Information Sciences 180 (2010) 4975–4998

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Element similarity measures in XML schema matching

Alsayed Algergawy [a,*,1], Richi Nayak [b], Gunter Saake [c]

[a] Department of Computer Science, University of Leipzig, Postfach 100920, 04009 Leipzig, Germany
[b] Queensland University of Technology, School of Information Systems, GPO Box 2434, Brisbane, Australia
[c] Faculty of Computer Science, Magdeburg University, 39106 Magdeburg, Germany

ARTICLE INFO

*Article history:*
Received 31 August 2009
Received in revised form 6 August 2010
Accepted 15 August 2010

*Keywords:*
XML
Schema matching
Element similarity measures

ABSTRACT

Schema matching plays a central role in a myriad of XML-based applications. There has been a growing need for developing high-performance matching systems in order to identify and discover semantic correspondences across XML data. XML schema matching methods face several challenges in the form of definition, adoption, utilization, and combination of element similarity measures. In this paper, we classify, review, and experimentally compare major methods of element similarity measures and their combinations. We aim at presenting a unified view which is useful when developing a new element similarity measure, when implementing an XML schema matching component, when using an XML schema matching system, and when comparing XML schema matching systems.



## 1. Introduction

The eXtensible Markup Language (XML) has emerged as a standard for information representation and exchange on the Web as well as on the Intranet due to its self-describing data capability and flexibility in organizing data [1,36]. As a result, data represented in XML are proliferating and efforts to develop high-performance techniques to efficiently manage large XML data have become vital. One of the biggest obstacles to develop such techniques is the identification of semantically similar elements (correspondences) between a pair or a group of XML schemas. The process of identifying semantic correspondences across XML schemas is called *XML schema matching*.

Schema matching, in general, plays a central role in many data-shared applications [53]: in *data integration* to identify and characterize inter-schema relationships between multiple (heterogeneous) schemas [26,22]; in *data warehousing* to map data sources to a warehouse schema [17,10]; in *E-business* to help map messages between different XML formats; in the *Semantic Web* to establish semantic correspondences between concepts of different web site ontologies [38,5,47]; in *data migration* to migrate legacy data from multiple sources into a new one [29]; in *data transformation* to map a source object to a target object, and in *XML data clustering* to determine semantic similarities between XML data [42,50,4].

Due to the complexity inherent in schema matching, it is mostly performed manually by a human expert. However, manual reconciliation tends to be a slow and inefficient process, especially in large-scale and dynamic environments. Therefore, the need for efficiently achieving the matching process has become inevitable. Consequently, a myriad of matching algorithms have been proposed and many systems for automatic schema matching have been developed. Some examples are Cupid [45], Similarity Flooding [48], COMA/COMA++ [24,25], LSD [27], OntoBuilder [33], PORSCHE [55], SMatch [35], Spicy [12,13], and XPrüM [6]. The common trait among these systems is that they all exploit schema element features (properties) as well as relationships between elements utilizing different element similarity measures.

* Corresponding author. Tel.: +49 17663217041.
*E-mail address:* algergawy@informatik.uni-leipzig.de (A. Algergawy).
[1] A. Algergawy partially worked on this paper while at Magdeburg University.

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.08.022

Despite the large number of such systems, a few studies have been conducted to compare and evaluate various element similarity measures independently of their matching systems. Results reported in [23,8] are used to compare the entire matching systems without considering individual element measures. A library of element level semantic matchers implemented within the S-Match system [35] is proposed in [34]. However, the library only considers the element features. A new machine learning-based approach to combine element measures (matchers) into ensembles has been developed [46]. The approach focuses on the ability to choose matchers that participate in an ensemble, without considering the evaluation of individual element measures. Recently, there are few works that survey approaches assessing the similarity between XML documents/schemas [58,37,60,20]. However, these works focus on measuring the similarity between the entire XML data, but not on the individual elements. A comparative study of similarity measures for XML schema elements can support (semi-) automatic and labor-intensive activities, such as XML schema integration [42], XML schema matching [62] and the critical step of matching services in Web service discovery [38].

