Value-Based Predicate XML Filtering System

Yi Mon Thet
University Of Computer Studies, Yangon, Myanmar
yimonthet.ucsy@gmail.com

Abstract

Nowadays, information dissemination applications are very popular and much of the data exchange over the Internet. Existing XML filtering techniques based on a publish/subscribe model on highly structured data marked up with XML tags. These techniques are efficient in filtering the documents of data centric XML but are not effective in filtering value-based predicates of document centric XML. In this paper, we propose a technique which does semantic matching of XML data and also handles value-based predicates. User Profiles are specified as XPath Twig queries. A query node is checked in OWL classes, if a node is found it is returned with its semantically related data. On the other hand, incoming XML document is parsed by SAX parser and a tree is built. After then matching XML document and user profiles. Therefore, the proposed method intends to provide not only exact match information but also semantic matched information from XML documents.

Keywords: XML document, value-based predicate, publish/subscribe model.

1. Introduction

XML has been used extensively in many applications as a de facto standard for information representatation and exchange over the internet. Publish-Subscribe system based on XML documents are evolving. XML document filtering plays an important role in Internet applications by enabling selective dissemination o f information [15]. In a typical publish-subscribe (pub-sub) system, whenever new content is produced, it is selectively delivered to interested subscribers. This has enabled new services such as alerting and notification services for user interested. There are many filtering mechanisms exist, they can provide semantic data and match the structure but most of these mechanisms can support value-based predicate processing especially equality operator. These existing mechanisms cannot or limit support for value-based predicate processing such as logical operators.

In this paper, we propose a method for value-based predicate processing (logical operators). In propose method, a user profiled specifies his/her interest through he/she subscribes. These subscriptions are then converted into XPath queries. A query node is checked in an ontology class, if the node is found in the class, then semantically related data i.e its sibling nodes are returned. These queries are then transformed into twig patterns. On the other hand, XML documents are parsed by SAX parser. SAX is the simple API for XML, originally a Java-only API. The parser converts the XML document into a tree structure. Matching of twig node and tree node takes place and only the matched information is delivered or displayed to the user. The value-based predicates are handled differently according to the operators in the twig patterns. We propose a method to provide logical operators such as AND, OR and NOT operators.

The key contribution of this paper is summarized as follows:

The most existing filtering methods focus on structure matching. There are some methods that focus on structure matching as well as value-based predicate selections, but they can provide value-based predicate such as equality operator. In real world, there are many queries with various operators such as equality operator, non equality operators, logical operators etc. In this paper, we propose a value-based predicate filtering method (divide and conquer strategy) for processing logical operators such as OR and NOT. In this approach, the first divide a twig query with OR predicate into multiple twig queries without OR predicates and then combines their intermediate results to get final results. For AND predicate, we divide twig query at AND node and processing individually and then combine their intermediate results in order to get final result. To evaluate twig query with NOT predicate is to divide it into multiple simple path queries (without not predicate) and evaluate each of the path queries individually and final result is derived by combing theirs individual results. For getting semantic, we follow the existing XML Filtering framework.

The rest of this paper is organized as follows. In the next section, we discuss the related papers with our work. In section 3, we present background of ontology and XML. We explain how to perform semantic query transformation processes using ontology with xml filtering system is depicted in section 4. Finally, section 5 we conclude our paper.

2. Related Work

We introduce some existing XML filtering methods. The earliest work of XML filtering was XFilter proposed by M. Altinel and M.J, Franklin in [9]. The XFilter engine was a Finite State Machine which used a sophisticated index structure and a modified finite state machine approach to quickly locate and examine relevant profiles. Y.Dia et al. [14] pointed out the weakness of XFilter that maked no attempt to eliminate redundant processing for similar queries. To address this problem, all path queries were represented as a single NFA and shared the common prefixes of the paths. In addition, they also proposed two methods for value-based predicate processing: Inline and Selection Postponed (SP). Inline performed early predicate evaluation before knowing if the structure was matched, and this early predicate evaluation did not prune future work. In contrast, SP performed structure matching to prune the set of queries for which predicate evaluation needed to be considered. C.Y. Chan et al. [2] proposed a novel index structure, called XTrie that provided the efficient filtering of XML documents based on XPath expressions (XPE). They also described the XPE decompositions and matching algorithms. In their approach, they firstly derived each path queries into sub-strings and indexed sub-strings by a trie-based data structure. This method could support both ordered and unordered matching of XML data. K. S. Candan, et al. [8] developed Adaptable XML Filtering, namely AFilter, with prefix-caching and suffix-clustering. Prefix catching was used to eliminate the redundant traversals of the StackBranch pointers. XPush [1] proposed the use of a modified deterministic pushdown automaton to simulate the execution of XPath filters and could handle predicates. XSQ [13] exploited the pushdown transducer to share the atomic predicates. This technique enabled the sharing of numeric and string constants. GFilter proposed in [11] was based on a novel Tree-of-Path (TOP) encoding scheme, which compactly represented the path matches for the entire documents. TOP encodings could be efficiently produced via shared bottom-up patch matching. MFilter proposed in[4] system not only improved filtering time by transforming

the multiple queries into prefix tree with node relation lists for the parent-child or ancestor-descendent relationship of query's elements but also provided semantic data by using ontology. Efficient processing of XML Twig Queries with OR-Predicates was proposed by H. Jiang, H. Lu and W. Wang [5]. They presented a merge-based algorithm for sorted XML data and an index-based algorithm for indexed XML data. They also presented that using indexes could significantly improve the performance for matching twig queries with OR-predicates, especially when the queries had large inputs but relatively small outputs.

