THE NATIONAL UNIVERSITY
of SINGAPORE


School of Computing
Computing 1, 13 Computing Drive, Singapore 117417

**TRA6/11**

# *Measuring XML Structured-ness with Entropy*

***Ruiming Tang, Huayu Wu and Stephane Bressan***

*June 2011*

# T e c h n i c a l  R e p o r t

## Foreword

*This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.*

OOI Beng Chin
Dean of School

# Measuring XML Structured-ness with Entropy

Ruiming Tang, Huayu Wu, and Stéphane Bressan

School of Computing, National University of Singapore
{tangruiming, wuhuayu, steph}@comp.nus.edu.sg

**Abstract.** XML is semi-structured. It can be used to annotate unstructured data, to represent structured data and almost anything in-between. Yet, it is unclear how to formally characterize, yet to quantify, structured-ness of XML. In this paper we propose and evaluate entropy-based metrics for XML structured-ness. The metrics measure the structural uniformity of path and subtrees, respectively. We empirically study the correlation of these metrics with real and synthetic data sets.

## 1 Introduction

XML is commonly qualified as a semi-structured data model. Indeed, the hierarchical and optional nature of element organization in XML make it a good choice for representing data ranging from annotated unstructured text to relational tables with fixed and prescribed schemata.

Being able to characterize and possibly quantify the absolute or relative "structured-ness" of an XML document or collection can be critical in many applications and at various stages of their life cycle (from design to tuning). For the sake of conciseness let us illustrate the potential benefits of a characterization of XML structured-ness with two example cases among numerous possible ones.

**Case 1:** There are two types of XML databases, i.e., XML-enabled relational databases and native XML databases. It has been studied that the efficiency of managing and querying XML data using the two types of XML databases highly depend on how structured an XML document is [13]. Generally, the more structured an XML document is, the more efficiently it can be managed and queried with an XML-enabled relational database; while the more irregular a document is, the more advantages emerged for a native XML database. Measuring the structured-ness of an XML document can help to determine which types of database should be used to store the document.

**Case 2:** Measuring similarity between XML documents is an essential building block for XML comparison, alignment, clustering and classification. However, computing the similarity among a set of XML documents is normally expensive. Using the degree of the structured-ness of each document, we can easily tell one document is far different from another, and thus avoid a lot of unnecessary similarity computations in many applications.

Surprisingly, the early enthusiasm for the promises brought by XML as a candidate standard for data management and interchange has not encouraged

attempts to formally and quantitatively define structured-ness. [11] and [2] address the issue in specific contexts for XML collections and DTDs, respectively.

In this paper we understand structured-ness as uniformity of the data representation (schema). Information theoretic entropy suggests itself as a measure of such uniformity, We devise, implement and compare two candidate categories of metrics evaluating uniformity of paths and uniformity of subtrees, respectively, and their variants. We empirically study the correlation of these metrics with the real and synthetic data sets. The structured-ness of the real data sets is manually annotated. The structured-ness of the synthetic data sets is given by construction.

The remainder of this paper is organized as follows: Section 2 presents the background and reviews related work. Section 3 develops our algorithms. Section 4 sets up the experiments to evaluate our algorithms. Section 5 concludes this paper.

## 2 Background and Related work

### 2.1 Background

**XML data model** An XML document can be modeled as an ordered labeled tree, without considering the ID references. In an XML document tree, each node represents an element, an attribute or a value in the corresponding XML document, and its label is the corresponding tag name, attribute name or value text. Each edge of an XML document tree represents a hierarchical relationship between the element and sub-element, element and attribute, element and value or attribute and value. Since in this paper we are only interested in the structure of an XML document, we ignore the value nodes which appear as leave of the document tree.

**Structured-ness of XML data** XML is a semi-structured data format, which contains both structured components that can be defined by schemas, and the flexibility to freely organize structured components and unstructured components in a hierarchy. The structured-ness of an XML document measures how flexible the document is, i.e., whether the document is regular with many structured components or irregular with a large amount of unstructured tags. The importance of knowing the structured-ness of an XML document is illustrated by the two examples in Section 1.

**Shannon Entropy** In the year of 1948, Claude E.Shannon introduced entropy to solve the problem of quantifying information. Shannon entropy is used to quantify the information contained in a random variable; in other words, it is a measure of information we lose if we do not know the value of the random variable. Assume there is a discrete random variable X that can take on possible values $x_1, x_2, ..., x_n$ and the Shannon entropy is: $H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$, where I(X) is the information content of X, and $p(x_i) = Pr(X = x_i)$ is the probability mass function of X.

