Structural Summary Tree with Complete Path Representation for Intelligent Query of Social-Web Documents

Hsu-Kuang Chang and King-Chu Hung

***Abstract*—Compiling documents in extensible markup language (XML) increasingly requires access to data services which provide both rapid response and the precise use of search engines. Efficient data service should be based on a skillful representation that can support low complexity and high precision search capabilities. In this paper, a novel complete path representation (CPR) associated with a modified inverted index is presented for the provision of efficient XML data services, where queries can be versatile in terms of predicates. CPR can completely preserve hierarchical information, and the new index is used to save semantic information. The CPR approach can provide template-based indexing for fast data search. An experiment is also conducted for the evaluation of the CPR approach.**

***Index Terms*—XML, DTD, complete path representation (CPR), structural summary tree (SST), versatile query.**

I. Introduction

By recommendation of the World Wide Web Consortium (W3C) [1], XML has served as a standard information description language widely used in computer communication. This facility demands skillful XML data representation for fast internet searches [2]. Since XML data can be uniquely described with a semantically structured tree (i.e., a hierarchical structure associated with relationships among nodes), both structure and semantics are significant elements of XML data representation and feature selection.

XML data representations can be categorized into string [3], [4] and path [5]-[15] groups. String representation can be derived with a preorder traversal algorithm; it requires dynamic programming for edit distance measurement. This approach, with its lack of structure information, may lead to indeterminate search results. Path approaches use sub-paths as the features, and represent each XML datum as a binary vector. An element in the binary vector denotes whether the datum involves a corresponding feature, where such features can be defined as tree nodes [6], two-node sub-paths (i.e., node-pair (NP)) [7], [8], or whole paths (WP) [9], [10]. To improve search efficiency, several path representation modifications have been proposed. Yang *et al*. [11] use the content instead of the leaf node for node representation. Liu *et al.* [12] present a hybrid definition that combines NP and WP for XML data description. Based on the determined finite automata, Mustafa *et al*. [13] and Lee *et al*. [14] present a path-embedded string representation. In order to improve the efficiency of common Xpath and principal component analysis, Li [15] presents a modified WP with limited-length paths. These approaches essentially only concern the service of simple queries (single path queries). Recently, based on XML QBE [16], XQuery [17], PathStack and TwigStack [18], and query pattern tree (QPT) [19], several query parsers [20]-[23] have been developed to facilitate compound-type queries, which produce versatile multi-path queries in terms of predicates. Thirteen compound-type queries are also designed in order to evaluate the performance of this CPR approach.

This paper is organized as follows: In Section II, the CPR is proposed for XML data description. In Section III, the modified inverted index is described. Section IV shows the performance evaluation results of handling versatile queries by using variant approaches. Finally, conclusions are drawn in Section V.

II. XML Data Representation Using Complete Path Elements

Any XML datum defined with the document type definition (DTD) can be modeled as an ordered label tree [3]. In this section, the hierarchical tree information is extracted by a pre-ordered traversal process performed with a document object model (DOM) API [1]. Following this, the CPE extraction based on the SSTs of XML data is described and the CPR is presented.

A. The Extraction of Structural Summary Tree of XML Document

SST is the XML tree skeleton commonly used for XML data representation [3]. The SST of an XML document can be extracted by four functional processes, as follows:

Step 1. This step performs a tree conversion using Java DOM (JDOM), where the tree element values are neglected. For the DTD-formatted XML datum of Example 1, the tree conversion process result is illustrated in Fig. 1.

Step 2. For efficient matching, we symbolize the name of the tree node with an abbreviated character order,

Manuscript received January 12, 2013; revised March 25, 2013.

This work was supported in part by the National Science Council of Taiwan under Grant NSC 101-2221-E-327-040.

H. K. Chang is with the Department of Information Engineering, I-Shou University, Taiwan, R. O. C. (e-mail: hkchang@isu.edu.tw).

K. C. Hung is with Institute of Engineering Science and Technology, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, R. O. C. (e-mail: kchung@nkfust. edu.tw).

as shown in Fig. 2.

