A Schema Matching-based Approach to XML Schema Clustering

Alsayed Algergawy, Eike Schallehn, and Gunter Saake
Computer Science Department
Magdeburg University
39106 Magdeburg
Magdeburg, Germany
{alshahat, eike, saake}@iti.cs.uni-magdeburg.de

ABSTRACT

The relationship between XML data clustering and schema matching is bidirectional. On one side, clustering techniques have been adopted to improve matching performance, and on the other side schema matching is the backbone of the clustering technique. This paper presents a new approach for clustering XML schema based on schema matching. In particular, we develop and implement an XML schema matching system, which determines semantic similarities between XML schemas based on the Prüfer sequence representation of schema trees. The proposed computation similarity algorithm makes use of the semantic meaning of XML elements as well as the hierarchical features of XML schemas. The computed similarities are then exploited by an agglomerative clustering algorithm to group similar schemas. Our experimental results show that the proposed approach is fast and accurate in clustering heterogeneous XML schemas.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering*

General Terms

Algorithms

Keywords

XML, clustering, schema matching, Prüfer sequences

1. INTRODUCTION

The extensible markup language (XML) has emerged as a standard for information representation and exchange on the Web. As a result, a huge amount of information is formatted in XML data and several tools have been developed to deliver, store, integrate, and query XML data [4, 10]. In order to analyze these data efficiently, a possible solution is to group similar XML data according to their semantics, content and structures. Grouping similar XML data across heterogeneous ones is known as XML data clustering. Clustering XML data plays a central role in many data application domains such as information retrieval, data integration, document classification, Web mining, and query processing.

In general there are two types of XML data— *XML documents* and *XML schemas*. An XML schema is the description of the structure and the legal building blocks for an XML document. A dozen of XML schema languages have been proposed [13]. Among them, XML DTD and XML Schema Definition (XSD) are commonly used. An XML document (document instance) represents a snapshot what the XML document contains [16]. Since the document definition outlined in a schema holds true for all document instances of that schema. Therefore, the result produced from the clustering of schemas will hold true for all document instances of those schemas, and can be reused for any other instances. However, clustering XML schemas is an intricate process and it differs significantly from clustering of flat data and text. The difficulties of clustering XML schemas are due to the following reasons [1]. (1) Clustering algorithms require the computation of similarity between different XML data. The computation of similarity among XML data has itself been known to be a difficult research problem. The heterogeneity in XML data presents many challenges to find similarity among the XML data. (2) The structural behavior of the XML data increases implicit dimensionality of the clustering algorithm, which leads to meaningless clusters.

Clustering is a useful technique for grouping data objects such that objects within a single group/cluster have similar features, while objects in different groups are dissimilar [12, 3]. Typically data clustering activity involves different steps. A prominent step is the similarity computation between pairs of data objects [12]. The similarity computation algorithms between XML data can be broadly classified into two categories depending on the exploited objects in similarity computation: (1) *tree-editing distance*, which exploits the whole XML schema (sub)tree or XML paths without considering elements' details [7, 8, 1, 5], (2) *schema matching*, which exploits the semantic and structural element' properties to determine similarity among XML schemas [14, 17, 16]. The tree-editing approaches have been proposed to cluster XML documents as well as they can be very expensive rendering them impractical for huge XML data.

Motivated by the above challenges, in this paper, we present

iiWAS 2008, November 24-26, 2008, Linz, Austria.

a new schema matching-based approach to XML schema clustering. In particular, we develop and implement a clustering framework, which consists of three phases. (1) *Pre-Processing*; XML schemas to be clustered are first parsed and represented as ordered labeled trees. In order to accelerate the clustering process, sequence representations for these schema trees are constructed using the Prüfer coding method [20]. (2) *Similarity Computation*; Similarity among XML schemas are determined exploiting both the semantic and structural information carried by the sequence representations of schema trees. (3) *Clustering*; Nested sets of data (hierarchies) are produced using a hierarchical clustering algorithm [12]. We carried out a set of experiments utilizing different real datasets to evaluate the proposed framework. Our experimental results show that the proposed framework is fast and accurate in clustering heterogenous XML data.