Another aspect of element similarity measures in the context of schema matching is *similarity measures combination*. Given that a variety of element features/relationships along with a variety of similarity measures are needed to correctly assess the element similarity, combining different similarity measures is far from being trivial [46]. As a result, almost all of existing matching systems have heavily relied on several combining methodologies [45,24,25,33]. To the best of our knowledge, all of these systems focus on using linear combination functions, which cannot sufficiently explore the interdependencies between different similarity measures.

Motivated by these challenges, in this paper, we aim to classify, review, and experimentally compare different element similarity measures in the context of XML schema matching and independently of their matching systems. We categorize element similarity measures guided by the following observation: a number of similarity measures make use of element internal features without considering its surrounds. On the other hand several element similarity measures exploit element relationships making use of element surrounds. We envisage that this study can be helpful when comparing XML schema matching systems, when developing a new element similarity measure, when implementing an XML schema matching component, and when using an XML schema matching system. Another aim of the study is to develop and evaluate rarely used element similarity measures in the schema matching context, such as element annotation and element sibling measures. Finally, the paper presents several methods to combine element similarity measures introducing a so-called nonlinear combination strategy.

To sum up, the main contributions of this study can be stated as follows:

- presenting a taxonomy of XML schema element measures based on the exploited information in the context of schema matching.
- discussing and evaluating rarely used element measures in the schema matching context, such as annotation and sibling measures.
- introducing several element measure combination strategies considering the nonlinear strategy.
- conducting an intensive set of experiments to compare between different element measures and their combinations using two different scenarios. The first scenario targets the matching quality of different element similarity measures utilizing two (small) XML schemas, whereas the second scenario addresses both matching quality and matching efficiency of element similarity measures using real-world data sets.

The remainder of the paper is structured as follows. We first present basic concepts and definitions used throughout the paper. The internal and external element similarity measures are introduced in Sections 3 and 4, respectively. Several strategies to combine element measures are introduced in Section 5. Section 6 presents the experimental evaluation. The concluding remarks and future directions are reported in Section 7.

## 2. Preliminaries

XML is a flexible representation language with self-explanatory tags that allow the storage of information in semi-structured format [1]. There are two kinds of XML data: *XML documents* and *XML schemas*. An XML schema provides the definitions and structure for XML documents. An XML document, on the other hand, contains the tag values represented in a structural format. Several XML schema languages have been proposed to describe the structure and the legal building blocks in XML documents [41]. Among them, XML Document Type Definition (DTD) and XML Schema Definition (XSD) are commonly used.

XML DTD was considered the de facto standard XML schema language until the arrival of XML XSD. Its main building block consists of *element* and *attribute*. It has limited capabilities compared to XSD. XSD aims to be more expressive than DTD and more usable by a wider variety of applications. It has many novel mechanisms, such as simple and complex types, rich datatype sets, occurrence constraints, inheritance, and others. Due to the wider popularity and usage of XSD [41], in this paper, we only consider XSD schemas. Unless clearly specified, we use the term "schema" to present the XML schema.

An XML schema can be modeled as a graph. It can also be represented as a tree by dealing with nesting and repetition problems using a set of predefined transformation rules [42]. Consequently, in the following, we represent XML schemas as rooted, labeled trees, called schema trees, defined as follows:

**Definition 1.** A schema tree ($ST$) is a rooted, labeled tree defined as a 3-tuple $ST = \{N_T, E_T, Lab_N\}$, where:

- $N_T = \{n_{root}, n_2, \ldots, n_n\}$ is a finite set of nodes, each of them is uniquely identified by an object identifier (OID), where $n_{root}$ is the tree root node. There are basically two types of nodes in a schema tree:
  1. *Element nodes*. These correspond to element declarations or complex type definitions.
  2. *Attribute nodes*. These correspond to attribute declarations.
- $E_T = \{(n_i, n_j) | n_i, n_j \in N_T\}$ is a finite set of edges, where $n_i$ is the parent of $n_j$. Each edge represents the relationship between two nodes.
- $Lab_N$ is a finite set of node labels. These labels are strings for describing the properties of the element and attribute nodes, such as *name, data type*, and *cardinality constraint*.

