A Quantitative Summary of XML Structures

Zi Lin[1], Bingsheng He[2] and Byron Choi[1]
[1] Nanyang Technological University.
{linzi,kkchoi}@ntu.edu.sg
[2] Hong Kong University of Science and Technology
saven@cs.ust.hk

Abstract. Statistical summaries in relational databases mainly focus on the distribution of data values and have been found useful for various applications, such as query evaluation and data storage. As XML has been widely used, e.g. for online data exchange, the need for (corresponding) statistical summaries in XML has been evident. While relational techniques may be applicable to the data values in XML documents, novel techniques are requried for summarizing the structures of XML documents. In this paper, we propose metrics for major structural properties, in particular, nestings of entities and one-to-many relationships, of XML documents. Our technique is different from the existing ones in that we generate a *quantitative summary* of an XML structure. By using our approach, we illustrate that some popular real-world and synthetic XML benchmark datasets are indeed highly skewed and hardly hierarchical and contain few recursions. We wish this preliminary finding shreds insight on improving the design of XML benchmarking and experimentations.

1 Introduction

eXtensible Markup Language (XML) is known to be a *flexible* [33] medium for online data exchange. The flexibility of XML is mainly[1] due to its capability of representing nested entities and one-to-many relationships in a tree. In comparison, the relational model is rigid and flat: neither nested entities nor one-to-many relationships are allowed in a single relation. In addition, the simplicity of the relational model has been one of its major strengths; implementations of the relational model have also been widely tested in industrial-strength applications. While XML repositories have been emerging (*e.g.* [17, 12, 21]), there has been an evident reservation on the advance from relational-based technology to XML-based technology. The host of work on reusing relational database systems for storing and querying XML (*e.g.* [31, 30, 11, 3, 14]) might reflect this standpoint. However, intuitively, relational systems are preferable *only when* an XML document is a mild generalization of relations. Otherwise, the impedance mismatch between the tree model and the relational model can become problematic where native XML/XML-based approaches should be adopted.

[1] For simplicity, we do not focus on the rich set of scalar data types (*e.g.* integers) in XML SCHEMAS.

In this paper, we define metrics for some structural properties of XML documents. We hope the metrics answer an informal question: Is an XML document "tree-like" or "relational-like"? Similar to effective techniques widely used in relational databases [24], we summarize XML structures using histograms and tree/graph structures and generate a quantitative summary from these structures, *i.e.*, our approach does not require DTDs/XML SCHEMAS. A goal of having a quantitative summary is that it may provide XML researchers (i.e. human) insights on XML documents. The advantage of quantitative approaches may be summarized by Lord Kelvin's remarks, *"When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of science."* As we shall see soon, we focus on the properties that cannot be derived from DTDs/XML SCHEMAS. Specifically, we examine two important structural properties of XML: (1) entity nestings and (2) one-to-many relationships.

Knowing the structures of XML datasets, in practice, can be fruitful to XML research, *e.g.* XML storage. For example, consider a simplified DBLP dataset [17] shown in Figure 1 and the `book-author` edges in the figure. For illustration purpose, assume that `book` is an entity and `author` is another entity. The `book-author` edges represent the relationship between `book` and `author`. Since DBLP is a real-world dataset, one may expect the distribution of the number of authors per book is a Gaussian distribution. However, we discovered the skewness (see Section 3 for the definition adopted) of this distribution is large. The result is shown in Figure 6. The x-axis and y-axis are the number of authors of a book and the number of books with x authors in DBLP document, respectively. Similar highly skewed distributions of `inproceeding-author`, `proceeding-author` and `article-author` are found. The implication of this finding is that if one simply "inlines" three authors into a `book` relation, such a `book` relation covers 97% of the books in DBLP and the remaining authors of the books can be stored in a small overflow relation. This storage scheme allows retrieving the relationship between `book` and `author` by a projection on a `book` relation and a join between `book` and the small overflow relation, as opposed to a join between all `book`s and all `author`s. We shall discuss some more research on XML storage in the related work section.

