Noname manuscript No.
(will be inserted by the editor)

Approximating the schema of a set of documents by means of resemblance

Alberto Abelló · Xavier de Palol · Mohand-Saïd Hacid

Received: date / Accepted: date

Abstract The WWW contains a huge amount of documents. Some of them share the same subject, but are generated by different people or even by different organizations. A semi-structured model allows to share documents that do not have exactly the same structure. However, it does not facilitate the understanding of such heterogeneous documents. In this paper, we offer a characterization and algorithm to obtain a representative (in terms of a resemblance function) of a set of heterogeneous semi-structured documents. We approximate the representative so that the resemblance function is maximized. Then, the algorithm is generalized to deal with repetitions and different classes of documents. Although an exact representative could always be found using an unlimited number of optional elements, it would cause an overfitting problem. The size of an exact representative for a set of heterogeneous documents may even make it useless. Our experiments show that, for users, it is easier and faster to deal with smaller representatives, even compensating the loss in the approximation.

Keywords Document · Design · XML

A. Abelló
Dept. de Llenguatges i Sistemes Informàtics, U. Politècnica de Catalunya
E-mail: aabello@essi.upc.edu

X. de Palol
Age Fotostock
E-mail: xdepalol@gmail.com

M-S. Hacid
LIRIS- UFR d'Informatique, U. Claude Bernard Lyon 1
E-mail: mshacid@liris.univ-lyon1.fr

1 Introduction

The Web is a powerful medium for human communication and an extraordinary source of information. Consequently, it has become a popular knowledge base, where people add documents (private, educational and organizational) and navigate through its contents. For scalability reasons, one important challenge is to distill those documents and extract valuable knowledge from them. There exist multiple formats for information sources, ranging from unstructured data to highly structured. As explained in [1], the term semi-structured data emerged to describe data that has some structure but neither regular, nor known a priori to the system. Hence, semi-structured documents are self-describing.

The importance of knowing the structure (schema) of a set of documents has been largely described in the literature. For example, [2] outlines its importance on integrating and analyzing the structure of the WWW. Besides, [3] points out that a known structure would also facilitate the storage and encourage queries. This is the key to improve the access methods to the data, thus enabling query optimization and data interchange among companies. As explained in [4], a simplistic approach taking the union or intersection of all documents does not work in practice, because it results too big or too small, respectively.

Without loss of generality, we consider a certain kind of semi-structured data, in particular, XML documents, which has been adopted as standard for data interchange, enabling the integration of heterogeneous information sources (notice that JSON can be easily mapped to this). A *well-formed* XML document is a document that conforms to the XML syntax rules in [5] (roughly, markups nest properly and attributes are unique). Moreover, a *valid* XML document is a docu-

The final publication is available at Springer via http://dx.doi.org/10.1007/s13740-018-0088-0

ment that is *well-formed* and also conforms to the rules of its grammar. A *schema* contains the declarations that provide such grammar for a class of documents. It determines the *element*s and *attribute*s that appear in a document, i.e., the name, type and constraints on every *element* and *attribute*. This is really important for the interchange of documents, since it represents the meaning of data. However, some automatically extracted documents (maybe coming from HTML) lack even this simple kind of schema.

As defined in [5], an XML document primarily consists of a nested hierarchy of *elements* with a single root. *Elements* can contain character data and *child element*s, in both cases the *element*s can have *attribute*s. The structure of *child elements* consists of a *sequence* list of *element*s. The standard states that *element*s in a *sequence* must be ordered.

The *choice* construct in a *schema* indicates that one, and only one, *element* in the *choice* list of contents should appear in the document (alternative elements). This construct is the key to find a perfect typing. With a grammar lacking it, we cannot find a *schema* common to a set of documents, and we have to approximate it. Otherwise, if we use *choice*, finding the *schema* is simply a question of finding the best grammar expression for each *element* (for example following a normal form like [6], or other approaches like [7]), so that all *element*s in the document belong to the corresponding grammar. Nevertheless, a perfect schema (i.e. one that is followed by all the documents) may cause an overfitting problem. Some works, like [8–10], have overcome overfitting by using clustering techniques to approximate typing. Such approximations are called inexact schemas in [11].

We aim at finding a common *schema* ("midpoint" - MP - from here on) for a set of *well-formed* semi-structured documents, avoiding the usage of optional *element*s when possible. Thus, we take an inexact approach based on the resemblance of documents. In particular, considering all approaches in [12], we use the resemblance family of functions in [2], which takes into account extra *element*s both in the document and in the *schema*. We could then redefine *valid* document as a document whose resemblance to its *schema* is above a given threshold. Our main contributions are the characterization of the MP in terms of a resemblance function and offering an efficient algorithm to obtain it. We have formalized *schema*s by means of Description Logics (DL). The rationale of our approach is grounded on its reasoning capabilities, but our results can be used outside its scope. Although our experiments show with DTDs, because of their simplicity, it also applies to any kind of XML or JSON schemas.

The structure of the paper is as follows: Next section reviews the work related to our method; Section 3 presents a formalization of XML by means of Description Logics and characterizes an MP; Section 4 shows a linear algorithm to obtain the MP without optional elements and repetitions; Sections 5 and 6 generalize the algorithm for those cases; and, finally, Section 7 concludes the paper. Appendix A shows some experimental results and user studies, and B contains the proofs of theorems.

2 Related work

Several authors worked on finding the schema of a set of semi-structured documents. Some, like [13], used Object Exchange Model (OEM). However, most of them worked on the generation of DTDs from XML data. For example, [14] applies heuristics to find a generalization of element descriptions. Another relevant result is [8], which explains how we can get an approximated typing for a set of objects. They find a set of types that cover most of the objects, but do not consider optional elements nor unnumbered repetitions. [15] describes an implementation of an algorithm to generate a DTD followed by an XML document. [9] classifies the documents in different classes and gets one DTD per class. This is a good solution if there are a few classes with not many documents or *element*s each. However, it may result in lots of different classes or optional *element*s for every class if we are dealing with really heterogeneous documents. [16] infers, in a more generic and generalizable approach, a deterministic regular expression from a sample of documents. [17,18] present another tool for the extraction of a DTD, in this case, by means of heuristic rules on the graph representation of the XML documents. [7] uses information theory to find the regular expressions of the DTD element by element. The regular expression of an element is so that it covers all appearances of it in the documents, and minimizes the number of bits needed to code the regular expression and the elements that follow it. The algorithm has high computational cost, so heuristics are also provided. [19] and [20] improve this work. In a completely different approach, [21] proposes a model-based technique to generate the underlying schema of a set of documents.

[8] pays attention to inexact schemas, outlining that the size of a perfect typing may be the order of the data set, prohibiting its use for query optimization and interfaces. Therefore, we are not searching a perfect typing but a human-friendly, computationally-tractable, and graphically-representable approximation. The main difference in this work with regard to previous authors is

that we do not use any heuristic, but a function. To this end, we should use some kind of resemblance or distance. The first option would be tree edit distance (like in [22] or [23]), but it results in high complexity (see [24]). Therefore, a promising option is structure similarity. [25] uses an internal graph structure that summarizing variants encountered in the data, which allows to detect structural outliers (patterns that occur only in few documents and might even be due to a errors during recording of the data). [8] uses Manhattan distance (i.e. the number of different descendants/ancestors of two *element*s). [26] shows different more elaborate resemblance measures. Among those, [9,27] use $\frac{|elem(d_1) \cap elem(d_2)|}{max(|elem(d_1)|,|elem(d_2)|)}$, while in [2], they use $\frac{|elem(d_1) \cap elem(d_2)|}{|elem(d_1) \cap elem(d_2)| + \alpha \cdot |elem(d_1) \setminus elem(d_2)| + \beta \cdot |elem(d_2) \setminus elem(d_1)|}$. We took this last measure, because it is more general, and allows to distinguish lack of *element*s in one side or another (i.e. either *schema* or documents).

Notice that we do not tackle the problem of finding the best (most concise) regular expression for every element in the *schema*, which can be done a posteriori by using [7], [19] or [20], but the problem of simplifying the *schema*. This can be used as a preprocess to reduce the cost of finding the best regular expression, avoid overfitting and eliminate unnecessary complexity in the result of those other algorithms, by automatically deciding what is representative and what is not in the original set of documents. To this aim, our approach is better than [8], because we do consider optional elements and repetitions, while it does not. [4] also identifies the smallest set of core attributes, but their approach is more complex and computationally expensive than the one we present here. Finally, [28] goes a step further by not finding a common *schema*, but trying to explain the different variants found in documents by means of association rules.

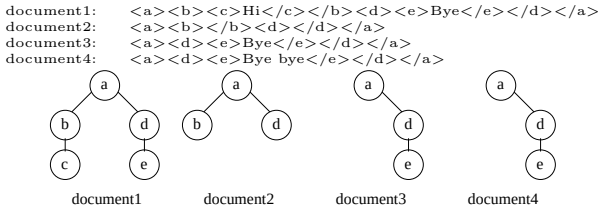


Fig. 1 Example of XML documents

3 Formalization of the problem

As we can see in [3], and exemplified in Figure 1, a document is thought as a rooted tree. A rooted tree is an acyclic graph ($\mathcal{N}$,$\mathcal{E}$), that has no more than one root. $\mathcal{N}$ is a set of nodes and $\mathcal{E}$ a set of edges. An edge e is an ordered pair of nodes (n_{source}, n_{target}). A node is a leaf if it is not the source of any edge in $\mathcal{E}$.

Since we only take into account *element* tags and their structure (not textual contents in the leaves), we are not actually interested in the whole documents, but in their *element*s and parent-child relationships. These can be obtained just by parsing the documents and eliminating textual data (leaving only *element* tags). Notice that one document will never contain *choice*, nor *unnumbered repetitions*, nor *optional elements*, nor *any*, because this is structural information that can only occur in a *schema*. How could we know, just from one document, that a present *element* may not be present or vice-versa? How could we decide that there is a potentially infinite repetition? We will consider that a document is a set of *element*s, each represented by the list of tags $(t_1, .., t_n)$ in the path from the root to it (see Section 6 to see how to deal with repetitions). The parent of an *element* is defined as $parent((t_1, .., t_n)) = (t_1, .., t_{n-1})$.

Regarding XML attributes, representing the information either as an attribute or a child is just a design decision. Thus, from here on, without loss of generality and just for the sake of simplicity, we will consider XML *attribute*s as simple content elements.

As stated in [5], *child elements* are ordered. Order is an important characteristic for documents. However, unordered data can be processed more efficiently in databases, so it is usually considered in that way. Therefore, from this point of view, we will assume order is not relevant in our case.