To summarize, the contributions of our work are:

- investigating the relationship between XML schema matching and XML data clustering,
- developing an XML schema clustering framework based on schema matching, and
- conducting a set of experiments using real datasets to validate the performance of our framework.

The paper is organized as follows. Section 2 introduces our XML clustering framework. Experiments evaluation is presented in Section 3. Section 4 discusses related work. The concluding remarks and open research directions are presented in Section 5.

2. THE CLUSTERING FRAMEWORK

In this section we shall describe the core parts of the proposed clustering framework. Figure 1 shows our clustering framework, which consists of three main phases—*Pre-processing, Similarity computation* and *Clustering*.

2.1 Pre-Processing

[1] This phase is concerned with the representation of heterogeneous XML schemas as sequence representations. To this end, it contains two steps: *Parsing* and *Prüfer Sequence Construction*. First, each XML schema is parsed using a SAX parser and represented internally as a rooted ordered labeled tree called the schema tree, wherein each schema component (element and/or attribute) is represented as a node, while edges are used to represent relationships between components. Each node in the schema tree carries the associated element properties. In the current implementation, we make use of the name, type/datatype element properties.

To efficiently cope with schema trees especially in huge XML data, we then construct a modified Prüfer sequence, called Consolidated Prüfer Sequence CPS [22], for each schema tree. The Prüfer sequence constructs a one-to-one correspondence between schema trees and CPSs. CPS of a schema tree consists of two sequences Number Prüfer Sequences NPS and Label Prüfer Sequences LPS. This allows us to capture schema tree semantic information in $LPSs$ while tree structure information in $NPSs$. CPSs are constructed by doing a *post-order traversal* that tags each node in the schema tree with a unique traversal number. NPS is then constructed iteratively by removing the node with the smallest traversal number and appending its parent node number to the already structured partial sequence. LPS is constructed similarly but by taking the node labels of deleted nodes instead of their parent node numbers.

[1]This subsection and the next one is based on our previous work. To keep this paper self contained, we give short notes about them. More details can be found in [2]

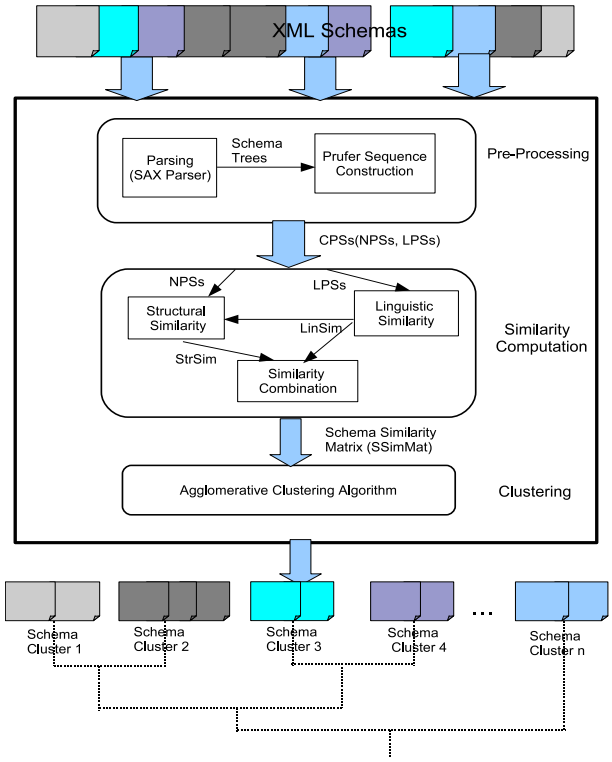


Figure 1: XML clustering framework architecture

2.2 Similarity Computation

This phase is concerned with the similarity computation between every XML schema pairs in order to form the similarity matrix, which will then be used by the clustering algorithm. For this purpose, we employ two different matchers: (1) *the linguistic matcher*, which exploits semantic schema information presented in $LPSs$; and (2) *the structural matcher*, which makes use of structural features of XML schemas carried by $NPSs$. Then, the two computed similarities are combined using the similarity combination step.