Step 3. Based on the pre-order traversal process [3], SST extraction requires two simplification procedures:

a) For each node, examine whether the current node's name is equal to an ancestor's name. If it is, set the current node's sub-tree to be a child of the ancestor; otherwise, check the next node. The purpose of this procedure is to remove nested sub-paths, as shown in Fig. 3.

b) Exhaustive searching based on a Hash table is applied for discovering and eliminating repeated branches. Fig. 4 shows the repeated branch elimination result where the simplified tree is the SST.

Step 4. For the extraction of CPEs, it is necessary to construct the adjacent-linked (AL) lists of the SSTs of all XML data. An AL list is a data structure that records the linking information of each node and facilitates the pre-order traversal process. The AL list of the SST in Fig. 4 is given in Table I where $\delta_i[n]$ denotes the nth head node of the ith XML datum.

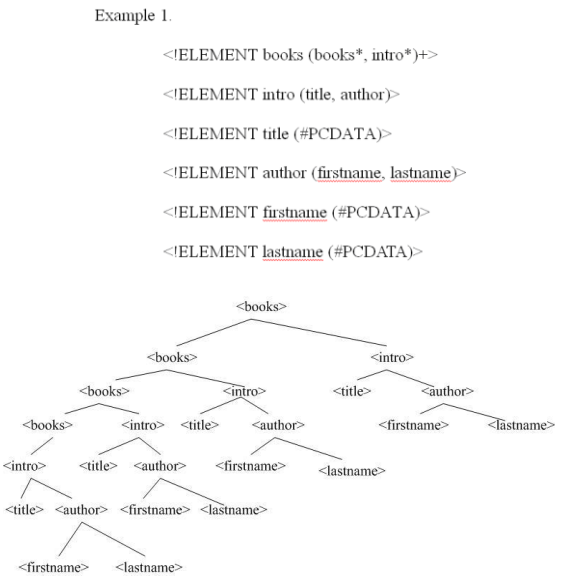


Fig. 1. The tree representation of Example 1 based on the JDOM.

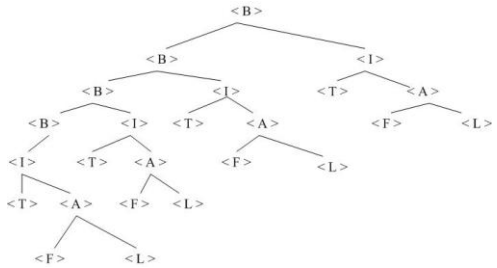


Fig. 2. The symbolization result of the XML tree of Example 1.

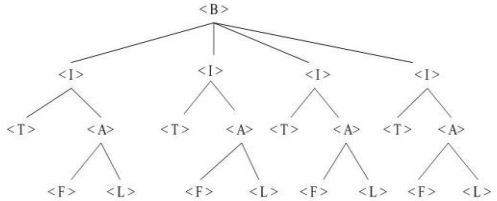


Fig. 3. The XML tree in Example 1 with nested nodes removed.

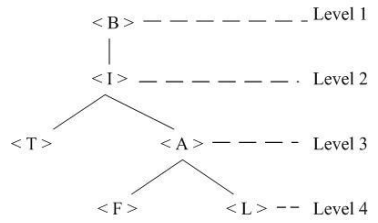


Fig. 4. The SST of Example 1 where the level of root is defined as 1 and increased towards the leaves.

TABLE I: THE AL LIST OF THE SST OF EXAMPLE 1

$\delta_i[0]$	$\delta_i[1]$	$\delta_i[2]$	$\delta_i[3]$	$\delta_i[4]$	$\delta_i[5]$
<B>	<I>	<T>	<A>	<F>	<L>
↓	↓	↓	↓	↓	↓
<I>	<T>	Nil	<F>	Nil	Nil
	↓		↓		
	<A>		<L>		

B. Complete Path Element Extraction and Representation

An example of level definition is shown in Fig. 4, where four *CP* sets: $CP_{L\text{-}1}$, $CP_{L\text{-}2}$, $CP_{L\text{-}3}$, and $CP_{L\text{-}4}$, can be defined. The elements of the four CP sets are shown in Table II.