There are two views of entropy. The first one is that entropy captures uncertainty in data, i.e., higher entropy indicates more unpredictability. The second

view is that entropy captures information content, i.e., higher entropy indicates more information. Thus we can conclude that given two distributions, the one that is closer to the uniform distribution has a higher entropy; if two distributions are both uniform distributions, the one that is more diverse has a higher entropy.

### 2.2 Related work

We have not identified any existing work on measuring the structured-ness of XML data. Theoretically, the lot of algorithms measuring similarity between XML documents can be extended for structured-ness measurement. In particular, an XML tree becomes a forest of trees if the root node and the edges between the root node and its children are removed. Then if all trees in the forest are quite similar to each other, the original XML document is considered with a high degree of structured-ness. Hereby, we review the existing work in XML similarity measurement in this section.

[10, 5, 12] measure the structural similarity between XML documents or XML document/DTD using tree edit distance. [10] aims to partition the XML document collection into several sets based on tree edit distance. [5] detects changes between several versions of an original XML document using tree edit distance. [5] matches XML documents and XML grammars for document classification using tree edit distance. [6, 7] measure the similarity between an XML document and an XML query using IR techniques. Traditional information retrieval(TF-IDF) deals with flat textual data. These works add some more information and modify the data structures in order to handle XML data which is in hierarchy. [3, 8]uses other techniques to measure similarity between XML documents. For instance, the authors of [3] describe the structure of an XML document as a set of paths, and in [8], they view the structure of an XML document as edges. Moreover, there exist some works which measures similarity using entropy. [11] measures the heterogeneity of XML collections using entropy while [2] proposes an entropy metric to measure the similarity between DTDs.

Our work focuses on measuring the structured-ness of a given XML document as a whole, instead of decomposing it into sub-documents and computing the similarity between these sub-documents.

## 3 Structured-ness measurement

We use entropy to measure the structured-ness of an XML document. If the XML document is more structured, it has a low entropy value; otherwise, a high entropy value should be assign to the document. We propose two types of entropy for structured-ness measurement: path-based entropy and subtree-based entropy.

### 3.1 Path-based entropy

The distribution of paths will affect the degree of structured-ness of an XML document. In particular, the more diverse the paths are, the less structured the document is. We define path entropy to measure the diversity of paths. When

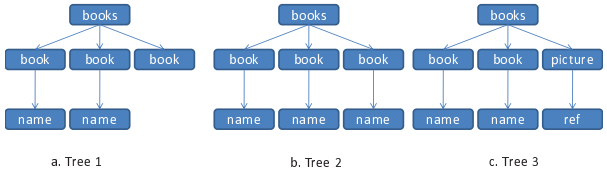



**Fig. 1.** Samples of three XML documents

all the paths[1] in an XML document are identical, i.e., the document is very structured, then the path diversity is of the lowest value, and so does the path entropy. When all the paths in the document are different, the diversity and the path entropy are in the highest values.

The formula is given in Equation 1.

$$Path\_entropy = -\sum_{i=1}^{n} p(X = x_i) \log_2 p(X = x_i) \tag{1}$$

where $n$ represents the number of unique paths. $p(X = x_i) = \frac{n(X=x_i)}{N}$, where $N$ is the number of paths in the document, and $n(X = x_i)$ is the number of paths which are the same as the unique path $x_i$. For instance, in Tree 1 in Fig. 1.a, there are three paths and only two unique paths. In this example, $p(X = x_1) = \frac{2}{3}$, $p(X = x_2) = \frac{1}{3}$, hence the path entropy of Tree 1 is $\frac{2}{3} * log_2\frac{3}{2} + \frac{1}{3} * log_2\frac{3}{1}$=0.92. Similarly, we can get the path entropy of Tree 2 in Figure 1.b is 0, and the path entropy of Tree 3 in Fig. 1.c has the same value as Tree 1.

### 3.2 Subtree-based entropy

In Fig. 2, we can see some drawbacks of the path entropy. For instance, the document in Fig. 2.a is more structured than the one in Fig. 2.b, so we expect that the path entropy of the document in Fig. 2.a is lower. Unfortunately it is higher.

