An Approach to Extracting Sub-schema Similarities from Semantically Heterogeneous XML Schemas

Pasquale De Meo and Giovanni Quattrone
Università Mediterranea di Reggio Calabria
Via Graziella, Località Feo di Vito
89122 Reggio Calabria, Italy
E-mail: demeo@unirc.it, quattrone@unirc.it

Giorgio Terracina
Dipartimento di Matematica
Università della Calabria
Via Pietro Bucci
87036 Rende (CS), Italy
E-mail: terracina@mat.unical.it

Domenico Ursino
Università Mediterranea di Reggio Calabria
Via Graziella, Località Feo di Vito
89122 Reggio Calabria, Italy
E-mail: ursino@unirc.it

Keywords: XML Schema, sub-schema similarities, interschema properties, semantically heterogeneous information sources

Received: November 7, 2007

This paper presents a semi-automatic approach to deriving sub-schema similarities from semantically heterogeneous XML Schemas. The proposed approach is specific for XML, almost automatic and light. It consists of two phases: the first phase selects the most promising pairs of sub-schemas, the second one examines them and returns only those which are similar. This paper describes the approach in all details and illustrates a large variety of experiments to test its performance. Furthermore, it presents a comparison between this approach and others which have already been proposed in the literature.

Povzetek: Opisano je iskanje podobnosti podshem v XML.

1 Introduction

The derivation of semantic matchings among concepts of different sources (known also as "schema matching" activity in the literature) has become a challenging issue in the field of Information Systems; as a matter of fact, their knowledge allows the improvement of source interoperability and plays a key role in various applications, such as data source integration, ontology matching, e-commerce, semantic query processing, data warehousing, source clustering and cataloguing, and so on.

In the past, most of the proposed approaches to deriving matchings were manual (1); today, due to the enormous number of available sources, it is widely recognized the need of semi-automatic techniques (4; 5; 11; 13; 20). Moreover, most of the matching derivation theory has been *developed to operate on classical, structured databases*, and the main focus has been on deriving similarities and dissimilarities between *single classes of objects* (e.g., two entities, two relationships, an entity and a relationship, and so on).

However, in the last few years, the Web is becoming the reference infrastructure for many applications conceived to handle the interoperability among different partners. Web sources are quite different from classical databases, since they are semi-structured. In order to make Web activities easier, World Wide Web Consortium (W3C) proposed XML (eXtensible Markup Language) as a new standard information exchange language, that aims at unifying representation capabilities, typical of HTML, and data management features, typical of classical DBMSs. In order to improve the capability of representing and handling the intensional component of XML sources, W3C proposed to associate XML Schemas with XML documents. An XML Schema can be considered as a catalogue of the information that can be found in the corresponding XML documents.

The exploitation of the semi-structured paradigm in general, and of XML in particular, makes it evident the necessity to develop new approaches to deriving semantic matchings; these approaches are quite different from the traditional ones. As a matter of fact, in semi-structured information sources, a concept is not generally expressed by a single class of objects but it is represented by a group of them; as an example, in XML, concepts are expressed by elements which can be, in their turn, described by sub-elements.

In such a situation, the emphasis shifts away from the extraction of *semantic correspondences between object classes* to the derivation of *semantic correspondences between portions of information sources* (i.e., sub-sources). We call *sub-schema* a self contained sub-source such that a concept represented therein is connected to other concepts of the sub-source by means of at least one relationship. We call *sub-schema similarity* a similarity between two sub-schemas belonging to different sources.

Due to its intrinsic complexity, the sub-schema similarity extraction problem goes beyond the classic problem of deriving semantic correspondences among single concepts belonging to different schemas and allows more complex relationships to be handled.

This paper aims at providing a contribution in this setting; in fact, it presents an approach to extracting similarities between XML sub-schemas. Our approach is characterized by the following features:

- *It is almost automatic*, in that it requires the user intervention only for validating obtained results; the present overwhelming amount of available information sources on the Web makes such a feature particularly relevant.

- *It has been specifically conceived for operating on XML Schemas*; in fact, the framework underlying our approach has been defined for directly covering the XML specificities (see, below, Section 2.1). With regard to this choice, we point out that XML source interoperability will play a more and more relevant role in the future; as a consequence, it will be more and more common the necessity to handle the interoperability of a group of information sources that are all XML-based. In this scenario, the possibility to exploit a technique specifically tailored for XML sources appears extremely useful; our approach has been conceived exactly for providing such a chance.

- *It is light*, since it does not exploit any threshold or weight; as a consequence, it does not need any tuning activity; in spite of this, obtained results are satisfactory, as pointed out in Section 3.

Our approach assumes the existence of an *Interschema Property Dictionary* (IPD), i.e., a catalogue storing relationships between *single concepts* represented in the involved XML Schemas. Specifically, it assumes that IPD stores the following properties: *(i) Synonymies*: a synonymy indicates that two concepts have the same meaning; *(ii) Hyponymies/Hypernymies*: given two concepts c_1 and c_2, c_1 is a hyponym of c_2 (which is, in its turn, a hypernym of c_1) if c_1 has a more specific meaning than c_2; *(iii) Overlappings*: an overlapping exists between two concepts if they are neither synonyms nor one a hyponym of the other but represent, to some extent, the same reality. In the literature, many approaches to deriving synonymies, hyponymies and overlappings have been proposed (see, for example, (4; 5; 13; 19)); any of them could be exploited for constructing IPD. However, in the prototype implementing our approach, we have adopted the technique described in (6) for deriving properties to be stored in IPD.

It is worth pointing out that the exploitation of IPD does not introduce scalability problems; in fact, even if IPD must be computed for each pair of XML Schemas into consideration, the worst case time complexity of its derivation is smaller than that associated with the extraction of sub-schema similarities (see (6), Theorems 2.2, 2.4 and 2.5 and Section 3.9).

Given an XML Schema, the number of possible sub-schemas that could be derived from it is extremely high; in certain circumstances it could be even exponential against the number of elements and attributes of the Schema. In order to avoid huge numbers of pairs of sub-schemas to be handled, we propose a heuristic technique for singling out only the most promising ones. A pair of sub-schemas is considered "promising" if the sub-schemas at hand include a large number of pairs of concepts whose similarity has been already stated (i.e., a large number of pairs of concepts for which a synonymy, a hyponymy or an overlapping has been already derived). In this way it is probable that the overall similarity of the promising pair of sub-schemas will be high.

After the most promising pairs of sub-schemas have been selected, they must be examined for detecting those ones that are really similar. The similarity degree associated with each pair of sub-schemas is determined by applying some matching functions defined on suitable bipartite graphs, constructed from the components of the sub-schemas into consideration (see below). The idea underlying the adoption of graph matching algorithms as the core step for "measuring" the similarity of two sub-schemas is motivated by the following reasoning: two sub-schemas can be detected to be similar only if it is possible to state that there exists a form of similarity (e.g., a synonymy, a hyponymy or an overlapping) for many of their elements. The graph matching algorithm is, thus, used to carry out such a verification.

It is worth pointing out that, in the past, we have presented in this journal an approach to extracting interschema properties from XML Schemas (see (6)); this approach was explicitly conceived to extract synonymies, homonymies, hyponymies and overlappings. The approach proposed in this paper derives another kind of interschema properties (i.e., sub-schema similarities), particularly important in the

current Internet era, by following the same guidelines followed in (6). As a consequence, the two papers, in the whole, define a new complete approach for uniformly extracting a large variety of interschema properties. In our opinion, this is a particularly important issue; in fact, as we pointed out also in (6), the capability of uniformly deriving distinct properties appears a crucial feature for a new interschema property derivation approach. As a matter of fact, different strategies for extracting distinct interschema properties could lead to different interpretations of the same reality, and this is a situation that should be avoided.

The outline of the paper is as follows: Section 2 provides a detailed illustration of our approach. Section 3 is devoted to present the results of several tests we have carried out for verifying its performance. In Section 4 we compare it with several approaches previously proposed in the literature. Finally, in Section 5, we draw our conclusions.

2 Approach description

2.1 Preliminary concepts

In this section we introduce some preliminary concepts that will be largely exploited in this paper. Preliminarily, we point out that XML Schemas are usually designed by adopting one of the following three classical techniques: *(i)* the "Russian doll" design, in which the schema structure mirrors the document structure; in particular, it defines one single global element whereas all other elements are local; *(ii)* the "Salami slice" approach, in which, contrarily to the Russian doll, all elements declarations are global; *(iii)* the "Venetian blind" technique, which defines one single global element, as the Russian doll design, but exploits named complex types and element groups instead of element declarations.

Since simple rules have been defined to switch among these three representations, in the following, for the sake of simplicity, we assume that XML Schemas have been designed with the "Salami slice" approach.

First of all we introduce the concept of *x-component*; it denotes an element or an attribute of a Schema S. Given two x-components x_S and x_T of an XML Schema S:

- x_S is defined $veryclose$ to x_T if and only if: *(i)* $x_T = x_S$, or *(ii)* x_T is an attribute of x_S, or *(iii)* x_T is a simple sub-element of x_S.
- x_S is defined $close$ to x_T if and only if x_T is a complex sub-element of x_S.
- x_S is defined $near$ to x_T if and only if x_S is either $veryclose$ or $close$ to x_T.
- x_T is defined $reachable$ from x_S if and only if there exists a sequence of k *distinct* x-components $x_1, x_2, \ldots, x_k$ such that $x_S = x_1$, x_1 is $near$ to x_2, x_2 is $near$ to x_3, ..., x_{k-1} is $near$ to x_k, $x_k = x_T$.

We now introduce the concept of *Connection Cost* $CC(x_S, x_T)$ from an x-component x_S to an x-component x_T. Specifically, *(i)* $CC(x_S, x_T) = 0$ if x_S is $veryclose$ to x_T; *(ii)* $CC(x_S, x_T) = 1$ if x_S is $close$ to x_T; *(iii)* $CC(x_S, x_T) = \mathcal{C}_{\mathcal{ST}}$ if x_T is $reachable$ from x_S and x_S is not $near$ to x_T; $CC(x_S, x_T) = +\infty$ if x_T is not $reachable$ from x_S. Here $\mathcal{C}_{\mathcal{ST}} = min_{x_A}\ (CC(x_S, x_A) + CC(x_A, x_T))$ for each x_A such that $reachable(x_S, x_A) = reachable(x_A, x_T) = true$.

We are now able to introduce the concept of neighborhood of an x-component, that plays a key role in our approach.

Definition 2.1. *Let S be an XML Schema and let x_S be an x-component of S. The* d^{th} neighborhood *of x_S is defined as:*

$$nbh(x_S, d) = \{x_T |\ x_T \textit{ is an x-component of } S, CC(x_S, x_T) \leq d\}$$ □

We call *significant neighborhoods* of x_S all neighborhoods $nbh(x_S, d)$ such that $nbh(x_S, d) \neq nbh(x_S, d-1)$.

As far as the previous concepts are concerned, the following propositions and the following theorem can be introduced; the interested reader can find the corresponding proofs in the Appendix available at the address `http://www.ing.unirc.it/ursino/informatica/Appendix.pdf`.

Proposition 2.1. Let S be an XML Schema; let x_S and x_T be two x-components of S; let m be the number of complex elements of S. If $CC(x_S, x_T) \neq +\infty$, then $CC(x_S, x_T) < m$. □

Proposition 2.2. Let S be an XML Schema; let x_S be an x-component of S; let m be the number of complex elements of S; then $nbh(x_S, d) = nbh(x_S, m-1)$ for each d such that $d \geq m$. □

Theorem 2.1. Let S be an XML Schema; let n be the number of x-components of S. The worst case time complexity for constructing all neighborhoods of all x-components of S is $O(n^3)$. □

Theorem 2.1 is particularly important since it guarantees that our approach is polynomial (see, below, Theorems 2.4 and 2.5). It could appear that a polynomial complexity to the degree of three for neighborhood derivation causes scalability problems for the whole approach. Actually, this is not the case. In fact, in an XML source exploited as a database, the intensional component (i.e., the schema-level information, corresponding, in our application context, to XML Schemas) is generally much smaller than the extensional one (i.e., the instance-level information, corresponding, in our application context, to XML documents); as a consequence, the number of involved x-components (i.e., n) is generally very small. Moreover, the derivation of the neighborhoods of a Schema S must be carried out once and for all when S is examined for the first time; derived neighborhoods can be, then, exploited each time a sub-schema

similarity extraction task involving S is performed. Only a change in the intensional component of S requires to update the corresponding neighborhoods; such a task, however, is uncommon and, in any case, it does not imply to re-compute, but simply to incrementally update, them.

A more detailed analysis concerning the scalability of our approach can be found in Section 3.9.

2.1.1 A case example

Consider the XML Schema S_1, shown in Figure 1, representing a University. Here, *professor* is $veryclose$ to *identifier* because *identifier* is an attribute of *professor*; analogously, *university* is $close$ to *professor* because *professor* is a complex sub-element of *university*; as a consequence, *university* is $near$ to *professor* and *professor* is $near$ to *identifier*; finally, *identifier* is *reachable* from *university* because *university* is $near$ to *professor* and *professor* is $near$ to *identifier*. As for neighborhoods, we have that:

> $nbh(university, 1)$ = *{university, professor, phd-student, paper, course, student, identifier, name, cultural_area, papers, advisor, thesis, research_interests, authors, type, volumes, pages, argument, duration, attended_by, taught_by, program, students, enrollment_year, attends}*

For instance, *professor* belongs to *nbh(university, 1)* because *CC(university, professor) = 1*. All the other neighborhoods can be determined analogously.

