Similarity of XML Schema Fragments Based on XML Data Statistics[*]

Irena Mlynkova and Jaroslav Pokorny
Charles University, Faculty of Mathematics and Physics, Department of Software Engineering
Malostranske nam. 25, 118 00 Prague 1, Czech Republic
{irena.mlynkova,jaroslav.pokorny}@mff.cuni.cz

Abstract

As XML has become a standard for data representation, it can be found in plenty of information technologies. A possible optimization of XML-based approaches can be exploitation of similarity of XML data.

In this paper we propose a technique for evaluating similarity of XML schema fragments focusing on two often omitted aspects – structural level of similarity and tuning of parameters of the similarity measure. In the former case we exploit the results of statistical analysis of real-world XML data. In the latter case we show that the tuning problem is a kind of constraints optimization problem and can be solved using corresponding approaches. We have analyzed (dis)advantages of two of them, genetic algorithms and simulated annealing, and in further experiments we show that appropriate tuning produces a more precise similarity measure.

1 Introduction

The XML [4] has become a standard for data representation and can be found in most areas of information technologies. A possible optimization of XML-based methods is exploitation of similarity of XML data. It enables to manage similar XML data in a similar manner or to extend approaches for particular XML data to the whole set of similar ones. But though the amount of existing similarity-based approaches is significant, there is still a space for further improvements.

In this paper we propose a similarity measure designed primarily for the purpose of enhancing of *user-driven* XML-to-relational storage strategies [10]. The main idea is to apply storage strategies specified for selected schema fragments to all similar fragments. But the key ideas can be simply extended to any appropriate similarity-based problem. Our method differs mainly in two aspects: Firstly, it focuses on structural similarity of the given schema fragments instead of semantics of element/attribute names used in most of existing works. Since the key aim of XML-to-relational storage strategies is to find the most efficient way XML data are stored into relations, the structural analysis has key impact. Secondly, we deal with tuning of parameters of the similarity measure, an aspect which is usually omitted. For this purpose we exploit the results of statistical analysis of real-world XML data [11] and we show that the tuning problem is a kind of constraints optimization problem and thus can be solved using respective approaches. We use two of them, genetic algorithms and simulated annealing, whose properties we analyze using an experimental implementation. Using further experiments we show that with appropriate tuning the similarity measure is much precise than a common "reasonable" setting usually used.

The rest of the paper is structured as follows: Section 2 overviews the existing related works. Section 3 describes the proposed similarity measure and Section 4 provides results of experimental testing. Finally, Section 5 provides conclusions.

2 Related Work

The number of existing works in the area of XML data similarity evaluation is significant. In case of document similarity we distinguish techniques expressing the similarity of two documents D_1 and D_2 by measuring how difficult is to transform D_1 into D_2 or vice versa (e.g. [13]) and techniques which specify a simple and reasonable representation of D_1 and D_2 that enables their efficient comparison and similarity evaluation (e.g. [14]). In case of similarity of document D and schema S there are also two types of strategies – techniques which measure the number of elements which appear in D but not in S and vice versa (e.g. [3]) and techniques which measure the closest distance between D and "all" documents valid against S (e.g. [12]). Fi-

[*]This work was supported by the National Programme of Research (Information Society Project number 1ET100300419).

nally, methods for measuring similarity of two XML schemes S_1 and S_2 exploit and combine various supplemental information and measures such as, e.g., predefined similarity rules, similarity of element/attribute names, equality of data types and structure, schema instances, thesauri, previous results, etc. (e.g. [5, 8])

For choosing the best XML-to-relational storage strategy the key information lies in structural analysis of XML data. Thus, the corresponding measure should focus on structural level. And since the best source of structural information are XML schemes, also the similarity should be evaluated primarily on schema level. In this area the key emphasis is put on the semantic similarity reflecting the requirements of corresponding applications (such as schema-integration systems [9], dissemination-based systems [1], etc.). But for the purpose of XML-to-relational storage strategies such techniques are inappropriate.