TABLE II: THE FOUR LEVEL COMPLETE PATHS OF THE SST OF EXAMPLE 1

Four CP sets	CP elements							
$CP_{L\text{-}1}$	/B/I/T	/B/I/A/F	/B/I/A/L	/B/I/A/~	/B/I/~/~	/B/I/~	/B/~/~/~	/B/~/~
$CP_{L\text{-}2}$	/~/I/T	/~/I/A/F	/~/I/A/L	/~/I/A/~	/~/I/~/~	/~/I/~		
$CP_{L\text{-}3}$	/~/~/A/F	/~/~/A/L	/~/~/T	/~/~/A/~				
$CP_{L\text{-}4}$	/~/~/~/F	/~/~/~/L						

Considering a database comprised of the three XML data shown in Fig. 5, the CPR can be found in Table III. In Fig. 5, there are two I nodes for both DOC 1 and 3. The two nodes with different children are distinct and cannot be merged. The two sub-paths /B/I/T in DOC 2 and /B/I/T/~ in DOC 1 have the same path length equal to 3, but have distinct distances from leaf node. The same distinctions also exist between the two elements /M/I/~ and /M/I/~/~ in the $CP_{L\text{-}1}$.

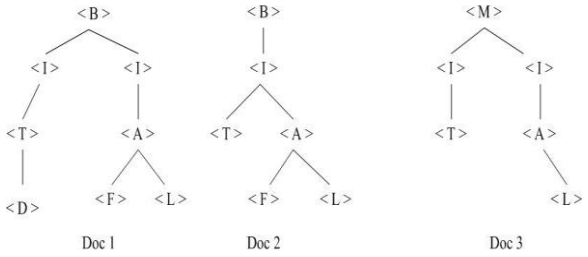


Fig. 5. The SSTs of three XML documents.

TABLE III: THE CPR FOR THE DESCRIPTION OF THE THREE XML DATA SHOWN IN FIG. 5

CPs	complete path elements															
$CP_{L\text{-}1}$	/B/I/T /D	/B/I/A /F	/B/I/A /L	/M/I/A/L	/B/I/T /~	/B/I/A /~	/B/I/~ /~	/B/I/~/~	/M/I/T	/M/~/~	/M/I/~	/M/I/A/~	/M/I/~/~	/M/~/~/~	/B/I/~	/B/I/T
$CP_{L\text{-}2}$	/~/I/T /D	/~/I/A /F	/~/I/A /L	/~/I/T /~	/~/I/A /~	/~/I/~ /~	/~/I/T									
$CP_{L\text{-}3}$	/~/~/T/D	/~/~/A/F	/~/~/A/L	/~/~/T/~	/~/~/A/~	/~/~/T										
$CP_{L\text{-}4}$	/~/~/~/D	/~/~/~/F	/~/~/~/L													

III. INDEXING THE COMPLETE PATH ELEMENTS

A CPE with the tree characteristic is a high dimensional feature. Traditional *B*-tree indexing [24, 27, 28] based on node relationships is suitable for WP, NP and twig queries,

but is inefficient for CPR, which regards each CPE as a feature element. In this section, a new index with feature similarity structure (FSS) is presented for CPE management.

The CPEs of Table III can be represented with a tree structure, as shown in Fig. 6, where *Pi* denotes a CPE subset with path length equal to *i*. The CPEs in Fig. 6 are inherent with the hierarchical information involving path length (*Pi*) and level (CP_{L-1}) that are available for inferring semantic relations, e.g., AD, SB and CN relationships. A B-tree index with a key design can achieve a balanced binary tree structure for efficiently indexing the elements of NP and WP, but cannot provide hierarchical information. To facilitate the inference of semantic information, the inverted index structure with additional fields is applied for CPE indexing.