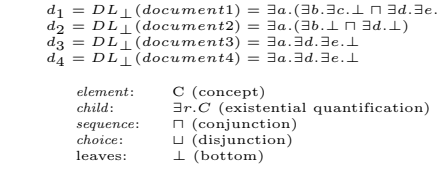


Fig. 2 DL representation of an XML document

3.1 Description Logics notation

We consider a set of documents as a knowledge base, which comprises two components, i.e. TBox (the terminology, we could recognize it as the schema) and ABox (the assertions about individuals, or instances). As explained in [29], the TBox contains concepts, and to define a formal semantics of the logic we use an interpretation $\mathcal{I}$. An interpretation is a pair $[\Delta^\mathcal{I}, \cdot^\mathcal{I}]$, where $\Delta^\mathcal{I}$

is the domain (a non-empty set), and $\cdot^{\mathcal{I}}$ is an interpretation function that assigns to every atomic concept A a set ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and to every atomic role r a binary relation ($r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). Inductively, this is extended to non-atomic concepts by the following definitions (where C and D are concepts, and r a role):

Bottom	$\bot^{\mathcal{I}}$	=	$\emptyset$
Top	$\top^{\mathcal{I}}$	=	$\Delta^{\mathcal{I}}$
Conjunction	$(C \sqcap D)^{\mathcal{I}}$	=	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$(C \sqcup D)^{\mathcal{I}}$	=	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Existential	$(\exists r.C)^{\mathcal{I}}$	=	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$

As exemplified in Figure 2, we will represent a *document* or piece of document by a concept "C". An unordered *sequence* of pieces of documents will be represented by a conjunction "$C \sqcap D$", while *choice* will be represented by a disjunction "$C \sqcup D$" (see Section 5 for its treatment). Finally, *children* will be represented by means of existential quantification "$\exists tag.C$". Actually, existential quantification allows the presence of more than one *element* of the same kind. Nevertheless, we do not consider such repetitions right now (see Section 6 for the treatment of repetitions). Leaves will be represented by bottom "$\bot$". However, in order to check the presence of a given *element* in a DTD, it is necessary to consider the possible existence of children. In this case, the chain of existentials ends with top "$\top$". Thus, *elements* are translated into DL as follows:

$$DL_{\bot}(()) = \bot; DL_{\bot}((t_1,..,t_n)) = \exists t_1..\exists t_n.\bot$$
$$DL_{\top}(()) = \top; DL_{\top}((t_1,..,t_n)) = \exists t_1..\exists t_n.\top$$

This formalization allows the usage of the following DL algorithms:

Subsumption (also known as "Query Containment" in other areas and noted "$C \sqsubseteq D$", if C is subsumed by D) shows whether one concept is more general than another (i.e. one set contains the other for all interpretations). For example, $d_1 \sqsubseteq d_3$.

$$C \sqsubseteq D \Leftrightarrow \forall \mathcal{I} : C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

Equivalence (noted "$C \equiv D$") shows that two concepts subsume each other. For example, $d_3 \equiv d_4$.

$$C \equiv D \Leftrightarrow C \sqsubseteq D \wedge D \sqsubseteq C$$

Least Common Subsumer (LCS) results in the subsumer of a set of concepts that is subsumed by any other subsumer of the set of concepts. For example, $lcs(d_2, d_4) = \exists a.\exists d.\top$. The usage of disjunction construct in the solution is not considered by LCS algorithms, because proves to be trivial (i.e. $lcs(C, D)$ would always be $C \sqcup D$).

$$L = lcs(C_1,..,C_n) \Leftrightarrow \forall i : C_i \sqsubseteq L \wedge \nexists D : (\forall i : C_i \sqsubseteq D \wedge D \sqsubseteq L)$$

Difference (non-standard operation defined in [30] and noted "$C - D$") is only defined if $C \sqsubseteq D$ and results in the concept characterized by the description in C not being in D. For example, $d_1 - d_3 = \exists a.\exists b.\exists c.\bot$.

$$S = C - D \Leftrightarrow D \sqcap S \equiv C \wedge \nexists S' : (D \sqcap S' \equiv C \wedge S \sqsubseteq S')$$

3.2 Characterization of the MP

Given a set of documents, we would like to find the *schema* that has the maximum number of common *element*s wrt that set, at the same time that minimizes the *element*s being in the *schema* not in the documents and those in the documents not in the *schema*. We will call such *schema* the MP of the set. In order to characterize the MP, we will use the resemblance family of functions used in [2].

$$r : (MP, setOfDocuments) \mapsto [0,1]$$

$$r(C,E) = \frac{w_c(C,E)}{w_c(C,E) + \alpha \cdot w_p(C,E) + \beta \cdot w_m(C,E)},\ \alpha, \beta \in \mathbb{R}^+$$

By instantiating α and β we get the concrete function we would like to use (notice that only if $\alpha = \beta$ the resemblance will be symmetric). Positive real values can be assigned to these parameters. They weight the importance of finding plus (*element*s in some documents that do not appear in the MP) and minus (*element*s in the MP that do not appear in some documents) *element*s respectively. The function relies now on three simpler ones that obtain respectively the size of common, plus, and minus *element*s.

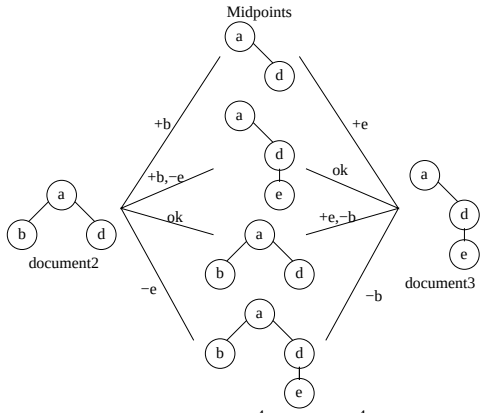
$$w_c(C,E) = \sum_{d \in E} size(lcs(C,d))$$

$$w_p(C,E) = \sum_{d \in E} (size(d) - size(lcs(C,d)))$$

$$w_m(C,E) = \sum_{d \in E} (size(C) - size(lcs(C,d)))$$

Any result in this paper does not depend on how we compute the size of an MP. We only assume that in the presence of *choice* the size is that of the smallest option ($size((t_1 \mid .. \mid t_n)) = min(size(t_1), .., size(t_n))$), and that the size of adding a non-optional *element* to an MP is always equal to the size of the MP plus the added *element* ($size(d) = size(d - t) + size(t)$). Therefore, from here on we will assume, in the examples, that every *element* contributes to the size with one unit ($size(d) = \#tags$)). For example, $size(d_1) = 5$ and $size(d_2) = size(d_3) = 3$. A general, more complex and accurate algorithm (still fulfilling these constraints) for obtaining the size of an MP is given in [2].

At this point, it is also important to notice that there may exist more than one *schema* maximizing the resemblance (i.e. more than one MP). For example, let be $\alpha = 2$ and $\beta = 3$. In this case, as we can see in Figure 3, all four resemblances coincide. The first candidate has two plus *elements*: "b" regarding d_2 and "e" regarding d_3. The second candidate has a perfect matching wrt d_3, and a plus and a minus *elements* wrt d_2. For the third candidate, it is the other way round. the last



$r(\exists a.\exists d.\bot, \{d_2, d_3\}) = \frac{4}{4+2\alpha+0\beta} = \frac{4}{4+4}$
$r(\exists a.\exists d.\exists e.\bot, \{d_2, d_3\}) = \frac{5}{5+\alpha+\beta} = \frac{5}{5+2+3}$
$r(\exists a.(\exists b.\bot \sqcap \exists d.\bot), \{d_2, d_3\}) = \frac{5}{5+\alpha+\beta} = \frac{5}{5+2+3}$
$r(\exists a.(\exists b.\bot \sqcap \exists d.\exists e.\bot), \{d_2, d_3\}) = \frac{6}{6+0\alpha+2\beta} = \frac{6}{6+6}$

Fig. 3 Example of multiple MPs

one has a minus *element* regarding each document: "e" for d_2 and "b" for d_3. Since this is the maximum resemblance, we can choose the MP of $\{d_2, d_3\}$ among those four candidates.

Midpoint	$\beta = 0$	$\beta \neq 0$
$\alpha = 0$	any	$\exists t.\bot$
$\alpha \neq 0$	$\bigsqcap_{d \in E} d$	?

Table 1 Trivial cases on finding an MP

4 Obtaining the MP of a set of documents

This section shows the possibility of finding an MP just based on the appearances of each *element* in the set of documents (we do not consider optional or repeated elements for the moment). First of all, it is important to show that depending on the values of α and β there are some trivial cases (summarized in Table 1). If $\alpha = 0$, we do not mind having extra *element*s in the documents wrt the MP. Therefore, among the multiple solutions to the problem, we find $\exists t.\bot$ (where "t" is the most frequent root tag in the documents). If $\beta = 0$, we do not mind having extra *element*s in the MP wrt every individual document. Therefore, $\bigsqcap_{d \in E} d$ is among the solutions. Both equaling zero means that just by matching some *element*s in some document we get maximum resemblance (i.e. $\forall w_c \neq 0 : \frac{w_c}{w_c+0w_p+0w_m} = 1$). From here on, we will only consider the non-trivial case where $\alpha \neq 0$ and $\beta \neq 0$.

The first question to answer is how we could know whether the point in the search space we are treating is better than another candidate or not. Surprisingly, it is not necessary to get all plus and minus *element*s. Thanks to Theorem 1, we know that all we need is the number of common *element*s between each of both candidate MPs and the set of documents E.

Theorem 1 *To decide whether the resemblance of a candidate C against a set of documents is better than that of another candidate C', it is only necessary to calculate the common* element*s between each document and the candidates (we do not need to calculate neither w_p, nor w_m).* [1]

Once we know that it is only necessary to compare the common *element*s, the next question is how we could improve the resemblance of a point in the search space. By Lemma 1, we know that if adding an *element* to the MP improves resemblance, all *element*s appearing the same number of times also improve it independently of their sizes. We may have thought that we have a set of possible improvements to investigate. Nevertheless, the *element*s with the same number of appearances do not generate alternative solutions, but all together belong to the same solution.

Lemma 1 *If adding an* element *e to a* child sequence *in the candidate increases its resemblance to the set of documents, adding all* element*s appearing in the same number of documents as e to the corresponding* child sequence *will also improve its resemblance independently of their sizes.* [2]

Finally, in Corollary 1, we show that *element*s appearing more times result in higher improvement of resemblance. As a special case of this, if an *element* improves resemblance, its parent improves resemblance even more. Thus, before adding an *element* to the result, all its ancestors should have been added (which otherwise could not have been avoided).

Corollary 1 *Independently of their size, an element e_1 appearing k_1 times in E improves the resemblance more than e_2 appearing k_2 times if $k_1 > k_2$.* [3]

From these theorems, we infer that we can incrementally build an MP for a set of documents from $\top$ (i.e. empty *schema*), by iteratively adding the most frequent *element* in the set of documents. Firstly, as we can see in Figure 4, we build a set of weighted elements (i.e. WE), whose contents are those elements e so that $DL_\top(e) \sqsupseteq \bigsqcap_{d \in E} d$, where e is weighted depending on its number of appearances in the set of documents. Once

[1] Proof has been moved to Appendix B.1.
[2] Proof has been moved to Appendix B.2.
[3] Proof has been moved to Appendix B.3.

```
                    // Phase 1: Count appearances of elements
WE := ∅;
// For each document
foreach d ∈ E do
    // For each of its elements
    foreach element : DL⊤(element) ⊒ d do
        // If the element appeared before
        if [element, k] ∈ WE
            // Increase its appearance
            then WE := WE \ {[element, k]} ∪ {[element, k + 1]};
        // Else
            // Initialize its appearance
            else WE := WE ∪ {[element, 1]};
            endif;
        endforeach;
    endforeach;
                // Phase 2: Build MP
// Initialize the empty document, and set the current appearance to the maximum
M := ⊤;
m :=| E |;
// While the appearance improves the resemblance
while ( m/(β·|E|) ≥ (Σ_{d∈E} size(lcs(M,d))) / (α·Σ_{d∈E} size(d)+β·|E|·size(M)) )
    // For each element appearing this amount of times
    foreach element ∈ getSubsetByWeight(WE, m) do
        // Add the element to MP
        add(M,element);
        endforeach;
    // Decrease the current number of appearance
    m := m − 1;
    endwhile;
```

Fig. 4 Algorithm to get an MP without optional elements

we have the weight of each *element*, we take the maximum possible weight (i.e. $\mid E \mid$) and check if it would improve resemblance from $\top$ to the set of documents. If this maximum weight improves the resemblance, we add all *element*s having such weight to the result (marking them as leaves by means of $\bot$) and get the next weight smaller than that.