3 Proposed Similarity Evaluation

The proposed similarity measure $sim(f_x, f_y) \in [0, 1]$ expressing similarity of two fragments f_x and f_y from space Φ of schema fragments, where 1 represents strong similarity and 0 strong dissimilarity, is based on a similar idea as most of the existing works [5,8]. It exploits a number of supplemental *matchers*, i.e. functions which evaluate similarity of a particular feature of the given schema fragments, such as, e.g. in our case, similarity of number of nodes of f_x and f_y, similarity of their depths, similarity of their contents, etc.

Definition 1 *A* matcher *is a function* $m : \Phi^2 \to [0, 1]$ *which evaluates similarity of a particular feature of schema fragments* $f_x, f_y \in \Phi$.

Then the partial results are aggregated into the resulting composite similarity value.

Definition 2 *A* composite similarity measure *is a function* $m_{comp} : [0, 1]^p \to [0, 1]$ *which aggregates results of p matchers into the total similarity value.*

The most common and verified [5] way of composition is usually a kind of weighted sum.

3.1 Matchers and Composite Measure

For definition of matchers m_1, m_2, ..., m_p we exploit most of the XML data characteristics developed in the analysis [11]. Since we want to describe the structure of the schema fragments as precisely as possible, their amount is significant. On the other hand, at this stage the versatility of the approach becomes evident, since in general any kind of matchers can be used depending on the requirements of corresponding application.

According to the scope the used characteristics can be divided into the following groups:

- *root* – characteristics of root node of the fragment, e.g. type of content (empty, text, element, mixed, etc.), element/attribute fan-out, etc.
- *subtree* – characteristics of the whole fragment, e.g. number of elements/attributes, number of unordered contents, depths, etc.
- *level* – characteristics of each level of the fragment, e.g. number of attributes, minimum/maximum fan-outs, etc.

Since each matcher should evaluate similarity of particular characteristic of fragments f_x and f_y, we transform their values to interval $[0, 1]$. For root characteristics we distinguish feature matchers and single-valued matchers. *Feature matchers* express the (in)equality of the value of i-th feature fea_i (e.g. type of content):

$$m_i^{fea}(f_x, f_y) = \begin{cases} 1 & fea_i(f_x) = fea_i(f_y) \\ 0 & otherwise \end{cases} \tag{1}$$

and they are combined into *composite feature matcher*:

$$m^{fea}(f_x, f_y) = \sum_{i=1}^{n} m_i^{fea}(f_x, f_y) \cdot w_i^{fea} \tag{2}$$

where $w_i^{fea} \in [0, 1]$, $\sum_{i=1}^{n} w_i^{fea} = 1$, and n is the number of feature matchers. *Single-valued matchers* express the difference between the value of j-th single-valued characteristic $value_j$ (e.g. element fan-out):

$$m_j^{single}(f_x, f_y) = \frac{1}{|value_j(f_x) - value_j(f_y)| + 1} \tag{3}$$

Subtree characteristics also involve single-valued characteristics (e.g. number of elements), hence we use single-valued matchers composed into *composite single-valued matchers* m^{single} similarly to (2). Multi-valued characteristics (e.g. allowed depths of a schema fragment) require similarity evaluation of two lists of values of arbitrary lengths. Thus, we supply the shorter list with zero values, we sort the lists in decreasing order, and we use *multi-valued matchers* which express the similarity of a j-th sorted sequence s_j:

$$m_j^{multi}(f_x, f_y) = \frac{\sum_{k=1}^{m} \frac{1}{|s_j(f_x)[k] - s_j(f_y)[k]| + 1}}{m} \tag{4}$$

where m is the length of the sequences and $seq_j(.)[k]$ expresses the k-th member of the sequence.

For level characteristics (e.g. minimum fan-out per level) we use *level matchers* which compose the results of single (3) or multi-valued (4) matchers at single levels and decrease their weights with the growing level:

$$m_j^{lev}(f_x, f_y) = \sum_{k=1}^{l} m_j^{single/multi}(f_x, f_y) \cdot (\frac{1}{2})^k \quad (5)$$

where l is the maximum of number of levels of f_x and f_y (assuming that the shallower one is again supplied with zero results of the parameters).

Finally, the resulting composite function m_{comp} is expressed as a weighted sum of all the matchers.