A schema tree, $ST$, is called an ordered labeled tree if a left-to-right order among siblings in $ST$ is given, otherwise it is called unordered schema tree. To compactly represent ordering between schema tree nodes, it is desirable to use a numbering scheme. The work of Dietz [21] is the original work on numbering schemes for trees. It labels each node in a schema tree with a pair of numbers, *(pre, post)*, which corresponds to the preorder and postorder traversal numbers of the node in the tree. In our design, without loss of generality, we choose to use the postorder traversal to uniquely number tree nodes.

**Example 1.** To describe the operations studied in this paper, we use the example presented in [27,6] that has been widely used in the literature. It describes two XML schemas that represent the organization in universities from different countries. Fig. 1(a), (b) show the schema trees of the two XML schemas, wherein each node is associated by its name label, such as "CSDeptUS", its OID such as $n_1$, and its corresponding postorder traversal number.

It should to be noted that XML schema provides mechanisms for uniqueness and reference constraints. For example, it allows users to define ID/IDREF attributes of elements, where an ID attribute uniquely identifies an element, and IDREF attributes refer to other elements that are explicitly identified by their ID attributes. ID/IDREF attributes provide more flexibility of the XML data model and extend the basic tree model to directed graphs. However, tree models are more commonly used by the XML data management community for several reasons. The first is that the general graph-based model significantly increases the complexity of the management of XML data. The second reason is that graph-shaped XML data with ID/IDREF attributes are not so common in practical applications as tree-shaped data [36].

**Definition 2.** Given a schema tree, an *Element* ($\mathcal{E}\ell$) is a singular data item that is the basis of the similarity measures. It may be an element node or an attribute node.

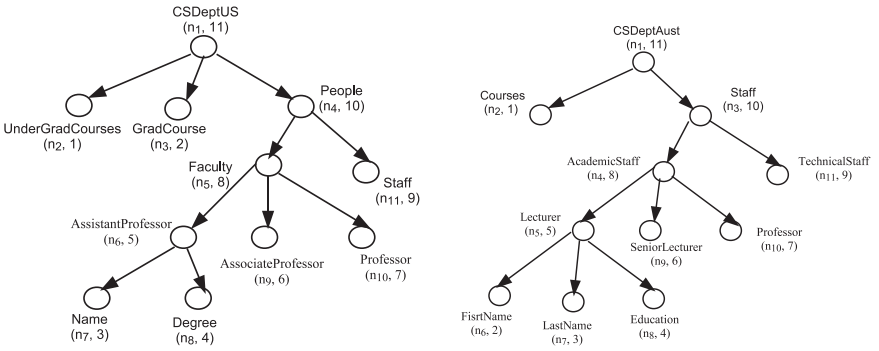
We characterize schema tree elements into:

- *Atomic elements*. These represent simple element or attribute nodes, which have no outgoing edges and represent leaf nodes in the schema tree. This provides the possibility to deal with both simple elements and attributes in the same way; and
- *Complex elements*. These represent complex element nodes, which are the internal nodes in the schema tree.

**Example 2.** As shown in Schema tree $ST1$ of Fig. 1 elements having OIDs $n_2$, $n_3$, $n_7$, $n_8$, $n_9$, $n_{10}$, and $n_{11}$ are atomic elements, while elements having OIDs $n_1$, $n_4$, $n_5$, and $n_6$ are complex elements.

Furthermore, there exist several relationships among schema tree elements that reflect the hierarchical nature of the schema tree. These relationships include:

- *parent–child (induced)* relationship, that is the relationship between each element node and its direct subelement/attribute node;




(a) Schema Tree ST1 (b) Schema Tree ST2

**Fig. 1.** Tree representation of XML schemas.

- *ancestor–descendant (embedded)* relationship, that is the relationship between each element node and its direct or indirect subelement/attribute/text nodes;
- *order* relationship among siblings.

**Definition 3.** The label set associated to each element (node) in a schema tree is called the element features. Each feature has an associated value.

For an XML schema, *names, types*, and *cardinality constraints* are the most commonly used features for XML elements and attributes. Consequently, these features are commonly used for assessing the similarity between schema elements. Fig. 1 shows that each schema tree element has a name feature represented by its value. We represent this using the dot (.) notation, such as $ST1.n_3.name$ = "*GradCourse*" and $ST2.n3.name$ = "*Staff*".

