Outlier Detection For XML Documents (Extended Abstract)

Giuseppe Manco and Elio Masciari

ICAR-CNR
{manco,masciari}@icar.cnr.it

Abstract. XML (eXtensible Markup Language) became in recent years the new standard for data representation and exchange on the WWW. This has resulted in a great need for data cleaning techniques in order to identify outlying data. In this paper, we present a technique for outlier detection that single out anomalies with respect to a relevant group of objects. We exploit a suitable encoding of XML documents that are encoded as signals of fixed frequency that can be transformed using Fourier Transforms. Outlier are identified by simply looking at the signal spectra. The results show the effectiveness of our approach.

1 Introduction

An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism [8]. There exist several approaches to the identification of outliers, namely, statistical-based [5], deviation-based [4], distance-based [3], density-based [6], projection-based [1], MDEF-based [12], and others. Abstracting from the specific method being exploited the general outlier detection task is the problem of identifying deviations from the general patterns characterizing a data set. Detecting outliers is important in many application scenarios, as an example it can be used for improving data cleaning approaches, where outliers are often data noise or errors diminishing the accuracy of data mining. Outlier detection is also the core of applications such as fraud detection, stock market analysis, intrusion detection, marketing, network sensors, and email spam detection, where irregular patterns entail special attention. Due to the increasing usage of semi-structured data models like XML (eXtensible Markup Language), that is the new standard for data representation and exchange on the Web there is a great need for outlier detection strategies entailed for such data. Although outlier detection methods are well established for relational data, adapting them directly to XML data is unfeasible because XML and relational data models differ in several aspects. First, XML data contain multiple levels of nested elements (or attributes) organized in a tree-based structure, whereas relational data models have a flat tabular structure. Indeed, the hierarchical structure of XML data induce an ordering lacking in relational data. Also, the modeling objectives for XML and relational data are different, and therefore different relationships are represented. In relational data models, the primary-foreign key relationships between entities form the basis for data normalization and referential integrity. On the contrary, relationships between the XML elements are encoded in hierarchies, often with direct semantic correspondence to the real-world relations such as containment and composition.

Despite its importance, XML outlier detection has not been paid the attention it deserves. There exists few works addressing structural and attribute outlier detection for XML. The main distinction is between class outlier and attribute outlier, i.e. respectively outlier based on the overall structure of the document and outlier based on univariate points that exhibit deviating correlation behavior with respect to other attributes[10]. In [10] an approach is presented for correlation based attribute outlier detection, while main approaches for class outlier detection try to adapt the techniques defined for the relational setting to the semistructured one. They have been mainly proposed for data cleaning purposes like in [15, 14, 16].

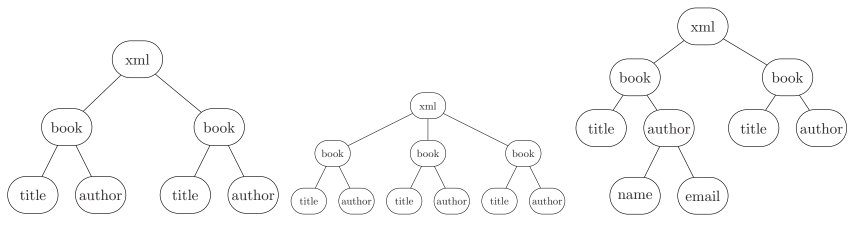
Our approach. We will tackle in this paper the class outlier detection problem. The basic intuition exploited in this paper is that an XML document has a "natural" interpretation as a time series (namely, a discrete-time signal), in which numeric values summarize some relevant features of the elements enclosed within the document. We can get an example evidence of this observation by simply indenting all the tags in a given document according to their nesting level. Indeed, the sequence of indentation marks (as they appear within the document rotated by 90 degrees), can be looked at as a time series, whose shape roughly describes the document's structure. Hence, a key tool in the analysis of time-series data is the use of the Discrete Fourier Transform (DFT): some useful properties of the Fourier transform, such as energy concentration or invariance under shifts, enable to analyze and manipulate signals in a very powerful way. The choice of comparing the frequency spectra follows both effectiveness and efficiency issues. Indeed, the exploitation of DFT leads to abstract from structural details which, in most application contexts, should not affect the similarity estimation (such as, e.g., different numbers of occurrences of a shared element or small shifts in the actual positions where it appears). This eventually makes the comparison less sensitive to minor mismatches. Moreover, a frequency based approach allows to estimate the similarity through simple measures (e.g., vector distances) which are computationally less expensive than techniques based on the direct comparison of the original document structures. To summarize, we propose to represent the structure of an XML document as a time series, in which each occurrence of a tag in a given context corresponds to an impulse. By analyzing the frequency spectra of the signals, we can hence state the degree of (structural) similarity between documents. It is worth noticing that the overall cost of the proposed approach is only $O(N \log N)$, where N is the maximum number of tags in the documents to be compared. Once defined an effective distance measure we will exploit a distance-based outlier detection algorithm in order to single out the outlying documents.