The FSS with feature similarity provides a template-based hierarchical query service. This service method can effectively reduce the searching complexity induced by the path element increment of CPR, compared to that of the NP and WP. Utilizing the one-to-one property of ρ_i, XML documents can be uniquely described with a feature vector (FV), defined as

$$FV_{DOC} = [\rho_0, \rho_1, \ldots, \rho_{N-1}], \rho_0 \in \{0,1\} \quad (1)$$

where ρ_i denotes the label of *i*th CPE, and *N* denotes the total number of CPEs. The element with $\rho_i = 1$ implies that the document involves the *i*th labeled CPE. This labeling provides a template-based hierarchical query service. Let *CPsT* (*l*, *i*) be an indexing template involving the CPEs of CP_{L-l} and P_i. An indexing template with (*l*, *i*) = (1, 1) can be defined with:

$$CPsT(1,1) = \begin{bmatrix} SW & 0 & 0 & 0 & 0 \\ \rho_{\#} & \rho_0 & \rho_1 & \rho_2 & \rho_3 \end{bmatrix}_{P_1}^{CP_{L-1}}$$

where SW denotes a switch. Setting a field of SW to one indicates that the corresponding CPE is selected. For example, an indexing template defined by:

$$CPsT(1,4) = \begin{bmatrix} SW & 1 & 0 & 1 & 1 \\ \rho_{\#} & \rho_{13} & \rho_{14} & \rho_{15} & \rho_{16} \end{bmatrix}_{P_4}^{CP_{L-1}}$$

will yield a response as:

ρ_{13} = */B/I/T/D* in Doc1
ρ_{15} = */B/I/A/L* in Doc1 and Doc2
ρ_{16} = */M/I/A/L* in Doc3.

Like the Region [22] and Dewey [26] methods, the CPE index can be easily updated with numerical labeling Updating the Dewey method is based on the extended Dewey labeling [25] [27], [28] which uses modular function to reserve even numbers for the insertion of new path elements.

The FSS with path length and level also allows the inference of semantic information. The path elements with AD relationships can be easily obtained from the CPEs with the path length field filled in *Pi* for $i \geq 3$, i.e., path length ≥ 3. For the example of Fig. 5, there are two kinds of AD relationship where A1 involves the path elements with one-generation AD, and A2 involves the path elements with two-generation AD. Note that these path elements are different from CPE, and are labeled with $\delta_0 \sim \delta_{11}$. SB and CN are relations among nodes, where these nodes have different descendants, but have the same father and grandfather node, respectively. For SB, the father nodes can be found in levels CP_{L-l} for $1 \leq l \leq L-1$. Furthermore, the search of CN nodes aims to verify whether their father nodes are inherent with a SB relationship. The tree structure index, including semantic information, is illustrated in Fig. 7, where SB and CN indexing requires fewer levels than the indexing of AD.

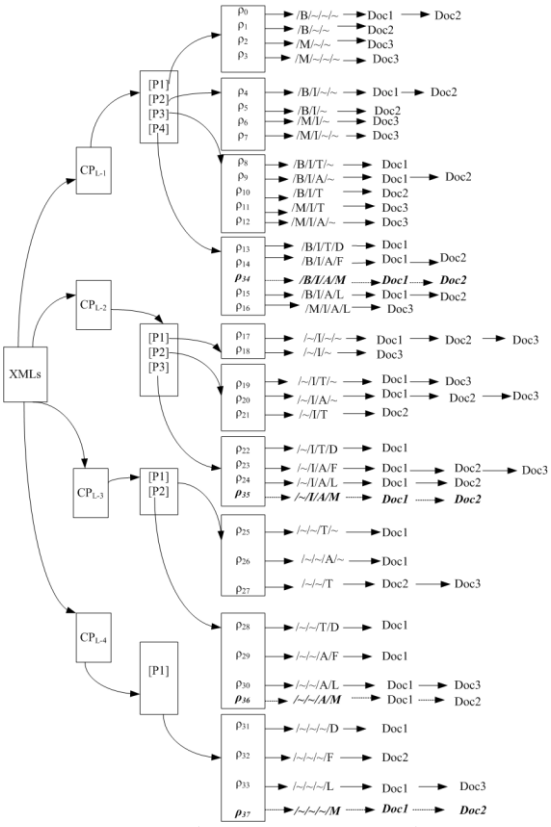


Fig. 6. The index structure of the tree representation of Fig. 5. Italic is a new path inserted.