**Definition 4.** A similarity measure, $Sim : ST1 \times ST2 \rightarrow \mathbb{R}$, is a generalized metric that quantifies the similarity between elements $\mathcal{E}\ell_1 \in ST1$ and $\mathcal{E}\ell_2 \in ST2$, such that:

$$\forall \mathcal{E}\ell_1 \in ST1, \quad \mathcal{E}\ell_2 \in ST2, \quad Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) \geqslant 0$$
$$\forall \mathcal{E}\ell \in ST, \quad Sim(\mathcal{E}\ell, \mathcal{E}\ell) = 1$$
$$\forall \mathcal{E}\ell_1 \in ST1, \quad \mathcal{E}\ell_2 \in ST2, \quad Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = Sim(\mathcal{E}\ell_2, \mathcal{E}\ell_1)$$

To accurately assess the similarity between pairs of schema elements, a similarity measure should exploit the features of elements as well as their relationships. It is represented as $Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)$, and its value is computed by the employed method. Usually, the similarity value ranges between 0 and 1, when the measure is normalized. The value of 0 means strong dissimilarity between elements, while the value of 1 means exact same elements. Given two elements, $\mathcal{E}\ell_1 \in ST1$, $\mathcal{E}\ell_2 \in ST2$, the similarity between them can be determined using the following equation:

$$Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = Combine(InterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2), ExterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)) \tag{1}$$

where $InterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)$ represents the internal similarity measure between two elements exploiting their features, $ExterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)$ represents the external similarity measure between them exploiting their hierarchal relationships, and *Combine* is an aggregation function used to combine and quantify the importance of *InterSim* measure and *ExterSim* measure.

In the following, we present different techniques used to determine the internal and external similarity between schema tree elements as well as different combining strategies. At the outset, it should be noted that it is not possible to include all measures of similarity. We are only presenting the common measures. These measures may not necessarily be the best in all situations as many of them are utilized according to the application used.

## 3. Internal element similarity measures

The internal element measures allow the schema elements to exploit their own features, such as their *names*, *data types*, *constraints*, *annotations*, and others in order to be compared with elements from different schema trees. Depending on the type of exploited feature, we present some of the commonly used internal measures.

### 3.1. Name similarity measure

In the absence of data instance, the element name is considered an important source of semantic information for schema matching [45]. Element names can be syntactically similar ("*Staff*", "*TechnicalStaff*") or semantically similar ("*People*", "*Staff*"). As a result, it is desirable to consider both *syntactic* and *semantic* measures to compute the degree of similarity between element names. In order to make element names comparable, they should be normalized into a set of tokens. The normalization process may have the following steps:

- *Tokenization*: The element name is parsed into a set of tokens using delimiters, such as punctuation, uppercase or special symbols, etc. e.g. *UnderGradCourses* → {*Under*, *Grad*, *Courses*}.
- *Expansion*: Abbreviations and acronyms are expanded. e.g. *Grad* → *Graduate*.
- *Elimination*: Tokens that are neither letters nor digits are eliminated and ignored during the comparison process.

After decomposing each element name into a set of tokens, the name similarity between the two sets of name tokens $T_1$ and $T_2$ can be determined as the average best similarity of each token with all tokens in the other set [45,52]. An example measure taken from [45,52] to compute the name similarity is as follows:

$$Nsim(T1, T2) = \frac{\sum_{t_1 \in T_1} [\max_{t_2 \in T_2} sim(t_1, t_2)] + \sum_{t_2 \in T_2} [\max_{t_1 \in T_1} sim(t_2, t_1)]}{|T1| + |T2|} \tag{2}$$

To determine the similarity between a pair of tokens, $sim(t_1, t_2)$, both syntactic and semantic measures can be used.

*3.1.1. Syntactic measures (string-based)*

Syntactic measures take the advantage of the representation of element names as strings (sequence of characters) to assess the similarity between two tokens. There are many methods to compare strings depending on the way the string is coded (as exact sequence of characters, an erroneous sequence of characters, a set of characters, etc.) [49,19,31]. Some of the most commonly used methods are discussed below.