2 Problem Statement and Overview of the Proposal

We begin by presenting the basic notation for XML documents that will be used hereafter. An XML document is characterized by *tags*, i.e., terms enclosed between angled brackets. Tags define the structure of an XML document and provide the semantics of the information enclosed. A tag is associated with a *tag name* (representing the term enclosed between angled brackets), and can appear in two forms: either as a *start-tag* (e.g., `<author>`), or as an *end-tag* (characterized by the `/` symbol, like, e.g., in `</author>`). Finally, a *tag instance* denotes the occurrence of a tag within a certain document. It is required that, in a well-formed XML document, tags are properly nested, i.e. each start-tag has a corresponding end-tag at the same level. Therefore, an XML document can be considered as an ordered tree, where each node (an *element*) represents a portion of the document, delimited by a pair of start-tag and end-tag instances, and denoted by the tag name associated with the instances. The structure of an XML document corresponds to the shape of the associated tree. In a tree, several types of structural information can be detected, which correspond to different refinement levels: for example, attribute/element labels, edges, paths, subtrees, etc. Defining the similarity among two documents essentially means choosing an appropriate refinement level and comparing the documents according to the features they exhibit at the chosen level. Different choices may result in rather dissimilar behaviors: in particular, comparing simple structural components (such as, e.g., labels or edges) allows for an efficient computation of the similarity, but typically produces loose-grain similarity values. On the other hand, complex structural components would make the computation of similarity inefficient, and hence unpractical.

Consider, for example, the documents represented in Fig. 1. If a comparison of nodes or edges is exploited, documents *book1* and *book2* appear to be similar, even though the subtrees rooted at the `book` element appear with different frequencies. Accounting for frequencies does not always help: for example, if the order of appearance of the subtrees of the `xml` element in *book3* were changed, the resulting tree still should have the same number of nodes, edges and even paths.

In principle, approaches based on tree-edit distance [11] can better quantify the difference between XML trees; however, they turn out to be too expensive in many applications contexts, as they are generally quadratic w.r.t. document sizes. Finally, notice that solutions based on detecting local substructures [9] to be used as features may be even hard to handle, for they showing two main disadvantages: first, they may imply ineffective representations of the trees in high dimensional spaces, and second, costly feature extraction algorithms are required.



(a) *book1* (b) *book2* (c) *book3*

Fig. 1. *book1* and *book2* have the same elements, but with different cardinality. By contrast, *book3* induces a different structure for the `author` element.

In our opinion, an effective notion of structural similarity should take into account a number of issues. First of all, it is important to notice that each document may induce a definition of the elements involved. Thus, an appropriate comparison between two documents should rely on the comparison of such definitions: the more different they are, the more dissimilar are the documents and this information has to be exploited for signaling candidate outliers. Our main objective is the development of an efficient method which is able to approximate the above features at best. Thus, we can state the problem of finding the structural similarity in a set of XML documents as follows. Given a set $D = \{d_1, \ldots, d_n\}$ of XML documents, we aim at building a similarity matrix S, i.e., a matrix representing, for each pair (d_i, d_j) of documents in D, an optimal measure of similarity s_{ij}. Here, optimality refers to the capability of reflecting the above described differences. Observe that we do not address here the problem of finding which parts of two documents are similar or not, as, e.g., tree-edit based techniques do. We propose a technique which is essentially based on the idea of associating each document with a time series representing, in a suitable way, both its basic elements and their relationships within the document. More precisely, we can assume a *preorder* visit of the tree-structure of an XML document. As soon as we visit a node of the tree, we emit an impulse encoding the information corresponding to the tag. The resulting signal shall represent the original document as a time series, from which relevant features characterizing a document can be efficiently extracted. As a consequence, the comparison of two documents can be accomplished by looking at their corresponding signals.

The main features of the approach can be summarized as follows: 1) Each element is encoded as a real value. Hence, the differences in the values of the sequence provide for an evaluation of the differences in the elements contained by the documents; 2) The substructures in the documents are encoded using different signal shapes. As a consequence, the analysis of the shapes in the sequences realizes the comparison of the definitions of the elements. 3) Context information can be used to encode both basic elements and substructures, so that the analysis can be tuned to handle in a different way mismatches which occur at different hierarchical levels.