As a side product of our investigation, our metrics shred insights on some properties of synthetic XML datasets. For example, we report that the XMARK dataset comprises skewed distributions of structures whereas the XBENCH dataset consists of mostly Gaussian or uniform distributions of structures. While no one has asserted that XML structures are supposed to be uniform, there does not appear obvious reasons for using highly skewed data for benchmarking either. Furthermore, XML algorithms, *e.g.* DTD validation in streaming XML [29] and updates through XML views [5], for recursive XMLs are more technically challenging than their counterparts for non-recursive XMLs. Unfortunately, [9] showed that real-world DTDs are often recursive. This paper reports, in quantitative terms,

that the recursive part, if any, of real-world XML documents is often tiny. Consider a simplified XMARK document shown in Figure 3 for example. The percentage of root-to-leaf paths with recursive elements is only 9.3% of the total number of all root-to-leaf paths in the XMARK document. Hence, algorithms for non-recursive XMLs may often work in practice or "survive" in experimental evaluations; but *problems may occur occasionally*.

The two main goals of this paper are (1) to define metrics for describing structural properties of an XML document, in order to study the informal concept of tree-ness of an XML document and (2) to survey a few popular XML repositories using our metrics. Applications of our metrics to specific research problems, *e.g.*, XPATH/XQUERY selectivity estimation and XML compression, are beyond the scope of this paper. For presentation simplicity, we omit the analysis on the scalar data in XML documents.

Contributions. The main contributions of this paper are listed below:

- We present metrics for describing the nestings of entities and the number of each kind of attributes of an entity in XML datasets;
- We apply the metrics on a few popular[2] real-world and synthetic XML datasets for experimentation, among other uses. We reveal that these datasets are highly skewed and hardly hierarchical and contain few recursions.

Organization. The remainder of the paper is organized as follows. Section 2 presents the background of the computation of our metrics. Section 3 presents each of our XML metrics in detail. We apply our metrics on a few popular real-world and synthetic XML datasets in Section 4. Section 5 discusses the related work on XML metrics and statistics. Conclusions and discussions on future works are presented in Section 6.

2 Preliminaries

In this section, we present some background information for the subsequent sections.

Prefix trees. The computation of our metrics of an XML document relies on the construction of the prefix tree of the document. Specifically, we associate histograms (structural information) to a prefix tree. A node in a prefix tree represents a prefix occurred in a document. First, a node in a prefix tree is associated with the support, *sup*, of the prefix in the document. Second, we define a *support ratio* between each pair of parent-child nodes (A, B) in the prefix tree, *i.e.*, sup_B/sup_A, to estimate the possible location of one-to-many relationships. There are three possible cases for the support ratio:

[2] According to Google scholars system, `http://scholar.google.com`, March 2006, the number of citations on these datasets is over 300.

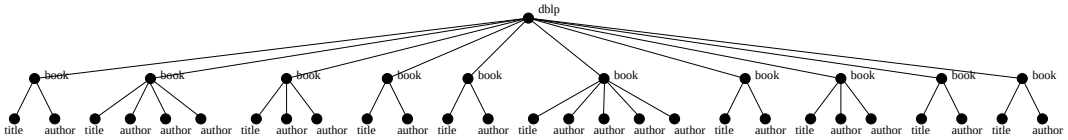

Fig. 1. Simplified DBLP document T_{dblp}

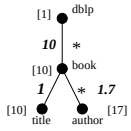

Fig. 2. Prefix tree of T_{dblp}, S_{dblp}

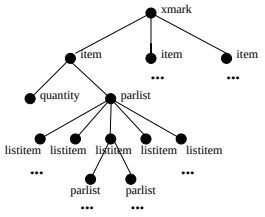

Fig. 3. Simplified XMARK document T_{xmark}

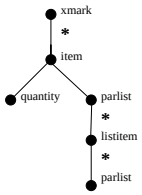

Fig. 4. Prefix tree of T_{xmark}, S_{xmark}

1. The support ratio is between 0 to 1. This implies B is probably A's optional child;
2. The support ratio is 1. This often implies a one-to-one relationship;
3. The support ratio is greater than one. This indicates a one-to-many relationship. We regard the edges in this class as *star edges*.

There are exceptions for these implications. Consider a pathological document in which one half of the A nodes do not have any B children and another half of the A nodes have exactly two B children. The support ratio indicates a false one-to-one relationship. However, such exceptions are rare, as we observed from our datasets.