The reason for the contradictions is that path-based entropy is only suitable for the case that each structured component is a path, instead of a subtree under the document root. However, this case is rare in practice. In this section, we propose to use subtree-based entropy to handle general cases.

**Exact subtrees entropy** The exact subtrees entropy measures the diversity of subtrees in the document. The basic idea is that the less diverse the subtrees are, the more structured the document is. We need to extract all the subtrees, and then compute the entropy based on their distribution. The formula is given in Equation 2.

$$exact\_subtrees\_entropy = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{2}$$

[1]This paper focuses on the structure of XML data, so when we mention *path* or *substree* the leaf values are not considered.

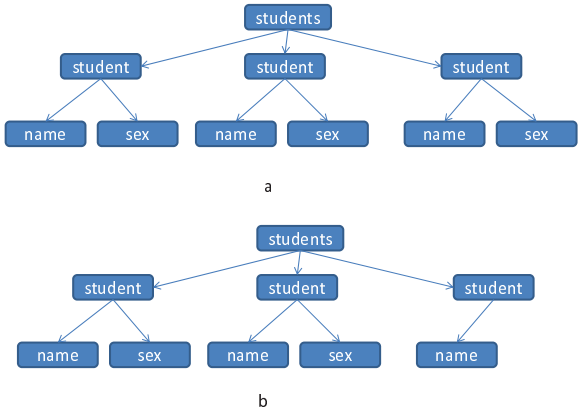



**Fig. 2.** Samples of two XML documents

where $n$ represents the number of categories. $p(i) = \frac{N_i}{N}$, where $N_i$ is the number of $subtree_i$ appearing in the document, $N$ is the total number of subtrees. In Fig. 2.a, there are 4 different kinds of subtrees in total while there are 5 in Figure 2.b. For the document in Figure 2.a, $p(1) = 0.3, p(2) = 0.3, p(3) = 0.3, p(4) = 0.1$, and entropy=$3 * 0.3 * log_2\frac{10}{3} + 0.1 * log_2\frac{10}{1}$=1.89546. For the document in 2.b, $p(1) = \frac{1}{3}, p(2) = \frac{2}{9}, p(3) = \frac{2}{9}, p(4) = \frac{1}{9}, p(5) = \frac{1}{9}$, and entropy=2.1964. This result reflects the fact that the document in Fig. 2.a is more structured than the one in Fig. 2.b.

Unfortunately, exact subtree entropy also has its drawbacks. This kind of entropy can only tell that these two subtrees are different, but cannot tell how much different they are.

**Similar subtrees entropy** The basic idea of similar subtrees entropy is that the difference between subtrees is better modeled as a numerical value rather than a Boolean value (yes/no). We use tree edit distance to measure this difference, after which we partition the subtrees into clusters based on their tree edit distances. We get the number of clusters and the size of each cluster. Finally, we compute similar subtrees entropy as the entropy of clusters

Consider an XML document with $n$ substrees. These subtrees are clustered into k clusters, i.e., $T$=$\{t_1, t_2, ..., t_k\}$. Assume the cluster $t_i$ contains $n_i$ subtrees, and $p_i = \frac{n_i}{n}$ denotes the ratio of the size of $t_i$ over the total size of the document, in terms of number of subtrees. Then the entropy of $T$ is the entropy of the cluster size distribution. The formula is given in Equation 3.

$$Similar\_subtrees\_entropy = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{3}$$

In the similar subtree entropy, there is one important concept, tree edit distance. In our similar subtrees entropy, we choose two different kinds of tree edit

distances: the first one is from [14], the second one is from [4]. In [14], for the first time, they introduce a non-exponential algorithm to compute the edit distance between ordered labeled trees, allowing insertion, deletion and updating of internal/leaf nodes. In [4], the authors restrict insertion and deletion to leaf nodes, and allow updating of nodes anywhere (in our version, we replace updating by one deletion and one insertion).

Clustering is a method to partition objects into groups to maximize the similarity within groups and the dissimilarity between groups. We choose the agglomerative clustering algorithm which is a bottom-up hierarchical clustering algorithm in [9].

There are three different metrics for measuring the distance between two clusters A and B: (1) max {d(x,y):$x \in A, y \in B$}; (2) min {d(x,y):$x \in A, y \in B$}; (3) $\frac{1}{|A||B|} \sum_{x \in A} \sum_{y \in B} d(x, y)$. We will use all of them for comparison.