1. *Edit distance*. The edit distance between two strings is the number of operations required to transform one string into the other. There are several algorithms that can be used to define or calculate this metric. In our implementation we make use of the following:
   - *Levenshtein distance*. The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. To compute the similarity degree between two tokens $t_1 \in T_1$ and $t_2 \in T_2$, the following equation is used:

$$sim_{edit}(t_1, t_2) = 1 - \frac{editDistance(t_1, t_2)}{max(|t_1|, |t_2|)} \tag{3}$$

     where $editDistance(t_1, t_2)$ is the minimum number of character insertion, deletion, and substitution operations that are needed to transform $t_1$ to $t_2$. Each edit operation is assigned a unit cost. The similarity value given by the Levenhstein formula is in [0,1], with the zero value denoting a dissimilarity and 1 denoting a total similarity.
   - *Jaro Similarity*. The Jaro measure of similarity between two strings is mainly used in the area of record linkage (duplicate detection). The higher the Jaro measure for two strings is, the more similar the strings are. The Jaro measure is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match. The Jaro similarity measure is calculated as [19]:

$$sim_{jaro}(t_1, t_2) = \frac{1}{3} \times \left( \frac{M}{|t_1|} + \frac{M}{|t_2|} - \frac{M - t}{M} \right) \tag{4}$$

     where $M$ is the number of matching characters and $t$ is the number of transpositions. A variant of this measure by Winkler uses the length $P$ of the longest common prefix of the two string. Let $P' = max(P, 4)$, the Jaro–Winkler measure is defined as [19]

$$sim_{jaro-winkler}(t_1, t_2) = sim_{jaro}(t_1, t_2) + \frac{P'}{10} \times (1 - sim_{jaro}(t_1, t_2)) \tag{5}$$

2. *N-gram distance*. N-grams are typically used in approximate string matching by sliding a window of length $n$ over the characters of a string to create a number of '$n$' length grams for finding a match. The '$n$' length grams are then rated as number of n-gram matches within the second string over the possible n-grams. An n-gram of size 1 is referred to as "uni-gram"; size 2 as "di-gram"; size 3 as "tri-gram"; and size 4 or more is called n-gram. The intuition behind the use of n-grams as a foundation for approximate string processing is that when two strings $t_1$ and $t_2$ are within a small edit distance of each other, they share a large number of n-grams in common. The *n-gram* between the two strings $t_1$ and $t_2$ is defined as

$$sim_{n\text{-}gram}(t_1, t_2) = \frac{2 \times |n\text{-}gram(t_1) \bigcap n\text{-}gram(t_2)|}{|n\text{-}gram(t_1)| + |n\text{-}gram(t_2)|} \tag{6}$$

   where *n-gram (t)* is the set of n-grams in $t$. In our implementation, we make use of the tri-gram measure. Hence, the above equation can be rewritten as follows

$$sim_{tri}(t_1, t_2) = \frac{2 \times |tri(t_1) \bigcap tri(t_2)|}{|tri(t_1)| + |tri(t_2)|} \tag{7}$$

   where $tri(t_1)$ is the set of tri-grams in the string $t_1$.

**Example 3.** Table 1 represents different string-based similarity measures used to compute the degree of similarity between the pair of element names, "UnderGradCourses"[2] and "Courses".

*3.1.2. Semantic measures (language-based)*

Element name semantics are considered as one of the main features of XML schema elements. Therefore, assessing the semantic similarity between element names is crucial in performing schema matching. Semantic measures are needed in two different situations: (i) when measuring the similarity between two syntactically-dissimilar but semantically similar element names, such as "*ST1.AssistantProfessor*" and "*ST2.Lecturer*" and (ii) when measuring the similarity between two syntactically similar but semantically dissimilar element names. To this end, several semantic similarity measures have been developed [44,54,15,32,59].

[2] Only tokenization is considered without expansion.