As pointed out in [4], it is hard for users to specify any parameter to predefine the frequency on the data. However, it should be noticed that we loop adding another subset of *element*s while their weight improves resemblance. Formally, if the parent of the *element* to be added was a leaf (i.e. $M \sqsubseteq DL_\bot(parent(element))$), we should remove it from the MP before adding the *element* as follows:

```
add(M: Concept, element: Element) {
  if M ⊑ DL⊥(parent(element))
    then
      M := (M − DL⊥(parent(element))) ⊓ DL⊥(element);
    else
      M := M ⊓ DL⊥(element);
    endif;
}
```

The first phase of the algorithm is really cheap in terms of complexity. Taking into account that the number of possible children of an *element* should be small, building the weighted tree can be considered linear in the number of different *element*s in the set of documents, because we can find an *element* in WE just searching the children of the previous *element* we modified/added to WE (assuming a depth first search of the document we are treating). Regarding the second phase of the algorithm, all calls to "getSubsetByWeight" can be done in linear time in the number of different *element*s if we keep the *element*s with the same weight in a list. Therefore, the space we need is linear in the number of different *element*s (not counting repetitions), and the time is also linear in the number of *element*s in the set of documents (counting repetitions).

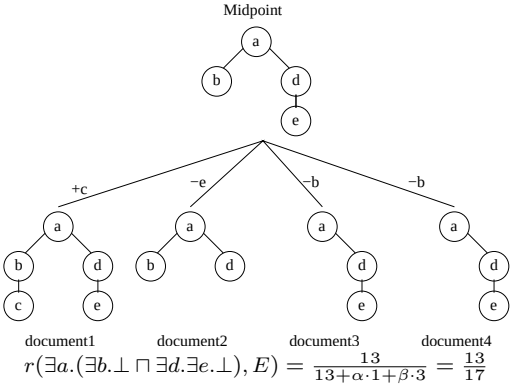
$$WE = \{[(a),4],[(a,b),2],[(a,b,c),1],[(a,d),4],[(a,d,e),3]\}$$

$$M_0 = \top \qquad \frac{4}{4\beta} \geq 0$$

$$M_1 = \exists a.\exists d.\bot \qquad \frac{3}{4\beta} \geq \frac{8}{14\alpha+2\cdot4\beta}$$

$$M_2 = \exists a.\exists d.\exists e.\bot \qquad \frac{2}{4\beta} \geq \frac{11}{14\alpha+3\cdot4\beta}$$

$$M_3 = \exists a.(\exists b.\bot \sqcap \exists d.\exists e.\bot) \qquad \frac{1}{4\beta} < \frac{13}{14\alpha+4\cdot4\beta}$$



$$r(\exists a.(\exists b.\bot \sqcap \exists d.\exists e.\bot), E) = \frac{13}{13+\alpha\cdot1+\beta\cdot3} = \frac{13}{17}$$

Fig. 5 Example of MP generated by the algorithm

If we ran this algorithm on the set of documents in Figure 1, it would result in the WE in Figure 5 (each pair consists of an *element* and the number of documents that contain it). Thus, in the first loop, condition evaluates true (for $\alpha = \beta = 1$, and every *element* contributing by one to the size), and we add the *element*s appearing four times. Since it still evaluates true, we add those appearing three times, and eventually twice. Since the condition evaluates false for weight equal one, the corresponding *element* does not belong to the solution. Notice that adding "(a,b)" to MP triggered the following operation "$(\exists a.\bot - DL_\bot((a))) \sqcap DL_\bot((a,b))$", resulting in "$\exists a.\exists b.\bot$".

As stated by Theorem 2, one of the possible MPs of the set of documents (which corresponds to that given by the previous algorithm) can be obtained by a conjunction of LCSs of subsets of the documents.

Theorem 2 *Given a set of documents* $E = \{d_1,...,d_n\}$, *and being* e_i elements *of the form* $(t_1, t_2, .., t_{l_i})$ *with* $l_i \geq 1$, *exists a collection of subsets of the set of documents, so that the MP (i.e. the* schema *that maximizes the resemblance) is the conjunction of the LCS of each of these subsets:* [4]

$$\exists S_1, ..., S_p \in \mathscr{P}(E) : \forall e_1, ..., e_q :$$

$$r(\bigsqcap_{i=1..q} DL_\bot(e_i), E) \leq r(\bigsqcap_{j=1..p} lcs(S_j), E)$$

[4] Proof has been moved to Appendix B.4.

Moreover, one of the MPs is such that the subsets of E that generate it are not subsets one of another.

Lemma 2 *There is a* schema *of the form* $\bigsqcap_{k=1..p} lcs(S_k)$ *maximizing the resemblance, so that for each pair of the sets of documents that generate it, they are not subsets one of another.*[5]

$$\forall 1 \leq i, j \leq p, i \neq j : (S_i \not\subseteq S_j)$$

Thus, it is easy to see that the upper bound of the number of sets that generate the MP is the number of possible subsets of E of size $\mid E \mid /2$.

Corollary 2 *There is a* schema *of the form* $\bigsqcap_{k=1..p} lcs(S_k)$ *maximizing the resemblance, so that the number of subsets of E that we need to generate it is smaller or equal than the number of subsets of E of size $\mid E \mid /2$.*

$$p \leq \left(\begin{array}{c} \mid E \mid \\ \frac{|E|}{2} \end{array} \right)$$

$$\begin{aligned} WE = \{\ & [\exists a.\top, \{d_1, d_2, d_3, d_4\}], \\ & [\exists a.\exists b.\top, \{d_1, d_2\}], \\ & [\exists a.\exists b.\exists c.\top, \{d_1\}], \\ & [\exists a.\exists d.\top, \{d_1, d_2, d_3, d_4\}], \\ & [\exists a.\exists d.\exists e.\top, \{d_1, d_3, d_4\}]\} \end{aligned}$$

$$M = lcs(d_1, d_2) \sqcap lcs(d_1, d_3, d_4)$$

Fig. 6 Obtaining the sets of documents that generate MP

It is easy to obtain the sets of documents whose LCSs generate the MP at a later stage (once we know MP) with a small modification of the previous algorithm. All we need is that "WE" keep the identifiers of the documents that contain every *element* instead of just a counter of them. The rationale of this is that the conjunction of the LCSs of the documents containing the leaves of the MP result in the MP. Figure 6 shows how this would result in our example. These documents related to each *element* could be used as a filtered input for a tool like [7] to generate the best grammar expression for that *element*.

5 Handling classes of documents

Until now, we assumed that we did not have the *choice* XML construct (i.e. disjunction "$\sqcup$" in terms of DL). In this section, we will study the possibility of using it to show the existence of different classes of documents.

[5] Proof has been moved to Appendix B.5.

Thus, resemblance needs to be redefined as follows, for k classes of documents:

$$\begin{aligned} M &= \bigsqcup_{i=1}^{k} M^i \\ E^i &= \{d \in E \mid \forall j \neq i, r(d, M^i) > r(d, M^j)\} \\ r(M, E) &= \frac{\sum_{i=1}^{k} w_c(M^i, E^i)}{\sum_{i=1}^{k} w_c(M^i, E^i) + \alpha \sum_{i=1}^{k} w_p(M^i, E^i) + \beta \sum_{i=1}^{k} w_m(M^i, E^i)} \end{aligned}$$

where M^i do not contain disjunctions. Sections 5.1 and 5.2 show, respectively, how w_p and w_m can be reduced by considering different classes of documents (adding a limited number of optional *element*s).

5.1 Reducing plus elements (in documents, not in MP)

We want to improve the resemblance to the whole set of documents by adding *element*s to the MP. Nevertheless, since we already reached the maximum resemblance, we could only worsen it. To solve this, we may consider those *element*s as optional. Adding an optional *element* will produce two classes of documents: Those containing the optional element (whose resemblance will be improved), and those that do not contain it (whose resemblance will not be modified).

It is easy to see that by just extending (strictly adding) the MP with optional *element*s, we will increase w_c, reduce w_p, and preserve w_m. The sum of all common and plus *element*s corresponds to the size of all documents together independently of the concept we are obtaining the distance to.

$$\forall C : w_c(C, E) + w_p(C, E) = \sum_{d \in E} size(d)$$

Thus, it is only necessary to consider how much w_c increases, i.e. how many documents match the optional *element*s and how big these are. The more documents matching those *element*s, the better; and the bigger the *element*s, also the better.

```
...                                          // Phase 3: Add optional leaves
// Get best element
e = nextByWeightAndSize(WE, m));
// While target resemblance is not reached and there are more elements
while (r(M, E) < target and e ≠ null)
    // Add an optional element to MP
    p := parent(e);
    M' := ⊔_{M_i ⊑ DL⊥(p)} (M_i ⊔ ((M_i − DL⊥(p)) ⊓ DL⊥(e)));
    M'' := ⊔_{¬(M_i ⊑ DL⊥(p))} (M_i ⊔ (M_i ⊓ DL⊥(e)));
    M := M' ⊔ M'';
    // Get next element
    e = nextByWeightAndSize(WE, m));
    endwhile;
```

Fig. 7 Algorithm for the selection of optional *element*s

Figure 7 shows the third phase of the algorithm in Figure 4. Once we got the *schema* of maximum resemblance, without any optional *element*, we may add optional *element*s also based on the number of appearance until we get the target resemblance ($target = 1$

would result in exact matching and overfitting). Every optional *element*, at worse, doubles the number of classes. Those already existing classes can be either extended with the current *element* or not, which is formalized by means of a disjunction. Formally, as in the second phase if the parent of the *element* to be added was a leaf (i.e. $M_i \sqsubseteq DL_\perp(p)$), it must be removed (by means of a difference) before adding the *element*.

Figure 8 exemplifies how we can improve the resemblance. In Figure 5, we stopped the second phase before adding those *element*s appearing only once. Therefore, $m = 1$ and the next *element* to be added is "(a, b, c)". Thus, we get a new MP being the disjunction of two classes, and we calculate the resemblance taking into account the best class for each document. In this way, numerator increases by a factor of one, while denominator increases (or even decreases if $\alpha > 1$) by a factor of $1 - \alpha$, thus, improving resemblance.

5.2 Reducing minus elements (in MP, not in documents)

The first problem with that improvement of resemblance is that the number of classes grows quickly on the number of optional *element*s, proving useful only when we are quite close to the target resemblance. Moreover, notice that, by using the algorithm in Figure 7 we will never modify w_m (which may not be zero after the second phase). Thus, we are not able to reach resemblance equal one if the MP after second phase contains non-optional *element*s that are absent from some documents. We should look for optional roots, besides optional leaves. In order to solve this, we should divide the documents into several classes, and obtain separately the MP of each of these classes. This way, increasing the number of classes, each of the MPs would eventually only contain *element*s that are present in all its corresponding documents (i.e. $w_m = 0$). The MP of the whole set of documents will be the disjunction of these partial MPs M^i.

We may trigger this phase of the algorithm if we do not reach the target resemblance with a given number of iterations in the third phase (i.e. a given small number of optional *element*s); or if $\frac{w_p}{w_m}$ after phase two is below a threshold; or it may even be triggered before hand, based on the number of appearances of *element*s at first level (i.e. if $\frac{|E|}{getMaxWeight(WE)}$ is above a threshold).

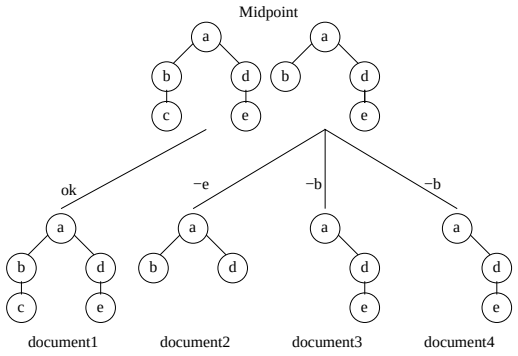
For this classification of documents, we may use an algorithm like "k-means" which is considered to need linear time (see [31]). If we take $k =\mid E \mid$, the problem becomes trivial, being $M = \bigsqcup_{d \in E} d$. Therefore, we are looking for a small k so that maximizes $r(M, E)$. For example, we may assume there are $\lceil \frac{|E|}{getMaxWeight(WE)} \rceil$ different kinds of documents, and generate such number of alternative subsets of *element*s.