3.2 Tuning of Weights

In existing works the weights of the matchers are set either without any argumentation or a machine-learning strategy is exploited. In our approach we use the "golden mean" exploiting the experience from the analysis of real-world XML schemes [11]. The basic idea is relatively simple: We use the same 98 real-world XML schemes divided into database (dat), document (doc), exchange (ex), report (rep), and research (res) category. Their characteristics are listed in Table 1.

Characteristic		dat	doc	ex	rep	res
Num. of schemes		31	18	38	4	7
Num. of elements	min	7	5	5	109	28
	max	76	377	523	3,213	250
Depth	min	2	4	2	3	5
	max	12	81	79	5	15

Table 1. Characteristics of XML schemes

The first two categories are similar to classical data-centric and document-centric ones, the other three are introduced in [11] to enable finer division. Then we prepare sample patterns of real schema fragments, e.g. data-centric, document-centric, unordered, recursive, etc., whose representation is in the particular categories known. Finally, we compute the number of occurrences of similar fragments within the schema categories and tune the parameters of the similarity measure so that the results correspond to the results of the analysis.

Note that this is the second stage where the algorithm can be modified to any purpose. It general it is possible to use any relevant information, i.e. knowledge of characteristics of any sample set of data.

3.2.1 Theoretical View of the Tuning Problem

In general the tuning problem and its solution can be described as follows: Let $c_1, c_2, ..., c_K$ denote the categories of schemes, $p_1, p_2, ..., p_P$ the sample patterns, and $(M_{i,j}^{rep})_{K \times P}$ the *representation matrix* which contains *real-world representation* of pattern p_j in category c_i, i.e. results of the analysis. Next let us have a search algorithm with parameters $par_1, par_2, ..., par_R$, where $\forall i : par_i \in [0, 1]$ and some subsets of the parameters have to fulfill constraints, such as, e.g., the sum of parameters which correspond to weights of a weighted sum must be equal to 1. With a setting of parameters the algorithm returns *calculated representation* $rep_{i,j}$ of pattern p_j in category c_i. The aim is to find the optimal setting of the parameters, where the sum of deviations of calculated and real-world representations

$$\Delta = \sum_{i=1}^{K} \sum_{j=1}^{P} |M^{rep}[i, j] - rep_{i,j}| \quad (6)$$

is minimal. This task is obviously a kind of a classical *constraints optimization problem (COP)* [2] – a problem of finding a solution in a *feasible region* (i.e. a space of all possible solutions), where the value of *objective function* (i.e. a function which determines quality of a solution) is optimal and the solution satisfies the given criteria. Since our feasible region is theoretically infinite, we search for a suboptimal solution of COP using two heuristics – *genetic algorithms* and *simulated annealing*. They enable to find a reasonable setting following the given requirements and influence the number of expensive evaluations.

Genetic Algorithms *Genetic algorithms (GA)* [6] are a part of evolutionary algorithms inspired by evolution biology. The idea is based on iterative improving of *initial population* P_0 of individuals using two simulations of natural processes – *crossover* and *mutation*. At i-th iteration the *fitness* f_{fit}, i.e. the quality of every individual of population P_i is evaluated, the best individuals are selected, and modified, i.e. crossed over and mutated to form a new population P_{i+1}. Crossover creates a new offspring by exchanging portions of two individuals. Mutation creates a new offspring by changing attributes of an existing one. Both the operations are performed with given probabilities P_{cross} and P_{mut} which influence the speed of convergence to the suboptimal solution. The algorithm terminates either if satisfactory *fitness level* F_{min} has been reached in population P_i or after N iterations.

In our case a single individual of a population corresponds to a single possible setting of parameters $par_1, par_2, ..., par_R$ and the fitness function evaluates the inverse value of Δ. Crossover and mutation are modified to ensure that the parameters fulfill the previously described conditions of weighted sums.

Simulated Annealing *Simulated annealing (SA)* [7] is inspired by the way metal cools and freezes into crystalline structure, where controlled cooling increases size of the crystals and thus reduces defects.