**Table 1**
Example of using string-based measures.

| $T_1$ | $T_2$ | $sim_{edit}(t_1,t_2)$ | $sim_{Jaro}(t_1,t_2)$ | $sim_{tri}(t_1,t_2)$ |
|---|---|---|---|---|
| Under | Courses | 0.2857 | 0.565 | 0 |
| Grad | Courses | 0.142 | 0.464 | 0 |
| Courses | Courses | 1.0 | 1.0 | 1.0 |
| $Nsim(T_1,T_2)$ | | 0.582 | 0.752 | 0.5 |

In contrast to syntactic measures, semantic measures are based on using Natural Language Processing (NLP) techniques to find the degree of similarity between schema tree element names. Most of these techniques rely heavily on the use of external sources, such as dictionaries and lexicons. In the schema matching context, these techniques can be categorized based on the nature of the exploited dictionary: *domain-specific* or *domain-independent*. Typically, WordNet is either used to simply find close relationships, such as synonyms between element names, or to compute some kind of semantic distance between them. The sMatch system [35] proposes a semantic schema matching exploiting the features in WordNet as a background knowledge source to return semantic relations (e.g. equivalence, more general) between element names rather than similarity values in the [0,1] range. Another possibility is to utilize a domain-specific user-defined dictionary. COMA++ [25] and PORSCHE [55] utilize a user-defined dictionary to get a similarity degree between element names. In our implementation we make use of two commonly used WordNet-based measures to assess the semantic similarity between schema elements [15,59]. It should be noted that other semantic-based similarity measures can be used, for instance measures explained in [39].

1. Wu and Palmer's similarity measure. This measure has been proposed to assess the semantic similarity between two concepts (element names) by identifying their lowest common ancestor [61]. The similarity of two concepts is defined by how closely they are in the taxonomy (hierarchy) and it can be calculated using the following equation:

$$sim_{WuPalmer}(t_1,t_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3} \tag{8}$$

where $N_1$ and $N_2$ are the numbers of IS-A links from $t_1$ and $t_2$ to their most specific common ancestor, $t_0$, and $N_3$ is the number of IS-A links from $t_0$ to the root of the taxonomy.

2. Lin's similarity measure. This measure has also been proposed to estimate the semantic similarity between concepts (element names) by taking into account the measure that should be both universal and theoretically justified [44]. The semantic similarity between two concepts, $t_1$ and $t_2$, in a taxonomy is a corollary of Lin's theorem of similarity

$$sim_{Lin}(t_1,t_2) = \frac{2 \times \log P(t_0)}{\log P(t_1) + \log P(t_2)} \tag{9}$$

where $t_0$ is the most specific common ancestor of the two concepts and $P(t_0)$ denotes the occurrence probability of words corresponding to concept $t_0$. This measure takes into account the information content of the two concepts ($\log P(t_1)$ and $\log P(t_2)$), as well as the information content of their most specific common ancestor ($\log P(t_0)$), in a way to increase with commonality and decrease with difference.

**Example 4.** Fig. 2 is a fragment of the WordNet. The numbers attached to selected concepts represent pre-computed concept frequencies based on a text corpus. The total word occurrences in the considered corpus is 300. The semantic similarities between two concepts "AssistantProfessor" and "lecturer" using the discussed methods are represented in Table 2.

It should be noted that invoking the external sources, such as WordNet, to determine the semantic similarity between element names makes the matching process slow, especially in the large-scale context. An arising conflict is the trade-off between matching quality and matching efficiency, where using the external sources may improve matching quality but decline matching efficiency. Different methods have been proposed to address this conflict. In the experimental evaluation section, we elaborate on this trade-off.

### 3.2. Data type similarity measure

Although the element name is considered a necessary source for determining the element similarity, it is sometimes insufficient. For example, the name similarity between two elements $ST1.n_9$ ("Staff") and $ST2.n_3$ ("Staff") in Fig. 1 equals 1.0. This is a false positive match, as these two elements are of different data types: the former is of an atomic type, and the latter one is of a complex type. This necessitates the need for using other schema information sources to compute the element similarity and to prune some of these false positive matches. The element data type is another schema information source that makes a contribution in determining the element similarity.