Figure 9 sketches the algorithm. In our case, we could benefit from the existence of WE to improve performance if it keeps the sets of documents that contain every *element*, instead of just counting them (as assumed at the end of Section 4). We should codify every set E^i as a list of bits $b_1 b_2 .. b_{|E|}$, where bit j shows whether the corresponding document contains the *element* or not ($d_j \sqsubseteq DL_\top(element)$). In this way, we could take k random disjoint chains of bits as seeds (s^i) for "k-means". Then, we can find the MP corresponding to each seed by running the second phase of our algorithm on WE using the corresponding mask of bits.

Figure 10 shows an example of clustering documents into two sets, and how this improves the resemblance. We take the first document as seed one, and the others as seed two (i.e. $\{d_1\}$ vs $\{d_2, d_3, d_4\}$). For each one of them, we obtain the MP (i.e. "$\exists a.(\exists b.\exists c.\perp \sqcap \exists d.\exists e.\perp)$" and "$\exists a.\exists d.\exists e.\perp$" respectively) by applying the algorithm in Figure 4, ANDing the corresponding seed to the sequence of bits of each *element* in WE. Obtaining the resemblance of each document to both MPs, we see that "d_2" is in the wrong class, because its resemblance to the first MP is higher, while it was related to the second one. Therefore, we perform a second iteration with one MP for "$\{d_1, d_2\}$" and another one for "$\{d_3, d_4\}$". Now every document is in the right class, so we have finished. Figure 11 graphically draws the result and resemblance calculation. The MP of E is the disjunction of both MPs, and the overall resemblance improves by reducing the denominator. It is easy to see that with three classes we had obtained the exact *schema*.

Notice also that resemblances can be obtained from WE by crossing it once per cluster (keeping all $\mid E \mid$ resemblances in memory). For example, lets see how to obtain in the first iteration of Figure 10 resemblances from "$\exists a.\exists d.\exists e.\perp$" to each document (i.e. "$r(M^2, d_1)$", "$r(M^2, d_2)$","$r(M^2, d_3)$", and "$r(M^2, d_4)$"). At the first step, we would take "(a)" that belongs to the MP. Since the sequence of bits indicates that it belongs to the four documents, it would increase all four common *element*s' counters (i.e. $w_c(M^2, d_1)$, $w_c(M^2, d_2)$, $w_c(M^2, d_3)$, and $w_c(M^2, d_4)$). The same would happen for "(a,d)". At the third step, we may consider "(a, d, e)" that also belongs to the MP. Since the sequence of bits indicates that it belongs to "$\{d_1, d_3, d_4\}$", it would increment common *element*s of these and minus of d_2 (i.e. $w_c(M^2, d_1)$, $w_m(M^2, d_2)$, $w_c(M^2, d_3)$, and $w_c(M^2, d_4)$). At the fourth step, we would take "(a, b)" that does not belong to the MP. Since the sequence of bits in-

$$WE = \{[(a),4],[(a,b),2],[(a,b,c),1],[(a,d),4],[(a,d,e),3]\}$$
$$M_4 = M_4^1 \sqcup M_4^2 = \exists a.(\exists b.\bot \sqcap \exists d.\exists e.\bot) \sqcup (\exists a.(\exists b.\exists c.\bot \sqcap \exists d.\exists e.\bot))$$
$$E^1 = \{d_2, d_3, d_4\}; E^2 = \{d_1\}$$



$$r(M_4,E) = \frac{w_c(M_4^1,E^1)+w_c(M_4^2,E^2)}{w_c(M_4^1,E^1)+w_c(M_4^2,E^2)+\alpha(w_p(M_4^1,E^1)+w_p(M_4^2,E^2))+\beta(w_m(M_4^1,E^1)+w_m(M_4^2,E^2))} = \frac{(9+5)}{(9+5)+\alpha\cdot(0+0)+\beta\cdot(3+0)} = \frac{14}{17}$$

Fig. 8 Example of resemblance improvement reducing plus *element*s

```
                        // Phase 0: Clustering documents
Choose k initial seeds  // (may be random)
do
    For each class get an MP
    Assign each document to its nearest MP
while (classes changed)
```

Fig. 9 K-means algorithm

dicates that it belongs to the first and second documents, it would increment plus *element*s of these (i.e. $w_p(M^2, d_1)$, and $w_p(M^2, d_2)$). We would follow this way also for the last *element* in WE.

Considering k-means iterations linear in the number of documents, we need to get $\mid E \mid$ MPs (whose cost is linear in the number of *elements*), and cross WE the same number of times to obtain the resemblance of the documents to those MPs (whose cost depends on the number of *elements* in WE, and the number of bits to be compared, i.e. $\mid E \mid$). Therefore, the cost of finding the MP considering *choice* construct and using k-means algorithm is $\mid E \mid \cdot((\sum_{d \in E} \mid d \mid) + (\mid WE \mid \cdot \mid E \mid))$.

6 Repeated *element*s

It is possible that the same tag appears at different places (i.e. having a different parent) in the same document (or even in different documents). If we consider that in this case all appearances of a tag share the same internal structure independently of their position in the document, we should start a previous process to find the MP of such tag (i.e. we should consider it as a whole document, get its internal structure, and treat it as a black box in the processing of the real document). This section does not deal with this kind of repetitions, but with one *element* that contains several others of the same kind in a *sequence*.

First of all, on talking about repetitions, it is important to distinguish between (a) unnumbered repetitions (i.e. $+$ in XML notation) and (b) numbered repetitions (i.e. a fixed number of children of the same kind in a *sequence*). The point is that we cannot decide whether a repetition is unnumbered or not without human participation. How could we decide (based on a finite set of finite documents) that there is a potentially infinite repetition of *element*s? Since we cannot, we should decide first if we are interested in generating numbered or unnumbered repetitions.

If we wanted to generate (a) numbered repetitions, we should just consider that each sibling *element* is a completely different one and we can use again the same algorithms. For example, in order to be able to treat the XML document "<a>brothersister", we should transform this document into "<a><b1>brother</b1><b2>sister</b2>". This would work specially well for ordered *element*s, where the position indicates which sibling they are. In the example, the first one would always be identified as $b1$, and the second as $b2$. Doing it this way, a different position indicates a different internal structure.

If we wanted to generate (b) unnumbered repetitions, another problem would appear. As stated before, since in databases we are dealing with unordered documents, repeated *element*s result in undistinguishable twins, which should have the same intensional internal structure. Otherwise, if the twins had a different structure, there would be a conceptual design problem in the documents. Even when dealing with semi-

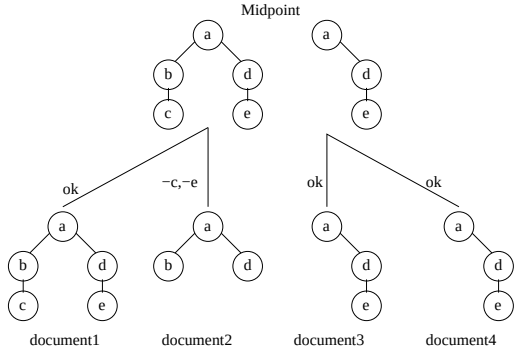
$$WE = \{[(a), \{1111\}], [(a,b), \{1100\}], [(a,b,c), \{1000\}], [(a,d), \{1111\}], [(a,d,e), \{1011\}]\}$$

	$s^1 = 1000, s^2 = 0111$	
	$WE^1 = \{[(a), \{1000\}], [(a,b), \{1000\}], [(a,b,c), \{1000\}], [(a,d), \{1000\}], [(a,d,e), \{1000\}]\}$	
Loop 1	$M_0^1 = \top$	$\frac{1}{1\beta} \geq 0$
	$M_1^1 = \exists a.(\exists b.\exists c.\bot \sqcap \exists d.\exists e.\bot)$	$\frac{0}{1\beta} < \frac{5}{5\alpha+5\cdot 1\beta}$
	$WE^2 = \{[(a), \{0111\}], [(a,b), \{0100\}], [(a,b,c), \{0000\}], [(a,d), \{0111\}], [(a,d,e), \{0011\}]\}$	
	$M_0^2 = \top$	$\frac{3}{3\beta} \geq 0$
	$M_1^2 = \exists a.\exists d.\bot$	$\frac{2}{3\beta} \geq \frac{6}{9\alpha+2\cdot 3\beta}$
	$M_2^2 = \exists a.\exists d.\exists e.\bot$	$\frac{1}{3\beta} < \frac{8}{9\alpha+3\cdot 3\beta}$
	$r(M^1, d_1) = \frac{5}{5+0\alpha+0\beta}$	$r(M^2, d_1) = \frac{3}{3+2\alpha+0\beta}$
	$r(M^1, d_2) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_2) = \frac{2}{2+1\alpha+1\beta}$
	$r(M^1, d_3) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_3) = \frac{3}{3+0\alpha+0\beta}$
	$r(M^1, d_4) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_4) = \frac{3}{3+0\alpha+0\beta}$

	$s^1 = 1100, s^2 = 0011$	
	$WE^1 = \{[(a), \{1100\}], [(a,b), \{1100\}], [(a,b,c), \{1000\}], [(a,d), \{1100\}], [(a,d,e), \{1000\}]\}$	
Loop 2	$M_0^1 = \top$	$\frac{2}{2\beta} \geq 0$
	$M_1^1 = \exists a.(\exists b.\bot \sqcap \exists d.\bot)$	$\frac{1}{2\beta} \geq \frac{6}{8\alpha+3\cdot 2\beta}$
	$M_2^1 = \exists a.(\exists b.\exists c.\bot \sqcap \exists d.\exists e.\bot)$	$\frac{0}{2\beta} < \frac{8}{8\alpha+5\cdot 2\beta}$
	$WE^2 = \{(a), \{0011\}], [(a,b), \{0000\}], [(a,b,c), \{0000\}], [(a,d), \{0011\}], [(a,d,e), \{0011\}]\}$	
	$M_0^2 = \top$	$\frac{2}{2\beta} \geq 0$
	$M_1^2 = \exists a.\exists d.\exists e.\bot$	$\frac{1}{2\beta} \geq \frac{6}{6\alpha+3\cdot 2\beta}$
	$M_2^2 = \exists a.\exists d.\exists e.\bot$	$\frac{0}{2\beta} < \frac{6}{6\alpha+3\cdot 2\beta}$
	$r(M^1, d_1) = \frac{5}{5+0\alpha+0\beta}$	$r(M^2, d_1) = \frac{3}{3+2\alpha+0\beta}$
	$r(M^1, d_2) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_2) = \frac{2}{2+1\alpha+1\beta}$
	$r(M^1, d_3) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_3) = \frac{3}{3+0\alpha+0\beta}$
	$r(M^1, d_4) = \frac{3}{3+0\alpha+2\beta}$	$r(M^2, d_4) = \frac{3}{3+0\alpha+0\beta}$

$$M = (\exists a.(\exists b.\exists c.\bot \sqcap \exists d.\exists e.\bot)) \sqcup (\exists a.\exists d.\exists e.\bot); E^1 = \{d_1, d_2\}; E^2 = \{d_3, d_4\}$$

Fig. 10 Example of dividing the documents into two classes



$$\begin{aligned} r(M,E) &= (w_c(M^1,E^1) + w_c(M^2,E^2))/(w_c(M^1,E^1) + w_c(M^2,E^2) \\ &\quad +\alpha \cdot (w_p(M^1,E^1) + w_p(M^2,E^2)) \\ &\quad +\beta \cdot (w_m(M^1,E^1) + w_m(M^2,E^2))) \\ &= \frac{(8+6)}{(8+6)+\alpha\cdot(0+0)+\beta\cdot(2+0)} = \frac{14}{16} \end{aligned}$$

Fig. 11 Graphical representation of classes and resemblance

structured data, two *element*s of the same (undistinguishable) class should share the same semi-structure. Thus, we have two different problems. The first one is deciding when there exists an unnumbered repetition, and the second one is how to find the internal structure common to all repetitions of the *element*.