2.2 Selection of the most promising pairs of sub-schemas

2.2.1 Overview

The first problem our approach must face is the extremely high number of possible sub-schemas that could be derived from an XML Schema S; in fact, this number might be exponential against the number of x-components of S.

In order to avoid huge numbers of pairs of sub-schemas to be examined, we have designed a heuristic technique for singling out only the most promising ones. This technique receives two XML Schemas S_1 and S_2 and an Interschema Property Dictionary IPD, storing synonymies, hyponymies and overlappings holding between complex elements of S_1 and S_2. The most promising pairs of sub-schemas are derived as follows: for each pair $\langle x_{1_j}, x_{2_k}\rangle$ belonging to IPD, such that $x_{1_j} \in S_1$ and $x_{2_k} \in S_2$, x_{1_j} and x_{2_k} are taken as the "seeds" for the construction of promising pairs of sub-schemas.

Specifically, our technique:

- considers the pairs $\langle nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma)\rangle$, such that $nbh(x_{1_j}, \delta)$, (resp., $nbh(x_{2_k}, \gamma)$) is a *significant neighborhood* (see Section 2.1) of x_{1_j} (resp., x_{2_k});
- derives a pair of sub-schemas $\langle prosub_{1j_\delta}, prosub_{2k_\gamma}\rangle$, from each pair $\langle nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma)\rangle$, such that $prosub_{1j_\delta}$ (resp., $prosub_{2k_\gamma}$) is obtained from $nbh(x_{1_j}, \delta)$ (resp., $nbh(x_{2_k}, \gamma)$) by removing from it those portions that are dissimilar with $nbh(x_{2_k}, \gamma)$ (resp., $nbh(x_{1_j}, \delta)$), i.e., those x-components not involved in semantic relationships with x-components of $nbh(x_{2_k}, \gamma)$ (resp., $nbh(x_{1_j}, \delta)$) - see below for more details.

2.2.2 Technical Details

In this section we formalize our technique for selecting the most promising pairs of sub-schemas. Specifically, given two XML Schemas S_1 and S_2, the set SPS of the most promising pairs of sub-schemas associated with them is obtained by calling a suitable function Φ as follows:

$$SPS = \Phi\left(S_1, S_2, IPD\right)$$

For each tuple $\langle x_{1_j}, x_{2_k}\rangle \in IPD$, Φ invokes a function Ψ for deriving the set of the most promising pairs of sub-schemas having x_{1_j} and x_{2_k} as their seeds. The formal definition of Φ is:

$$\Phi\left(S_1, S_2, IPD\right) = \bigcup_{\langle x_{1_j}, x_{2_k}\rangle \in IPD} \Psi\left(S_1, S_2, x_{1_j}, x_{2_k}, IPD\right)$$

The function Ψ receives two XML Schemas S_1 and S_2, two complex elements $x_{1_j} \in S_1$ and $x_{2_k} \in S_2$ and an Interschema Property Dictionary IPD; for each pair of significant neighborhoods $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$, Ψ calls a function ξ, which extracts the most promising pair of sub-schemas $\langle prosub_{1j_\delta}, prosub_{2k_\gamma}\rangle$ associated with it. Ψ can be defined as follows:

$$\Psi\left(S_1, S_2, x_{1_j}, x_{2_k}, IPD\right) = \bigcup_{\substack{0\le\delta<\mu(S_1)\\ 0\le\gamma<\mu(S_2)}} \xi\left(S_1, S_2, nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma), \nu\left(IPD, nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma)\right)\right)$$

Here, the function μ receives an XML Schema and returns the number of its complex elements. The function ν receives an Interschema Property Dictionary IPD and two neighborhoods $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$; it returns the set $IPD_{\delta\gamma} \subseteq IPD$ of interschema properties involving only pairs of x-components belonging to both $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$.

The function ξ receives two XML Schemas S_1 and S_2, two neighborhoods $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$ and the set $IPD_{\delta\gamma}$, as constructed by ν; in order to extract the most promising pair of sub-schemas $\langle prosub_{1j_\delta}, prosub_{2k_\gamma}\rangle$, associated with $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$, ξ activates functions ζ, θ and π for pruning $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$ in such a way as to eliminate the most dissimilar portions. ξ can be formalized as follows:

$$\xi\left(S_1, S_2, nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma), IPD_{\delta\gamma}\right) = \langle\zeta(\theta(nbh(x_{1_j}, \delta), \pi(S_1, IPD_{\delta\gamma})), S_1), \zeta(\theta(nbh(x_{2_k}, \gamma), \pi(S_2, IPD_{\delta\gamma})), S_2)\rangle$$

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
            <xs:attribute name="identifier" type="xs:ID"/>
            <xs:attribute name="name" type="xs:string"/>
            <xs:attribute name="cultural_area" type="xs:string"/>
            <xs:attribute name="papers" type="xs:IDREFS"/>
            <xs:attribute name="advisor" type="xs:IDREF"/>
            <xs:attribute name="thesis" type="xs:string"/>
            <xs:attribute name="research_interests" type="xs:string"/>
            <xs:attribute name="authors" type="xs:IDREFS"/>
            <xs:attribute name="type" type="xs:string"/>
            <xs:attribute name="volumes" type="xs:integer"/>
            <xs:attribute name="pages" type="xs:integer"/>
            <xs:attribute name="argument" type="xs:string"/>
            <xs:attribute name="duration" type="xs:duration"/>
            <xs:attribute name="attended_by" type="xs:IDREFS"/>
            <xs:attribute name="taught_by" type="xs:IDREFS"/>
            <xs:attribute name="program" type="xs:string"/>
            <xs:attribute name="students" type="xs:IDREFS"/>
            <xs:attribute name="enrollment_year" type="xs:date"/>
            <xs:attribute name="attends" type="xs:IDREFS"/>
    <xs:element name="professor">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="name"/>
            <xs:attribute ref="cultural_area"/>
            <xs:attribute ref="papers"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="phd-student">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="advisor"/>
            <xs:attribute ref="thesis"/>
            <xs:attribute ref="research_interests"/>
            <xs:attribute ref="papers"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="paper">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="type"/>
            <xs:attribute ref="volumes"/>
            <xs:attribute ref="pages"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="course">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="name"/>
            <xs:attribute ref="argument"/>
            <xs:attribute ref="duration"/>
            <xs:attribute ref="attended_by"/>
            <xs:attribute ref="taught_by"/>
            <xs:attribute ref="program"/>
            <xs:attribute ref="students"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="student">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="name"/>
            <xs:attribute ref="enrollment_year"/>
            <xs:attribute ref="attends"/>
        </xs:complexType>
    </xs:element>
    <!-- root -->
    <xs:element name="university">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="professor" maxOccurs="unbounded"/>
                <xs:element ref="phd-student" maxOccurs="unbounded"/>
                <xs:element ref="paper" maxOccurs="unbounded"/>
                <xs:element ref="course" maxOccurs="unbounded"/>
                <xs:element ref="student" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 1: The XML Schema S_1

Here, the function π receives an XML Schema S_h, $h \in \{1, 2\}$, and the set $IPD_{\delta\gamma}$, computed by ν; it returns the set $AtLeastOne$ of the complex elements belonging to S_h and involved in at least one property of $IPD_{\delta\gamma}$.

The function θ receives a neighborhood $nbh(x_S, d)$, associated with an XML Schema S, and the set $AtLeastOne$ as computed by the function π. It constructs a set of x-components $XSet_{S_d} \subseteq nbh(x_S, d)$ by removing from $nbh(x_S, d)$ each complex element x_R (along with all its sub-elements and attributes) that satisfies both the following conditions: *(i)* $x_R \notin AtLeastOne$; *(ii)* for each complex element x_{R_i} such that $reachable(x_R, x_{R_i}) = true$ and $x_{R_i} \in nbh(x_S, d)$, $x_{R_i} \notin AtLeastOne$.

In other words a complex element x_R, belonging to $nbh(x_S, d)$, is not inserted in $XSet_{S_d}$ if both it and all complex elements in $nbh(x_S, d)$ reachable from it are not involved in any interschema property stored in $IPD_{\delta\gamma}$. Note that the two conditions above guarantee that if x_R is not inserted in $XSet_{S_d}$, then no x-components reachable from it are inserted therein. In fact, if the two conditions above are valid for x_R, then they must be also valid for all x-components reachable from it.

The function ζ receives the set of x-components $XSet_{S_d}$ returned by θ and constructs a sub-schema $prosub_{S_d}$ taking into account the initial structure of S. Specifically, $prosub_{S_d}$ is constructed from $XSet_{S_d}$ in such a way that the following two conditions hold: *(i)* it must contain all, and only, the x-components of $XSet_{S_d}$; *(ii)* all x-components of $XSet_{S_d}$ must preserve, in $prosub_{S_d}$, the same hierarchical organization they have in S[1].

The next theorems state the worst case time complexity for computing all promising pairs of sub-schemas, as well as an upper bound to the number of promising pairs of sub-schemas returned by the function Φ. Their proofs can be found in the Appendix available at the address `http://www.ing.unirc.it/ursino/informatica/Appendix.pdf`.

Theorem 2.2. Let S_1 and S_2 be two XML Schemas. Let IPD be the Interschema Property Dictionary associated with S_1 and S_2; let m be the maximum between the number of complex elements of S_1 and S_2; let n be the maximum between the number of x-components of S_1 and S_2. The worst case time complexity for computing, by means of the function Φ, the set SPS of the most promising pairs of sub-schemas associated with S_1 and S_2 is $max\{O(m^7), O(m^4 \times n^2)\}$. □

Theorem 2.3. Let S_1 and S_2 be two XML Schemas; let IPD be the corresponding Interschema Property Dictionary; let m be the maximum between the number of complex elements of S_1 and S_2. The maximum cardinality of SPS is $O(m^4)$. □

As for these two theorems, all considerations about the value of n, that we have drawn after Theorem 2.1, are still valid. Moreover, since in an XML document the number of attributes and simple elements is generally much greater than the number of complex elements, the value of m is even much smaller than that of n.

[1]Note that the sub-schema $prosub_{S_d}$ obtained by the function ζ is a well-formed and self contained XML Schema because of the function θ. In fact, this function constructs $XSet_{S_d}$ in such a way that, if an x-component is not inserted in $XSet_{S_d}$, then no x-components reachable from it are inserted therein.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="first_name" type="xs:string"/>
    <xs:attribute name="last_name" type="xs:string"/>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="roles" type="xs:string"/>
    <xs:attribute name="research" type="xs:string"/>
    <xs:attribute name="argument" type="xs:string"/>
    <xs:attribute name="budget" type="xs:string"/>
    <xs:attribute name="funds" type="xs:string"/>
    <xs:attribute name="responsibles" type="xs:IDREFS"/>
    <xs:attribute name="termination" type="xs:date"/>
    <xs:attribute name="authors" type="xs:IDREFS"/>
    <xs:attribute name="title" type="xs:string"/>
    <xs:attribute name="volume" type="xs:integer"/>
    <xs:attribute name="pages" type="xs:integer"/>
    <xs:attribute name="year" type="xs:date"/>
    <xs:attribute name="booktitle" type="xs:string"/>
    <xs:attribute name="address" type="xs:string"/>
    <xs:attribute name="publisher" type="xs:string"/>
    <xs:attribute name="chief" type="xs:IDREF"/>
    <xs:attribute name="people" type="xs:IDREF"/>
    <xs:attribute name="projects" type="xs:IDREFS"/>
    <xs:attribute name="locations" type="xs:string"/>
    <xs:attribute name="labs" type="xs:string"/>
    <xs:element name="article">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="journal"/>
                <xs:element ref="conference"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="researcher">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="first_name"/>
            <xs:attribute ref="last_name"/>
            <xs:attribute ref="type"/>
            <xs:attribute ref="roles"/>
            <xs:attribute ref="research"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="project">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="argument"/>
            <xs:attribute ref="budget"/>
            <xs:attribute ref="funds"/>
            <xs:attribute ref="responsibles"/>
            <xs:attribute ref="termination"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="journal">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="title"/>
            <xs:attribute ref="volume"/>
            <xs:attribute ref="pages"/>
            <xs:attribute ref="year"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="conference">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="title"/>
            <xs:attribute ref="booktitle"/>
            <xs:attribute ref="address"/>
            <xs:attribute ref="year"/>
            <xs:attribute ref="pages"/>
            <xs:attribute ref="publisher"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="department">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="chief"/>
            <xs:attribute ref="people"/>
            <xs:attribute ref="projects"/>
            <xs:attribute ref="locations"/>
            <xs:attribute ref="labs"/>
        </xs:complexType>
    </xs:element>
    <!-- root -->
    <xs:element name="university">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="article" maxOccurs="unbounded"/>
                <xs:element ref="project" maxOccurs="unbounded"/>
                <xs:element ref="researcher" maxOccurs="unbounded"/>
                <xs:element ref="department" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 2: The XML Schema S_2

x-component of S_1	*x-component of* S_2	*interschema property typology*
university	university	synonymy
professor	researcher	overlapping
phd-student	researcher	overlapping
paper	article	synonymy
paper	journal	hyponymy
paper	conference	hyponymy

Table 1: The Interschema Property Dictionary IPD associated with S_1 and S_2

2.2.3 A case example (cnt'd)

Consider the XML Schemas S_1 and S_2, associated with a University and illustrated in Figures 1 and 2. Consider the corresponding Interschema Property Dictionary IPD shown in Table 1 [2].