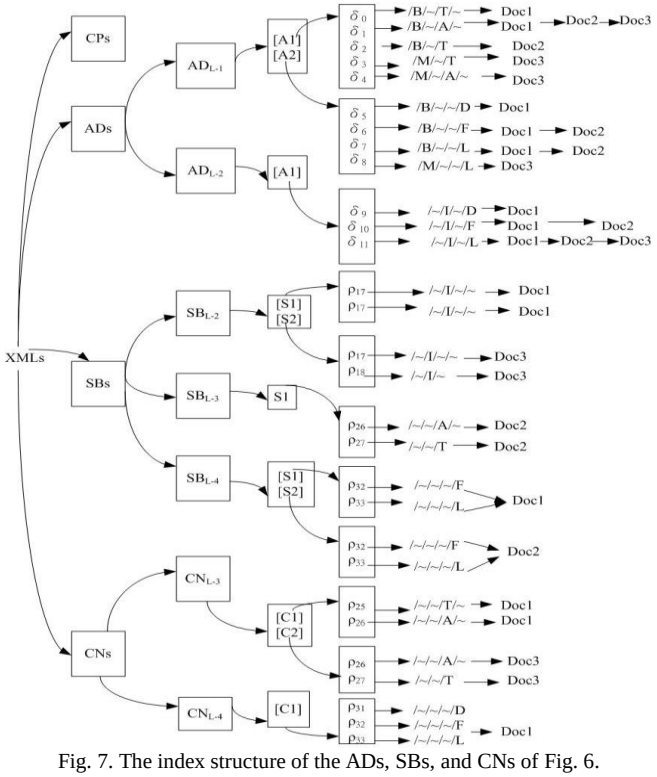


Fig. 7. The index structure of the ADs, SBs, and CNs of Fig. 6.

IV. Experiment

For the data service efficiency analysis of CPR, an experiment using the simple dataset of Fig. 5 was performed. In this dataset, WP and NP have 6 and 10 feature elements respectively. For CPR, the feature elements of CPE and AD relation are 34 and 13, respectively. Some queries shown in Table IV are designed for the simulation of versatile client requests. These queries can be categorized into CPE (TPQ_1~TPQ_8), AD (TPQ_9~TPQ_A), and SB&CN (TPQ_B~TPQ_D) groups, where TPQ_1~TPQ_5 belong to WP and NP types. TPQ_D is special due to the distinct I nodes. Decoded with the query parser [23], these statements can be translated into compound tree-pattern queries. Two commonly used indices: searching complexity and accuracy, are applied for performance evaluation. The searching complexity (SC) is defined with the total checking times required for matching all of the query paths. Here, we suppose that all of the path elements (in dataset) fitting query conditions should be checked in each query path matching. For TPQ_1, there are four query paths with level=1 and path length=4. The level and path length determine the selection of the query template: $CPsT(1,4)$, where four path elements: ρ_{13}~ρ_{16}, satisfy the conditions. Considering exhaustive matching, each query path should be matched four times. Thus the query service of CPR requires a complexity of SC=4×4=16, and the SW fields of ρ_{13}~ρ_{16} will be set to 1:

$$CPsT(1,\ 4)=\begin{bmatrix} SW & 1 & 1 & 1 & 1 \\ \rho_{\#} & \rho_{13} & \rho_{14} & \rho_{15} & \rho_{16} \end{bmatrix}_{P_4}^{CP_{L-1}}.$$