In a sense, the analysis of the way the signal shapes differ can be interpreted as the detection of different definitions for the elements involved in the documents. Moreover, the analysis of the frequencies of common signal shapes can be seen as the detection of the differences between the occurrences associated with a repetition marker. In this context, the proposed approach can be seen as an efficient technique, which can satisfactorily evaluate how much two documents are similar w.r.t. the structural features previously discussed. Notably, the use of time-series for representing complex XML structures, combined with an efficient frequency-based distance function, is the key for quickly evaluating structural similarities: if N is the maximum number of tags in two documents, they can be compared in only $O(N \log N)$ time. In particular, the use of DFT supports the above described notion of similarity: if two documents share many elements having a similar definition, they will be recognized as similar, even when there are repeated and/or optional sub-elements. Indeed, working on frequency spectra makes the comparison less sensitive to the differences in the number of occurrences of a given element and to small shifts in the actual positions where it occurs in the documents. The details on representing an XML document as a signal are omitted here due to space limitations, a complete explanation can be found in [7].

3 Comparing Documents using DFT

Once defined a proper document encoding, we can now detail the similarity measures for XML documents, sketched in section 1. As already mentioned, we can assume that we are visiting the tree-structure of an XML document d (using a preorder visit) starting from an initial time t_0. We also assume that each tag instance occurs after a fixed time interval Δ. The total time spent to visit the document is $N\Delta$, where N is the size of $tags(d)$. During the visit, as we find a tag, we produce an impulse which depends on a particular tag encoding function e and on the overall structure of the document (i.e., the document encoding function enc). As a result of the above physical simulation, the visit of the document produces a signal $h_d(t)$, which usually changes its intensity in the time interval $[t_0, t_0 + N\Delta)$. The intensity variations are directly related to the opening/closure of tags:

$$h_d(t) = \begin{cases} [enc(d)](k) & \text{if } t_0 + k\Delta \leq t < t_0 + (k+1)\Delta \\ 0 & \text{if } t < t_0 \text{ or } t \geq t_0 + N\Delta \end{cases}$$

Comparing such signals, however, might be as difficult as comparing the original documents. Indeed, comparing documents having different lengths, requires the combination of both resizing and alignment operations. Moreover, the intensity of a signal strongly depends on the encoding scheme adopted, which can in turn depend from the context (as in the case, e.g., of the multilevel encoding scheme).

In order to compare two documents d_i and d_j, hence, we can exploit the properties of the corresponding transforms. In particular [2, 13], a possibility is to exploit that, by Parseval's theorem, energy (total power) is an invariant in the transformation (and hence the information provided by the encoding remains unchanged in the transform). However, a more effective discrimination can exploit the difference in the magnitude of frequency components: in a sense, we are interested (i) in abstracting from the length of the document, and (ii) in knowing whether a given subsequence (representing a subtree in the XML document) exhibits a certain regularity, no matter where the subsequence is located within the signal. In particular, we aim at considering as (almost) similar documents exhibiting the same subtrees, even if they appear at different positions. Now, as the encoding guarantees that each relevant subsequence is associated with a group of frequency components, the comparison of their magnitudes allows the detection of similarities and differences between documents. Observe that measuring the energy of the difference signal would result in a low similarity value. On the other side, if the phases of the documents' transforms are disregarded, documents are more likely to be considered as similar.

A viable approximation can be the interpolation of the missing coefficients starting from the available ones. It is worth noticing that the approximation error due to interpolation is inversely proportional to $\min(N_{d_i}, N_{d_j})$: the more elements are available in a document d, the better the `DFT` approximates the (continuous) Fourier Transform of the signal $h_d(t)$, and consequently the higher is the degree of reliability of interpolation. As a practical consequence, the approach is expected to exhibit good results with large documents, providing poorer performances with small documents.

Definition 1. *Let d_1, d_2 be two XML documents, and enc a document encoding function, such that $h_1 = enc(d_1)$ and $h_2 = enc(d_2)$. Let `DFT` be the Discrete Fourier Transform of the (normalized) signals. We define the* Discrete Fourier Transform *distance of the documents as the approximation of the difference of the magnitudes of the `DFT` of the two encoded documents:*

$$dist(d_1, d_2) = \left(\sum_{k=1}^{M/2} \left(\left| [\tilde{DFT}(h_1)](k) \right| - \left| [\tilde{DFT}(h_2)](k) \right| \right)^2 \right)^{\frac{1}{2}}$$

where $\tilde{DFT}$ is an interpolation of `DFT` to the frequencies appearing in both d_1 and d_2, and M is the total number of points appearing in the interpolation, i.e., $M = N_{d_i} + N_{d_j} - 1$ points. □

3.1 Outlier Identification

Once defined a technique to state the similarity between two XML documents we need to define a strategy that, exploiting such a technique, identifies anomalies in the data set.