Consider a simplified DBLP XML document shown in Figure 1 as an example. Its prefix tree is shown in Figure 2. We show the support of a node in a square bracket. Support ratios are placed next to the edges. We mark a star with an asterisk sign. Similarly, we show the stars of the prefix tree of the simplified XMARK dataset (Figure 3) in Figure 4.

Finally, for each star edge (A, B), we build a histogram for the number of B nodes of an A node.

Structural properties. In this paper, we focus on two major structural properties allowed by XML. These two properties can only be determined by the document instances, not DTDs/XML SCHEMAS.

First, we study the nestings of entities in an XML document. In the absence of user specifications, one could at best "estimate" entities in XML instances. In this paper, we assume that a star edge is an indication of a one-to-many relationship between two entities since such edges in the document instance cannot be naturally represented by a single relation. A complication here is that entities can be recursively defined in XML. Consider the XMARK dataset as

an example. A `parlist` subtree may contain `parlist` subtrees. A survey [9] on DTDs shows that a large number of real-world DTDs are recursive. We investigate whether the XML documents are indeed recursive.

Second, we investigate the number of each kind of one-to-many relationship. Such relationship can be modeled by $A \rightarrow B^*$ in DTDs, where A and B are two element types. Obviously, the number of B nodes of an A node can only be known from document instances. [3]

Statistics used. Similar to the techniques developed for relational databases, we use a few statistical terms to describe the histograms (or distributions) stored in a prefix tree. Specifically, given a distribution, we compute its minimum, maximum, average and variance. Initially, we expect some structural distributions of real-world XML datasets are normal, *i.e.* Gaussian. As mentioned in Introduction, our benchmark XML datasets are hardly normal. We found that many structural distributions are skewed. To understand the distribution, we adopt a definition of the skewness of a distribution. These numbers form the basis of our quantitative summary.

3 The Metrics

In this section, we present our metrics for XML structures. Then, we describe its meaning and possible implications of each metric. For simplicity, we often refer root-to-leaf paths to as (simple) paths.

Our metrics are listed below. When applicable, we compute the minimum, maximum, variance and average of these metrics, which quantify the structure of an XML document.

1. *The number of paths;*
2. *The length of a path;*
3. *The number of star edges in the prefix tree;*
4. *The number of star edges of a path;*
5. *The number of recursive/non-recursive element types;*
6. *The number of recursive elements of a path;*
7. *The number of a particular kind of star edges of a node;*
8. *The skewness of the number of a particular kind of star edges of a node.*

The first three metrics are some (arbitrary) basic counts of an XML document. We shall discuss the next five metrics in more detail.

The number of star edges of a path. A path p in a document must also appear in the prefix tree. We count the number of edges in p that have a star edge correspondence in the prefix tree. The number of star edges in a path implies

[3] XML SCHEMAs allow specifying the min- and max-occurrence constraints of a repetition. However, these constraints do not accurately describe the multiplicities in a conforming document instance.

the number of nested one-to-many relationship in a tree. The larger the number of star edges is, the more the tree and a relation mismatch.

The number of recursive/non-recursive element types. This metric measures the number of recursive and non-recursive element types in a document.

The number of recursive elements of a path. This metric attempts to quantify the recursiveness of an XML document to some extent. When elements can be recursively defined, *e.g.* `parlist` is defined in terms of `parlist`, the star hierarchy can only be known from the document. We define the number of recursive elements in a path p in a document T to be $\sum_{e \in R}$ (the count of e in p - 1), where R is the set of recursive element types in T. Algorithms for non-recursive XML may not work on documents with a large number of recursive elements in paths.

The number of a particular kind of star edges of a node. This metric measures the multiplicity of each kind of star edges of a node. Specifically, given a star edge (A, B) in a prefix tree, this metric counts the number of B children of an A node. The edges are often modeled by $A \rightarrow B^*$ in DTDs and a one-to-many relationship between A and B in ER diagrams.

The skewness of the number of a particular kind of star edges of a node. The distributions of the number of a particular kind of star edges of a node, i.e. the previous metric, of our XML benchmark datasets have a large variance. Hence, we investigate the distribution of the numbers obtained by the previous metric. In particular, we compute the skewness of the distributions, since data compression algorithms often work effectively on skewed data. The skewness is defined as $\sum_{i=1}^{n}(x_i - \bar{x})^3/(n\text{-}1)\sigma^3$, where n, x_i, $\bar{x}$ and σ are the size, an individual value, the mean and the standard derivation of a distribution, respectively.