In order to construct SPS, first the function Φ is activated. For each tuple of IPD, Φ calls the function Ψ. In order to show the behaviour of Ψ, we consider its application to the pair of complex elements $\langle university_{[S_1]}, university_{[S_2]} \rangle$ [3].

For each pair $\langle nbh(university_{[S_1]}, \delta), nbh(university_{[S_2]}, \gamma) \rangle$ of significant neighborhoods, Ψ activates the function ξ. In order to illustrate the behaviour of ξ, we consider its application to $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$; $nbh(university_{[S_1]}, 1)$ has been shown in the previous section; $nbh(university_{[S_2]}, 2)$ is as follows:

> $nbh(university_{[S_2]}, 2)$ = *{university, article, project, researcher, department, journal, conference, ID, first_name, last_name, type, roles, research, argument, budget, funds, responsibles, termination, authors, title, volume, pages, year, booktitle, address, publisher, chief, people, projects, locations, labs}*

For this pair of neighborhoods the set $IPD_{\delta\gamma}$, returned by the function ν, is equal to IPD. ξ activates θ for pruning $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$ in such a way as to remove the most dissimilar portions. As an example, the complex element $student_{[S_1]}$ and all its attributes are pruned from $nbh(university_{[S_1]}, 1)$ because: *(i)* $student_{[S_1]}$ is not involved in any interschema property of $IPD_{\delta\gamma}$; *(ii)* there does not exist any complex element x_{R_i} such that $reachable(student_{[S_1]}, x_{R_i}) = true$, $x_{R_i} \in nbh(university_{[S_1]}, 1)$, and x_{R_i} is involved in some interschema property of $IPD_{\delta\gamma}$.

The final sets of x-components returned by the function θ, when applied on $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$, are:

> *{university, professor, phd-student, paper, identifier, name, cultural_area, papers, advisor, thesis, research_interests, authors, type, volumes, pages}*

[2] As previously pointed out, we have chosen to construct IPD by applying the approaches described in (6); however, any other approach proposed in the literature for deriving synonymies, hyponymies and overlappings among elements of different XML Schemas could be exploited.

[3] Here and in the following, whenever necessary, we use the notation $x_{[S]}$ for indicating the x-component x of an XML Schema S.

{university, article, researcher, journal, conference, ID, first_name, last_name, type, roles, research, authors, title, organization, year, booktitle, address, pages, publisher}

After this, ξ activates ζ that constructs the promising sub-schemas corresponding to $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$.

The final promising pair of sub-schemas corresponding to $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$ returned by ξ is illustrated in Figure 3. All the other promising pairs of sub-schemas can be determined analogously.

2.3 Derivation of sub-schema similarities

In the previous section we have seen how the most promising pairs of sub-schemas can be determined. In this section we illustrate how these pairs can be analyzed in order to derive sub-schema similarities. Before describing this task in detail some preliminary considerations are needed.

Applications possibly exploiting sub-schema similarities (and, more in general, interschema properties) are extremely heterogeneous. Some of them (i.e., the most critical ones) require, for each pair of involved sub-schema similarities, a high level of trustworthiness; in other words they require the correspondences between the elements belonging to the involved sub-schemas to be precisely and unambiguously determined. In order to achieve this guarantee, it is necessary to pay the price of filtering out the weakest sub-schema similarities, i.e., those involving sub-schemas whose elements might have a form of similarity different from synonymy. In fact, synonymy represents the strongest form of similarity; it is the only one capable of guaranteeing that the similar elements belonging to the involved sub-schemas can be precisely and unambiguously mapped each other.

By contrast, other (possibly non critical) applications could prefer to have a more complete picture of existing sub-schema similarities and, therefore, could choose to consider also the weakest ones. As previously pointed out, while a strong similarity requires most of the elements of the corresponding schemas to be related by a synonymy, a weak similarity can accept other kinds of similarity property, e.g., hyponymies and overlappings; it can also exist between two schemas characterized by quite a different structure. As a consequence of these reasonings, even if weak similarities cannot be considered in critical applications, in non-critical scenarios they can provide a richer vision of the reality.

Clearly, the two exigencies outlined above (i.e., strength and breadth of discovered similarities) are divergent and, consequently, it appears extremely difficult to satisfy both of them simultaneously.

In order to address this issue, our technique allows the derivation of two levels of sub-schema similarities, namely strong similarities, that guarantee a strong correspondence between the x-components of similar sub-schemas, and weak similarities, that allow the existence of less characterizing semantic relationships between the corresponding x-components. Specifically, *strong sub-schema similarities* are derived by taking only synonymies into account; *weak sub-schema similarities* cannot be derived with the only support of synonymies but need also the contribution of hyponymies and overlappings.

Our technique for deriving sub-schema similarities between two XML Schemas S_1 and S_2 receives the set SPS of the most promising pairs of sub-schemas and the Interschema Property Dictionary IPD associated with S_1 and S_2 and selects two sets of pairs of similar sub-schemas, namely:

$$SSS_{strong} = \rho_{strong}(SPS, IPD)$$
$$SSS_{weak} = \rho_{weak}(SPS, IPD)$$

Here, the function ρ_{strong} derives the strong sub-schema similarities, whereas the function ρ_{weak} extracts the weak ones.

2.3.1 Derivation of strong similarities

ρ_{strong} operates by computing the objective function associated with a maximum weight matching defined on a suitable bipartite graph. Specifically, let $\langle prosub_{1j_\delta}, prosub_{2k_\gamma}\rangle \in SPS$ be a promising pair of sub-schemas; let $BG_{\delta\gamma} = \langle NSet, ESet\rangle$ be the bipartite graph associated with $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$. $NSet = PSet \cup QSet$ is the set of nodes of $BG_{\delta\gamma}$; there is a node in $PSet$ (resp., $QSet$) for each complex element of $prosub_{1j_\delta}$ (resp., $prosub_{2k_\gamma}$). $ESet$ is the set of edges of $BG_{\delta\gamma}$; in $ESet$ there exists an edge $\langle p, q\rangle$ between two nodes $p \in PSet$ and $q \in QSet$ if and only if, in IPD, there exists a synonymy between the element corresponding to p and that corresponding to q.

The maximum weight matching on $BG_{\delta\gamma}$ is the set $ESet^* \subseteq ESet$ such that, for each node $x \in NSet$, there exists at most one edge of $ESet^*$ incident onto x and $|ESet^*|$ is maximum (the interested reader is referred to (15) for details about the maximum weight matching problem). The objective function we associate with the maximum weight matching is $\chi_{BG} = \frac{2|ESet^*|}{|PSet|+|QSet|}$. Here $|ESet^*|$ represents the number of matches associated with $BG_{\delta\gamma}$, as well as the number of synonymies involving $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$. $2|ESet^*|$ indicates the number of matching nodes in $BG_{\delta\gamma}$, as well as the number of similar complex elements present in $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$. $|PSet| + |QSet|$ denotes the total number of nodes in $BG_{\delta\gamma}$ as well as the total number of complex elements associated with $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$. Finally, χ_{BG} represents the share of matching nodes in $BG_{\delta\gamma}$, as well as the share of similar complex elements present in $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$.

We assume that $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$ are similar if $\chi_{BG} > \frac{1}{2}$. Such an assumption derives from the consideration that two sets of objects can be considered similar if the number of similar elements is greater than the number

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:attribute name="identifier" type="xs:ID"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="cultural_area" type="xs:string"/>
    <xs:attribute name="papers" type="xs:IDREFS"/>
    <xs:attribute name="advisor" type="xs:IDREF"/>
    <xs:attribute name="thesis" type="xs:string"/>
    <xs:attribute name="research_interests" type="xs:string"/>
    <xs:attribute name="authors" type="xs:IDREFS"/>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="volumes" type="xs:integer"/>
    <xs:attribute name="pages" type="xs:integer"/>
    <xs:element name="professor">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="name"/>
            <xs:attribute ref="cultural_area"/>
            <xs:attribute ref="papers"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="phd-student">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="advisor"/>
            <xs:attribute ref="thesis"/>
            <xs:attribute ref="research_interests"/>
            <xs:attribute ref="papers"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="paper">
        <xs:complexType>
            <xs:attribute ref="identifier"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="type"/>
            <xs:attribute ref="volumes"/>
            <xs:attribute ref="pages"/>
        </xs:complexType>
    </xs:element>
    <!-- root -->
    <xs:element name="university">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="professor" maxOccurs="unbounded"/>
                <xs:element ref="phd-student" maxOccurs="unbounded"/>
                <xs:element ref="paper" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:attribute name="ID" type="xs:ID"/>
    <xs:attribute name="first_name" type="xs:string"/>
    <xs:attribute name="last_name" type="xs:string"/>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="roles" type="xs:string"/>
    <xs:attribute name="research" type="xs:string"/>
    <xs:attribute name="authors" type="xs:IDREFS"/>
    <xs:attribute name="title" type="xs:string"/>
    <xs:attribute name="organization" type="xs:string"/>
    <xs:attribute name="year" type="xs:date"/>
    <xs:attribute name="booktitle" type="xs:string"/>
    <xs:attribute name="address" type="xs:string"/>
    <xs:attribute name="pages" type="xs:integer"/>
    <xs:attribute name="publisher" type="xs:string"/>
    <xs:element name="article">
        <xs:complexType>
            <xs:choice>
                <xs:element ref="journal"/>
                <xs:element ref="conference"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="researcher">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="first_name"/>
            <xs:attribute ref="last_name"/>
            <xs:attribute ref="type"/>
            <xs:attribute ref="roles"/>
            <xs:attribute ref="research"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="journal">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="title"/>
            <xs:attribute ref="organization"/>
            <xs:attribute ref="year"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="conference">
        <xs:complexType>
            <xs:attribute ref="ID"/>
            <xs:attribute ref="authors"/>
            <xs:attribute ref="title"/>
            <xs:attribute ref="booktitle"/>
            <xs:attribute ref="address"/>
            <xs:attribute ref="year"/>
            <xs:attribute ref="pages"/>
            <xs:attribute ref="publisher"/>
        </xs:complexType>
    </xs:element>
    <!-- root -->
    <xs:element name="university">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="article" maxOccurs="unbounded"/>
                <xs:element ref="researcher" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 3: The promising pair of sub-schemas associated with $nbh(university_{[S_1]}, 1)$ and $nbh(university_{[S_2]}, 2)$

of the dissimilar ones or, in other words, if the number of similar elements is greater than half of the total number of elements.

The following theorem states the worst case time complexity for computing all strong similarities. Its proof can be found in the Appendix at the address `http://www.ing.unirc.it/ursino/informatica/Appendix.pdf`.

Theorem 2.4. Let S_1 and S_2 be two XML Schemas; let IPD be the corresponding Interschema Property Dictionary; let m be the maximum between the number of complex elements of S_1 and S_2. The worst case time complexity for computing SSS_{strong} is $O(m^7)$. □

With regard to this result, the same reasoning about the extremely small number of complex elements in an XML Schema, that we have presented after Theorems 2.2 and 2.3, is still valid.

2.3.2 Derivation of weak similarities

ρ_{weak} receives SPS and IPD and returns weak sub-schema similarities. We call them "weak" because, differently from ρ_{strong}, which takes only synonymies into account, ρ_{weak} considers also overlappings and hyponymies, that are weaker properties than synonymies in the representation of concept similarities.

When we introduce hyponymies and overlappings in the computation of sub-schema similarities we must consider that, often, more than one element of a schema could be hyponymous or overlapping with an element of the other schema.

A consequence of this reasoning is that, in order to derive weak sub-schema similarities, it is not suitable to apply maximum weight matching techniques; in fact, they would associate an element of a schema with at most one element of the other schema.

Taking into account the reasoning above, ρ_{weak} has been defined as follows. Let $\langle prosub_{1j_\delta}, prosub_{2k_\gamma}\rangle \in SPS$ be a promising pair of sub-schemas; let $BG'_{\delta\gamma} = \langle NSet', ESet'\rangle$ be a bipartite graph associated with $prosub_{1j_\delta}$ and $prosub_{2k_\gamma}$. Here, $NSet' = PSet' \cup QSet'$ is the set of nodes of $BG'_{\delta\gamma}$; there is a node in $PSet'$ (resp., $QSet'$) for each complex element of $prosub_{1j_\delta}$ (resp., $prosub_{2k_\gamma}$). $ESet'$ is the set of edges of $BG'_{\delta\gamma}$;

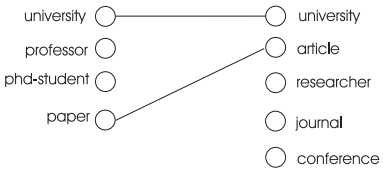

Figure 4: The bipartite graph $BG_{\delta\gamma}$ associated with the promising pair of sub-schemas illustrated in Figure 3

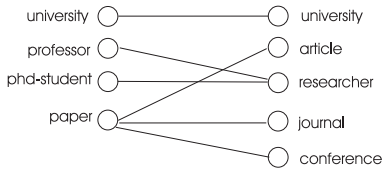

Figure 5: The bipartite graph $BG'_{\delta\gamma}$ associated with the promising pair of sub-schemas illustrated in Figure 3

in $ESet'$ there exists an edge $\langle p, q\rangle$ between two nodes $p \in PSet'$ and $q \in QSet'$ if and only if, in IPD, a synonymy, a hyponymy or an overlapping holds between the element corresponding to p and that corresponding to q.