The complete algorithm of similar subtree entropy is given in Algorithm 1.

**Algorithm 1**: similar subtree entropy Algorithm

**Data**: $file$ : an XML document
**Result**: The entropy for the document $E$

```
Parse file, and get all the subtrees;
N ← the number of subtrees;
for i = 1; i <= N; i++ do
    for j = 1; j <= N; j++ do
        compute tree editing distance between subtree_i and subtree_j, and store it
        in the distance matrix
    end
end
min ← minimum value in the distance matrix;
while min <= threshold do
    merge two closest clusters into one;
    update the distance matrix;
    min ← minimum value in the distance matrix;
end
compute entropy based on distribution of fraction of objects in each cluster
```

## 4 Performance experiment

### 4.1 Experiment setup

We generate our synthetic data sets using ToXGene([1]). During synthetic data generation, we use "?" in DTDs to control the structured-ness of each XML document. "?" in the DTDs means that this element either appears once or does not appear. In the same depth, the XML generated by this DTD will be more structured-ness if there are less "?". It will affect the structured-ness of the document more seriously if "?" is in the level closer to the root. Based on the observations above, we can design several DTDs which can guarantee that

the XML documents generated by them are ranked from the most structured to least structured. The DTDs are presented in Fig. 3. We can see that Fig. 3.a is the most structured case, while Fig. 3.j is the least structured case. From Fig. 3.a to Figure 3.j, the structured-ness of each document decreases. The "*" beside the element "student" means that the number of students can be changed.

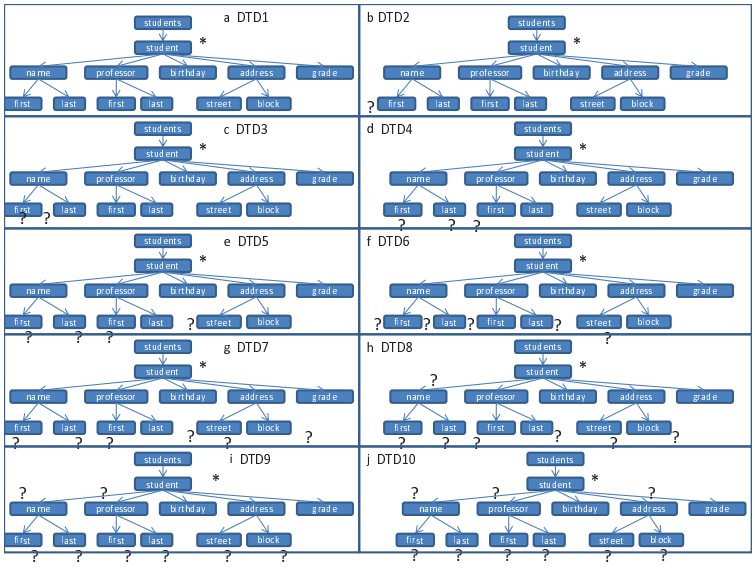


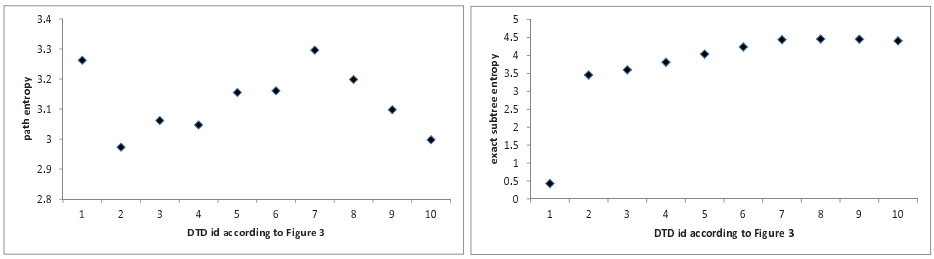

**Fig. 3.** DTDs used for generating synthetic data

From our manual judgment, the entropy of each group of documents generated by DTDs in Fig. 3 should keep increasing from Fig. 3.a to Fig. 3.j due to the increase of the number of "?" in these DTDs. We calculate all these kinds of entropy when the number of students is 50.