2.2.1 Linguistic Matcher

This step aims at determining initial similarity values between schema trees' nodes based on the semantic properties of nodes. In the current implementation, we make use both of the name property and of the type/datatype property. To compute name similarity between two element'names represented as two strings s_1 and s_2, we first break each string into a set of tokens T_1 and T_2 using a customizable tokenizer using punctuation, upper case, special symbols, and digits, e.g. UnderGradCourses $\rightarrow$ {Under, Grad, Courses}. We then determine the name similarity between the two sets of name tokens T_1 and T_2 as the the average best similarity of each token with a token in the other set.

To measure the string similarity between every two tokens, $sim(t_1, t_2)$, we use two string similarity measures. The first one is $sim_{edit}(t_1, t_2) = \frac{max(|t_1|,|t_2|)-editDistance(t_1,t_2)}{max(|t_1|,|t_2|)}$, where *editDistance($t_1$, t_2)* is the minimum number of character insertion and deletion operations needed to transform one string to the other. The second is based on the number of different trigrams in the two strings: $sim_{tri}(t_1, t_2) = \frac{2\times|tri(t_1)\bigcap tri(t_2)|}{|tri(t_1)|+|tri(t_2)|}$, where *tri($t_1$)* is the set of trigrams in t_1.

To enhance the matching result and to prune some of false positive candidates, we propose to exploit type/datatype of nodes. We make use of built-in XML datatypes hierarchy [2] in order to compute datatype compatibility coefficients. Based on XML schema datatype hierarchy, we build a datatype compatibility table as the one used in [15]. After computing datatype compatibility coefficients, we can adjust name similarity values. The result of the above process is a linguistic similarity matrix $LinSim$.

2.2.2 Structural Matcher

The linguistic matcher considers only the label information and ignores the structural information. There can be multiple match candidates, which differ in structure but have the same label. The structural matcher prunes these false positive candidates by considering the structural information presented in the *NPS* sequence.

Our structural matcher is motivated by the fact that the most prominent feature for an XML schema is its hierarchical structure, and is based on the node context, which is reflected by its ancestors and its descendants. The descendants of an element include both its immediate children and the leaves of the subtrees rooted at the element. The immediate children reflect its basic structure, while the leaves reflect the element's content. In the current implementation, we consider three kinds of node contexts depending on its position in the schema tree: *child context, leaf context* and *ancestor context*. The context of a node is the combination of its ancestor context, its child context, and its leaf context. Two nodes are structurally similar if they have similar contexts.

To measure the structural similarity between two nodes from two different XML schemas, we compute respectively the similarity of their child, leaf, and ancestor contexts. Both the child context and the leaf context depend on set comparison while the ancestor context is based on path comparison [2].

The child context. The representation of a schema tree as Prüfer sequence facilitates the determination of structural properties of the schema tree. Each entry in the *CPS* presents an edge from the parent node *NPS* to its immediate child node *LPS*. Therefor, we could easily obtain the number of immediate children of a non-leaf node from the *NPS* sequence by counting its post-order traversal number in the sequence, and also we could identify these children. To obtain the child context similarity between two nodes, we compare the two child context sets for the two nodes. To this end, we first extract the child context set for each node from NPS and LPS sequences. Second, we get the linguistic similarity between each pair of children in the two sets. Third, we select the matching pairs with maximum similarity values. And finally, we take the average of best similarity values.