To face these two problems, we should modify the parser of documents that capture the *element*s. If the parser finds a repetition, it should use a special mark showing the presence of sibling *element*s, and indicating the number s of them existing in the corresponding parent. However, because of syntactical issues, we need to visit all the descendants of the parent before we know the number of twins. Therefore, we should keep in memory the whole set of *element*s of each document, to generate s, without a second pass.

If there exists only one repetition of "t" in a million of documents, substituting "t" by "t+" does improve the resemblance, which is clearly not representative of the million documents. The problem is that once the tag "t" belongs to the MP, replacing it by "t+" increases w_c, decreases w_p, but does not affect w_m. This means that we should never consider "t" instead of "t+" if only one document contains a repetition. This is quite drastic and unrealistic. Therefore, we took the approach of changing "t" by "t+" if there are m documents con-

taining repetitions of "t", and *element*s appearing m times improve resemblance.

Thus, in order to decide whether the MP should show a repetition or not, if the first phase of the algorithm (Figure 4) finds the special parsing mark, then $(t_1, .., t_n)$ and $(t_1, .., t_n+)$ should both be increased in WE. During the second phase, on adding $(t_1, .., t_n+)$, we should change t_n into t_n+ from M. Notice that the appearance of $(t_1, .., t_n)$ will always be higher than that of $(t_1, .., t_n+)$, because we always increase the counter of the first (if we find the *element*), while only increase the latter if there is a repetition of that *element*. The second phase does not need to be modified.

Regarding the problem of finding the internal structure of repetitions, the second phase does not need any modification, either. When, during the first phase of the algorithm, we find a subelement in any of the repetitions, we should just increase the counter of the corresponding *element* in $\frac{1}{s}$, where s is the number of siblings in the parent *element*. By doing this, we avoid overweighting the subelements of repetitions, and keep the basic idea of the algorithm still true (i.e. a child cannot appear more times in the documents than its parent).

document5: <a>Single
document6: <a>Twin2 $Twin1$
document7: <a>FlatTwin<c>ComTwin</c>

$WE = \{[(a), 3], [(a, b), 3], [(a, b+), 2], [(a, b, c), 0.5]\}$

Fig. 12 Example of documents with repetitions

Figure 12 exemplifies how repetitions are treated by the algorithm. In this case, "(a)" weights three, because appears in three documents. The same happens for "(a,b)", because three documents contain such element. Moreover, there are two documents containing repetitions of "(a,b)", which is recorded by the appearance of "(a,b+)". Finally, "(a,b,c)" appears once in one document. Nevertheless, it is part of a repetition of two twins, so that its weight is $\frac{1}{2} = 0.5$.

7 Conclusions and future work

Along this paper, we have studied the possibility of approximating the *schema* of a set of documents. Based on a given measure of resemblance, we are able to find the MP of the set. We began by considering only a restricted class of *schema*s (having neither repetition nor *choice*), and it has been generalized to any *schema*. Thus, we are able to approximate the collection of documents as much as we want (by considering a limited number of alternative classes). Taking this to the extreme, we would get an exact matching. Besides that, we can also find the relevant documents in the original set with regard to the generated MP, which would ease the obtaining of the best grammar expression of each element. The general algorithm using k-means clustering is quadratic in the number of documents and linear in the number of different elements, while it is only linear in the whole number of elements (counting repetitions) if not considering the *choice* construct (i.e. without clustering).

Our experiments in Appendix A show that these approximated *schema*s are much simpler and ease user understanding. People spend less time and, if resemblance is high enough, they make less mistakes (even compensating the loss in the approximation).

Acknowledgements

We would also like to thank Age Fotostock for allowing us to use their files for testing purposes. Special thanks also to Toni Gutzens for his help on implementing the prototype and Jon Bosak for allowing us to use his documents for testing purposes.

References

1. S. Abiteboul, Querying semi-structured data, in: Proc. of 6th Int. Conf. Database Theory (ICDT'97), Vol. 1186 of LNCS, Springer, 1997, pp. 1–18.
2. E. Bertino, G. Guerrini, M. Mesiti, A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications, Information Systems 29 (1) (2004) 23–46.
3. S. Abiteboul, P. Buneman, D. Suciu, Data on the Web - From Relations to Semistructured Data and XML, Morgan Kaufmann, 2000.
4. L. Wang, O. Hassanzadeh, S. Zhang, J. Shi, L. Jiao, J. Zou, C. Wang, Schema management for document stores, Proceedings of the VLDB Endowment 8 (9) (2015) 922–933.
5. W3C, Extensible Markup Language (XML) 1.0, 3rd Edition (February 2004).
6. J. Albert, D. Giammarresi, D. Wood, Normal Form algorithms for extended Context-Free Grammars, Theoretical Computer Science 267 (1-2) (2001) 35–47.
7. M. Garofalakis, A. Gionis, R. Rastogi, S. Sechadri, K. Shim, XTRACT: Learning Document Type Descriptors from XML Document Collections, Data Mining and Knowledge Discovery 7 (1) (2003) 23–56.
8. S. Nestorov, S. Abiteboul, R. Motwani, Extracting schema from semistructured data, in: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1998), ACM, 1998, pp. 295–306.
9. I. Sanz, J. Pérez, R. Berlanga, M. Aramburu, XML Schemata Inference and Evolution, in: Proc. of 14th Int. Conf. on Databases and Expert Systems Applications (DEXA'03), Vol. 2736 of LNCS, Springer, 2003, pp. 109–118.

10. R. Nayak, W. Iryadi, XML schema clustering with semantic and hierarchical similarity measures, Knowl.-Based Syst. 20 (4) (2007) 336–349.
11. J. Widom, Data Management for XML: Research Directions, IEEE Data Engineering Bulletin 22 (3) (1999) 44–52.
12. G. Guerrini, M. Mesiti, I. Sanz, An overview of similarity measures for clustering xml documents, Web Data Management Practices: Emerging Techniques and Technologies (2007) 56 – 78.
13. K. Wang, H. Liu, Schema discovery for semistructured data, in: 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97), 1997, pp. 271–274.
14. J. Hegewald, F. Naumann, M. Weis, Xstruct: Efficient schema extraction from multiple and large XML documents, in: Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA, 2006, p. 81.
15. J.-S. Jung, D.-I. Oh, Y.-H. Kong, J.-K. Ahn, Extracting Information from XML Documents by Reverse Generating a DTD, in: Proc. of the EurAsia-ICT 2002, Vol. 2510 of LNCS, Springer, 2002, pp. 314–321.
16. G. J. Bex, W. Gelade, F. Neven, S. Vansummeren, Learning deterministic regular expressions for the inference of schemas from XML data, TWEB 4 (4) (2010) 14:1–14:32.
17. D.-H. Moh, E.-P. Lim, W.-K. Ng, Re-engineering Structures from Web Documents, in: 5th ACM Conf. on Digital Libraries (DL 2000), ACM, 2000, pp. 67–76.
18. D.-H. Moh, E.-P. Lim, W.-K. Ng, DTD-Miner: A Tool for Mining DTD from XML Documents, in: Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000), IEEE Computer Society, 2000, pp. 144–151.
19. A. V. Leonov, R. R. Khusnutdinov, Study and Development of the DTD Generation System for XML Documents, Programming and Computer Software 31 (4) (2005) 197–210.
20. J.-K. Min, J.-Y. Ahn, C.-W. Cung, Efficient extraction of schemas for XML documents, Information Processing Letters 85 (2003) 7–12.
21. J. L. C. Izquierdo, J. Cabot, Discovering implicit schemas in JSON data, in: Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings, 2013, pp. 68–83.
22. U. Boobna, M. de Rougemont, Correctors for XML Data, in: Proc. of 2nd Int. XML Database Symposium (XSYM'04), Vol. 3186 of LNCS, Springer, 2004, pp. 97–111.
23. T. Dalamagas, T. Cheng, K.-J. Winkel, T. Sellis, A methodology for clustering XML documents by structure, Information Systems 31 (2006) 187–228.
24. Z. Zhang, D. Shasha, Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems, SIAM Journal on Computing 18 (6) (1989) 1245–1262.
25. M. Klettke, U. Störl, S. Scherzinger, Schema extraction and structural outlier detection for json-based nosql data stores, in: Datenbanksysteme für Business, Technologie und Web (BTW), 16. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 4.-6.3.2015 in Hamburg, Germany. Proceedings, 2015, pp. 425–444.
26. V. Batagelj, M. Bren, Comparing resemblance measures, Journal of Classification 12 (1) (1995) 73–90.
27. W. Lian, D. Cheung, N. Mamoulis, S.-M. Yiu, An efficient and scalable algorithm for clustering xml documents by structure, IEEE Transactions on Knowledge and Data Engineering 16 (1) (2004) 82–96.
28. E. Gallinucci, M. Golfarelli, S. Rizzi, Schema profiling of document-oriented databases, Information Systems 75 (2018) 13 – 25.
29. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook, Cambridge University Press, 2003.
30. G. Teege, Making the Difference: A Substraction Operation for Description Logics, in: Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94), Morgan Kaufmann, 1994, pp. 540–550.
31. V. Estivill-Castro, J. Yang, Fast and Robust General Purpose Clustering Algorithms, in: Proc. of 6th Pacific Rim Int. Conf. on Artificial Intelligence (PRICAI 2000), Vol. 1886 of LNCS, Springer, 2000, pp. 208–218.

A Experimental study

We have conducted the experiments in two different directions. On the one hand, we have analyzed the usefulness of the obtained *schema*, and how informative and useful it is with regard to the perfect-matching (in Section A.1). On the other hand, we have also tested the performance of the algorithm to find one MP (shown in Section A.2), by means of randomly generated documents. The performance on finding different classes has not been analyzed, because it strictly depends on the cost of the clustering algorithm used (i.e. k-means in our case), and this is out of the scope of this work.

A.1 Usefulness

This section scrutinizes the usefulness of approximating the *schema* of a set of documents instead of using the perfect-matching. Four different sets of real-life documents have been used in the experiment (in Sections A.1.1, A.1.2, A.1.3 and A.1.4), so that an approximated *schema* has been generated for each of them[6]. These have been used in a user study of usability against the perfect ones (in Section A.1.5).

A.1.1 Religious texts

The documents in this section are authored by Jon Bosak[7], and correspond to four religious texts (i.e. "The Old Testament", "The New Testament", "The Quran", and "The Book of Mormon"). LHS of Figure 13 shows the MP of the four documents under consideration. Characteristics of the documents and parameters are as follows:

Number of documents: 4
Real number of classes: 4
Used number of classes: 1

With those parameters, we obtain a resemblance of 47.6%. This would mean that documents are quite different, and we should define different classes of documents. However, this does not make sense if we only have four documents. Surprisingly, we would get 93.5% (not 100%) of resemblance for four different classes. This is because not even the repetitions of the same element inside a document share the same structure.

Notice that elements like "suracoll", "sura" and "witness" are not considered relevant. Nevertheless, some that were optional, like "coverpg", "titlepg", and "preface") appear enough times to be in the output.

[6] $\alpha = \beta = 1$ unless explicitly said otherwise.