### 4.2 Experimental evaluation on synthetic data

The experimental evaluation results are presented in Fig. 4 and 5. The x axis represents the corresponding DTD ID according to Fig. 3, the y axis is for the average entropy value. Each point in Fig. 4 and Fig. 5 is average entropy value of 1000 documents generated by the corresponding DTD. Fig. 4.a and Fig. 4.b are path entropy and exact subtree entropy when fixing the number of students to 50 and varying DTDs respectively. Fig. 5 is the similar subtree entropy. Fig. 5.a to Fig. 5.c are results for using the first kind of tree edit distance in similar subtree entropy, and Fig. 5.d to Fig. 5.f are results for using the second kind of tree edit distance(these two kinds of tree edit distance are introduced in the section of "similar subtree entropy"). In Fig. 5.a to Fig. 5.c, they are three different metrics we use(minimum value, mean value and maximum value) when we do

the clustering(mentioned in the section"similar subtree entropy"), and in Fig. 5.d to Fig. 5.f, the situation is the same.




(a) path entropy (b) exact subtree entropy

**Fig. 4.** Path-based entropy and exact subtree entropy

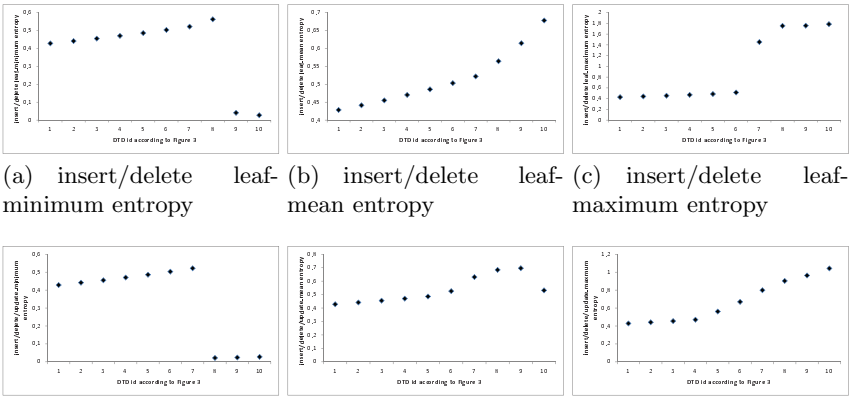
From Fig. 4.a, we can see that the path entropy does not work well. The key reason is that path entropy defines structured based on paths, but now the synthetic documents are ranked from structured to unstructured based on subtrees. In Fig. 4.b and Figure 5, we can see that the kinds of subtree-based entropy works better, except for the two cases that use "minimum distance" as clustering parameter. In 5.a and 5.d, the entropy value suddenly goes down when there comes the most unstructured documents. The reason is that when we construct synthetic documents, we put more and more "?" in DTDs which causes the nodes of unstructured documents less than structured ones. Thus in the more unstructured case, the tree edit distance between subtrees becomes smaller since the size of subtrees shrinks. To make things worse, the "minimum distance" is more likely to merge different groups compared to "maximum distance" and "mean distance". In this case, the algorithm will generate less clusters than it is supposed to do. However, the exact subtree entropy and other cases of the similar subtree entropy works acceptable since they almost fit to our manual judgement.

We also do the experiments varying the number of students from 10 to 100. We find that the results are almost the same as the case of 50. Thus we will not show the results due to the space limitation.

### 4.3 Experiment evaluation on real data

We get ten different XML documents from INEX and XML repository, whose sizes are from 1K to 334K. INEX: 427.xml(334K), 536.xml(166K), 2008-topics.xml(142K); XML repository: 321gone.xml(24K), ebay.xml(35K), nation.xml(5K), region.xml(1K), supplier.xml(29K), ubid.xml(20K), yahoo.xml(25K). We measure Pearson correlation coefficient within kinds of subtree-based entropy. Table 1 is the Pearson correlation coefficient within subtree based entropy.

As shown in Table 1, the Pearson's correlation coefficient is almost 1 between any two kinds of entropy within the same tree edit distance method(one is insert/delete/update, the other one is insert/delete leaf). When looking into the



(a) insert/delete leaf-minimum entropy (b) insert/delete leaf-mean entropy (c) insert/delete leaf-maximum entropy

(d) insert/delete/update-minimum entropy (e) insert/delete/update-mean entropy (f) insert/delete/update-maximum entropy

**Fig. 5.** Similar subtree entropy

entropy values, we can see the entropy values of insert/delete/update-minimum, maximum, and mean are almost the same for most of the documents. The same thing happens for entropy of insert/delete leaf. We can conclude that, sometimes when computing entropy for real XML data, it does not matter choosing which parameter to do the clustering(minimum distance, maximum distance, or mean distance).