The leaf context. First we notice that nodes whose post-order numbers do not appear in the *NPS* sequence are atomic nodes. From this notice and from the child context we could recursively obtain the leaf context for a certain node. To compute the leaf context similarity between two nodes, we compare their leaf context sets. To this end, first, we extract the leaf context set for each node. Second, we determine the gap between each node and its leaf context set. Third, we apply the cosine measure between two gap vectors.

The ancestor context. For a non-atomic node, we obtain the ancestor context by scanning the *NPS* sequence from left to right and identifying the numbers which are greater than post-order number of the node until the first occurrence of the root node. While scanning from left to right, we ignore nodes whose post-order numbers are less than post-order numbers of already scanned nodes. For a leaf node, the ancestor context is the ancestor context of its parent union the parent itself. To compute the ancestor similarity between two nodes n_1 and n_2, first we extract each ancestor context from the NPS sequence, say path P_1 for n_1 and path P_2 for n_2. Second, we compare two paths. To compare two paths, we use three of four scores established in [6].

The result of the structural matcher is a structural similarity matrix $StrSim$.

2.2.3 Schema Similarity Matrix

Given a set of XML schemas $S = \{S_1, S_2, ..., S_n\}$, we construct an $n \times n$ schema similarity matrix $SSimMat$. Each entry in the matrix $SSimMat[i][j]$ represents the similarity between schema S_i and schema S_j. For every schema pairs, we sum up (weighted sum) all element similarity values computed by the linguistic matcher and the structural matcher.

2.3 Clustering

There are many techniques for clustering algorithms among them are hierarchical clustering algorithms [12]. Hierarchical clustering solutions are in the form of trees called *dendrograms*, which provide a view of the data at different levels of abstraction. The consistency of clustering solutions at different levels of granularity allows flat partitions of different granularity to be extracted during data analysis, making them ideal for interactive exploration and visualization [11]. Two primary methods to obtain hierarchical clustering solutions: *agglomerative algorithms* and *partitional algorithms*.

In agglomerative algorithms, objects are initially assigned to its own cluster and then the pairs of clusters are repeatedly merged until the whole tree is formed. However, partitional algorithms can also be used via a sequence of repeated bisections. The partitional algorithms are well-suited for clustering large datasets due to their relatively low computational requirements. However, the agglomerative algorithms outperform partitional algorithms. For this, in our implementation we make use of another hierarchical clustering algorithm called the *constrained agglomerative algorithm* [11].

[2] `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`

We use the wCluto[3] a web-enabled data clustering application for clustering XML schemas. In order to make use of wCluto, we first perform the similarity computation phase to obtain the schema similarity matrix, which is used as the input for wCluto. Then, the hierarchical clustering algorithm is selected and its parameters are tuned.

3. EXPERIMENTAL EVALUATION

In this section we describe the experiments that we have carried out to evaluate our proposed framework. In the rest of this section we first describe the used datasets and our experimental methodology, followed by a description of the experimental results.

3.1 Dataset

We used a total of six different datasets, whose general characteristics are summarized in Table 1. The datasets have been obtained from different domains [4] [5] [6] and represent different characteristics. Each domain consists of a number of different categories that have structural and semantic differences. The XML schemas from the same domain also vary in structures and semantics.

Table 1: Data set details

Domain	*No. of schemas*	*No. of nodes*	*No. of levels*
Auction	4	35/39	4/6
Mondial	7	11-	4/8
Financial	2	14/14	3/6
TPC-H	10	8/45	2/6
GeneX	2	75/85	3/8
University	25	8-20	3/7

3.2 Experimental Methodology and Metrics

The different datasets are first extracted and modified to be ready for the clustering framework. The current implementation supports only clustering XSD schemas, hence we transformed DTDs into XSDs. The quality of clustering solutions have been verified using two common measures: (1) FScore as an external measure, and (2) intra-clustering similarity and inter-clustering similarity as internal measures.