External construction of prefix trees. When the amount of memory available is larger than the size of the prefix tree S of the input document T and the histogram N of star edges, one can easily compute S, the star edges and Metrics 1, 2, 5 and 6 in one pass of T. Metrics 3 can then be derived from S in the size of S. The remaining metrics are determined by a second pass of T followed by a scan on S and N. The overall complexity for computing all metrics is $2|T|+2|S|+|N|$.

When the prefix tree S does not fit into memory, we apply a simple divide-and-conquer method to construct S, while keeping the structure of depth first traversal. We outline the construction method as follows. (1) We traverse the document in depth first manner and maintain a root-to-current-node p as the traversal proceeds. (2) We construct a tree T_i for each M consecutive edges encountered during the traversal, where M is the amount of memory available. Initially, T_i contains p only. The edges in p are annotated as *edges in the previous subtrees*. The next M edges are added to T_i in a straightforward manner as in the internal construction of the prefix tree. (3) For each T_i, we build its prefix tree S_i. We assume that each S_i fit into memory comfortably. (4) We merge two consecutive prefix trees, S_i and S_{i+1}, by traversing the two trees "in parallel".

Note that during the merge, only two edges, and their associated histograms, are needed to be stored in memory. The edges with annotations in S_{i+1} are not merged as they already appeared in the previous subtrees. This operation requires $|S_i| + |S_{i+1}|$. We obtain S of T by merging S_is iteratively. Once S is constructed, even its size may be larger than the memory size, the metrics can still be computed in two passes of S. The total number of scans on T for computing all metrics is $1 + \log(|T|/M)$.

4 A Survey on XML Benchmarks

In this section, we apply our metrics on a few popular XML datasets. The main goal of this section is to illustrate how these metrics are useful to understand XML datasets and the current state of experimentations conducted by the XML community. Hence, we present, compare and visualize numbers obtained from different XML datasets, as opposed to presenting individual summary of numbers.

XML benchmark datasets. We first describe our real-world datasets. DBLP is the XML version of DBLP Computer Science bibliography datasets [17]. It contains bibliography information of conference papers, articles, books, master and PhD theses, etc. NASA is the dataset converted from legacy flat file format by NASA XML project [21]. SP is a curated protein sequence database SWISSPROT [12]. Next, we describe the synthetic datasets used. We used two XML benchmark datasets, namely XMARK [28] and XBENCH [34], denoted as XK and XB, respectively. XMARK datasets contain synthetic auction transactions. The XMARK generator [27] allows users to vary the size of the generated dataset by providing a scaling factor. We used a few scaling factors to generate our synthetic datasets. We noted that the structural properties of these datasets remained roughly the same as the dataset size varies. Hence, we report the results from XMARK datasets with scaling factor 1. For XBENCH, we used the four example XBENCH datasets, namely TC/SD, TC/MD, DC/SD and DC/MD, shipped with the data generator [32]. (TC, DC, SD and MD stand for text-centric, data-centric, single document and multiple documents, respectively.) Note that [32] may also take templates, that describe the abstract structure of the synthetic data, as an input of data generation. When XBENCH produces multiple datasets, we concatenate them into a single dataset before checking its structural properties.

Results. We apply the first two metrics on the XML datasets. We present the numbers for these datasets in Table 1.

There has been a large body of work on storing XML as an edge table [14] in relational databases. The number of joins required for evaluating an XPATH descendant step in the absence of XML indexes, is bounded by the depth of a document. Table 1 shows that the longest path in the datasets surveyed is often small. One non-trivial fact is that the variance of the length of simple paths of our datasets is very small. Consider DBLP as an example. While the length of the longest simple path is 6, the variance of the length of paths is close to zero. This indicates that the paths in DBLP are "regular". There is one exception: The

Table 1. Simple paths in the benchmark XML datasets

Dataset	DBLP	NASA	SP	XK	XB TC/SD	XB TC/MD	XB DC/SD	XB DC/MD
# of paths	7.5M	473K	2.0M	1.2M	250K	34K	158K	225
Minimum length	3	3	3	4	4	4	3	3
Maximum length	6	8	5	12	8	8	8	5
Average length	3.3	6.2	3.7	6.3	6.9	6.1	5.8	4.1
Variance	0.001	1.5	0.48	3.96	0.45	0.29	1.9	0.29

simple paths in XMARK are often lengthy (12) and the variance of their length is 3.96. This shows the paths in XMARK is rather complex.