It should be noted that an element in a schema tree is of either atomic or complex type. We therefore distinguish the type similarity between them, i.e. the possibility that two complex (or atomic) elements from two different schema trees are sim-

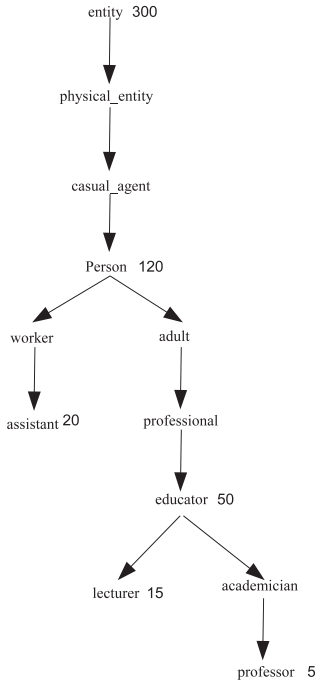



**Fig. 2.** A fragment of WordNet.

**Table 2**
Example of using semantic measures.

| $T_1$ | $T_2$ | $sim_{WuPalmer}(t_1, t_2)$ | $sim_{Lin}(t_1, t_2)$ |
|---|---|---|---|
| Assistant | Lecturer | $\frac{2\times3}{2+4+2\times3} = 0.5$ | $\frac{2\times log(\frac{120}{300})}{\log(\frac{20}{300})+\log(\frac{15}{300})} = 0.321$ |
| Professor | Lecturer | $\frac{2\times6}{2+1+2\times6} = 0.8$ | $\frac{2\times logb(\frac{50}{300})}{\log(\frac{5}{300})+\log(\frac{15}{300})} = 0.505$ |
| $Nsim(T_1, T_2)$ | | 0.7 | 0.443 |

| Type1 | Type2 | Tsim |
|---|---|---|
| string | string | 1.0 |
| string | decimal | 0.2 |
| decimal | float | 0.8 |
| float | float | 1.0 |
| float | integer | 0.8 |
| integer | short | 0.8 |
| ... | ... | ... |

**Fig. 3.** Data type similarity table.

ilar is higher than that one complex (atomic) element and one atomic (complex) are similar. Furthermore, atomic elements in a schema tree have XML built-in data types, such as string, integer and many others. To assess the type similarity between atomic elements, we make use of the XML built-in data type hierarchy.[3] One method is to build a data type similarity table as used in [45,52,51]. Fig. 3 illustrates that elements having the same data types or belonging to the same data type category have the possibility to be similar and their type similarities (*Tsim*) are high. For elements having the same data types, the type similarity is set to 1.0, while the type similarity of elements having different data types but belonging to the same category (such as integer and short) is set to 0.8.

[3] http://www.w3.org/TR/xmlschema-2/.

| | * | + | ? | none |
|---|---|---|---|---|
| * | 1 | 0.9 | 0.7 | 0.7 |
| + | 0.9 | 1 | 0.7 | 0.7 |
| ? | 0.7 | 0.7 | 1 | 0.8 |
| none | 0.7 | 0.7 | 0.8 | 1 |

**Fig. 4.** Cardinality constraint similarity table.

**Example 5.** Fig. 1 depicts that the element $ST_1.n_9$ has a string data type (atomic element) and element $ST_2.n_3$ has a complex type (complex element). The data type similarity between two elements Tsim (string,complex) yields a value of 0 increasing the possibility that the two elements are not similar.

### 3.3. Constraint similarity measure

Other schema information sources of the element that make a contribution in assessing the element similarity are its constraints. The cardinality (occurrence) constraint is considered the most significant. The *minOccurs* and *maxOccurs* in the XML schema define the minimum and maximum occurrence of an element that may appear in XML documents. A cardinality table for DTD constraints has been proposed in [42]. The authors of [52] adapt this table for constraint similarity of XML schemas. Fig. 4 shows the cardinality constraint similarity, where "none" is equivalent to minOccurs = 1 and maxOccurs = 1, "?" is equivalent to minOccurs = 0 and maxOccurs = 1, "*" is equivalent to minOccurs = 0 and maxOccur = unbounded, and "+" is equivalent to minOccurs = 1 and maxOccurs = unbounded.

The cardinality constraint similarity table illustrated in Fig. 4 only represents specific values of minOccurs and maxOccurs. To consider other combination values of them, we utilize the formula proposed in [59]. The following example gives more explanation.