The complexities required for serving TPQ_2~TPQ_9 are evaluated in Table V, where the symbol '-' denotes that this representation method cannot serve the query. For TPQ_A, there are three 1-level query paths involving two one-generation AD and one two-generation AD. The level and AD relations determine the selection of two query templates: *ADsT(1,1)* and *ADsT(1,2)*, where the former has five elements (δ_0~δ_4), and the latter has four elements (δ_5~δ_8). Also considering exhaustive search, the SC of TPQ_A can be found as SC = 5 * 2 + 4 = 14. The query templates are set by:

$$ADsT(1,1)=\begin{bmatrix} SW & 0 & 1 & 1 & 0 & 0 \\ \delta_{\#} & \delta_0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix}_{A_1}^{AD_{L-1}},$$

$$ADsT(1,2)=\begin{bmatrix} SW & 1 & 0 & 0 & 0 \\ \delta_{\#} & \delta_5 & \delta_6 & \delta_7 & \delta_8 \end{bmatrix}_{A_2}^{AD_{L-1}}.$$

For $TPQ_{B\sim D}$, the level and semantic relations will determine the selection of the three query templates: *SBsT(3)*, *CNsT(3)*, and *SBsT(2)*. By using exhaustive search, the SC of the three queries can be found as SC = 4(2×2), 9(3×3), and 4(2×2) respectively. The query templates are set by:

$$SBsT(3)=\begin{bmatrix} SW & 1 & 1 \\ \rho_{\#} & \rho_{26} & \rho_{27} \end{bmatrix}_{}^{SB_{L-3}},$$

$$CNsT(3)=\begin{bmatrix} SW & 1 & 1 & 1 \\ \rho_{\#} & \rho_{25} & \rho_{26} & \rho_{27} \end{bmatrix}_{}^{CN_{L-3}}, \text{ and}$$

$$SBsT(2)=\begin{bmatrix} SW & 1 & 1 \\ \rho_{\#} & \rho_{17} & \rho_{18} \end{bmatrix}_{}^{SB_{L-2}}.$$

TABLE IV: Some Queries for the Simulation of Versatile Client Requests

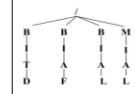
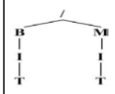
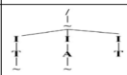
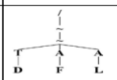
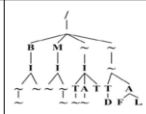
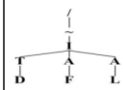
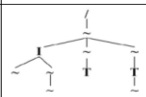
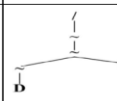
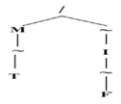
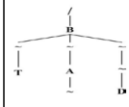
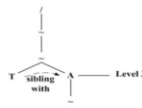
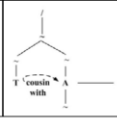
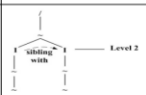
Query	Description	Tree Pattern Query	Query	Description	Tree Pattern Query
TPQ_1	Find documents with four 4-length whole paths.		TPQ_2	Find documents with two 3-length whole paths.	
TPQ_3	Find documents with three level 2 NPs.		TPQ_4	Find documents with three level 3 NPs.	
TPQ_5	Find documents with the NPs in all levels.		TPQ_6	Find documents with three 3-length path elements in level 2.	
TPQ_7	Find documents with nodes I and T in level 2 and 3.		TPQ_8	Find documents with leaf nodes T and D.	
TPQ_9	Find documents with M being ancestor of T and I being ancestor of F.		TPQ_A	Find documents with B being ancestor of T, A, and D.	
TPQ_B	Find documents with T and A being sibling on level 3.		TPQ_C	Find documents with T and A being cousin on level 3.	
TPQ_D	Find documents with I and I being sibling on level 2.				

TABLE V: A Comparison of the XML Data Service Performances of WP, NP, and CPR Approaches for the Queries Given in Table IV

Tree Pattern Queries	SC			SA		
	WP	NP	CPR	WP	NP	CPR
TPQ_1	16	-	16	S	F	S
TPQ_2	10	-	10	S	F	S
TPQ_3	-	9	9	F	S	S
TPQ_4	-	9	9	F	S	S
TPQ_5	-	34	34	F	S	S
TPQ_6	-	-	9	F	F	S
TPQ_7	-	-	10	F	F	S
TPQ_8	-	-	6	F	F	S
TPQ_9	-	-	8	F	F	S
TPQ_A	-	-	14	F	F	S
TPQ_B	-	-	4	F	F	S
TPQ_C	-	-	9	F	F	S
TPQ_D	-	-	4	F	F	S