To study the nestings of star edges, we apply the third and the fourth metrics on the benchmark XML datasets. The number of star edges in the prefix tree of DBLP, NASA, SP, XK, XB TC/SD, XB TC/MD, XB DC/SD and XB DC/MD are 52, 26, 95, 314, 12, 13, 5 and 9, respectively. Except for DBLP, SWISSPROT and XMARK, the number of star edges in the prefix tree is far fewer than 50. Recall that a star can be modeled by a one-to-many relationship in ER diagrams. When a relation is created to capture a one-to-many relationship, the number of relations required is small [31]. Therefore, most of these XML datasets can be efficiently stored and queried by using mature relational technology.

The number of star edges in a prefix tree is not sufficient for describing the hierarchy of (or the nestings in) the datasets. We apply the fourth metric – the number of star edges on paths – on the benchmark datasets. For the discussion purpose, we present the breakdown of the numbers in Table 2.

Table 2. The number of star edges in simple paths of the benchmark XML datasets

Dataset	DBLP	NASA	SP	XK	XB TC/SD	XB TC/MD	XB DC/SD	XB DC/MD
# of paths w. 1 star	3.1M	22.5K	9	521K	713	143	59.4K	160K
# of paths w. 2 stars	4.4M	102K	1.06M	452K	14K	2035	82.9K	65K
# of paths w. 3 stars	226	227K	972K	176K	224K	2623	15.2K	0
# of paths w. 4 stars	0	14.5K	0	57K	12K	24.5K	0	0
# of paths w. 5 stars	0	108K	0	4.9K	0	4113	0	0
# of paths w. 6 stars	0	0	0	0	0	341	0	0

The maximum number of star edges on paths is 6. Note also that the number of stars on a path is at least one in practice. For example, consider the `dblp` dataset again. The root node, `dblp`, of `dblp` has many `book` children and `dblp-book` is a star edge. The breakdown shows that the number of star edges on paths exhibits a Gaussian distribution. Also the average number of star edges on paths is small. That is, these datasets can be considered as a mild generalization of relations, not similar to a tall tree.

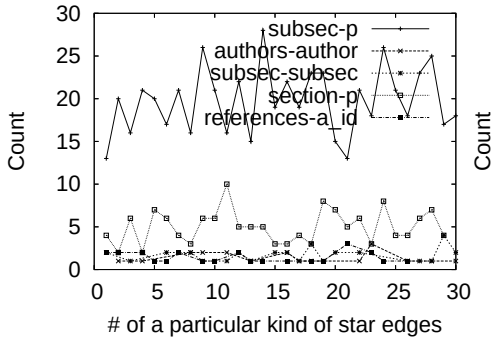

Fig. 5. The five distributions with the highest variance on the number of star edges (XBENCH TC/MD)

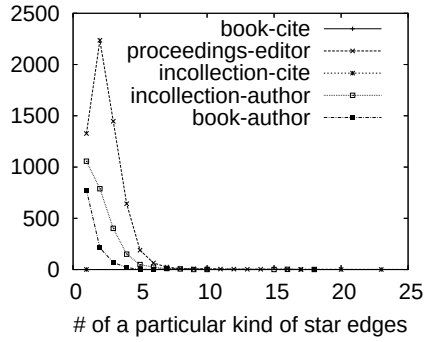

Fig. 6. The five distributions with the highest variance on the number of star edges (DBLP)

Then, we used the fifth metric to measure the number of recursive element types in the XML benchmark datasets. These datasets, including the text centric (TC) XBENCH datasets, contain only one or two recursive element types. The recursive elements in DBLP, NASA, XMARK and XBENCH TC/MD are (`sub` and `sup`), (`para`), (`listitem` and `parlist`) and (`subsec`), respectively.