Let η_p and η_q be the sets of nodes of $PSet'$ and $QSet'$ involved in at least one interschema property; specifically, $\eta_p = \{p \in PSet'$ such that at least one edge of $BG'_{\delta\gamma}$ is incident onto it$\}$ and $\eta_q = \{q \in QSet'$ such that at least one edge of $BG'_{\delta\gamma}$ is incident onto it$\}$; we assume that $prosub_{1_{j\delta}}$ and $prosub_{2_{k\gamma}}$ are weakly similar if $\chi'_{BG'} = \frac{|\eta_p|+|\eta_q|}{|PSet'|+|QSet'|} > \frac{1}{2}$. Such an assumption indicates that two sub-schemas are weakly similar if at least half of their elements are someway related by an interschema property. The justification underlying such an assumption is analogous to that we have seen for strong similarities.

The following theorem states the worst case time complexity for computing all strong similarities. Its proof can be found in the Appendix at the address `http://www.ing.unirc.it/ursino/informatica/Appendix.pdf`.

Theorem 2.5. Let S_1 and S_2 be two XML Schemas; let IPD be the corresponding Interschema Property Dictionary; let m be the maximum between the number of complex elements of S_1 and S_2. The worst case time complexity for computing SSS_{weak} is $O(m^6)$. □

2.3.3 A case example (cnt'd)

Consider the XML Schemas illustrated in Figures 1 and 2 and the promising pair of sub-schemas derived in Section 2.2.3 and illustrated in Figure 3.

For this pair, $BG_{\delta\gamma}$ is shown in Figure 4; the objective function χ_{BG} computed on it is equal to $\frac{4}{9} < \frac{1}{2}$; as a consequence, we can conclude that the sub-schemas of the pair are not strongly similar.

For the same pair, $BG'_{\delta\gamma}$ is shown in Figure 5; the value of $\chi'_{BG'}$, computed by ρ_{weak}, is equal to $\frac{9}{9} > \frac{1}{2}$, which allows us to conclude that a weak similarity holds between the two sub-schemas into consideration.

3 Experimental results

3.1 Introduction

In this section we provide a detailed description of the experiments we have carried out in order to test the performance of our approach. Specifically, in Section 3.2 we describe the characteristics of the sources exploited in our experimental tests. The adopted accuracy measures are illustrated in Section 3.3; the results we have obtained by applying these measures are presented in Section 3.4. In Section 3.5 we compare the accuracy of our approach with that achieved by some other approaches previously proposed in the literature. Section 3.6 illustrates our study about the role of our heuristics for the extraction of promising pairs of sub-schemas in the improvement of the efficiency of our approach. An analysis about the improvements of our approach against manual "naive" approaches, based on identifying synonyms and expanding around them for constructing sub-schemas, is illustrated in Section 3.7. The robustness of our approach is estimated in Section 3.8. Finally, Section 3.9 is devoted to discuss our experimental results about its scalability.

3.2 Characteristics of the exploited sources

In our tests we have exploited a large variety of XML Schemas associated with disparate application contexts, such as Biomedical Data, Project Management, Property Register, Industrial Companies, Universities, Airlines, Scientific Publications and Biological Data.

Specifically, we have compared all pairs of schemas within a particular domain. Biomedical Schemas have been derived from various sites; one of these sites has been `http://www.biomediator.org`. Schemas concerning Project Management, Property Register and Industrial Companies have been derived from Italian Central Government Office sources and are shown at the address `http://www.mat.unical.it/terracina/tests.html`. Schemas concerning Universities have been downloaded from the Web address `http://anhai.cs.uiuc.edu/archive/domains/courses.html`. Schemas concerning Airlines have been found in (26). Schemas concerning Scientific Publications have been supplied by the authors of (17). Finally, Biological Schemas have been downloaded from the addresses `http://smi-web.stanford./`

edu.projects/helix/pubs/ismb02/schemas/, http://www.cs.toronto.edu/db/clio/testSchemas.html and http://www.genome.ad.jp/kegg/genes.html.

Examined sources were characterized by the following properties: *(i) Number of schemas*: we have considered 30 XML Schemas whose characteristics are reported in Table 2. *(ii) Maximum depth of schemas*: for each domain we have computed the maximum depth of the involved schemas; it is shown in the third column of Table 2. In our opinion, this parameter is particularly interesting for an approach specifically conceived for XML Schemas; in fact, the maximum depth is an indicator of the complexity of the sub-schemas that can be generated. *(iii) Size of schemas*: the size of the evaluated XML Schemas, i.e., the number of their elements and attributes, are shown in the fourth column of Table 2. The size of a test schema is relevant because it influences the quality of obtained results; in fact, as mentioned in (8), the bigger the input schemas are, the greater the search space for candidate pairs is, and the lower the quality of obtained results will be.

The *total number of pairs of schemas we have compared* for each domain is shown in the last column of Table 2.

3.3 Accuracy Measures exploited in our experimental tests

All accuracy measures adopted in our experimental tests have been computed according to the following general framework: *(i)* a set of experts has been asked to identify the sub-schema similarities existing among involved schemas; *(ii)* sub-schema similarities among the same schemas have been determined by running our algorithm; *(iii)* the sub-schema similarities provided by the experts and those returned by our algorithm have been compared and accuracy measures have been computed.

The number of experts that have been involved in manually solving the match tasks is as follows: 6 for Biomedical Data, 3 for Project Management, 3 for Property Register, 4 for Industrial Companies, 4 for Universities, 2 for Airlines, 2 for Scientific Publications and 7 for Biological Data.

Let A be the set of sub-schema similarities provided by the experts and let C be the set of sub-schema similarities returned by our approach; two basic accuracy measures are:

- *Precision* (hereafter *Pre*), that specifies the share of correct sub-schema similarities detected by the system among those it derived. It is defined as: $Pre = \frac{|A \cap C|}{|C|}$.
- *Recall* (hereafter *Rec*), that indicates the share of correct sub-schema similarities detected by the system among those the experts provided. It is defined as: $Rec = \frac{|A \cap C|}{|A|}$.

Precision and Recall are typical measures of Information Retrieval (see (29)). Both of them fall within the real interval $[0, 1]$; in the ideal case (i.e., when $A \equiv C$) they are both equal to 1.

Property Typology	*Average Precision*	*Average Recall*	*Average F-Measure*	*Average Overall*
weak similarities	0.89	0.77	0.83	0.67
strong similarities	0.92	0.72	0.81	0.66

Table 3: Accuracy measures of our approach for weak and strong similarities

However, neither Precision nor Recall alone can accurately measure the quality of an interschema property extraction algorithm; in order to improve the quality of results, it appears necessary the computation of a joint measure of them. Two very popular measures satisfying these requirements are:

- *F-Measure* (3; 29), that represents the harmonic mean between Precision and Recall. It is defined as: $F\text{-}Measure = 2 \cdot \frac{Pre \cdot Rec}{Pre + Rec}$.
- *Overall* (9; 22), that measures the post-match effort needed for adding false negatives and removing false positives from the set of similarities returned by the system to evaluate. It is defined as: $Overall = Rec \cdot (2 - \frac{1}{Pre})$.

F-Measure falls within the interval $[0, 1]$ whereas Overall ranges between $-\infty$ and 1; the higher they are, the better the accuracy of the tested approach will be.

3.4 Discussion of obtained results

As for the evaluation of Precision and Recall associated with our approach we considered particularly interesting to compute them by distinguishing weak and strong similarities. Before the experiments we expected that passing from weak to strong similarities would have caused an increase of the Precision and a decrease of the Recall of our approach. This intuition was motivated by considering that strong similarities are a subset of the weak ones, obtained from them by eliminating the most uncertain ones; this should cause the set of strong similarities to be more precise than the set of the weak ones. However, this filtering task could erroneously discard some valid similarities; for this reason the set of strong similarities could have a lower Recall w.r.t. the set of the weak ones.

In order to verify this intuition and, possibly, to quantify it, we have applied our approach on all pairs of XML Schemas belonging to the same application domain and we have computed Precision, Recall, F-Measure and Overall for each pair into consideration; after this, we have computed the average values of all obtained measures for weak and strong similarities; they are reported in Table 3. From the analysis of this table we can draw the following conclusions:

- As for weak similarities, *(i)* Precision is quite high, even if our approach returns some false positives; *(ii)* Recall is quite high, given the specificity of the interschema property typology we are studying in this

Application context	*Number of Schemas*	*Maximum depth of Schemas*	*Minimum, Average and Maximum Number of x-components*	*Minimum, Average and Maximum Number of complex elements*	*Total Number of Comparisons*
Biomedical Data	6	8	15 - 26 - 38	4 - 8 - 16	15
Project Management	3	4	37 - 40 - 42	6 - 7 - 8	3
Property Register	2	4	64 - 70 - 75	14 - 14 - 14	1
Industrial Companies	5	4	23 - 28 - 46	6 - 8 - 9	10
Universities	5	5	15 - 17 - 19	3 - 4 - 5	10
Airlines	2	4	12 - 13 - 13	4 - 4 - 4	1
Scientific Publications	2	6	17 - 18 - 18	8 - 9 -9	1
Biological Data	5	8	70 - 136 - 262	21 - 41 - 103	10

Table 2: Characteristics of the XML Schemas exploited for testing the performance of our approach

paper (see below); as a consequence, our approach returns most of the valid properties or, in other words, it returns a very small number of false negatives.

– If we consider the strong similarities, *(i)* the set of similarities returned by our approach contains a smaller number of false positives w.r.t. the previous case; specifically, Precision increases about 4%; *(ii)* Recall decreases of about 6% w.r.t. the previous case; in other words, a certain increase of false negatives can be observed.

All these experiments confirm our original intuition about the trend of Precision and Recall when passing from weak to strong similarities.

Note that in the weak similarity context a user is willing to accept false positives if this allows him to obtain a wide set of similarities. On the contrary, in the strong similarity context, a user is willing to receive an incomplete set of similarities by the system but he desires that proposed properties are (almost surely) correct. These observations fully agree with the trend of Precision and Recall registered in our tests.

As a final remark about this experiment, we observe that the not particularly high values of Recall are explained by considering that: *(i)* the possible number of sub-schema similarities might be exponential against the number of x-components of the corresponding XML Schemas; *(ii)* we have used a heuristics for selecting the most promising pairs of sub-schemas.

After this, we have computed the variation of our accuracy measures in presence of a variation of the dimension of the input schemas. Specifically, given two XML Schemas S_1 and S_2 such that $n_1 = |XCompSet(S_1)|$ and $n_2 = |XCompSet(S_2)|$, we have computed the average values of Precision, Recall, F-Measure and Overall for the extraction of weak and strong similarities for different values of the number of involved x-components $n_t = n_1 + n_2$.

The obtained results are shown in Figures 6, 7, 8 and 9. From their analysis it is possible to conclude that all our accuracy measures slightly decrease in presence of an increase of n_t. Such quite an intuitive result confirms observations and results presented in (8).

Finally, we have verified if the accuracy of our approach depends on the application domain which the test XML Schemas belong to. The value of each accuracy measure for a domain has been determined by computing the accuracy measure for all possible pairs of XML Schemas belonging to the domain and, then, by averaging these values. The results we have obtained are shown in Figures 10, 11, 12 and 13. From the analysis of these figures it is possible to conclude that the accuracy of our approach is quite independent of the application domain (the only, quite significant, differences can be found in the biological domain). As far as our experiments are concerned, we have obtained the best accuracy for the Airlines domain; here, Precision reaches its best value, i.e., 0.94, obtained for the derivation of strong similarities; Recall, F-Measure and Overall are maximum for the extraction of weak similarities and are 0.79, 0.84 and 0.71, respectively. The worst accuracy results have been obtained in the Biological domain; here, Precision is maximum for the strong similarity derivation and is 0.86; Recall, F-Measure and Overall reach their best values for the weak similarity extraction and are 0.70, 0.76 and 0.55, respectively.

3.5 Comparison of the accuracy of our approach with that achieved by some related approaches

In this section we report the results of some experimental tests aiming to compare the accuracy of our approach with that achieved by other approaches already presented in the literature.

Our experimental comparison has been inspired by the ideas and methodologies proposed in (8). The authors of (8) considered the following systems: *Autoplex* (2), *Automatch* (3), *COMA* (9), *Cupid* (20), LSD (10), GLUE (12), *SemInt* (18) and *SF* (*Similarity Flooding*) (22); they ran each of these prototypes on the same data sources. For each prototype, Precision, Recall, F-Measure and Overall were computed; these measures were averaged across all input data sources.

We believe that the authors of (8) have provided a meaningful survey which can help us to objectively assess the accuracy of our system with that achieved by the systems mentioned above. To this purpose, we ran our system on the same data sources exploited in (8) and computed the Precision, the Recall, the F-Measure and the Overall achieved by it. Obtained results are reported in Table 4 [4]. Before discussing them, we point out that the accuracy

[4] It is worth pointing out that the values of the accuracy measures of the other systems reported in this table are exactly those specified in (8).