SA starts with the *initial state* s_0 which is iteratively improved. The quality of a state s_i is evaluated using its *energy* $E(s_i)$ which needs to be minimized. At i-th iteration the current state s_i is replaced with a random "nearby" state s_{i+1} whose choice depends on a global parameter T called *temperature* which is gradually decreased during the process. The probability P_{mov} of moving from state s_i to s_{i+1} is expressed as a function of T, $E(s_i)$, and $E(s_{i+1})$:

$$P_{mov} = \begin{cases} 1 & E(s_i) > E(s_{i+1}) \\ exp(\frac{E(s_i)-E(s_{i+1})}{T}) & otherwise \end{cases} \tag{7}$$

SA terminates either after a certain number of iterations N or if a state with satisfactory energy E_{min} is reached. The main advantage of SA is its ability to avoid trapping at local optimum, since SA does not accept only states which improve the current optimum, but the probability P_{mov} and temperature T ensure that at the beginning the state changes almost arbitrarily, but the changes decrease as T goes to zero.

In our case each state represents a single setting of parameters par_1, par_2, ..., par_R and the energy E evaluates Δ. The neighboring states are defined by random change of a single parameter, whereas others are recomputed to fulfill conditions of weighted sums.

4 Experimental Tests

With correct setting of parameters of both the algorithms (i.e. P_{cross} and P_{mut} of GA and T and P_{mov} of SA [10]) we can analyze their quality and behavior. Table 2 overviews the quality of the suboptimums expressed using the result of Δ and numbers of iterations necessary for reaching the suboptimums. Moreover, the algorithms start either with random initial population P_0 and state s_0 or with average-producing weights (denoted as avg). Apparently for both GA and SA the results are better when we start with a reasonable setting than with a random one. But, though the values of the two algorithms do not differ too much, the results of GA are better in both quality and efficiency. It is probably caused by its ability to improve a population of possible settings than a single one.

Having the weights tuned according to the knowledge of structure of real-world data (denoted as SimTuned), we want to analyze its quality. Since the existing works focus on semantic similarity and omit

Characteristic		**Result of Δ**			
		min	**avg**	**med**	**max**
P_0 (GA)	random	0,013	1,176	0,673	3,959
	avg	0,001	0,652	0,463	3,441
s_0 (SA)	random	0,082	17,318	11,764	55,719
	avg	0,061	9,412	6,595	40,519
Characteristic		**Number of iterations**			
		min	**avg**	**med**	**max**
P_0 (GA)	random	1	17,2	19	30
	avg	5	20,9	22,5	30
s_0 (SA)	random	8	39,8	38	80
	avg	2	38.7	37	80

Table 2. Behavior of GA and SA

the tuning at all, comparison with any of them would be misleading. But we can compare the tuning with the usually used reasonable setting to the average-producing weights (denoted as and SimAvg). For this purpose we use the approach introduced in [5] which compares results of an algorithm with results of manual processing representing the optimum. Let R be the set of manually determined schema fragments similar to the given schema pattern and P the set of fragments determined by the similarity measure. Then I denotes the set of *true positives*, i.e. fragments correctly identified by the mesure, $F = P \backslash I$ denotes *false matches*, i.e. fragments identified incorrectly, and $M = R \backslash I$ denotes *false negatives*, i.e. not identified fragments, and thus

- $Precision = \frac{|I|}{|P|} = \frac{|I|}{|I|+|F|}$ evaluates the reliability of the measure,
- $Recall = \frac{|I|}{|R|}$ represents the share of real matches that is found, and
- $Overall = 1 - \frac{|F|+|M|}{|R|} = \frac{|I|-|F|}{|R|}$ represents a combined measure of the post-match effort necessary to remove false and add missed matches.

The lower the values are, the less precise the similarity measure is.

We have selected 5 XML schemes representing the 5 categories and a sample set of 10 data-centric and 10 document-centric schema patterns and we have manually identified the set R and using both SimAvg and SimTuned the set P. Finally, within the categories we have computed average values of Precision, Recall, and Overall which are depicted in Figure 1.