FScore is a trade-off between two popular information retrieval metrics, precision P and recall R. Precision considers the rate of correct matches in the generated solution, and recall considers the rate of correct matches in the model solution. Given a cluster C_i, let TP be the number of XML data in C_i which are similar (correctly clustered), FP be the number of documents C_i which are not similar (misclustered), FN be the number of documents which are not in C_i but should be. The precision and recall of a cluster C_i are defined as $P_i = \frac{TP}{TP+FP}$, and $R_i = \frac{TP}{TP+FN}$.

FScore combining precision and recall with equal weights for the given cluster C_i is defined by, $FScore_i = 2 \times \frac{P_i \times R_i}{P_i + R_i}$. The FScore of the overall clustering approach is defined as the sum of the individual class FScores weighted differently according to the number of XML data in the class

$$FScore = \frac{\sum_{i=1}^{k} n_i \times FScore_i}{n} \quad (1)$$

[3]http://cluto.ccgb.umn.edu/cgi-bin/wCluto/wCluto.cgi
[4]http://www.dbis.informatik.uni-goettingen.de/Mondial/
[5]http://www.cs.washington.edu/research/xmldatasets/
[6]http://www.cs.toronto.edu/db/clio/testSchemas.html

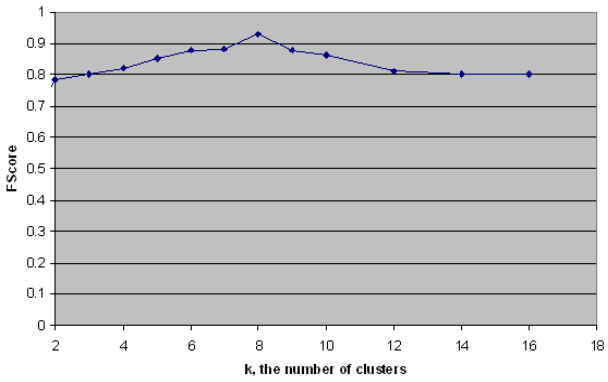


Figure 2: FScore

where k, n_i and n are the number of clusters, the number of XML data in a cluster C_i, and the number of XML data respectively. A good clustering solution has the FScore value closer to one.

The internal clustering solution quality measures are evaluated by calculating the average inter and intra-clustering similarity. The intra-clustering similarity measures the cohesion within a cluster, how similar the XML data within a cluster are. This is computed by measuring the similarity between each pair of data within a cluster, and the intra-clustering similarity of a clustering solution is determined by averaging all computed similarities taking into account the number of XML data within each cluster

$$IntraSim = \frac{\sum_{i=1}^{k} n_i \times IntraSim(C_i)}{n}. \quad (2)$$

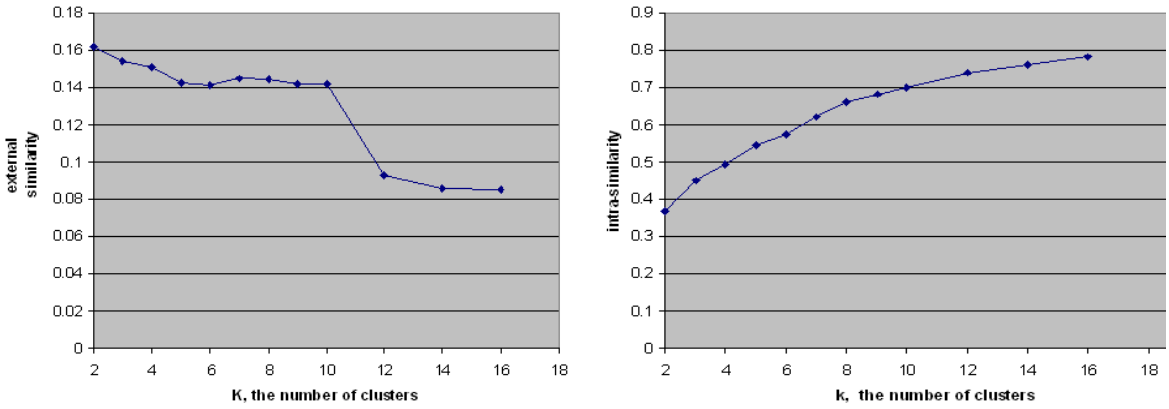
The larger the values of intra-clustering similarity ($IntraSim$), the better the clustering solution is. The inter-clustering similarity measures the separation among different clusters. It is computed by measuring the similarity between two clusters. A good clustering solution has lower inter-clustering similarity values.