Table 3. The number of recursions in simple paths in XML benchmark datasets

Dataset	DBLP	NASA	SP	XK	XB TC/SD	XB TC/MD	XB DC/SD	XB DC/MD
# of paths w. 1 recursion	15	110	0	0	0	4113	0	0
# of paths w. 2 recursions	0	0	0	112K	0	341	0	0
% of recursive paths	0%	0%	0%	9.3%	0%	13%	0%	0%

Next, we used the sixth metric to illustrate the recursions in the paths of the benchmark datasets. The results are presented in Table 3. We found that although DTDs may often be recursive, the recursions in the document instances are often simple. The only fairly recursive XML datasets in our benchmark datasets are XMARK and XBENCH TC/MD datasets. The number of recursions in the paths of XMARK is always 2 while that of XBENCH TC/MD is mostly 1. Due to the simplicity of the benchmark datasets, algorithms for non-recursive XML datasets may continue to work on these datasets. However, these datasets are insufficient to show the benefits of algorithms for recursive XML datasets.

We applied the seventh metric on the XML benchmark datasets. We visualize our results for discussion purposes. For each star edge (A, B) in the prefix tree, we obtain a distribution – the number of B children of an A node. Except for the datasets generated by XBENCH, such distributions of the datasets are highly

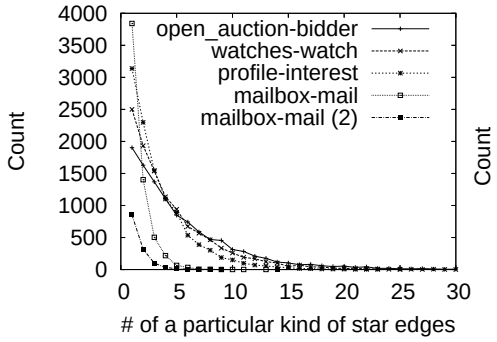

Fig. 7. The five distributions with the highest variance on the number of star edges (XMARK)

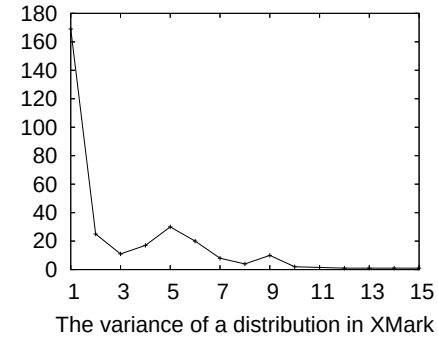

Fig. 8. The variances of the distribution of star edges in XMARK

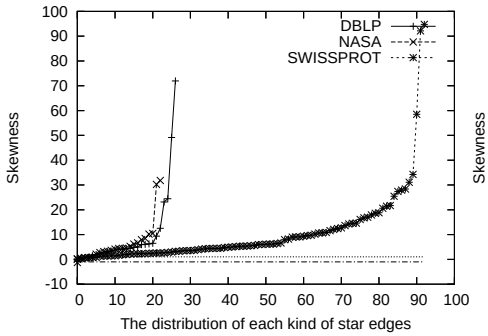

Fig. 9. Skewness of star edges in real-world dataset variance

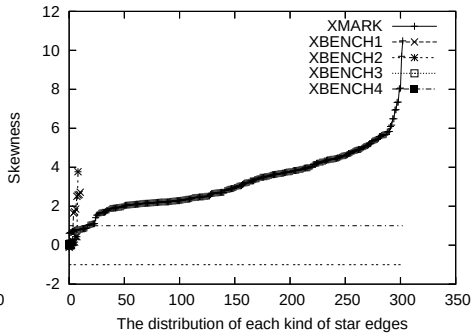

Fig. 10. Skewness of star edges in synthetic dataset

skewed. For example, we show five of such distributions with the highest variance in XBENCH in Figure 5. The figure shows that XML structures in XBENCH appear random or uniform. In comparison, as shown in Figure 6, such distributions in DBLP are highly skewed.

Another counter-intuitive finding is that while XMARK is a synthetic dataset, such distributions in XMARK are highly skewed also, shown in Figure 7. The non-zero variance of the distributions in XMARK dataset is shown in Figure 8. The x-axis is the value of variance of a distribution and the y-axis shows the number of distributions with a particular variance. The figure shows that there are many distributions with a large variance. In addition, a few distributions with a large variance are found in DBLP, NASA and SWISSPROT datasets.