[7] http://www.oasis-open.org/cover/bosakShakespeare200.html

```
<!-- DTD for testaments    J. Bosak -->
<!-- Early versions 1992-1998 -->
<!-- Major revision Copyright (c) Jon Bosak September 1998 -->
<!ENTITY % plaintext "#PCDATA|i">
<!ELEMENT tstmt     (coverpg?,titlepg?,preface?,(bookcoll|suracoll)+)>
<!ELEMENT coverpg   ((title|title2)+, (subtitle|p)*)>
<!ELEMENT titlepg   ((title|title2)+, (subtitle|p)*)>
<!ELEMENT title     (%plaintext;)*>
<!ELEMENT title2    (%plaintext;)*>
<!ELEMENT subtitle  (p)+>
<!ELEMENT preface   ((ptitle|ptitle0)+,p+,witlist?)+>
<!ELEMENT witlist   (witness)+>
<!ELEMENT ptitle    (%plaintext;)*>
<!ELEMENT ptitle0   (%plaintext;)*>
<!ELEMENT witness   (%plaintext;)*>
<!ELEMENT bookcoll  (book|sura)+>
<!ELEMENT book      (bktlong,bktshort,epigraph?,bksum?,chapter+)>
<!ELEMENT suracoll  (sura+)>
<!ELEMENT sura      (bktlong,bktshort,epigraph?,bksum?,v+)>
<!ELEMENT bktlong   (%plaintext;)*>
<!ELEMENT bktshort  (%plaintext;)*>
<!ELEMENT bksum     (p)+>
<!ELEMENT chapter   (chtitle,chstitle?,epigraph?,chsum?,(div+|v+))>
<!ELEMENT chtitle   (%plaintext;)*>
<!ELEMENT chstitle  (%plaintext;)*>
<!ELEMENT div       (divtitle, v+)>
<!ELEMENT divtitle  (%plaintext;)*>
<!ELEMENT chsum     (p)+>
<!ELEMENT epigraph  (%plaintext;)*>
<!ELEMENT p         (%plaintext;)*>
<!ELEMENT v         (%plaintext;)*>
<!ELEMENT i         (%plaintext;)*>
```

```
<!-- Automatically generated DTD -->

<!ELEMENT tstmt    (bookcoll,coverpg,preface,titlepg)>
<!ELEMENT coverpg  (subtitle+,title,title2)>
<!ELEMENT titlepg  (title, title2)>
<!ELEMENT title    (#PCDATA)>
<!ELEMENT title2   (#PCDATA)>
<!ELEMENT subtitle (p)>
<!ELEMENT preface  (p+,ptitle)>
<!ELEMENT ptitle   (#PCDATA)>
<!ELEMENT bookcoll (book+)>
<!ELEMENT book     (bktlong,bktshort,chapter+)>
<!ELEMENT bktlong  (#PCDATA)>
<!ELEMENT bktshort (#PCDATA)>
<!ELEMENT chapter  (chtitle,v+)>
<!ELEMENT chtitle  (#PCDATA)>
<!ELEMENT v        (#PCDATA)>
<!ELEMENT p        (#PCDATA)>
```

Fig. 13 Religious texts

A.1.2 Shakespeare texts

The documents in this section are authored by Jon Bosak[8], and show a set of the plays of William Shakespeare. LHS of Figure 14 shows the DTD of the thirty-seven documents under consideration. Characteristics of the documents and parameters are as follows:

Number of documents: 37
Real number of classes: 27
Used number of classes: 1

With those parameters, we obtain the *schema* at RHS of Figure 14, resulting in a resemblance of 90%. This would mean that the 37 kinds of documents are quite similar. The maximum resemblance would be for 37 classes, which results in 96%. As in Section A.1.1, this is because not even the repetitions of the same element inside a document share the same structure. Just by adding optional elements to the *schema* in Figure 14, we get a 98.5% of resemblance.

Notice that some optional elements like "EPILOGUE", "PROLOGUE" and "INDUCT" do not appear in the MP, because only few documents contain them (for instance, "INDUCT" appears in just 2 out of 37 documents). Thus, it seems clear that to understand the meaning of a "PLAY", "INDUCT" is not relevant. The same can be said on "ACT". If you want to explain your child what an act is, you would just say that it has a title, and one or more scenes. Only if s/he is really interested and old enough, you would point out that it may contain subtitles, prologue and epilogue.

On the other hand, "PERSONA" and "PGROUP" are important enough to appear in the MP. They are not optional nor a *choice*, because most documents contain them. All documents contain "PERSONA", and only four documents do not contain "PGROUP" inside "PERSONAE".

A.1.3 Response/Request documents

The documents in this section are authored by Age Fotostock, which is an imagery agency in all areas (both rights protected and royalty free). Age provides a technical hosting platform (THP) for the sharing of images among imagery agencies around the world. In this example, four classes of documents are provided, which correspond to a licensed protocol for B2B image sharing. First, an agency needs to request the existing resolutions for one or more images (classes one and three). Secondly, an agency requests an specific high-resolution file for one or more images (classes two and four). The documents have been extracted from the service log files of the company. Thus, there was no available DTD. Characteristics of the documents and parameters are as follows:

Number of documents: 189
Real number of classes: 6
Used number of classes: 4

If we try to find the MP of these documents, it results in a resemblance of 36.8%. This means that there exist completely different kinds of documents. By looking for four classes, we obtain those in Figure 15, which results in a resemblance of 99.3%. Looking for less than four classes results in unrealistic DTDs where some of those four are united. This effect can be avoided by increasing β above one (1.2 was enough in our experiments).

A.1.4 Photo documents

The documents in this section are also authored by Age Fotostock, and are an stratified random extraction of the imagery database. Thus, again, there was no available DTD. Characteristics of the documents and parameters are as follows:

Number of documents: 2497
Real number of classes: $\sim$100
Used number of classes: 1

All elements found in the MP (those at LHS of Figure 16) appear in more than two thousand documents. Thus, it looks reasonable to consider them as mandatory. On the other hand, optional elements (those new at RHS of Figure 16) appear in less than one thousand. The resemblance obtained to the MP is 91%, which increases up to 97.4% if we also consider the optional elements. This means that there is only one class of documents, that can be well described by the obtained MP.

[8] http://www.oasis-open.org/cover/bosakShakespeare200.html

```
<!-- DTD for Shakespeare J. Bosak    1994.03.01, 1997.01.02 -->
<!-- Revised for case sensitivity 1997.09.10 -->
<!-- Revised for XML 1.0 conformity 1998.01.27 (thanks to Eve Maler) -->
<!ENTITY amp "&#38;#38;">
<!ELEMENT PLAY     (TITLE, FM, PERSONAE,SCNDESCR,PLAYSUBT,INDUCT?,
  PROLOGUE?,ACT+,EPILOGUE?)>
<!ELEMENT TITLE    (#PCDATA)>
<!ELEMENT FM       (P+)>
<!ELEMENT P        (#PCDATA)>
<!ELEMENT PERSONAE (TITLE,(PERSONA | PGROUP)+)>
<!ELEMENT PGROUP   (PERSONA+,GRPDESCR)>
<!ELEMENT PERSONA  (#PCDATA)>
<!ELEMENT GRPDESCR (#PCDATA)>
<!ELEMENT SCNDESCR (#PCDATA)>
<!ELEMENT PLAYSUBT (#PCDATA)>
<!ELEMENT INDUCT   (TITLE, SUBTITLE*,(SCENE+|(SPEECH|STAGEDIR|SUBHEAD)+))>
<!ELEMENT ACT      (TITLE, SUBTITLE*,PROLOGUE?, SCENE+, EPILOGUE?)>
<!ELEMENT SCENE    (TITLE, SUBTITLE*,(SPEECH | STAGEDIR | SUBHEAD)+)>
<!ELEMENT PROLOGUE (TITLE, SUBTITLE*,(STAGEDIR | SPEECH)+)>
<!ELEMENT EPILOGUE (TITLE, SUBTITLE*,(STAGEDIR | SPEECH)+)>
<!ELEMENT SPEECH   (SPEAKER+,(LINE | STAGEDIR | SUBHEAD)+)>
<!ELEMENT SPEAKER  (#PCDATA)>
<!ELEMENT LINE     (#PCDATA | STAGEDIR)*>
<!ELEMENT STAGEDIR (#PCDATA)>
<!ELEMENT SUBTITLE (#PCDATA)>
<!ELEMENT SUBHEAD  (#PCDATA)>
```

```
<!-- Automatically generated DTD -->
<!ELEMENT PLAY     (ACT+,FM,PERSONAE,PLAYSUBT,SCNDESCR,TITLE)>
<!ELEMENT TITLE    (#PCDATA)>
<!ELEMENT FM       (P+)>
<!ELEMENT P        (#PCDATA)>
<!ELEMENT PERSONAE (PERSONA+,PGROUP+,TITLE)>
<!ELEMENT PGROUP   (GRPDESCR,PERSONA+)>
<!ELEMENT PERSONA  (#PCDATA)>
<!ELEMENT GRPDESCR (#PCDATA)>
<!ELEMENT SCNDESCR (#PCDATA)>
<!ELEMENT PLAYSUBT (#PCDATA)>
<!ELEMENT ACT      (SCENE+,TITLE)>
<!ELEMENT SCENE    (SPEECH+,STAGEDIR+,TITLE)>
<!ELEMENT SPEECH   (LINE,SPEAKER)>
<!ELEMENT SPEAKER  (#PCDATA)>
<!ELEMENT LINE     (#PCDATA)>
<!ELEMENT STAGEDIR (#PCDATA)>
```

Fig. 14 Shakespeare texts

```
<!-- Class 1 -->
<!ELEMENT request        (getResolutions,login,password)>
<!ELEMENT getResolutions (imagecode)>
<!ELEMENT imagecode      (#PCDATA)>
<!ELEMENT login          (#PCDATA)>
<!ELEMENT password       (#PCDATA)>

<!-- Class 3 -->
<!ELEMENT response    (resolutions,status)>
<!ELEMENT resolutions (imagecode,resolution+,status)>
<!ELEMENT imagecode   (#PCDATA)>
<!ELEMENT resolution  (#PCDATA)>
<!ELEMENT status      (#PCDATA)>
```

```
<!-- Class 2 -->
<!ELEMENT request         (getUrlHires,login, password)>
<!ELEMENT getUrlHires     (imagecode,imageresolution,imagesaveas,imageusage)>
<!ELEMENT imagecode       (#PCDATA)>
<!ELEMENT imageresolution (#PCDATA)>
<!ELEMENT imagesaveas     (#PCDATA)>
<!ELEMENT imageusage      (#PCDATA)>
<!ELEMENT login           (#PCDATA)>
<!ELEMENT password        (#PCDATA)>

<!-- Class 4 -->
<!ELEMENT response  (status,urlHires)>
<!ELEMENT urlHires  (imagecode,status)>
<!ELEMENT imagecode (#PCDATA)>
<!ELEMENT status    (#PCDATA)>
```

Fig. 15 Requests/Responses (Four classes of documents)

```
<!-- Without optional elements -->
<!ELEMENT photo          (agency,photographer,general_data,imagecode,
    license_terms,original_code)>
<!ELEMENT agency         (code,collection,name)>
<!ELEMENT code           (#PCDATA)>
<!ELEMENT collection     (#PCDATA)>
<!ELEMENT name           (#PCDATA)>
<!ELEMENT photographer   (name)>
<!ELEMENT name           (#PCDATA)>
<!ELEMENT general_data   (creation_date,documentation,photocrypt)>
<!ELEMENT creation_date  (#PCDATA)>
<!ELEMENT documentation  (description,keywording)>
<!ELEMENT description    (#PCDATA)>
<!ELEMENT keywording     (#PCDATA)>
<!ELEMENT photocrypt     (resolution+)>
<!ELEMENT resolution     (#PCDATA)>
<!ELEMENT imagecode      (#PCDATA)>
<!ELEMENT license_terms  (#PCDATA)>
<!ELEMENT original_code  (#PCDATA)>
```

```
<!-- With optional elements -->
<!ELEMENT photo            (additional_info?,agency,photographer,general_data,
  imagecode,license_terms,original_code)>
<!ELEMENT additional_info  (model_release?,property_release?,usage_conflict?)>
<!ELEMENT model_release    (#PCDATA)>
<!ELEMENT property_release (#PCDATA)>
<!ELEMENT usage_conflict   (country*)>
<!ELEMENT country          (iso?,name?)>
<!ELEMENT iso              (#PCDATA)>
<!ELEMENT name             (#PCDATA)>
<!ELEMENT agency           (code,collection,name)>
<!ELEMENT code             (#PCDATA)>
<!ELEMENT collection       (#PCDATA)>
<!ELEMENT name             (#PCDATA)>
<!ELEMENT photographer     (code?,name)>
<!ELEMENT code             (#PCDATA)>
<!ELEMENT name             (#PCDATA)>
<!ELEMENT general_data     (cdrom?,creation_date,documentation,photocrypt)>
<!ELEMENT cdrom            (#PCDATA)>
<!ELEMENT creation_date    (#PCDATA)>
<!ELEMENT documentation    (description,keywording,large_description?)>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT keywording       (#PCDATA)>
<!ELEMENT large_descr      (#PCDATA)>
<!ELEMENT photocrypt       (resolution+)>
<!ELEMENT resolution       (#PCDATA)>
<!ELEMENT imagecode        (#PCDATA)>
<!ELEMENT license_terms    (#PCDATA)>
<!ELEMENT original_code    (#PCDATA)>
```

Fig. 16 Photos (without/with optional elements)

A.1.5 Usability test

In this section, we study the comprehensibility of approximated *schema*s against perfect-matching. First of all, let's see the usability and maintainability measures defined in [22]:

Size is the number of nodes in the graph representing the *schema*.