Definition 2 (Fourier Based XML Outlier). *Given a set of XML documents S, a positive integer k, and a positive real number R, a documents $d \in S$ is a $DB(k, R)$-outlier, or a distance-based outlier with respect to parameters k and R, if less than k objects in S lie within distance R from o w.r.t. our distance metric.*

The threshold values R and k has to be chosen depending on the scenario being monitored. Once defined our notion of outlier, we can design an effective method for outlier detection.

Algorithm 1 *Function* Compute Outlier:

```
***INPUT:***    a set of XML documents $\mathcal{S} = \{d_1, \cdots, d_n\}$, a pair of threshold
values $R$ and $k$, an XML document $d_{new}$;
***OUTPUT:***   $Yes$ if $d_{new}$ is an outlier no otherwise;
***begin***
   $temp = 0$
   ***For each*** $d_i \in \mathcal{S}$ ***do***
      $dist = computeDFTDistance(d_i, d_{new})$
      ***if*** $dist > R$
         $temp = temp + 1$
   ***if*** $temp > k$ ***return*** $Yes$
      ***return*** $No$;
***end***
```

Function *computeDFTDistance* evaluates the *DFT* distance between the XML documents being analyzed and the XML documents previously collected. If the computed distance is greater than the threshold distance set by the user we will increase an auxiliary variable *temp*. If *temp* is greater than the threshold value k the document is marked as outlying.

Proposition 1. *Algorithm 1 works in time* $O(|S|\dot{N}\log(N))$.

The running time can be trivially computed by observing that for each document being analyzed we have to compute the Fourier Transformation and this operation is performed in $O(N\log(N))$ time that is the dominant operation for the Algorithm.

4 Experimental Results

In this section, we present the experiments we performed to assess the effectiveness of the proposed approach in detecting outliers. To this purpose, a collection of tests is performed, and for each test some relevant groups of homogeneous documents (*document classes*) are considered. The direct result of each test is a similarity matrix representing the degree of similarity for each pair of documents in the data set and the number of detected outliers. The evaluation of the results relies on some *a priori* knowledge about the document classes being used that was obtained by domain experts or available from the datasets providers. We performed several experiments on a wide variety of real datasets. More in detail we report here due to space limitations the results on the following data.

The documents used belong to two main classes:1) `Astronomy`, a data set containing 217 documents extracted from an XML-based metadata repository, that describes an archive of publications owned by the *Astronomical Data Center* at NASA/GSFC (`http://adc.gsfc.nasa.gov/`); 2) `Sigmod`, a data set composed of 51 XML documents containing issues of SIGMOD Record. Such documents were obtained from the XML version of the ACM SIGMOD Web site.

For each class we added some outlying documents by perturbating the original DTD's. We compared our approach with the one proposed in [16], we refer to it as *Noise*. In order to perform a simple quantitative analysis we produce for each test a similarity matrix, aimed at evaluating the resulting *neighbors* similarities (i.e., the average of the values computed for documents belonging to the same class), and to compare them with the *outer* similarities (i.e., the similarity computed by considering only documents belonging to different classes). To this purpose, values inside the matrix can be aggregated according to the class of membership of the related elements: given a set of documents belonging to n prior classes, a similarity matrix S about these documents can be summarized by a $n \times n$ matrix CS, where the generic element $CS(i, j)$ represents the average similarity between class i and class j.

$$CS(i,j) = \begin{cases} \frac{\sum_{x,y \in C_i, x \neq y} DIST(x,y)}{|C_i| \times (|C_i|-1)} & \text{iff } i = j \\ \frac{\sum_{x \in C_i, y \in C_j} DIST(x,y)}{|C_i| \times |C_j|} & \text{otherwise} \end{cases}$$

where $DIST(x, y)$ is the chosen distance metric ($Noise$ metric or our $Fourier$ metric).

The above definition is significant since we normalize the metric value so eventually we can use different approaches, this will allow us to compare performance in the ideal setting for any approach.

The higher are the values on the diagonal of the corresponding CS matrix w.r.t. those outside the diagonal, the higher is the ability of the similarity measure to separate different classes. In the following we report a similarity matrix for each dataset being considered, as it will be clear the reported results show that our technique is quite effective for outlier detection. In particular, the similarity matrix will give an intuition about the ability of the approach to catch the neighboring documents for each class while the number of outliers detected for each dataset is reported in a separate table. We used for the experiments the following parameter values: as k value the maximum number of documents supposed to belong to each class and as R value the average distance between documents belonging to the same class.