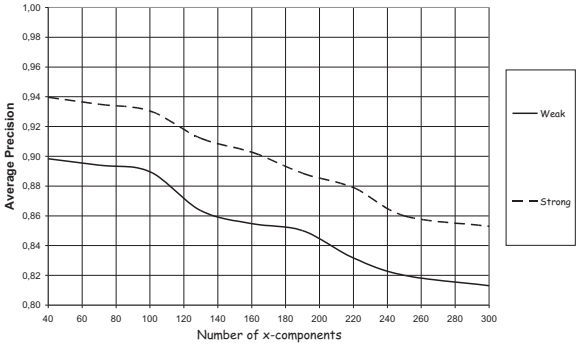

Figure 6: Variation of the Average Precision when the dimension n_t of involved XML Schemas grows

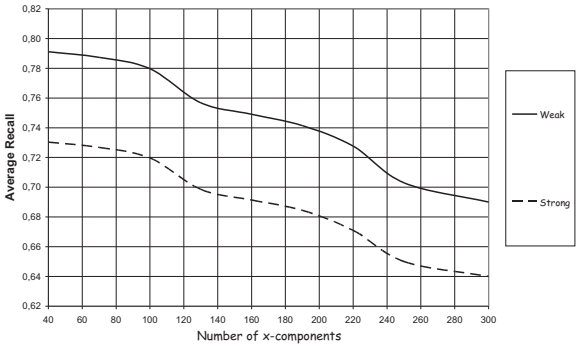

Figure 7: Variation of the Average Recall when the dimension n_t of involved XML Schemas grows

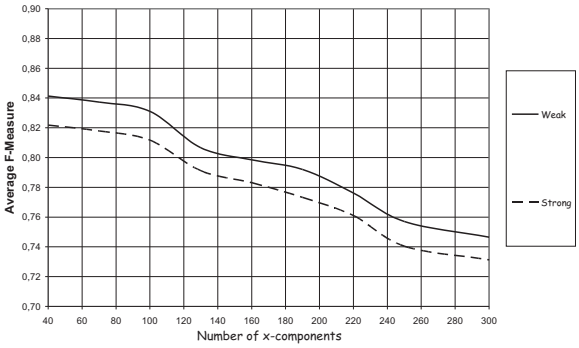

Figure 8: Variation of the Average F-Measure when the dimension n_t of involved XML Schemas grows

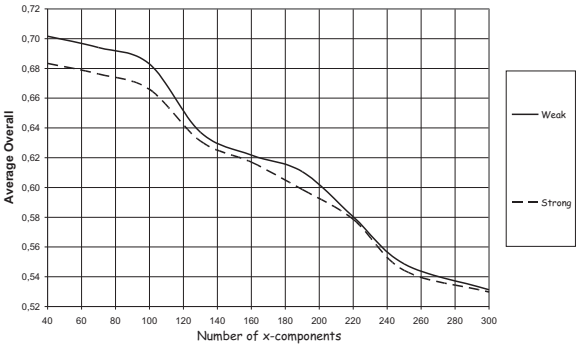

Figure 9: Variation of the Average Overall when the dimension n_t of involved XML Schemas grows

System	*Precision*	*Recall*	*F-Measure*	*Overall*
Our system (*weak*)	0.88	0.78	0.84	0.67
Our system (*strong*)	0.94	0.71	0.81	0.66
Autoplex & Automatch	0.84	0.82	0.82 & 0.72	0.66
COMA	0.93	0.89	0.90	0.82
Cupid	–	–	–	–
LSD	~ 0.80	0.80	~ 0.80	~ 0.60
GLUE	~ 0.80	0.80	~ 0.80	~ 0.60
SemInt	0.78	0.86	0.81	0.48
SF	–	–	–	~ 0.60

Table 4: Comparison of the accuracy of our approach with that of the other approaches evaluated in (8)

measures of the other approaches described in (8) concern both the derivation of sub-schema similarities and the extraction of similarities between single concepts; this last problem is simpler and, generally, the corresponding task shows better accuracy measures, especially for Recall.

From the analysis of Table 4 we can conclude that:

- If strong similarities are computed, our approach achieves the highest Precision and the lowest Recall. This result confirms the main findings emerging from Section 3.4 stating that if strong sub-schema similarities are computed then only few similarities are found but, generally, they are characterized by a high level of reliability. As for weak similarities, our approach achieves the third highest value of Precision and its Recall is high if we consider that we are extracting sub-schema similarities that are complex properties. This result points out the great flexibility of our approach which can be adapted to prioritize Precision over Recall (or vice versa) depending on the exigencies of the application context which it operates in.
- Approaches like *Autoplex*, *Automatch*, LSD, GLUE (12) and *SemInt* (18) use machine learning techniques and, as will be clear in Section 4, analyze data instances along with a wealth of auxiliary information (i.e., data type information or key constraints) to derive semantic matchings. This explains the high values of Precision and Recall achieved by them. The flip side of the coin is that they require a meaningful human effort to provide an initial set of training examples; moreover, if these examples are incomplete and/or incorrect, the accuracy achieved by these approaches may drastically decrease (7; 8; 10; 11).
- Approaches like *SF* and *COMA* may yield highly accurate results; however, their accuracy strongly depends on the feedbacks provided by human operators.

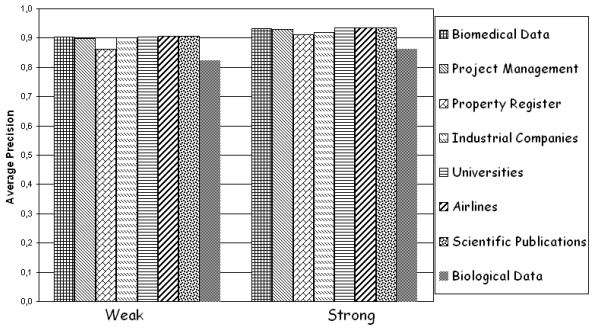

Figure 10: Average Precision of our approach in different application domains

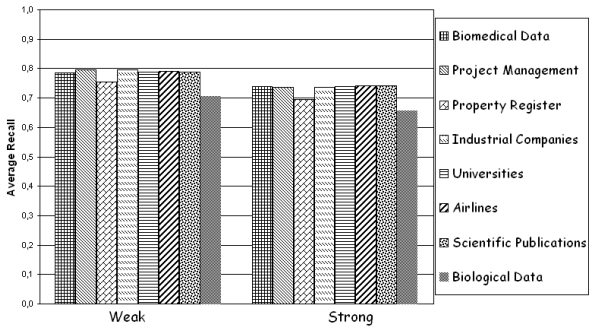

Figure 11: Average Recall of our approach in different application domains

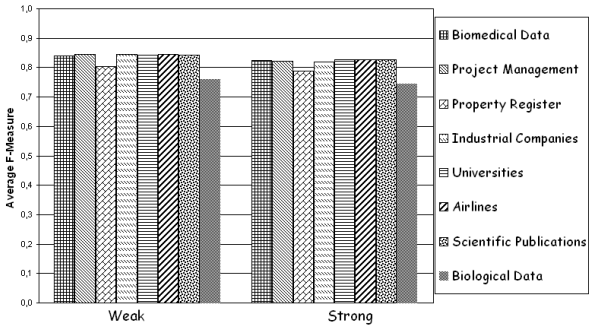

Figure 12: Average F-Measure of our approach in different application domains

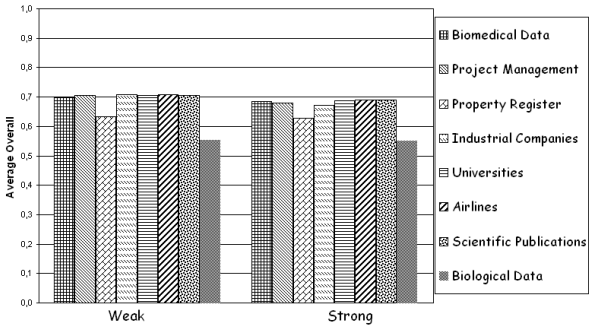

Figure 13: Average Overall of our approach in different application domains

In fact, *SF* iteratively computes semantic matchings; a human user is asked to check the matchings generated at each iteration and to fix a threshold beyond which no further iteration must be performed. Clearly, this threshold influences the number and the quality of discovered matchings.

COMA is a flexible library of matchers; a user exploiting *COMA* can select the matchers best fitting both his needs and the scenario he is operating in and can suitable combine them to improve the quality of obtained results. Clearly, the combination strategy adopted by him influences the overall quality of the results returned by *COMA*.

3.6 Role of our heuristics for the extraction of the most promising pairs of sub-schemas

In order to evaluate the role of our heuristics for the extraction of the most promising pairs of sub-schemas in the improvement of the efficiency of our approach, we have implemented a simple prototype that receives an XML Schema and evaluates the number of possible pairs of (well-formed and self contained) sub-schemas that can be derived from it. The prototype has been exploited for computing the following parameter:

Application context	*Average* $E_{Promising}$
Biomedical Data	2.47×10^{-8}
Project Management	1.61×10^{-10}
Property Register	9.18×10^{-13}
Industrial Companies	2.36×10^{-8}
University	3.88×10^{-5}
Airlines	5.07×10^{-4}
Scientific Publications	3.66×10^{-5}
Biological Data	1.97×10^{-20}

Table 5: Values of $E_{Promising}$ for the various application domains

$$E_{Promising} = \frac{\textit{Number of promising pairs of sub-schemas}}{\textit{Number of possible pairs of sub-schemas}}$$

We have carried out some tests for evaluating this parameter; Table 5 shows the average values we have obtained in the various application domains; these values have been computed by following a procedure analogous to that previously illustrated for accuracy measures.

From the analysis of this table it is possible to conclude that the value of $E_{Promising}$ is extremely low in all application domains; this confirms the importance, for our approach, of the task that singles out the most promising pairs of sub-schemas. The results shown in this table, coupled with the results about the accuracy measures reported previously, allow us to conclude that the extraction of the most promising pairs of sub-schemas plays a fundamental

role for obtaining a scalable approach, applicable on real cases and producing good results. Such an idea is further enforced if we consider that the number of possible sub-schemas in an XML Schema might be exponential against the number of its x-components and, consequently, neither a manual approach nor an automatic one, exhaustively examining all pairs of sub-schemas, might be applied.

3.7 Improvement w.r.t. "naive" approaches

This class of experiments has been performed for verifying the improvements of our approach against manual, "naive" ones; here we use the term "naive" for indicating an approach that is capable of constructing only immediate and quite simple pairs of similar sub-schemas; it is generally based on identifying synonyms and expanding around them for constructing sub-schemas. Usually, a "naive" approach is little time expensive, but it tends to detect only immediate sub-schema similarities, whereas it tends to exclude many complex and potentially significant similarities. In our opinion, a comparison between our approach and the "naive" one is useful to identify the capabilities of our approach of finding complex sub-schema similarities that could be discovered by a human expert only spending a great amount of time in the analysis of the involved Schemas.

In order to carry out such a comparison, we applied our approach to our test schemas by following the guidelines illustrated in the previous experiments. For each considered pair of sub-schemas we asked human experts to determine the number N_{Naive} of sub-schema similarities, identified by our approach, that, in their opinion, had a "naive" structure. After this, we have computed the following parameter:

$$R_{Naive} = \frac{N_{Naive}}{N_{Total}}$$

where N_{Total} indicates the total number of sub-schema similarities derived by our approach. Clearly, the lower R_{Naive} is, the higher the improvement caused by our approach will be.

Table 6 shows the average value of R_{Naive} for weak and strong sub-schema similarities in the various application domains. The average values have been computed by following the same guidelines illustrated in the previous experiments.

From the analysis of this table we can observe that the best values of R_{Naive} can be obtained for domains characterized by large information sources, such as Biological Data and Property Register. This fact is explained by considering that, if involved information sources have a great number of x-components, the structures of the sub-schemas can be more complex and, consequently, N_{Total} can be extremely greater than N_{Naive}.

This result is confirmed by Figure 14 that illustrates the values of the average R_{Naive} for the extraction of weak and strong similarities for different values of the dimension of the input XML Schemas, i.e., of the parameter n_t, measuring the dimension of involved XML Schemas (see Section 3.4).

Application context *Application context*	R_{Naive} *for weak similarities*	R_{Naive} *for strong similarities*
Biomedical Data	0.53	0.62
Project Management	0.36	0.43
Property Register	0.21	0.28
Industrial Companies	0.51	0.60
University	0.61	0.70
Airlines	0.64	0.72
Scientific Publications	0.51	0.63
Biological Data	0.08	0.09

Table 6: Values of R_{Naive} for the extraction of weak and strong sub-schema similarities in the various application domains

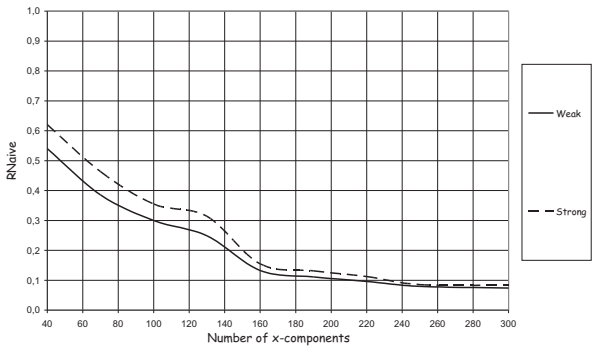

Figure 14: Variation of R_{Naive} for the extraction of weak and strong similarities when the parameter n_t, measuring the dimension of involved XML Schemas, grows

From the analysis of this figure we can observe that R_{Naive} significantly decreases, for both weak and strong sub-schema similarities, when n_t grows. This implies that, for large XML Schemas, the fraction of "naive" sub-schema similarities identified by our approach is actually negligible if compared with the total number of similarities it derives. This further confirms that a "naive" approach, even if it is less time expensive, might exclude many potentially significant similarities.