3. BACKGROUND

Web Ontology Language (OWL) is used for semantic matching. 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). In recent years the development of ontologies has been moving from the realm of Artificial-Intelligence laboratories to the desktops of domain experts. Ontologies have become common on the World-Wide Web.

Ontologies are usually expressed in a logical-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. The element required for the semantic web is the web ontology language (OWL), which can formally describe the semantics of classes and properties used in Web documents. It is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL adds more vocabulary for describing properties and classes, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes.

An XML database is essentially a tree database. Accordingly, XML queries specify tree-shaped search patterns, called twig patterns, which may be accompanied by additional predicates imposed on the contents or attribute values of the data tree node, XML queries are thus called twig queries.

User profiles can also have value-based predicates. Examples of value-based predicate are given in following figures.

Q1: // market/stock [code="IBM"]

Figure 1: Query with equality based predicates

Q2: //market/stock[sell price>25]

Figure 2: Query with non-equality operators

Q3: //market/stock [(code="IBM" and sell price>25) or code="HP"]

Figure 3: Query with logical OR operator

The use of value-based predicate is that a user can also specify exact information he/she needs from the system. Existing filtering approaches can be broadly classified into the following categories namely (1) Automaton-based approaches, (2) Sequence-based approaches, (3) Stack-based approaches and (4) Other approaches. The existing XML filtering systems can be categorized as follows:

Table1. Existing XML Filtering Systems

Filtering System	Filtering Name	Filtering Mechanism	Characteristics	Twig Support
Automata-based System	XFilter	FSM based	-	No

Automata-based System	YFilter	NFA/DFA	Detection of Common Prefix	Yes
Sequence-based System	FiST	Subsequence Matching	Ordered Matching	Yes
Stack-based System	AFilter	Stack	Exploitation of Prefix-Suffix commanalit-ies	No
Other Approach-es	XTrie	Indexing	Ordered Matching, Substring Indexing	Yes

4. FILTERING MECHANISM

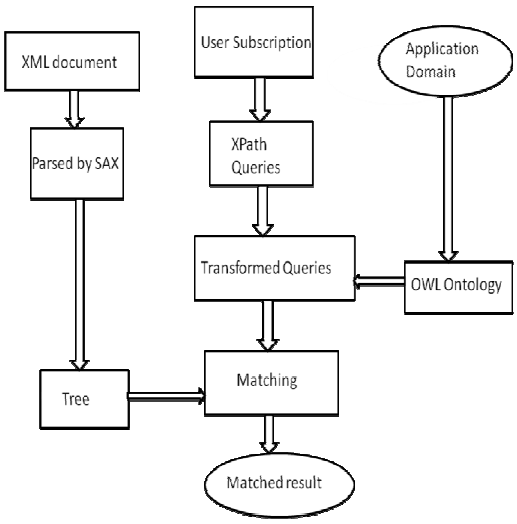

Figure 1: Architecture of Filtering Engine

In our proposed system, user specifies interest by logging into the system and then subscribing for the content. The user interests are then converted into XPath queries. The query is checked in the ontology (OWL) class and if it is present, its sibling elements are returned by using class-subclass relationship of the OWL class. Thus a single query gets converted into multiple transformed queries. The queries are then converted into twig pattern. On the other hand, XML document is parsed by SAX parser and built a tree. Finally, matching twig node and tree node is performed. Then matching results (semantic and exact matched results) are then produced to particular users.

5. Conclusion

In this paper, we present a XML document filtering system for multiple queries, which is based on ontology for getting semantic information. By using predicate user queries and ontology, our system provides exact matched information and the semantic matched information.

6. References

[1] 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.

[2] B. Ludascher, P. Mukhopadhay, Y. Apakonstantinou, A transducer-based XML query processor, in: Proceedings of the 28th VLDB Conference, Hong Kong, China 2002,pp. 227-238.

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

[4] E. Chaw Htoon, Thi Thi Soe Nyunt. 2009. M-Filter: Semantic XML Data Filtering System for Multiple Queries. Eight IEEE/ACIS International Conference on Computer and Information Science (1-3June2009), 1167-1171. DOI=10.1109/ICS.2009.

[5] H. Jiang, H. Lu and W. Wang, " Efficient Processing of XML Twig Queries with OR-Predicates ", 2004.2

[6] J. Kwon, P. RaoB. Moon, S. Lee (2005) FiST: scalable XML document filtering by

sequencing twig patterns, in: Proceeding of the 31st VLDB Conference, Trondheim, Norway, pp. 217-228

[7] J. Kwon, P. Rao, B. Moon, S. Lee. 2007. Value-based Predicate Filtering of Streaming XML Data. International Conference on Multimedia and Ubiquitous Engineering, (2007), 289-293 DOI=10.1109/MUE.2007.216.

[8] K. Selcuk Candan, Wang-Pin Hsiung, Songting Chen, Jun'ichi Tatemura and Divyakant Agrawal (2006). AFilter: Adaptable XML Filtering with Prefix-Caching and Suffix-Clustering, ACM.

[9] M. Altinel and M.J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In VLDB, 53-64,2002.s

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

[10]P. Silvasti, "XML Document Filtering Automaton", VLDBA, ACM, New Zealand, 2008, pp.1666-1671.

[11] 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.

[12] S.R. Cho, "T-SIX: An Indexing System for XML Siblings", 8th International Workshop on Web and Databases, Maryland, June, 2005.

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

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

[15] Y. Diao, M. J. Franklin (2008). XML PUBLISH/SUBSCRIBE.

[16] Y.Diao, H. Zhang, M.J. Franklin, "NFA-based Filtering for Efficient and Scalable XML routing." 2002.