Measuring Effectiveness for Astronomy. For this dataset our prior knowledge is the partition of the documents into two classes. As it is easy to see in Figure 2(a) and (b) $Fourier$ outperforms $Noise$ by allowing a perfect assignment of the proper neighboring class to each document.

$Noise$	Class 1	Class 2
Class 1	0.9790	0.8528
Class 2	0.8528	0.9915

(a)

$Fourier_{2D}$	Class 1	Class 2
Class 1	1	0.6250
Class 2	0.6250	1

(b)

Fig. 2. $Noise$ and $Fourier$ similarity matrices for $Astronomy$ dataset

In Figure 3 the number of detected outlier is reported. The actual number of outliers for each class is 7 so as it is easy to see $Fourier$ exactly detected all the outliers in the dataset, such a result is quite understandable considering that the similarity matrix for $Fourier$ exactly recognized the neighboring documents.

$Method$	Class 1	Class 2
$Noise$	5	4
$Fourier$	7	7

Fig. 3. $Noise$ and $Fourier_{2D}$ number of detected outliers for $Astronomy$ dataset

Measuring Effectiveness for Sigmod. In this case there were 3 main classes as it is shown in Figure 4(a) and (b). Also in this case $Fourier$ outperforms $Noise$. As we can see, differences among the various classes are marked with higher precision by $Fourier$. This is mainly due to the fact that our approach is quite discriminative since it takes into account all the document features. For this dataset the number of actual outliers was 8 for each class, as it is easy to see in Figure 5 $Fourier$ still outperforms $Noise$ for this dataset.

Noise	Class 1	Class 2	Class 3
Class 1	0.9986	0.7759	0.7055
Class 2	0.7759	0.9889	0.7566
Class 3	0.7055	0.7566	0.9920

(a)

Fourier	Class 1	Class 2	Class 3
Class 1	0.9885	0.7439	0.7108
Class 2	0.7439	0.9899	0.7223
Class 3	0.7108	0.7223	0.9874

(b)

Fig. 4. *Noise* and *Fourier* similarity matrices for *Sigmod* dataset

Method	Class 1	Class 2	Class 3
Noise	6	4	5
Fourier	7	8	8

Fig. 5. *Noise* and *Fourier* number of detected outliers for *Sigmod* dataset

5 Conclusion

In this paper we addressed the problem of detecting outliers in XML data. The technique we have proposed is mainly based on the idea of representing a document as a signal. Thereby, the similarity between two documents can be computed by analyzing their Fourier transforms thus defining a distance measure that can be exploited to define distance based outliers. Experimental results showed the effectiveness of the approach in detecting outlying XML documents.

References

1. C.C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *SIGMOD01*, 2001.
2. R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *FODO'93*, pages 69–84, 1993.
3. F. Angiulli and F. Fassetti. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. *TKDD*, 3(1), 2009.
4. A. Arning, C. Aggarwal, and P. Raghavan. A linear method for deviation detection in large databases. In *KDD'96*, page 164169, 1996.
5. V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley & Sons, 1994.
6. M.M. Breunig, H. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD00*, 2000.
7. S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. Fast detection of xml structural similarity. *IEEE TKDE*, 17(2):160–175, 2005.
8. D. Hawkins. *Identification of Outliers. Monographs on Applied Probability and Statistics.* Chapman & Hall, 1980.
9. H. Kashima and T. Koyanagi. Kernels for semi-structured data. In *Procs. Int. Conf. on Machine Learning (ICML'02)*, pages 291–298, 2002.
10. Judice L. Y. Koh, Mong Li Lee, Wynne Hsu, and Wee Tiong Ang. Correlation-based attribute outlier detection in xml. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1522–1524, Washington, DC, USA, 2008. IEEE Computer Society.
11. A. Nierman and H.V. Jagadish. Evaluating structural similarity in XML documents. In *Procs. 5th Int. Workshop on the Web and Databases (WebDB 2002)*, 2002.
12. S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *ICDE'03*, page 315326, 2003.
13. D. Rafiei and A. Mendelzon. Efficient retrieval of similar time series. In *FODO'98*, 1998.
14. Choh Man Teng. Polishing blemishes: Issues in data correction. *IEEE Intelligent Systems*, 19:34–39, 2004.
15. Melanie Weis and Felix Naumann. Dogmatix tracks down duplicates in xml. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 431–442, New York, NY, USA, 2005. ACM.
16. Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: a quantitative study of their impacts. *Artif. Intell. Rev.*, 22:177–210, November 2004.