3.8 Robustness analysis

3.8.1 Robustness against structural dissimilarities

XML is inherently hierarchical; it allows nested, possibly complex, structures to be exploited for representing a domain. As a consequence, two human experts might model the same reality by means of two XML Schemas characterized by deep structural dissimilarities. We have performed a robustness analysis of our approach, devoted to verify if it is resilient to structural dissimilarities.

Before describing our experimental tests about this issue, we point out that the particular features of our approach make it intrinsically robust for a specific case, that is very common in practice. Specifically, if the typology of an x-component x_{1_j} of an XML Schema S_1 is changed from "simple element" to "attribute", or vice versa, no

modifications of the sub-schema similarities involving x-components of S_1 occur. This result directly derives from the definitions of $veryclose$ and *neighborhood* (see Section 2.1).

There are further structural modifications that could influence the results of our approach and for which it is not intrinsically robust; for these cases an experimental measure of its robustness appears necessary. As an example, consider Figure 15 illustrating two portions of XML Schemas representing persons; in the first XML Schema, the concept "Person" is represented by means of a nested hierarchical structure; on the contrary, in the second XML Schema, the same concept is represented by means of a flat structure.

In order to determine the robustness of our approach in the management of these cases, for each pair of XML Schemas into consideration, we have progressively altered the structure of one of them by transforming a certain percentage of its x-components from a nested structure to a flat one. For each of these transformations, we have derived the sub-schema similarities associated with the "modified" versions of the XML Schemas and we have computed the corresponding average values of the accuracy measures. Specifically, for each pair of XML Schemas within each domain, we have considered five cases, corresponding to a percentage of flattened x-components (hereafter FXP - *Flattened X-component Percentage*) equal to: *(a)* 0%; *(b)* 7%; *(c)* 14%; *(d)* 21%; *(e)* 28%. The results we have obtained are shown in Figures 16, 17, 18 and 19.

From the analysis of these figures it is possible to observe that our approach shows a good robustness against increases of FXP. In fact, even if structural dissimilarities occur, the changes in the accuracy measures are generally quite small. However, we stress that if the increases of FXP would be significantly greater than those considered above, the changes in the accuracy measures could be significant; this behaviour is correct since it guarantees that our approach shows the right degree of sensitivity to changes to the structure of involved XML Schemas.

3.8.2 Robustness against errors in IPD

In this experiment we have tested the effects of errors and inaccuracies in the IPD received in input by our approach. We have performed the experiment as follows. First, we have exploited the approaches described in (6) for constructing IPD[5]; then, we have asked experts to validate IPD properties in such a way as to remove any possible error.

After this, we have performed some variations on the correct IPD and, for each of them, we have computed the Average Precision, the Average Recall, the Average F-Measure and the Average Overall; this activity has been performed by following the same guidelines illustrated for the previous experiments. Specifically, we have performed two different typologies of variations on IPD; in a first series of experiments we have discarded a certain percentage of correct properties from the correct IPD, without adding any new wrong property; in a second series of experiments, we have added a certain percentage of wrong properties to the correct IPD, without removing any existing correct property. Variations we have carried out on IPD are: *(a)* 10% of correct properties have been filtered out; *(b)* 20% of correct properties have been filtered out; *(c)* 30% of correct properties have been filtered out; *(d)* 50% of correct properties have been filtered out; *(e)* 10% of wrong properties have been added; *(f)* 20% of wrong properties have been added; *(g)* 30% of wrong properties have been added; *(h)* 50% of wrong properties have been added.

[5]We point out again that any other approach conceived for deriving synonymies, hyponymies and overlappings could be exploited for constructing IPD.

Case	*Average Precision*	*Average Recall*	*Average F-Measure*	*Average Overall*
No errors	0.89	0.77	0.83	0.67
(a)	0.88	0.71	0.79	0.62
(b)	0.88	0.65	0.75	0.56
(c)	0.87	0.60	0.71	0.51
(d)	0.87	0.51	0.64	0.43
(e)	0.82	0.77	0.79	0.59
(f)	0.75	0.76	0.76	0.51
(g)	0.69	0.76	0.72	0.42
(h)	0.58	0.76	0.66	0.22

Table 7: Variation of the Average Precision, the Average Recall, the Average F-Measure and the Average Overall w.r.t. possible errors in IPD for the extraction of weak similarities

Case	*Average Precision*	*Average Recall*	*Average F-Measure*	*Average Overall*
No errors	0.92	0.72	0.81	0.65
(a)	0.91	0.67	0.77	0.61
(b)	0.91	0.62	0.74	0.56
(c)	0.90	0.58	0.71	0.52
(d)	0.90	0.50	0.64	0.44
(e)	0.86	0.72	0.78	0.60
(f)	0.80	0.71	0.75	0.54
(g)	0.75	0.71	0.73	0.47
(h)	0.65	0.71	0.68	0.33

Table 8: Variation of the Average Precision, the Average Recall, the Average F-Measure and the Average Overall w.r.t. possible errors in IPD for the extraction of strong similarities