Complexity is defined as the number of edges, plus one, minus the number of nodes (i.e. the number of edges that should be removed to obtain a tree, which would mean zero complexity).

Depth is the maximum depth of the graph representing the *schema*.

Fan-In is the maximum number of children among the *element*s of the *schema*.

Fan-Out is the maximum number of parents among the *element*s of the *schema*.

Table 2 shows those metrics in the *schema*s considered in previous sections (from A.1.2, we use the approximated *schema* without optional elements). Just notice that only in one case (i.e. A.1.3) these metrics are not affected. In this case, only the optionality of some elements has changed. In the other three pairs of *schema*s, we can observe that all us-

	Size	Complexity	Depth	Fan-In	Fan-out
A.1.1 (perfect)	27	34	5	6	5
A.1.1 (approx.)	17	7	4	4	2
A.1.2 (perfect)	22	42	5	9	7
A.1.2 (approx.)	16	12	4	6	4
A.1.3 (perfect)	15	5	3	4	4
A.1.3 (approx.)	15	5	3	4	4
A.1.4 (perfect)	24	5	4	7	4
A.1.4 (approx.)	16	2	3	6	2

Table 2 Usability metrics as in [22]

ability metrics have been improved, which means that the approximated *schema* is much simpler in any sense.

In order to demonstrate the efficiency and effectiveness of the approximated *schema*s, a user study has been conducted. Some *schema*s (some perfect and other approximated) have been given to each individual, besides a list of five randomly chosen *element*s from the perfect one (notice that some of these may not be present in the approximated one) for each *schema*. The individual had to answer the number of paths leading to each *element*, the depth of each one of these paths and whether the path must be in every document or not (i.e. the optionality of the path). In the approximated *schema*, the queried *element* may not be present or if present, it may have been considered non-optional (when, actually, it is optional). Any of these cases has also been considered a user error. Another important point in the study is that the mistakes have been weighted by the importance of the element (i.e. a mistake in an optional leaf that only appears from time to time counts proportionally less than a mistake in the root or any other mandatory *element*).

	Perfect			Approx.			T-test	Resem.
	#	avg	stdev	#	avg	stdev		
A.1.1	7	394	132	9	176	30	0.002	47.6%
A.1.2	9	319	110	7	196	129	0.03	90%
A.1.3	8	342	149	10	298	144	0.27	99.3%
A.1.4	9	264	109	8	187	81	0.06	91%

Table 3 Efficiency

Table 3 shows the average time expressed in seconds (column "#" shows the number of individuals that answered every questionnaire). In all four cases, the approximated *schema* results in less average time for users to answer. A T-test has been done in order to discard that those results have been obtained by chance. For the first one, second one and fourth one, there is a probability of 0.2%, 3% and 6% respectively of obtaining these results just by chance. Only in the third one (that has a really high similarity) the probability of obtaining these results by chance can be considered high (i.e. 27%).

	Perfect			Approx.			T-test	Resem.
	#	avg	stdev	#	avg	stdev		
A.1.1	8	4.12	0.61	9	3.3	0.10	0.003	47.6%
A.1.2	9	4.17	0.62	8	3.74	0.74	0.11	90%
A.1.3	8	4.43	0.47	10	4.71	0.23	0.08	99.3%
A.1.4	9	4.57	0.51	8	4.12	0.83	0.11	91%

Table 4 Effectiveness

Regarding the effectiveness, Table 4 summarizes the experimental results. Each of the five queried *elements* counts by one, so the maximum score would be 5 (meaning answers for all five *elements* are correct) and the minimum would be 0 (meaning that the five elements are mandatory and all five answers are wrong). In the first case, since the resemblance was really low (i.e. 47.6%), users made much more mistakes working on the approximated *schema* and this is not by chance (0.3% in the T-test). Third and fourth cases have similar resemblances, and so are the experimental results. Users make more mistakes with the approximated *schema* and there is an 11% of probability of making such mistakes by chance. However, the most surprising result is the third one. In this case, the effectiveness is improved with the approximated *schema* (with a probability of only 8% of being by chance). Notice that in this case, the resemblance is really high.

	Perfect			Approx.			T-test
	#	avg	stdev	#	avg	stdev	
Efficiency	9	319	110	8	313	118	0.45
Effectiveness	9	4.17	0.62	8	4.48	0.51	0.14

Table 5 Efficiency and effectiveness for A.1.2 with 95% of resemblance

Thus, to check the influence of different resemblances, we clustered the documents in A.1.2 into five clusters and generated a new approximated *schema*, whose resemblance to the perfect one is now 95% (instead of 90% as the previous approximation). Giving this new *schema* to eight people, we obtained the results in Table 5. As in the previous experiment, when the resemblance is high, there is not any relevant difference between the average time users spend in understanding the *schema*, either perfect or approximated. However, as also happened in the other case of high resemblance, the effectiveness has been improved (just a 14% of having this improvement by chance), since individuals have more correct answers.

These experiments corroborate our intuition that giving simpler *schema*s to the users, they will spend less time working with them. If these *schema*s are too different (resemblance 91% or bellow in our experiments), users make mistakes due to the information loss we have in the approximation of the *schema*. However, if the resemblance we obtain is high enough (95% or above in our experiments) the effectiveness of users compensates the loss in the approximation. Thus, spending the same time, users understand the contents of the *schema* much better and make less mistakes.

A.2 Performance

This section contains experimental performance results obtained from a self generated data set. These sets have been generated randomly from the *schema* in the LHS of Figure 17, parametrizing the probability of appearance of repeated and optional *element*s as follows:

- $+$ $0.25 * 0.75^{n-1}$ (being n the number of repetitions)
- $*$ $0.25 * 0.75^{n}$ (being n the number of repetitions)
- $?$ 0.75
- $|$ $\frac{k}{\frac{n(n+1)}{2}}$ (for the *element* at position k in a choice of n *element*s)

In the LHS of Figure 17, we have the original *schema*, while its RHS shows the obtained MP from one thousand randomly generated documents. Resulting resemblance is 65%. However, we should not analyze it in this case, because the purpose of this test is just to corroborate the linear behaviour of the algorithm and the documents used are senseless. Characteristics of the set and parameters used are as follows:

```
<!-- Original DTD -->
<!ELEMENT root (a,b+,c*,d?)>
<!ELEMENT a (e,f)>
<!ELEMENT b (g,h)>
<!ELEMENT c (i|j?)>
<!ELEMENT d (#PCDATA)>
<!ELEMENT e (#PCDATA)>
<!ELEMENT f (k?,l)>
<!ELEMENT g (m|n|o)>
<!ELEMENT h (#PCDATA)>
<!ELEMENT i (p|q)>
<!ELEMENT j (r,s)>
<!ELEMENT k (#PCDATA)>
<!ELEMENT l (#PCDATA)>
<!ELEMENT m (#PCDATA)>
<!ELEMENT n (#PCDATA)>
<!ELEMENT o (#PCDATA)>
<!ELEMENT p (#PCDATA)>
<!ELEMENT q (#PCDATA)>
<!ELEMENT r (#PCDATA)>
<!ELEMENT s (#PCDATA)>
```

```
<!-- Automatically generated DTD -->
<!ELEMENT r (a,b+,c)>
<!ELEMENT a (e,f)>
<!ELEMENT b (g,h)>
<!ELEMENT c (#PCDATA)>
<!ELEMENT e (#PCDATA)>
<!ELEMENT f (l)>
<!ELEMENT g (#PCDATA)>
<!ELEMENT h (#PCDATA)>
<!ELEMENT l (#PCDATA)>
```

Fig. 17 Random documents

Number of documents: 1000
Real number of classes: ∼180
Used number of classes: 1

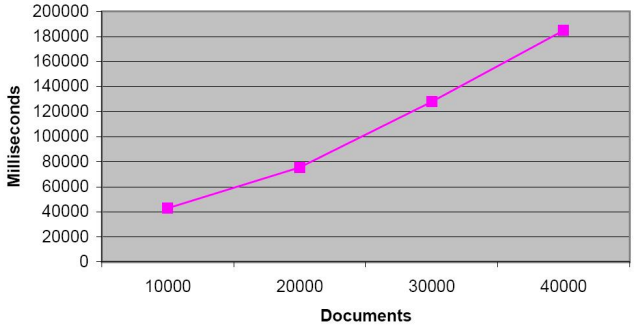


Fig. 18 Performance on getting an MP

Figure 18 shows the time (in milliseconds) to obtain the MP of 10, 20, 30, and 40 thousand documents following the previous DTD. We can see that, as expected, it increases linearly on the number of documents.

B Proofs

This section contains the proofs of the different theorems ("s" stands for "size", to make equations shorter).

B.1 Proof of Theorem 1

Proof Let be C and C' two *schema*s so that $r(C,E) \geq r(C',E)$. Expanding equations, we get:

$$\frac{w_c(C,E)}{w_c(C,E)+\alpha\cdot w_p(C,E)+\beta\cdot w_m(C,E)} \geq \frac{w_c(C',E)}{w_c(C',E)+\alpha\cdot w_p(C',E)+\beta\cdot w_m(C',E)}$$

$$\frac{\sum_{d\in E} s(lcs(C,d))}{\sum_{d\in E} s(lcs(C,d)) + \alpha\cdot(s(d)-s(lcs(C,d))) + \beta\cdot(s(C)-s(lcs(C,d)))} \geq \frac{\sum_{d\in E} s(lcs(C',d))}{\sum_{d\in E} s(lcs(C',d))+\alpha\cdot(s(d)-s(lcs(C',d)))+\beta\cdot(s(C')-s(lcs(C',d)))}$$

By crossing denominators,

$$(\sum_{d\in E} s(lcs(C,d)))\cdot (\sum_{d\in E} s(lcs(C',d)) + \alpha\cdot(s(d)-s(lcs(C',d))) + \beta\cdot(s(C')-s(lcs(C',d)))) \geq (\sum_{d\in E} s(lcs(C',d)))\cdot (\sum_{d\in E} s(lcs(C,d)) + \alpha\cdot(s(d)-s(lcs(C,d))) + \beta\cdot(s(C)-s(lcs(C,d))))$$

Simplifying $(\sum_{d\in E} s(lcs(C,d)))\cdot(\sum_{d\in E} s(lcs(C',d)))$ at both sides results in:

$$(\textstyle\sum_{d\in E} s(lcs(C,d)))(\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\sum_{d\in E} s(C')) \geq (\sum_{d\in E} s(lcs(C',d)))(\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\sum_{d\in E} s(C))$$