3.3 Experimental Results

Figure 2 illustrates the FScore of the datasets over 16 different clustering solutions. With $k = 2$, all the 25 schemas from the university domain are in one group, while the other schemas from the other domains are in the second group. This results in a high FScore at $k = 2$. As k increase as FScore increases until the best FScore occurs at $k = 8$. When the process reaches the 12-clustering solutions, the clustering quality is stabilized. Also, Fig. 2 shows that the quality (FScore) of our proposed algorithm ranges between 79% and 93%, i. e. it is almost accurate.

The better clustering solution is the one having both higher intra-clustering similarity and lower inter-clustering similarity. Figure 3 supports this fact, such that the figure shows that as the clustering process continues, clusters are further decomposed into smaller sub clusters that contain highly similar schemas. Figure 3 also illustrates that as the number of clusters increases, the average intra-clustering similarity increases while the average inter-clustering similarity decreases.

4. RELATED WORK



Inter-clustering similarity

Intra-clustering similarity

Figure 3: Internal quality measures

The relationship between XML schema clustering and schema matching is bidirectional. From the view point of using clustering to support schema matching, research in this direction depends heavily on the fact that it is easier to find element correspondences between schemas that are contextually similar. [18] develops a clustered-based approach to schema matching. The approach clusters schemas based on their contextual similarity, then clusters attributes of schemas within the same schema cluster. Then attributes across different schema clusters are clustered using statistical information gleaned from existing attributes clusters to find attribute correspondences among different schemas. However, the approach deals only with flat schemas. [21] proposes a clustered schema matching technique. Clustering is used to identify clusters in the large schema repository which are likely to produce mappings for a personal schema. Other approaches, which make use of clustering to identify element correspondences in the context of integrating heterogeneous data sources, can be found in [23, 19].

From the other side point of view, research on clustering XML data is gaining momentum. XML data clustering can be broadly classified into two categories based on the data to be clustered: clustering XML documents and clustering XML schemas. Many approaches have been developed in the context of XML document clustering [9], while a little work is done in the context of XML schema clustering [14, 17]. [14] proposes an integration strategy, called XClust, that involves the clustering of DTDs. A matching algorithm, based on the semantic and structural properties of schema' elements has been proposed. [17] also develops a framework, called XMine, to cluster XML schemas (both DTTs and XSDs). XMine makes use of semantic, syntactic and structural properties of schema' elements.

Both XClust and XMine, as our proposed framework, represent XML schemas as rooted (ordered) labeled trees. However, we extend the tree representation of XML schemas into sequence representation in order to efficiently deal with schema elements instead of traversing schema trees many times. Moreover, the two clustering frameworks make use of WordNet to determine semantic (synonyms) similarity. XMine additionally implements a user defined dictionary in order to identify abbreviations and finally makes use of syntactic string functions (string edit distance) to compare between element names if no semantic relationships. In contrast, we use only simple string functions in order to determine initial similarity values for the structural matcher. Our structural matcher is similar to the one in [14]. They both depend on the node context utilizing both ancestor and descendant contexts of a node. However, our approach benefits from the sequence representation of schema trees.

5. SUMMARY AND FUTURE WORK

The growing number of heterogeneous XML schemas has raised a number of issues concerning how to represent and manage semi-structure data. We developed and implemented a schema matching-based XML schema clustering framework. The proposed framework consists of three phases: Pre-processing; to represent XML schemas as sequence representations, Similarity computation; to determine the similarity across XML schemas, and Clustering; to group similar XML schemas into clusters using the hierarchical clustering algorithm. Experimental evaluation showed that our proposed framework is almost accurate with FScore ranging between 80% and 93%.