We then apply the next metric to check the skewness of distributions of XML structures. The skewness of the distributions (with non-zero variance) of the real-world and synthetic datasets are presented in Figure 9 and Figure 10, respec-

tively. Figure 9 shows that the number of skewed distributions (skewness > 1) is significant. We sort the distributions according to their skewness. That is, the x-axis is a canonical number of a distribution. The figure shows that the percentage of skewed distributions in DBLP, NASA and SWISSPROT are 82%, 100% and 94%, respectively. Datasets that are generated by XBENCH are not highly skewed as it is illustrated with Figure 10. When one uses highly skewed datasets for experimentations, the results are prone to be biased.

Summary and recommendations. While XMARK and DBLP datasets appear popular in the XML research community, we found that the majority of these XML datasets are mild generalization of relations - not "tree-like". Furthermore, these datasets are highly skewed. Except XBENCH TC/MD, the other three XBENCH datasets are also a mild generalization of relations. The XBENCH datasets are not skewed. All the datasets are hardly hierarchical and contain a small number of recursions. To conduct fair experiments on algorithms for recursive XML (*e.g.* [2]), one needs to derive a highly recursive schema and datasets by using "XBENCH-like" XML generator.

5 Related Work

There has been a host of work on XML summary structures for optimizing XML query evaluation [19, 20, 16]. These approaches and ours are orthogonal: Their approaches are graph-based whereas our approach focuses on producing quantitative summaries. There have also been works on deriving statistical summaries of XML structures [7, 23, 15] for estimating selectivities of a query workload. [7] counts the number of simple paths in XML documents and determines the correlation between paths for estimating the selectivity of a given query workload. In comparison, our approach does not require query workloads as our focus is not selectivity estimation. [23] proposes statistical synopses for XML for path query selectivity estimation. Such synopses are designed for query processors, as opposed to offering a structural summary for human. STATIX [15] builds histograms on entities of an XML SCHEMA and generates the optimal relational storage of an XML document for a pariticular query workload. In comparison, our approach does not require an XML SCHEMA. We derive a prefix tree from an XML document and use it as the "schema" of the XML document. While [15] stated that some real-world XMLs are highly skewed, our result is more comprehensive and informative.

In addition to query evaluation, the structure of XML documents influences the design of XML storage scheme. For example, since XML is flexible, heuristic algorithms [11, 3] has been proposed to mine the optimal storage for XML. The storage subsequently affects query evaluations. Another stream of work is the XML query algorithms [4, 13, 25, 22, 6] that assume a specific physical layout or XML index structure. Although encouraging performances have been reported, it was not clear how the performance of a system may change as the structure of documents changes. Recently, [26] proposes a microbenchmark for understanding

the strengths and weaknesses of an XML system. Compression can be understood as a space-efficient storage. Existing XML compression techniques [18, 6, 10, 8] utilize properties of both scalar data and structures of an XML document. Since (real-world) XML documents are often skewed, XML compressions have been shown effective.

Finally, surveys on real-world DTDs and XML SCHEMAS are presented in [9] and [1], respectively. However, nestings of entities and the number of occurrences of star edges can only be computed from document instances.

6 Conclusions and Future Works

In this paper, we presented quantitative metrics for XML structures. We derived statistics from a prefix tree of XML structures and used simple paths and star edges as the basis of our metrics. These metrics are developed to answer our informal question stated in Introduction: whether an XML structure is tree-like or relational-like. We applied our metrics on a few popular XML datasets for experimental evaluations, among others, for XML research. Our result is that the structures of these XML documents are highly skewed, non-hierarchical and mostly non-recursive. That is, the datasets are relational-like.

In the future, we will use our metrics to aid the design of our ongoing native XML system [6, 10].

Acknowledgements. We would like to thank Daxin Jiang for discussions on statistics and databases and our colleagues in CAIS at NTU for providing technical discussions. This work is supported by CoE startup grant M58020002.601001.