This can also be written like

$$\frac{\sum_{d\in E} s(lcs(C,d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\mid E\mid\cdot s(C)} \geq \frac{\sum_{d\in E} s(lcs(C',d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\mid E\mid\cdot s(C')}$$

which shows that all we need to compare are common elements (in numerator), and the sizes of both *schema*s (in denominator). □

B.2 Proof of Lemma 1

Proof Let be C and C' two *schema*s and e an *element* so that $C = C' \sqcap DL_{\perp}(e)$, and $r(C,E) > r(C',E)$ (i.e. C contains one more *element* and this improves resemblance). Retaking the inequality at the end of proof of Theorem 1:

$$\frac{\sum_{d\in E} s(lcs(C,d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\mid E\mid\cdot s(C)} > \frac{\sum_{d\in E} s(lcs(C',d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\mid E\mid\cdot s(C')}$$

Which by adding and subtracting $s(lcs(C',d))$ to every term in the left numerator, and $s(C')$ to the left denominator results in:

$$\frac{\sum_{d\in E}(s(lcs(C',d))+(s(lcs(C,d))-s(lcs(C',d))))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot|E|\cdot(s(C')+(s(C)-s(C')))} > \frac{\sum_{d\in E} s(lcs(C',d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot|E|\cdot s(C')}$$

And reordering sums, we obtain

$$\frac{\sum_{d\in E} s(lcs(C',d))+\sum_{d\in E}((s(lcs(C,d))-s(lcs(C',d))))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot|E|\cdot(s(C)-s(C'))} > \frac{\sum_{d\in E} s(lcs(C',d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta|E|\cdot s(C')}$$

Which (given that $\frac{a+b}{c+d} \geq \frac{a}{c}$ iif $\frac{b}{d} \geq \frac{a}{c}$) is true if and only if

$$\frac{\sum_{d\in E}((s(lcs(C,d))-s(lcs(C',d))))}{\beta\cdot\mid E\mid\cdot((s(C)-s(C')))} > \frac{\sum_{d\in E} s(lcs(C',d))}{\alpha\cdot\sum_{d\in E} s(d)+\beta\cdot\mid E\mid\cdot s(C')}$$

Since we assume that the size of adding a non-optional *element* to a *schema* is always equal to the size of the *schema* plus the added *element*, we can transform the left hand side as follows:

$$\frac{\sum_{d\in E}((s(lcs(C,d))-s(lcs(C',d))))}{\beta\cdot\mid E\mid\cdot((s(C)-s(C')))} = \frac{\sum_{d\sqsubseteq DL_{\top}(e)} s(e)}{\beta\cdot\mid E\mid\cdot s(e)} = \frac{\sum_{d\sqsubseteq DL_{\top}(e)} 1}{\beta\cdot\mid E\mid}$$

Therefore, either adding an *element* or not does not depend on the size of the *element*, but on the number of times it appears in the documents. Thus, if adding an *element* is worthwhile, so it is adding any set of *element*s appearing the same number of times. □

B.3 Proof of Corollary 1

Proof Since by hypothesis, $k_1 > k_2$, then $\frac{k_1}{\beta \cdot |E|} > \frac{k_2 \cdot s(e_2)}{\beta \cdot |E| \cdot s(e_2)}$. Therefore, if e_2 improved the resemblance (i.e. by proof of Lemma 1, we know that $\frac{k_2}{\beta \cdot |E|} \geq \frac{\sum_{d \in E} s(lcs(C,d))}{\alpha \cdot \sum_{d \in E} s(d) + \beta \cdot |E| \cdot s(C)}$), then e_1 improves it even more:

$$\frac{k_1}{\beta \cdot \mid E \mid} \geq \frac{\sum_{d \in E} s(lcs(C,d)) + (k_2 \cdot s(e_2))}{\alpha \cdot \sum_{d \in E} s(d) + \beta \cdot \mid E \mid \cdot s(C) + (\beta \cdot \mid E \mid \cdot s(e_2))}$$

. □

B.4 Proof of Theorem 2

Proof By hypothesis, let's suppose that there is an MP $M = \bigsqcap_{i=1..q} DL_\perp(e_i)$ that maximizes the resemblance and is not a conjunction of LCSs. Let's define $E_C = \{d \in E \mid d \sqsubseteq C\}$, and divide the proof in three steps:

Step 1: Every *element* in M subsumes some document in E (i.e. $\forall i \in 1..q : E_{DL_\top(e_i)} \neq \emptyset$)

Let's suppose not (proof by contradiction), i.e. $\exists i = 1..q : E_{DL_\top(e_i)} = \emptyset$. We can remove the last k tags from e_i until there exists some document d with an *element* matching e'_i (being $e'_i = (t_1, t_2, .., t_{l_i - k})$). Now, $d \sqsubseteq DL_\top(e'_i)$. Let be $M' = DL_\perp(e_1) \sqcap ... \sqcap DL_\perp(e_{i-1}) \sqcap DL_\perp(e'_i) \sqcap DL_\perp(e_{i+1}) \sqcap ... \sqcap DL_\perp(e_q)$. It is easy to see that $w_c(M,E) = w_c(M',E)$, $w_p(M,E) = w_p(M',E)$, and $w_m(M,E) > w_m(M',E)$. So,

$$r(M,E) \leq r(M',E)$$

which means they are either equal (if $\beta = 0$) or contradicts the hypothesis of $r(M,E)$ being the maximum resemblance. Therefore, we can assume that $\forall i = 1..q : E_{DL_\top(e_i)} \neq \emptyset$.

Step 2: All *element*s e_i in M are leafs of $lcs(E_{DL_\top(e_i)})$ (i.e., $lcs(E_{DL_\top(e_i)}) \sqsubseteq DL_\perp(e_i)$, which is only possible if every e_i is a leaf of some document)

Let's suppose not, because exists e_i so that the corresponding chain of existentials in $lcs(E_{DL_\top(e_i)})$ is longer than $DL_\perp(e_i)$ (notice that it can never be shorter, by construction of $E_{DL_\top(e_i)}$ and definition of the LCS). This means that e_i is not a leaf of any document in E.
Let's call e_L to $(t_1, t_2, .., t_{l_i}, .., t_{l_i+k})$ so that it results in the corresponding chain of existentials of $lcs(E_{DL_\top(e_i)})$, and let be $M' = DL_\perp(e_1) \sqcap ... \sqcap DL_\perp(e_{i-1}) \sqcap DL_\perp(e_L) \sqcap DL_\perp(e_{i+1}) \sqcap ... \sqcap DL_\perp(e_q)$. Therefore, since e_L must be present in all documents in $E_{DL_\top(e_i)}$, we can see the following equalities:

$$w_c(M', E) = w_c(M, E) + \mid E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e_L)) - s(DL_\perp(e_i)))$$

$$w_p(M', E) = w_p(M, E) + \mid E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e_i)) - s(DL_\perp(e_L)))$$

$$w_m(M', E) = w_m(M, E) + \mid E \setminus E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e_L)) - s(DL_\perp(e_i)))$$

Notice that $\forall d \in E \setminus E_{DL_\top(e_i)} : s(lcs(d, DL_\top(e_L))) = s(lcs(d, DL_\top(e_i)))$, because if exists a document with an *element* e' so that $DL_\top(e_L) \sqsubseteq DL_\top(e') \sqsubset DL_\top(e_i)$, by definition it belongs to $E_{DL_\top(e_i)}$.
By hypothesis, $r(M,E) \geq r(M',E)$ (M maximizes resemblance). Thus, expanding both resemblances ("e_i" stands for "$DL_\perp(e_i)$", and "e_L" stands for "$DL_\perp(e_L)$"),

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha w_p(M,E) + \beta w_m(M,E)}$$

$$\geq$$

$$\frac{w_c(M,E) + \mid E_{e_i} \mid \cdot (s(e_L) - s(e_i))}{(w_c(M,E) + \alpha w_p(M,E) + \beta w_m(M,E)) + ((1-\alpha) \cdot \mid E_{e_i} \mid + \beta \mid E \setminus E_{e_i} \mid)(s(e_L) - s(e_i))}$$

Let be $e'_i = (t_1, t_2, .., t_{l_i - 1})$. Since, as stated before, by hypothesis, e_i is not a leaf in any document, $\forall d \in E \setminus E_{DL_\top(e_i)} : s(lcs(d, DL_\top(e'_i))) = s(d, DL_\top(e_i))$. Thus, as before, we can define $M'' = DL_\perp(e_1) \sqcap ... \sqcap DL_\perp(e_{i-1}) \sqcap DL_\perp(e'_i) \sqcap DL_\perp(e_{i+1}) \sqcap ... \sqcap DL_\perp(e_q)$:

$$w_c(M', E) = w_c(M, E) + \mid E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e'_i)) - s(DL_\perp(e_i)))$$

$$w_p(M', E) = w_p(M, E) + \mid E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e_i)) - s(DL_\perp(e'_i)))$$

$$w_m(M', E) = w_m(M, E) + \mid E \setminus E_{DL_\top(e_i)} \mid \cdot (s(DL_\perp(e'_i)) - s(DL_\perp(e_i)))$$

By hypothesis, $r(M,E) \geq r(M',E)$ (M maximizes resemblance). Thus, expanding both resemblances ("e_i" stands for "$DL_\perp(e_i)$", and "e'_i" stands for "$DL_\perp(e'_i)$"),

$$\frac{w_c(M,E)}{w_c(M,E) + \alpha w_p(M,E) + \beta w_m(M,E)}$$

$$\geq$$

$$\frac{w_c(M,E) - \mid E_{e_i} \mid \cdot (s(e_i) - s(e'_i))}{(w_c(M,E) + \alpha w_p(M,E) + \beta w_m(M,E)) - ((1-\alpha) \cdot \mid E_{e_i} \mid + \beta \mid E \setminus E_{e_i} \mid)(s(e_i) - s(e'_i))}$$

However, both inequalities (i.e. $\frac{a}{b} \geq \frac{a+ck}{b+dk}$ and $\frac{a}{b} \geq \frac{a-ck'}{b-dk'}$) are not possible at the same time, because $s(DL_\perp(e_L)) - s(DL_\perp(e_i))$ and $s(DL_\perp(e_i)) - s(DL_\perp(e'_i))$ (which correspond to k and k' respectively) are both strictly positive numbers. Therefore, the hypothesis is not true and e_i must be a leaf of $lcs(E_{B_i})$.

Step 3: All *element*s in $lcs(E_{DL_\top(e_i)})$ are also elements of M (i.e. $M \sqsubseteq lcs(E_{DL_\top(e_i)})$)

This means that e_i appears exactly in $\mid E_{DL_\top(e_i)} \mid$ documents, and by definition of LCS, all other elements in $lcs(E_{DL_\top(e_i)})$ appear at least in those documents. Therefore, by Lemma 1, all those *element*s also belong to M.

Therefore, since $E_{DL_\top(e_i)}$ is never empty (as shown in step 1), and being $M \sqsubseteq \bigsqcap lcs(E_{DL_\top(e_i)}) \sqsubseteq \bigsqcap DL_\perp(e_i)$ (as shown in steps 2 and 3), then by the equivalence of M ($M \equiv \bigsqcap DL_\perp(e_i)$) we get that $M \equiv \bigsqcap lcs(E_{DL_\top(e_i)})$. □

B.5 Proof of Lemma 2

Proof Let's suppose not (i.e. $S_i \subseteq S_j$). Then, by definition of LCS, we get that $lcs(S_i) \sqsubseteq lcs(S_j)$

1. If $lcs(S_i) \equiv lcs(S_j)$ then we can remove one of them from the *schema*.
2. If $lcs(S_i) \sqsubset lcs(S_j)$ then $lcs(S_i) \sqcap lcs(S_j) \equiv lcs(S_i)$. Therefore, we can remove S_j from the *schema*.

□