Tables 7 and 8 present the values of Precision, Recall, F-Measure and Overall we have obtained for the extraction of weak and strong similarities in all these tests. These results show that our system is quite robust w.r.t. errors and inaccuracies in IPD. In fact, its accuracy significantly decreases only for cases *(d)* and *(h)*; i.e., when the correct properties of IPD that are filtered out or the wrong properties of IPD that are added are greater than 30%. This shows, also, that our system presents a good sensitivity in addition to a satisfying robustness.

```
<xs:element name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="address"/>
        </xs:sequence>
        <xs:attribute name="first_name" type="xs:string"/>
        <xs:attribute name="last_name" type="xs:string"/>
        <xs:attribute name="gender" type="xs:string"/>
        <xs:attribute name="birthdate" type="xs:date"/>
    </xs:complexType>
</xs:element>
<xs:element name="address">
    <xs:complexType>
        <xs:attribute name="city" type="xs:string"/>
        <xs:attribute name="state" type="xs:string"/>
        <xs:attribute name="country" type="xs:string"/>
        <xs:attribute name="zip" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
    <xs:complexType>
        <xs:attribute name="first_name" type="xs:string"/>
        <xs:attribute name="last_name" type="xs:string"/>
        <xs:attribute name="gender" type="xs:string"/>
        <xs:attribute name="birthdate" type="xs:date"/>
        <xs:attribute name="city" type="xs:string"/>
        <xs:attribute name="state" type="xs:string"/>
        <xs:attribute name="country" type="xs:string"/>
        <xs:attribute name="zip" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

Figure 15: Example of "nested" and "flat" structures

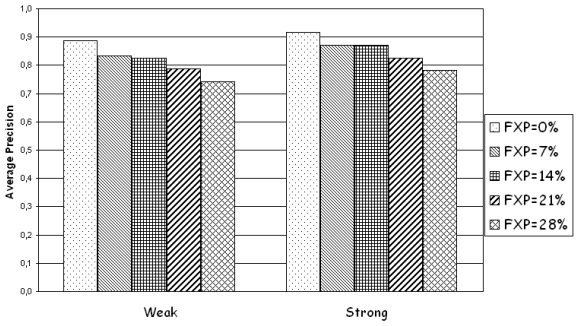

Figure 16: Average Precision of our approach for various values of FXP

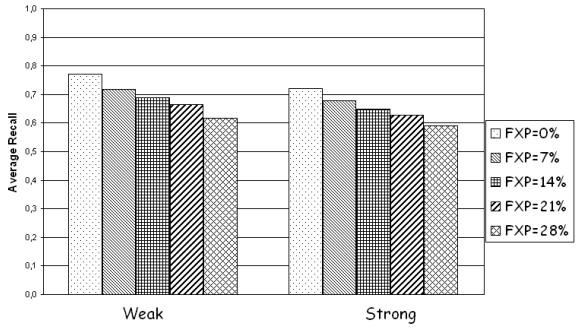

Figure 17: Average Recall of our approach for various values of FXP

3.9 Scalability Issues

3.9.1 Analysis of the cardinality of SPS

One of the most important factors that may influence the scalability of our system is the number of the most promising pairs of sub-schemas, i.e., the cardinality of SPS. In the previous sections we have shown that our heuristics for the construction of SPS allow both a very high accuracy of results to be maintained and the number of pairs of sub-schemas into examination to be significantly reduced. In this section we analyze how this number grows when the number of complex elements of the input Schemas grows. Specifically, Figure 20 plots the increase of the cardinality of SPS against m, i.e., the maximum between the number of complex elements of S_1 and that of S_2.

From the analysis of this figure we can observe that this increase is much lower than that we could expect from the theoretical worst case analysis (see Theorem 2.3). This result is quite interesting because it further confirms that the number of promising sub-schemas generated by our approach is large enough to yield accurate results (see Section 3.4) but small enough to prevent a untenable computational effort.

This result depends on the following factors:

– In order to construct SPS, we need to apply the function Ψ, introduced in Section 2.2, on all pairs of complex elements belonging to the Interschema Property Dictionary (IPD) associated with the input sources. In real cases, a complex element is involved in a very

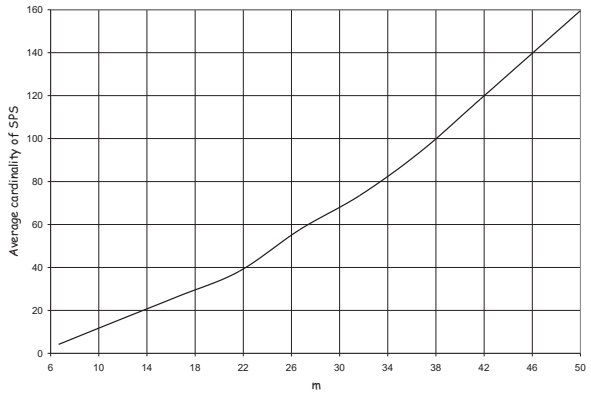

Figure 20: Cardinality of SPS against the number m of complex elements

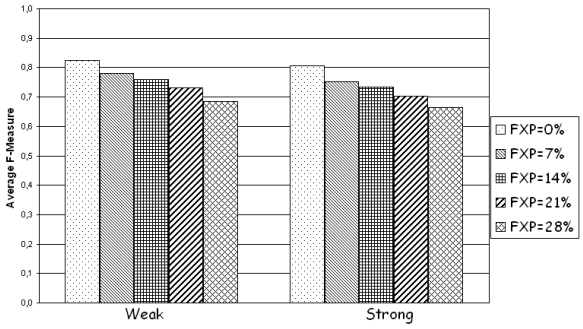

Figure 18: Average F-Measure of our approach for various values of FXP

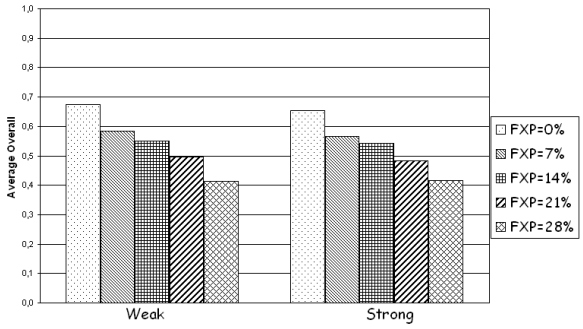

Figure 19: Average Overall of our approach for various values of FXP

low number of interschema properties and, consequently, the cardinality of IPD is much less than the overall number of possible pairs of complex elements existing between S_1 and S_2. As a consequence, the number of times our approach needs to call the function Ψ is significantly lower than that we could expect from the theoretical analysis, and this produces a significant time saving.

– In order to construct a promising sub-schema we need to apply also functions ξ and θ (see Section 2.2). Recall that θ receives a pair $\langle nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma)\rangle$, where x_{1_j} (resp., x_{2_k}) is a complex element of S_1 (resp., S_2) and δ (resp., γ) is an integer ranging from 0 to m. It may be that θ receives a pair of neighborhoods $\langle nbh(x_{1_j}, \delta), nbh(x_{2_k}, \gamma)\rangle$ such that no elements of $nbh(x_{1_j}, \delta)$ share any interschema property with any element of $nbh(x_{2_k}, \gamma)$. In this case $nbh(x_{1_j}, \delta)$ and $nbh(x_{2_k}, \gamma)$ would be completely "pruned" by θ and, then, ξ would not return any promising sub-schema. This contributes to reduce the overall number of promising sub-schemas w.r.t. the theoretical upper bound specified by Theorem 2.3.

3.9.2 Analysis of the average size of a promising sub-schema

A second, important, factor that can influence the scalability of our approach concerns the average cardinality of promising sub-schemas; in fact, in order to derive sub-schema similarities, our approach computes some matchings on suitable bipartite graphs constructed starting from the complex elements of the involved sub-schemas (see Section 2.3). According to the reasoning illustrated in Section 2.2, measuring the average cardinality of a promising sub-schema is equivalent to measure the average cardinality of the set of complex elements generated by applying the function θ.

In Figure 21 we plot the Average Cardinality AC of a promising sub-schema against the number n of x-components of the corresponding XML Schema. From the analysis of this figure we can observe that AC depends on n in a sub-linear fashion. This result is encouraging because it shows that the size of the promising sub-schemas generated by our approach does not "explode" when the number of x-components of the corresponding input Schemas grows; this influences the scalability of our approach positively.

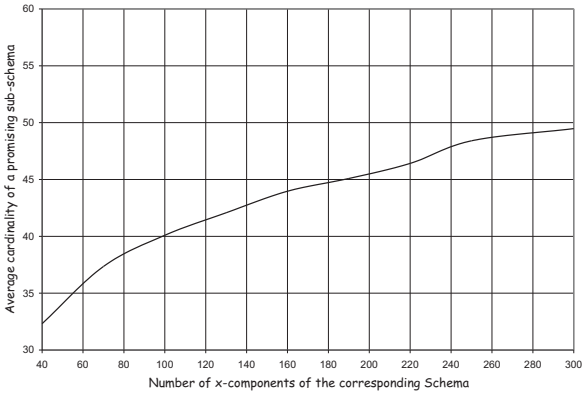

Figure 21: Average Cardinality against the number n of involved x-components

This behaviour can be justified by the following reasoning: if the overall number of x-components of an XML Schema grows then both the number of its complex elements and that of its simple elements and attributes grows. However, in real cases, this growth is not "balanced", in the sense that the number of complex elements does not grow as quick as the number of simple elements and attributes; as a consequence, if n becomes large, we expect that the number of complex elements of the Schema into consideration grows slowly, whereas the number of its simple elements/attributes increases significatively. This implies that the Interschema Property Dictionary IPD associated with input Schemas grows slowly in presence of an increase of n because IPD contains only pairs of complex elements and they must be also semantically related. Now, the number of interschema properties stored in IPD has a substantial impact on the pruning activity performed by θ (see Section 2.2) and, ultimately, on the average cardinality of a promising sub-schema; this impacts on the scalability of

our approach positively.

3.9.3 Analysis of the Response Time

A third important parameter that we have considered in order to evaluate the scalability of our approach is its Response Time. To this purpose we have conducted an experimental study on our test XML Schemas to compute the increase of the Response Time caused by an increase of the sizes of schemas. All these tests have been performed on a machine with a Pentium IV 3 GHz CPU and 1 Gb of RAM.

This experiment was carried out as follows: given two XML Schemas S_1 and S_2 such that $n_1 = |XCompSet(S_1)|$ (resp., $n_2 = |XCompSet(S_2)|$) and m_1 (resp., m_2) is the number of complex elements of S_1 (resp., S_2), we have computed the average values of the Response Time of our approach against the values of $n_t = n_1 + n_2$ and $m_t = m_1 + m_2$. The obtained results are shown in Figures 22 and 23.

From the analysis of Figure 22 we can observe that the increase of the Response Time against m_t is much "softer" than that we could expect from the theoretical, worst case, analysis (see Theorems 2.4 and 2.5). In our opinion this result is even more important if we consider that:

- in real XML Schemas the number of complex elements is generally very low;
- even when the number of complex elements in one or both of the involved XML Schemas is quite high (e.g., $m_t \simeq 50$) the time necessary to our system to determine sub-schema similarities is quite low (e.g., at most some minutes for $m_t \simeq 50$);
- the extraction of sub-schema similarities is generally an activity to be performed offline.

All these considerations are further strengthened by Figure 23 where we analyze the increase of the Response Time of our system against the increase of n_t i.e., against the increase of the total number of x-components belonging to the involved XML Schemas (which is a reliable and precise indicator of the complexity of the involved Schemas).

All the reasonings above allow us to conclude that our approach is scalable and really adequate in those contexts characterized by numerous and large information sources.

Finally, Figures 22 and 23 show also that the Response Time for deriving the strong properties is comparable with that required for extracting the weak ones. This confirms the theoretical results illustrated in Theorems 2.4 and 2.5.

3.9.4 Analysis of the percentage of Response Time spared by a human expert

Finally, an interesting study, someway related to scalability, regards the computation of the average percentage of time spared by the experts by applying our approach and validating its results w.r.t. doing the same task manually.

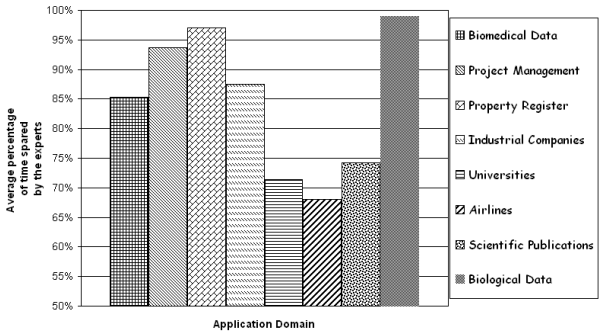

Figure 24: Average percentage of time spared by the experts by applying our approach and validating its results than doing the same task manually

Actually, this percentage is not a direct measure of the scalability of our approach; however, in our opinion, it provides a precise idea of the positive impact of our approach on the daily life of a human expert working in this application context.

The obtained results are shown in Figure 24. From the analysis of this figure we can see that the exploitation of our system really allows experts to save a great amount of time, especially in those domains involving large source schemas, such as the Property Register domain (97% of spared time) and the Biological Data domain (99% of spared time).

4 Related works

4.1 Introduction

The problem of deriving interschema properties is also called *schema matching* or *ontology alignment* in the Information Systems and Artificial Intelligence research communities; the corresponding algorithms are known as matchers (27). The problem of extracting semantic similarities between two single elements of different schemas or ontologies is often referred as *1:1* matching, whereas the problem of deriving similarities between two groups of elements or attributes is also known as *1:n*, *n:1* or, more in general, *m:n* matching.

In the literature various classification criteria have been proposed for comparing schema matching approaches (see, for example, (27)). They allow approaches to be examined from various points of view. In the following we report those criteria appearing particularly interesting in our context and exploit them to compare our approach with the other ones already presented in the past. The most common of these criteria are the following:

- *Schema Types*: Some matching algorithms can operate only on a specific kind of data source (e.g., XML, relational, and so on); these approaches are called *specific* in the following. On the contrary, other approaches are able to manage various kinds of data source; we

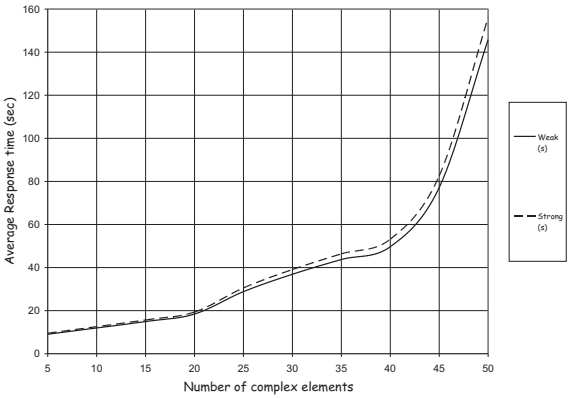

Figure 22: Response Time of our approach against the number m_t of involved complex elements

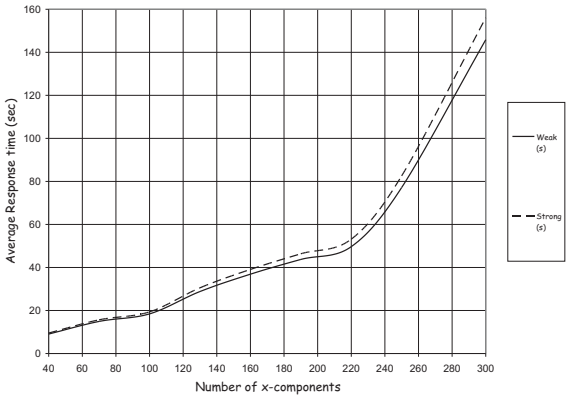

Figure 23: Response Time of our approach against the number n_t of involved x-components

call them *generic* in the following. A generic approach is usually more versatile than a specific one because it can be applied on data sources characterized by heterogeneous representation formats. On the contrary, a specific approach can take advantage of the peculiarities of the corresponding data model.

– *Instance-Based versus Schema-Based*: In order to detect interschema properties, matching approaches can consider data instances (i.e., the so-called *extensional information*) or schema-level information (i.e., the so-called *intensional information*). The former class of approaches is called instance-based; the latter one is known as schema-based. An intermediate category is represented by *mixed approaches*, i.e. those ones exploiting both intensional and extensional information.

  Instance-based approaches are generally very precise because they look at the actual content of the involved sources; however, they are quite expensive since they must examine the extensional component of the involved sources; moreover, the results of an instance-based approach are valid only for the sources it has been applied to. On the contrary, schema-based approaches look at the intensional information only and, consequently, they are less expensive; however, they could be also less precise; the results of a schema-based approach are valid for all sources conforming to the considered schemas.

– *Exploitation of Auxiliary Information*: Some approaches could exploit auxiliary information (e.g., dictionaries, thesauruses, and so on) for their activity; on the contrary, this information is not needed in other approaches. Auxiliary information represents an effective way to enrich the knowledge that an approach can exploit. However, in order to maintain its effectiveness, the time required to compile and/or retrieve it must be negligible w.r.t. the time required by the whole approach to perform its matches. For this reason, pre-built or automatically computed auxiliary information would be preferred to the manually provided one.

– *Individual versus Combinatorial*: An individual matcher exploits just one matching criterion; on the contrary, combinatorial approaches integrate different individual matchers to perform schema matching activities. Combinatorial matchers can be further classified as: *(i) hybrid matchers*, if they directly combine several schema matching approaches into a unique matcher; *(ii) composite matchers*, if they combine the results of several independently executed matchers; they are sometimes called *multi-strategy approaches*. The individual matchers are simpler, and consequently less time-consuming, than the combinatorial ones; however, the results they obtain are often not very accurate.

4.2 Some related approaches

In (23) the authors propose a logic-based matcher called *SKAT* (Semantic Knowledge Articulation Tool). In *SKAT* the user has to initially specify matching and mismatching relationships existing between two ontologies/schemas. After this, the system exploits a set of *first-order logic rules* to refine available relationships and derive new semantic matchings. These matchings can be approved or rejected by the user. Obtained results can be reused in the subsequent schema matching activities.

In (22) the *Similarity Flooding* (SF) algorithm, capable of operating on a wide variety of data sources, is proposed. SF is a graph-based matcher; first it converts input schemas into labeled graphs; then it uses a *fixpoint computation* to determine semantic matchings between the nodes of the graphs; these matchings are refined by means of specific software modules called *filters*. Generated matchings are checked by the human experts at each iteration of the fixpoint computation.

In (20) *Cupid*, a system for deriving interschema properties among heterogeneous information sources, is presented. *Cupid* takes an external thesaurus as input; its approach consists of two phases, named *linguistic* and *structural*. *Cupid* exploits sophisticated techniques, taking into account various characteristics of involved schemas; as a consequence, it is particularly suited when the precision of results is compulsory and the involved schemas are not numerous.

In (7) the authors propose the iMAP prototype. iMAP operates in two phases: the first one exploits Artificial Intelligence techniques (like *Bayesian Network* or *beam search*) to generate a set of rough matchings; the second one uses auxiliary information (like domain integrity constraints, past matchings, etc.) for refining these matchings. Interesting properties of iMAP are its *modularity* and its *extensibility*, since new matching algorithms might be easily embodied in it. It is worth observing that in iMAP the required user effort is (quite) limited; in this aspect it follows the same philosophy of our approach.

In (18) the system *SemInt* is presented. *SemInt* operates on relational schemas. First it associates each attribute with a coefficient (*signature*), computed by taking into account both intensional and extensional information. Then, it exploits the signatures of the attributes of the first schema to train a neural network that is used to cluster similar attributes of the first schema. Finally, it feeds the neural network with the signatures of the attributes of the second schema to find the attributes (resp., the groups of attributes) of the first schema best mapping the attributes (resp., the groups of attributes) of the second schema.

In (9) the authors describe a graph-based system called COMA (COmbining MAtch). It first transforms input schemas into rooted, directed, acyclic graphs; then, it activates different schema matching algorithms on these graphs; finally, it suitably combines the results produced by each algorithm to generate accurate matchings. Interestingly enough, COMA allows a user to specify the matching strategy, i.e., to choose the algorithms for performing the schema matching task.