## 5 Conclusion and future work

In this paper, we introduce a method measuring XML structured-ness within a single XML document using entropy. To the best of our knowledge, this is the first work on measuring the structured-ness of XML data. We propose two different groups of entropy, i.e., the path-based entropy and the subtree-based entropy. The experiment results show that the subtree-based entropy work better than path entropy for more general XML data.

As we know, computing tree edit distance is a very expensive operation. In our future work, we will try to find another metric to measure the distance between subtrees instead of tree edit distance.

## References


1. D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons. ToXgene: An Extensible Template-Based Data Generator for XML. In *WebDB*, 2002.
2. D. Basci and S. Misra. Entropy metric for xml dtd documents. *ACM SIGSOFT Software Engineering Notes*, 33(4), 2008.
3. D. Buttler. A Short Survey of Document Structure Similarity Algorithms. In *International Conference on Internet Computing*, pages 3–9, 2004.
4. S. S. Chawathe. Comparing Hierarchical Data in External Memory. In *VLDB*, 1999.

| entropy x | entropy y | Pearson's correlation |
|---|---|---|
| exact subtree | insert/delete/update-minimum | 0.217305 |
| exact subtree | insert/delete/update-maximum | 0.185191 |
| exact subtree | insert/delete/update-mean | 0.192933 |
| exact subtree | insert/delete leaf-minimum | -0.045959 |
| exact subtree | insert/delete leaf-maximum | 0.309342 |
| exact subtree | insert/delete leaf-mean | -0.066557 |
| insert/delete/update-minimum | insert/delete/update-maximum | 0.998775 |
| insert/delete/update-minimum | insert/delete/update-mean | 0.999257 |
| insert/delete/update-minimum | insert/delete leaf-minimum | 0.498007 |
| insert/delete/update-minimum | insert/delete leaf-maximum | 0.491377 |
| insert/delete/update-minimum | insert/delete leaf-mean | 0.487187 |
| insert/delete/update-maximum | insert/delete/update-mean | 0.999936 |
| insert/delete/update-maximum | insert/delete leaf-minimum | 0.476003 |
| insert/delete/update-maximum | insert/delete leaf-maximum | 0.458947 |
| insert/delete/update-maximum | insert/delete leaf-mean | 0.465916 |
| insert/delete/update-mean | insert/delete leaf-minimum | 0.481393 |
| insert/delete/update-mean | insert/delete leaf-maximum | 0.466889 |
| insert/delete/update-mean | insert/delete leaf-mean | 0.471148 |
| insert/delete leaf-minimum | insert/delete leaf-maximum | 0.929183 |
| insert/delete leaf-minimum | insert/delete leaf-mean | 0.999720 |
| insert/delete leaf-maximum | insert/delete leaf-mean | 0.922643 |

**Table 1.** correlation coefficient between subtree-based entropy for real data


5. G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in xml documents. In *ICDE*, pages 41–52, 2002.
6. N. Fuhr and K. Großjohann. XIRQL: An XML Query Language Based on Information Retrieval Concepts. *ACM Trans. Inf. Syst.*, 22(2):313–356, 2004.
7. T. Grabs and H.-J. Schek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In *XML and Information Retrieval Workshop at SIGIR*, 2002.
8. H.-P. Kriegel and S. Schönauer. Similarity Search in Structured Data. In *DaWaK*, pages 309–319, 2003.
9. T. Kurita. An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition*, 24(3):205–209, 1991.
10. A. Nierman and H. V. Jagadish. Evaluating structural similarity in xml documents. In *WebDB*, pages 61–66, 2002.
11. I. Sanz, M. Mesiti, G. Guerrini, and R. B. Llavori. An Entropy-Based Characterization of the Heterogeneity of XML Collections. In *DEXA Workshops*, 2008.
12. J. Tekli, R. Chbeir, and K. Yétongnon. Structural similarity evaluation between xml documents and dtds. In *WISE*, pages 196–211, 2007.
13. H. Wu, T. W. Ling, B. Chen, and L. Xu. Twigtable: Using semantics in xml twig pattern query processing. *Journal on Data Semantics*, 2011.
14. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.