References

1. G. J. Bex, F. Neven, and J. V. den Bussche. DTDs versus XML Schema: A Practical Study. In *WebDB*, pages 79–84, 2004.
2. P. Bohannon, B. Choi, and W. Fan. Incremental evaluation of schema-directed XML publishing. In *SIGMOD*, 2004.
3. P. Bohannon, J. Freire, P. Roy, and J. Simeon. From XML schema to relations: A cost-based approach to XML storage. In *ICDE*, 2002.
4. P. A. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner. MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In *SIGMOD*, pages 479–490, 2006.
5. V. P. Braganholo, S. B. Davidson, and C. A. Heuser. From XML view updates to relational view updates: old solutions to a new problem. In *VLDB*, 2004.
6. P. Buneman, B. Choi, W. Fan, R. Hutchison, R. Mann, and S. Viglas. Vectorizing and querying large xml repositories. In *ICDE*, pages 261–272, 2005.
7. Z. Chen, H. V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. Ng, and D. Srivastava. Counting twig matches in a tree. In *ICDE*, 2001.
8. J. Cheney. Compressing XML with multiplexed hierarchical PPM models. In *Data Compression Conference*, 2001.
9. B. Choi. What are real DTDs like. In *WebDB*, pages 43–48, 2002.

10. B. Choi. Document decomposition for XML compression: A heuristic approach. In *DASFAA*, pages 202–217, 2006.
11. A. Deutsch, M. F. Fernandez, and D. Suciu. Storing semistructured data with STORED. In *SIGMOD*, pages 431–442. ACM Press, Jun. 1999.
12. ExPASy. Swiss-prot and TrEMBL. Available at `http://www.expasy.ch/sprot/`.
13. T. Fiebig, S. Helmer, C.-C. Kanne, G. Moerkotte, J. Neumann, R. Schiele, and T. Westmann. Anatomy of a native XML base management system. *VLDB Journal*, 11(4):292–314, Dec. 2002.
14. D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. *IEEE Data Engineering Bulletin*, 22(3):27–34, 1999.
15. J. Freire, J. R. Haritsa, M. Ramanath, P. Roy, and J. Siméon. StatiX: making XML count. In *SIGMOD Conference*, pages 181–191, 2002.
16. R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for efficient indexing of paths in graph structured data. In *ICDE*, 2002.
17. M. Ley. DBLP Bibliography. Available at `http://www.informatik.uni-trier.de/~ley/db/`, Mar 2005.
18. H. Liefke and D. Suciu. XMILL: An efficient compressor for XML data. In *SIGMOD*, 2000.
19. J. McHugh and J. Widom. Query optimization for XML. In *VLDB*, 1999.
20. T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, 1999.
21. National Aeronautics and Space Administration. The NASA XML project. Available at `http://xml.nasa.gov/xmlwg/index.htm`.
22. S. Paparizos, S. Al-Khalifa, A. Chapman, H. V. Jagadish, L. V. S. Lakshmanan, A. Nierman, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A native system for querying XML. In *SIGMOD*, 2003.
23. N. Polyzotis and M. N. Garofalakis. Statistical synopses for graph-structured XML databases. In *SIGMOD*, 2002.
24. V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, pages 294–305, 1996.
25. S. Prakash, S. S. Bhowmick, and S. K. Madria. Efficient recursive XML query processing in relational database systems. In *ER*, pages 493–510, 2004.
26. K. Runapongsa, J. Patel, H. Jagadish, Y. Chen, and S. Al-Khalifa. The Michigan benchmark: Towards XML query performance diagnostics, 2003.
27. A. Schmidt. XMark – an XML benchmakr project. Available at `http://monetdb.cwi.nl/xml/generator.html`, 2003.
28. A. Schmidt, F. Waas, M. Kersten, M. J. Carey, I. Manolescu, and R. Busse. XMark: A benchmark for XML data management. In *VLDB*, pages 974–985, 2002.
29. L. Segoufin and V. Vianu. Validating streaming xml documents. In *PODS*, pages 53–64, 2002.
30. J. Shanmugasundaram, E. Shekita, and J. Kiernan. A general technique for querying XML documents using a relational database system. *SIGMOD Record*, 30(3):20–26, 2001.
31. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. *VLDB Journal*, pages 302–314, 1999.
32. ToXGene. The ToX XML generator. Available at `http://www.cs.toronto.edu/tox/toxgene/`, 2005.
33. W3C. Extensible Markup Language (XML). Available at `http://www.w3.org/XML/`.
34. B. B. Yao, M. T. Ozsu, and N. Khandelwal. XBench benchmark and performance testing of XML DBMSs. In *ICDE*, pages 621–633, 2004.