Searching accuracy (SA) is defined with two bi-levels: Success and Fail, indicating whether or not the document can be found. With WP and NP element queries, the documents satisfying the conditions of TPQ_1~TPQ_5 can easily be retrieved for the WP and NP approaches respectively. For queries TPQ_6~TPQ_8 that request sub-paths starting from different levels, neither NP nor WP can handle these queries due to a lack of level information. The experiment clearly shows that NP and WP are subsets of the CPR. Nevertheless, with hierarchical template search, the increased feature elements do not reduce the searching efficiency of CPR at all. With the semantic relation inference capability, CPR can also easily serve the queries with inherent AD, SB and CN relationships. The SC of TPQ_9 and TPQ_D are shown in Table V. However, neither WP nor NP can handle these queries due to a lack of level and path length information.

V. Conclusion

In this paper, a new XML data representation called CPR is presented as a means of providing an efficient and versatile query service. CPR uses complete path elements as XML data description features. In association with a modified inverted index, the CPR approach can preserve both structure and semantic information, as well as provide a template-based indexing for fast XML data search. Performance evaluation results show that the CPR can be an efficient kernel for XML data service.

References

[1] The document object model. [Online]. Available: http://www.w3.org/DOM/

[2] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "A methodology for clustering XML documents by structure," *Information Systems*, vol. 31, no. 3, pp. 187-228, 2006.

[3] T. Dalamagas *et al.*, "Clustering XML Documents using Structural Summaries," *EDBT Work-shop on Clustering Information over the Web* (*ClustWeb04*), Heraklion, Greece, 2004.

[4] A. Nierman and H. V. Jagadish, "Evaluating structural similarity in XML documents," presented at Fifth International Workshop on the Web and Databases (WebDB 2002).

[5] S. Flesca *et al.*, "Fast detection of XML structural similarity," *IEEE Trans. on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 160-175, February 2004.

[6] W. Lian *et al.*, "An efficient and scalable algorithm for clustering XML documents by structure," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 82-96, January 2004.

[7] M. Kozielski, "Improving the results and performance of clustering bit-encoded XML documents," presented at Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06).

[8] J.-S. Yuan, X.-Y. Li, and L.-N. Ma, "An improved XML document clustering using path feature," presented at Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 2, pp. 400-404, 2008.

[9] H.-P. Leung, F.-L. Chung, S. C. F. Chan, and R. Luk, "XML document clustering using common Xpath," in *Proc. the International Workshop on Challenges in Web Information Retrieval and Integration*, Tokyo, April 2005, pp. 91-96.

[10] A. Termier, M.-C. Rousset, and M. Sebag, "Treefinder: a first step towards XML data mining," in *Proc. IEEE International Conf. on Data Mining*, Maebashi, December 2002, pp. 450-457.

[11] J.-W. Yang, W.-K. Cheung, and X.-O. Chen, "Learning the kernel matrix for XML document clustering," in *Proc. the 2005 IEEE International Conf. on e-Technology, e-Commerce and e-Service* (*EEE'05*), Hong Kong, April 2005, pp. 353-358.

[12] J.-H. Liu, J. T. L. Wang, W. Hsu, and K. G. Herbert, "XML clustering by principal component analysis," in *Proc. the 16th IEEE International Conf. on Tools with Artificial Intelligence* (*ICTAI'04*), Boca Raton, November 2004, pp. 658-662.

[13] J.-W. Lee, K. Lee, and W. Kim, "Preparation for semantic-based XML mining," presented at the 2001 IEEE International Conference on Data Mining, San Jose, pp. 345-352, November 2001.

[14] M. H. Qureshi, Kozielski, and M. H. Samadzadeh, "Determining the complexity of XML documents," in *Proc. the International Conf. on Information Technology: Coding and Computing* (*ITCC'05*), vol. II–vol. 02, pp. 416-421.