6. REFERENCES

[1] C. C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. J. Zaki. Xproj: a framework for projected structural clustering of XML documents. In *KDD 2007*, pages 46–55, 2007.

[2] A. Algergawy, E. Schallehn, and G. Saake. A prufer sequence-based approach for schema matching. In *BalticDB&IS2008*. Estonia, 2008.

[3] P. Berkhin. Survey of clustering data mining techniques. In *Accrue Software, Inc.*, pages 1–56, 2002.

[4] E. Bertino and E. Ferrari. XML and data integration. *IEEE Internet Computing*, 5(6):75–76, 2001.

[5] E. Bertino, G. Guerrini, and M. Mesiti. Measuring the structural similarity among XML documents and DTDs. *Intelligent Information Systems*, 30(1):55–92, 2008.

[6] D. Carmel, N. Efraty, G. M. Landau, Y. S. Maarek, and Y. Mass. An extension of the vector space model for querying xml documents via XML fragments. *SIGIR Forum*, 36(2), 2002.

[7] I. Choi, B. Moon, and H.-J. Kim. A clustering method

based on path similarities of XML data. *DKE*, 60:361–376, 2007.

[8] T. Dalamagasa, T. Cheng, K.-J. Winkel, and T. Sellis. A methodology for clustering XML documents by structure. *Information Systems*, 31:187–228, 2006.

[9] G. Guerrini, M. Mesiti, and I. Sanz. *An Overview of Similarity Measures for Clustering XML Documents. Web Data Management Practices: Emerging Techniques and Technologies*. IDEA GROUP, 2007.

[10] M. Hassler and A. Bouchachia. Searching XML documents: Preliminary work. In *INEX2005*, pages 119–133, 2005.

[11] Y. B. Idrissi and J. Vachon. Evaluation of hierarchical clustering algorithms for document datasets. In *the 11th International Conference on Information and knowledge Management*, pages 515–524, 2002.

[12] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[13] D. Lee and W. W. Chu. Comparative analysis of six XML schema languages. *SIGMOD Record*, 9(3):76–87, 2000.

[14] M. L. Lee, L. H. Yang, W. Hsu, and X. Yang. Xclust: Clustering XML schemas for effective integration. In *CIKM 2002*, pages 292–299, 2002.

[15] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB 2001*, pages 49–58. Roma, Italy, 2001.

[16] R. Nayak. Fast and effective clustering of XML data using structural information. *Knowledge and Information Systems*, 14(2):197–215, 2008.

[17] R. Nayak and W. Iryadi. XML schema clustering with semantic and hierarchical similarity measures. *Knowledge-Based Systems*, 20:336–349, 2007.

[18] J. Pei, J. Hong, and D. A. Bell. A novel clustering-based approach to schema matching. In *4th ADVIS*, pages 60–69, 2006.

[19] C. Pluempitiwiriyawej and J. Hammer. Element matching across data-oriented XML sources using a multi-strategy clustering model. *Data & Knowledge Engineering*, 48:297–333, 2004.

[20] H. Prufer. Neuer beweis eines satzes uber permutationen. *Archiv fur Mathematik und Physik*, 27:142–144, 1918.

[21] M. Smiljanic, M. van Keulen, and W. Jonker. Using element clustering to increase the efficiency of XML schema matching. In *ICDE Workshops 2006*, pages 45–54, 2006.

[22] S. Tatikonda, S. Parthasarathy, and M. Goyder. LCS-TRIM: Dynamic programming meets XML indexing and querying. In *VLDB'07*, pages 63–74, 2007.

[23] H. Zhao and S. Ram. Clustering schema elements for semantic integration of heterogeneous data sources. *Journal of Database Management*, 15(4):88–106, 2004.