processing of XPath queries with predicates, in: Proceedings of the 2003 ACM-SIGMOD Conference, ACM Press, San Diego, CA, 2003, pp. 419–430.

[8] T. R. Gruber: A Translation Approach to Portable Ontology Specifications. in: Knowledge Acquisition. vol. 6, no. 2, 1993. pp199-221

[9] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology".

[10] Y. Diao, H. Zhang and M. J. Franklin, "NFA-based Filtering for Efficient and Scalable XML Routing".

[11] O. Corby, R. Dieng-Kuntz, C. Faron-Zucker and F. Gandon, "Ontology-based Approximate Query Processing for Searching the Semantic Web with Corese", INRIA, July 2005.

[13] T. S. Li, T. Han and G. N. Chen, "XML Query based on Ontology", IEEE Intelligent Informatics Bulletin vol.6 No.2, Nov 2005.

[14] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language W3C Working Draft. Technical Report WD-xquery-20050404, World Wide Web Consortium.

[15] D. Megginson. SAX: A Free API for Event-based XML Parsing. Available: http://www.saxproject.org, 2005.

[16] BLIP XML records. http://dblp.uni-trier.de/xml/ http://www.acm.org/sigmod/dblp/db/about/dblp.dtd

[17] S. Chen, H. G. Li and J. Tatemura, GFilter: "Scalable Filtering of Multiple Generalized-Tree-Pattern Queries over XML Streams", IEEE Transactions on Knowledge and Data Engineering 2008.