[15] X.-Y. Li, "Using clustering technology to improve XML semantic search," in *Proc. the Seventh International Conf. on Machine Learning and Cybernetics*, July 2008, vol. 5, pp. 2635-2639.

[16] S. Zhang, J. T. L. Wang, and K. G. Herbert, "Xml query by example," *International Journal of Computational Intelligence and Applications*, vol. 2, no. 3, pp. 329-337, 2002.

[17] J. Robie and R. Hat, "XML Processing and data integration with Xquery," *IEEE Internet Computing*, vol. 11, no. 4, pp. 62-67, August 2007.

[18] N. Bruno, N. Koudas, and D. Srivastava, "Holistic twig joins: optimal XML pattern matching," in *Proc. the SIGMOD Conference*, 2002, pp. 310-321.

[19] Q.-K. Zhao, L. Chen, S.-S. Bhowmick, and S.-K. Madria, "XML structural delta mining," *Issues and challenges. Data and Knowledge Engineering*, vol. 59, no. 3, pp. 627-651, 2006.

[20] S. Chen, H. G. Li., J. Tatemura, W. P. Hsiung, D. Agrawal, and K. S. Candan, "Twig2 Stack: bottom-up processing of generalized tree-pattern queries over XML documents," in *Proc. the VLDB Conf.*, 2006, pp. 283-294.

[21] N. Bruno, N. Koudas, and D. Srivastava, "Holistic twig joins: optimal XML pattern matching," in *Proc. the SIGMOD Conf.*, 2002, pp. 310-321.

[22] J. Lu, T. W. Ling, C. Y. Chan, and T. Chen, "From region encoding to extended Dewey: on efficient processing of XML twig pattern matching," in *Proc. the VLDB Conf.*, 2005, pp. 193-204.

[23] S. K. Izadi, T. Härder, and M. S. Haghjo, "S3: Evaluation of tree-pattern XML queries supported by structural summaries," *Data & Knowledge Engineering*, vol. 68, issue 1, pp. 126-145, Jan. 2009.

[24] B. Catania and A. Maddalena, "XML document indexes: a classification," *IEEE Internet Computing*, vol. 9, no. 5, pp. 64-71, October 2005.

[25] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury, *ORDPATHs: Insert-Friendly XML Node Labels*, SIGMOD, 2004, pp. 903-908.

[26] S. Tatarinov, K. S. Viglas, J. Beyer, E. Shanmugasun-daram, J. Shekita, and C. Zhang, *Storing and Querying Ordered XML Using a Relational Database System*, SIGMOD, 2002, pp. 204-215.

[27] T. Harder, M. P. Haustein, C. Mathis, and M. Wagner, "Node labeling schemes for dynamic XML documents reconsidered," *Data and Knowledge Engineering*, vol. 60, no. 1, 2007, pp. 126-149.

[28] M. P. Haustein and T. Harder, "An efficient infrastructure for native transactional XML processing," *Data and Knowledge Engineering*, vol. 61, no. 3, 2007, pp. 500-523.

Hsu-Kuang Chang received the B.S. degree in Computer Science from New York Institute of Technology in 1989 and the M.S. degree in Computer Science from New York Polytechnic University in 1991. He is a Ph.D. candidate in National Kaohsiung First University of Science and Technology, Taiwan. He is also a lecturer in the Department of Information Engineering, I-Shou University, Taiwan. His research interests include data mining, multimedia database, and information retrieval.

King-Chu Hung was born in Tainan, Taiwan, R. O. C., on March 29, 1959. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kong University, Tainan, in 1980, 1982, and 1988, respectively.

In 1988, he was an associate professor with the Institute of Computer Science and Electronic Engineering, National Central University, Taiwan. From 1989 to 1995, he was an Associate Researcher and Technology Supervisor with Chung Shan Institute of Science and Technology (CSIST), Lung Tan, Taiwan. He was with the Department of Electronics Engineering, I-Shou University, Dashu, Taiwan, during 1995-1999. He is currently a Professor of Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan. His current research interests include computer vision, mathematics, image compression, VLSI, wavelets, and error control coding.