As can be seen, the SimAvg approach is apparently less precise than SimTuned in all the categories. The quality of both the measures is correlated with the amount of information we had for tuning. The best results can be found for categories dat, doc, and ex since the amount of schemes highly exceeds the amount in

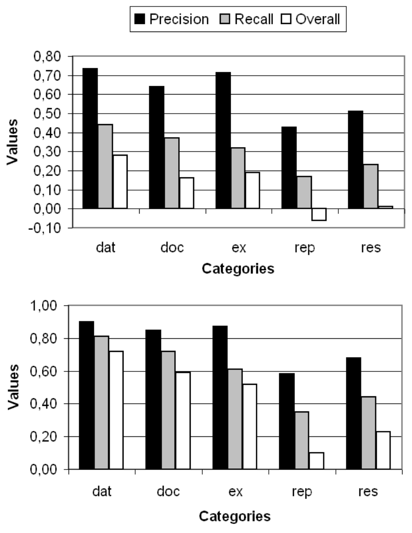

Figure 1. Results for SimAvg **and** SimTuned

the other two (see Table 1). Considering the Precision and Recall parameters, the reliability and share of real matchers exceeds 60% in the first three categories. It is not as good as in [5], where these values often exceed 75%, but none of the similarity measures focussed on structural similarity and also the match tasks were different. Considering the Overall parameter the worst results are again in case of rep and res categories, whereas in case of SimAvg and rep the value is even negative. This denotes that the number of false positives exceeds the number of true positives and thus the post-match effort is too high.

5 Conclusion

There are two main contributions of our paper. Firstly, we have proposed a similarity measure focusing on structural level which is not very common in existing works. Secondly, using experimental tests we show that with tuning of weights based on reliable information, the corresponding similarity measure has much better characteristics than the commonly used average-producing ones. Our approach can be viewed as a compromise between machine-learning techniques and the straightforward setting on the basis of user experience.

Our future work will focus on exploiting the semantics of element/attribute names. The similarity can be searched not only on structural level, but also using a thesaurus or similar user-given information. Although our proposal emphasizes structural similarities related to efficiency of database processing, it is worth testing whether the semantics of the names carries additional important information useful for this purpose too.

References

[1] M. Altinel and M. J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In *VLDB'00: Proc. of the 26th Int. Conf. on Very Large Data Bases*, pages 53–64, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[2] R. Bartak. *On-Line Guide to Constraint Programming.* 1998. `http://kti.mff.cuni.cz/~bartak/constraints/`.

[3] E. Bertino, G. Guerrini, and M. Mesiti. A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its Applications. *Inf. Syst.*, 29(1):23–46, 2004.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition).* W3C, 2006.

[5] H. H. Do and E. Rahm. COMA – A System for Flexible Combination of Schema Matching Approaches. In *VLDB'02: Proc. of the 28th Int. Conf. on Very Large Data Bases*, pages 610–621, Hong Kong, China, 2002. Morgan Kaufmann Publishers Inc.

[6] J. H. Holland. *Adaptation in Natural and Artifical Systems.* University of Michigan Press, Ann Arbor, MI, USA, 1975.

[7] S. Kirkpatrick, C. D. G. Jr., and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[8] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *VLDB'01: Proc. of the 27th Int. Conf. on Very Large Data Bases*, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[9] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *ICDE'02: Proc. of the 18th Int. Conf. on Data Engineering*, page 117, Washington, DC, USA, 2002. IEEE.

[10] I. Mlynkova. *UserMap – an Enhancing of User-Driven XML-to-Relational Mapping Strategies.* Technical report 2007/3. Charles University, Prague, Czech Republic, 2007.

[11] I. Mlynkova, K. Toman, and J. Pokorny. Statistical Analysis of Real XML Data Collections. In *COMAD'06: Proc. of the 13th Int. Conf. on Management of Data*, pages 20–31, New Delhi, India, 2006. Tata McGraw-Hill Publishing Company Limited.

[12] P. K. Ng and V. T. Ng. Structural Similarity between XML Documents and DTDs. In *ICCS'03: Proc. of the Int. Conf. on Computational Science*, pages 412–421. Springer Berlin / Heidelberg, 2003.

[13] A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *WebDB'02: Proc. of the 5th Int. Workshop on the Web and Databases*, pages 61–66, Madison, Wisconsin, USA, 2002.

[14] Z. Zhang, R. Li, S. Cao, and Y. Zhu. Similarity Metric for XML Documents. In *FGWM'03: Proc. of Workshop on Knowledge and Experience Management*, Karlsruhe, Germany, 2003.