(32) proposes a schema matching approach operating as follows: first it represents a schema by means of a rooted graph; in this way it can uniformly manage different data source typologies; after this, it combines four different techniques for computing semantic similarities between the elements of the two schemas; this last information is further exploited to derive *m:n* matchings.

In (31) an approach to deriving *1:1* and *1:n* semantic matchings holding among Web query interfaces (i.e., among data sources containing the results of the execution of queries posed through Web interfaces) is proposed. First, it derives *1:1* matchings by means of a hierarchical agglomerative clustering algorithm. After this, it extracts *1:n* matchings by applying a suitable clustering algorithm on derived *1:1* matchings.

In (16) an algorithm exploiting data mining techniques for deriving interschema properties holding among Web query interfaces is presented. This approach first translates involved sources in a suitable format; after this, it derives matchings by analyzing the co-occurrence patterns of attributes belonging to involved sources. Differently from most of the schema matching approaches proposed in the literature, the approach of (16) simultaneously examines all involved schemas.

In (11) the system CGLUE is proposed. It exploits machine learning techniques for deriving both *1:1* and *1:n* semantic matchings between two given ontologies O_1 and O_2. CGLUE receives an initial set of matchings (*training matches*) from the user; then it exploits suitable artificial intelligence techniques (e.g. Bayesian learner) to derive new interschema properties. These techniques are implemented on specific software modules called *learners*. Each learner independently operates on input schemas and generates its set of matchings; the results obtained by each learner are, then, combined to produce the final set of interschema properties.

In (14) the authors propose a schema matching approach particularly suited for Web sources. It first derives *1:1* matchings by solving a matching problem on a suitable weighted bipartite graph; in this task, several parameters (e.g., constraints associated with data types and ranges, linguistic similarities, etc.) are taken into account. After this, it derives *1:n* matchings by applying a polynomial-time heuristic algorithm on previously derived *1:1* matchings.

In (21) the MAFRA (ontology MApping FRAmework) prototype, capable of extracting mappings among distributed ontologies in the Semantic Web, is presented. MAFRA derives both *1:1* and *1:n* matchings as follows. First it represents available ontologies as RDF schemas; then, it adopts a composite approach, taking into account both structural and linguistic matchings, for deriving interschema properties. In order to carry out its activities, MAFRA requires quite a limited human intervention.

In (30) an approach to deriving *m:n* matchings is proposed. It first represents each input schema by means of a graph; after this, it asks the user to provide some basic similarities and dissimilarities. Finally, it derives similarities by taking into account information provided by users, as well as structural and linguistic information; this last is constructed with the support of a suitable thesaurus.

In (24; 25) the system *DIKE* is presented; it is devoted to extract interschema properties from E/R schemas. DIKE has been conceived to operate with quite a small number of information sources; as a consequence, it privileges accuracy to computation time. The extraction task is graph-based and takes into account the "context" of the concepts into examination; it exploits a large variety of thresholds and weights in order to better adapt its behaviour to the sources which it must operate on; these thresholds and weights must be tuned during a training phase.

4.3 Contribution of our approach

We are now able to illustrate the main novelties introduced by our approach w.r.t. the previous ones illustrated above. These novelties can be summarized as follows:

- For each pair of sub-schemas into examination, our approach analyzes both interschema properties and the structural relationships holding among the complex elements stored therein; structural relationships are modelled and handled by means of the *reachable* function.

  In the literature some *schema-based approaches* consider both the similarity of the elements belonging to promising sub-schemas and their structural relationships; however, the notion of similarity considered in these approaches is less rich and expressive than that emerging from the usage of interschema properties. For instance, *Cupid* (20) considers only lexical matchings stored in a thesaurus and the "adjacency" of schema elements (e.g., if an element X is a sub-element of an element Y); *Similarity Flooding* (22) defines an ad-hoc graph matching algorithm which uses a string-matching technique to determine the similarity of two groups of schema elements; MAFRA (21) and the approach of (14) represent input Schemas as graphs and use linguistic and structural constraints to derive *1:1* and *1:m* matchings; finally, the approach of (30) considers structural properties of input schemas (represented as graphs) and uses information extracted from a thesaurus to find sub-schema similarities.

  By contrast, *instance-based approaches*, like iMAP (7), *SemInt* (18), the approach of (31), the approach of (16) and CGLUE (11), perform a detailed analysis of the extensional component of each data source. This analysis is quite complex and refined because it considers not only the similarities existing among single elements but also complex co-occurrence patterns involving concepts belonging to different schemas. This analysis yields accurate and important results because it is often able to derive interesting semantic correspondences which would be usually neglected by a traditional schema-based approach. However, these approaches require a significant *data preparation* phase (as in (16; 31)) and a, often long, *training* phase (as in iMAP (7), *SemInt* (18), and CGLUE (11)).

  Our approach tries to overcome the shortcomings characterizing schema-based and instance-based approaches, while preserving their merits. Specifically, unlike most of schema-based approaches, it considers interschema properties, instead of lexical similarities or string matchings, as the basic properties for the computation of sub-schema similarities. The exploitation of interschema properties allow our approach to achieve a great accuracy since these properties are able to capture the semantic correspondences that would be usually neglected by lexical-based matchings (because two terms might be conceptually equivalent even though they have different names), or to discard semantic matchings that are erroneously recognized by lexical approaches (because two terms might represent different real-world concepts even though they are associated with the same, or at least quite similar, names). In addition, analogously to schema-based approaches, our own is scalable (see Section 3.9); this important feature derives from the fact that it mainly manages schema-based information.

  Unlike instance-based approaches, our approach does not inspect the extensional component of involved data sources and does not need a training phase. Owing to these reasons, it requires a less computational effort and a much more reduced human intervention. In spite of this fact, experimental tests performed in Section 3 show that the accuracy achieved by it is fully satisfactory and comparable with that obtained by instance-based approaches.

- Our approach conceptually separates the derivation of *1:1* matchings and the extraction of *1:m* and *m:n* matchings. In fact, it proposes a technique for the derivation of sub-schema similarities (i.e., *1:n* and *m:n* matchings) which is separate and independent of (although conceptually uniform with) the approach for the extraction of *1:1* matchings described in (6). Our sub-schema similarity derivation approach simply requires an Interschema Property Dictionary as input and does not oblige the user to apply the approach of (6) for constructing it.

  As a consequence, our approach can take advantage of the fact that some *1:1* matching derivation techniques are competitive in some scenarios, whereas other, conceptually different, techniques operate well in other different scenarios. A human expert could select the schema matching technique producing the best results in his scenario and could apply it to derive *1:1* matchings; then, he could use these matchings to derive new sub-schema similarities.

  In the literature, some approaches (e.g., (7; 11; 16; 18; 22; 23; 30; 32)) explicitly designed to derive *m:n* and *1:n* matchings regard *1:1* matchings as special cases of *m:n* matchings. Other approaches (e.g., (9; 14; 20; 21; 24; 25)) propose a two-phase technique: first they derive *1:1* matchings and, then, exploit these matchings, along with other support information derived during the first phase, for extracting *m:n* matchings. As previously pointed out, our approach follows a third philosophy that does not consider *1:1* matchings as special cases of *1:n* and *m:n* matchings and, at the same time, in order to derive *m:n* matchings, it does not need any further information derived during the computation of *1:1* matchings.

– Our approach considers two kinds of sub-schema similaritie, namely, *strong similarities*, computed starting from synonymies, and *weak similarities*, computed by taking also hyponymies and overlappings into account. Strong similarities detected by our system are usually few and characterized by a high level of trustworthiness; on the contrary, weak similarities are able to provide a wide picture of the semantic relationships between two schemas, even if this picture might contain some sub-schema similarities that could be not completely reliable in some application contexts.

  A clear distinction between strong and weak similarities is not present in any of the approaches described in Section 4.2.

  As a consequence of this distinction, our system is characterized by a *great flexibility*; in fact, according to the operating scenario, a human expert could prefer to manage a small set of highly reliable sub-schema similarities or, alternatively, he could want to consider a wide set of sub-schema similarities, some of which could be not precise.

5 Conclusions

In this paper we have presented a semi-automatic approach to deriving sub-schema similarities between XML Schemas; we have shown that our approach is specialized for XML sources, is almost automatic and "light". It consists of two steps: the first one selects a set of promising pairs of sub-schemas, whereas the second one computes sub-schema similarities.

We have pointed out that our approach is part of a more general framework that allows a uniform derivation of similarities and dissimilarities among concepts and groups of concepts represented in semantically heterogeneous XML Schemas. We have also presented the experimental results we have obtained by applying our approach on some, quite variegate, XML Schemas. Finally, we have examined various other related approaches previously proposed in the literature and we have compared them with ours by pointing out their similarities and differences.

At present we are working on the development of an XML Schema integration approach taking sub-schema similarities into account. In the future, we plan to study the possibility to make our sub-schema similarity derivation technique more refined by taking into account the "context" which the sub-schemas into consideration are involved in, in such a way as to define their semantics in a more precise fashion.

In addition, we plan to develop techniques exploiting sub-schema similarities in other application contexts such as those we have mentioned in the Introduction.

Finally, we argue that several other semantic relationships, already studied for single concepts could be extended to sub-schemas. In the future, we plan to verify if this intuition is really feasible and, in the affirmative case, to define suitable approaches.

References

[1] C. Batini and M. Lenzerini. A methodology for data schema integration in the entity relationship model. *IEEE Transactions on Software Engineering*, 10(6):650–664, 1984.

[2] J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proc. of the International Conference on Cooperative Information Systems (CoopIS 2001)*, pages 108–122, Trento, Italy, 2001. Lecture Notes in Computer Science, Springer.

[3] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. of the International Conference on Advanced Information Systems Engineering (CAiSE 2002)*, pages 452–466, Toronto, Canada, 2002. Lecture Notes in Computer Science, Springer.

[4] S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2):277–297, 2001.

[5] C.E.H. Chua, R.H.L. Chiang, and E.P. Lim. Instance-based attribute identification in database integration. *The International Journal on Very Large Databases*, 12(3):228–243, 2003.

[6] P. De Meo, G. Quattrone, G. Terracina, and D. Ursino. An approach for extracting interschema properties from XML schemas at various severity levels. *Informatica*, 31:217–232, 2007.

[7] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. `iMAP`: Discovering complex semantic matches between database schemas. In *Proc. of the ACM International Conference on Management of Data (SIGMOD 2004)*, pages 383–394, Paris, France, 2004. ACM Press.

[8] H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proc. of the International Workshop on Web, Web-Services, and Database Systems*, pages 221–237, Erfurt, Germany, 2002. Lecture Notes in Computer Science, Springer.

[9] H. Do and E. Rahm. COMA- a system for flexible combination of schema matching approaches. In *Proc. of the International Conference on Very Large Databases (VLDB 2002)*, pages 610–621, Hong Kong, China, 2002. VLDB Endowment.

[10] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. of the ACM International Conference on Management of Data (SIGMOD*

2001), pages 509–520, Santa Barbara, California, USA, 2001. ACM Press.

[11] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the Semantic Web. *The International Journal on Very Large Databases*, 12(4):303–319, 2003.

[12] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proc. of the ACM International Conference on World Wide Web (WWW 2002)*, pages 662–673, Honolulu, Hawaii, USA, 2002. ACM Press.

[13] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The International Journal on Very Large Databases*, 14(1):50–67, 2005.

[14] A. Gal, G. Modica, and H.M. Jamil. Improving Web Search with Automatic Ontology Matching. Technical Report TR-IDB-2002-09, Department of Computer Science, Mississippi State University, 2002.

[15] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18:23–38, 1986.

[16] B. He, K. Chen-Chuan Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *Proc. of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2004)*, pages 148–157, Seattle, Washington, United States, 2004. ACM Press.

[17] M.L. Lee, L.H. Yang, W. Hsu, and X. Yang. XClust: clustering XML schemas for effective integration. In *Proc. of the ACM International Conference on Information and Knowledge Management (CIKM 2002)*, pages 292–299, McLean, Virginia, USA, 2002. ACM Press.

[18] W. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[19] J. Madhavan, P.A. Bernstein, A. Doan, and A.Y. Halevy. Corpus-based schema matching. In *Proc. of the IEEE International Conference on Data Enginnering (ICDE 2005)*, pages 57–68, Tokyo, Japan, 2005. IEEE Computer Society Press.

[20] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. of the International Conference on Very Large Data Bases (VLDB 2001)*, pages 49–58, Roma, Italy, 2001. Morgan Kaufmann.

[21] A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA - a MApping FRAmework for distributed ontologies. In *Proc. of the International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002)*, pages 235–250, Siguenza, Spain, 2002. Lecture Notes in Computer Science, Springer.

[22] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. of the IEEE International Conference on Data Engineering (ICDE 2002)*, pages 117–128, San Jose, California, USA, 2002. IEEE Computer Society Press.

[23] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proc. of Fusion'99*, Sunnyvale, California, USA, 1999.

[24] L. Palopoli, D. Saccà, G. Terracina, and D. Ursino. Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):271–294, 2003.

[25] L. Palopoli, G. Terracina, and D. Ursino. Experiences using DIKE, a system for supporting cooperative information system and data warehouse design. *Information Systems*, 28(7):835–865, 2003.

[26] K. Passi, L. Lane, S.K. Madria, B.C. Sakamuri, M.K. Mohania, and S.S. Bhowmick. A model for XML Schema integration. In *Proc. of the International Conference on E-Commerce and Web Technologies (EC-Web 2002)*, pages 193–202, Aix-en-Provence, France, 2002. Lecture Notes in Computer Science, Springer.

[27] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[28] E. van der Vlist. Using W3C XML Schema. `http://www.xml.com/pub/a/2000/11/29/schemas/part1.html`, 2001.

[29] C.J. Van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[30] G. Wang, J.A. Goguen, Y.K. Nam, and K. Lin. Critical points for interactive schema matching. In *Proc. of the Asia-Pacific Web Conference on Advanced Web Technologies and Applications (APWeb 2004)*, pages 654–664, Hangzhou, China, 2004. Lecture Notes in Computer Science, Springer.

[31] W. Wu, C.T. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the Deep Web. In *Proc. of the ACM International Conference on Management of Data (SIGMOD 2004)*, pages 95–106, Paris, France, 2004. ACM Press.

[32] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proc. of IEEE International Conference on Database Systems for Advanced Applications (DASFAA '03)*, pages 39–46, Kyoto, Japan, 2003. IEEE Computer Society.