**Example 6.** Given two elements $\mathcal{E}\ell_1 \in ST_1$ and $\mathcal{E}\ell_2 \in ST_2$ having cardinality constraints minOccurs = 0, maxOccurs = 3 and minOccurs = 1, maxOccurs = 10, respectively. The combination of the cardinality constraint values does not appear in the cardinality similarity table. In this case, the formula presented in [59] is used as follows: $CSim(minOccurs = 0, minOccurs = 1) = 1 - \frac{|0-1|}{|0|+|1|} = 0$ and $CSim(manOccurs = 3, minOccurs = 10) = 1 - \frac{|3-10|}{|3|+|10|} = 0.46$. To get the cardinality similarity between the two elements, an aggregation function is then used to combine both computed values.

### 3.4. Annotation similarity measure

The XML element *annotation* provides human- and machine-targeted annotations of schema elements. The annotation element is a part of schema components and appears at the beginning of schema constructors. The element is a container for *documentation* and *appInfo* elements that contain additional information and are dedicated to holding human readable documentation and machine readable information, respectively.[4]

To enhance the internal element similarity, we can capture the document information about schema elements existing in the annotation element. To this end, we make use of a token-based similarity measure. According to the comparison made in [19], the *TF-IDF* (Term Frequency–Inverse Document Frequency) ranking performed best among several token-based similarity measures. Therefore, we consider the *TF–IDF* measure in our study.

*TF–IDF* is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. It is used here to assess the relevance between element annotations. In this case, the content of the annotation element can be considered as multisets (or bags) of words. Given the annotations of two schema elements represented as texts $\mathcal{T}_1$ and $\mathcal{T}_2$, the *TF-IDF* measure between them is given by [19]:

$$TF - IDF(\mathcal{T}_1, \mathcal{T}_2) = \sum_{w \in \mathcal{T}_1 \cap \mathcal{T}_2} V(w, \mathcal{T}_1) \cdot V(w, \mathcal{T}_2) \tag{10}$$

where,

$$V(w, \mathcal{T}_1) = \frac{V\prime(w, \mathcal{T}_1)}{\sqrt{\sum_w V'(w, \mathcal{T}_1)^2}}$$

and

$$V'(w, \mathcal{T}_1) = \log(TF_{w,\mathcal{T}_1} + 1).\log(IDF_w)$$

where, $TF_{w,\mathcal{T}_1}$ is the frequency of the word $w$ in $\mathcal{T}_1$, and $IDF_w$ is the inverse of the fraction of names in the corpus that contain the word $w$.

[4] http://www.w3.org/TR/xmlschema-1/.

*Putting all together.* In the schema matching context, a few matching systems use only one of the internal measures to produce an initial matching result, such as the Similarity Flooding system uses a simple string matcher to compare common prefixes and suffixes of literals [48]. However, the majority of these systems makes use of multiple string-based methods. For example, COMA++ [25] and BtreeMatch [30] utilize Edit distance and n-gram as name similarity measures. A few matching systems exploit both syntactic-based and semantic-based name measures, such as XS3 [59]. Moreover, some of these systems make use of element data type and element cardinality constraints, such as COMA++ and XS3. In this situation, the arising question is how to combine these different measures [28,43]. The internal similarity measure between two elements can be defined as follows.

**Definition 5.** The internal similarity of two elements $\mathcal{E}\ell_1 \in ST_1$ and $\mathcal{E}\ell_2 \in ST_2$ is the combination of name similarity (*Nsim*), data type similarity (*Tsim*), constraint similarity (*Csim*), and annotation similarity (*Asim*):

$$\begin{aligned} InterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = Combine_I(&Nsim(\mathcal{E}\ell_1.name, \mathcal{E}\ell_2.name) \\ &Tsim(\mathcal{E}\ell_1.type, \mathcal{E}\ell_2.type) \\ &Csim(\mathcal{E}\ell_1.card, \mathcal{E}\ell_2.card) \\ &Asim(\mathcal{E}\ell_1.annotation, \mathcal{E}\ell_2.annotation)) \end{aligned}$$

where $Combine_I$ is a function that combines the various internal similarity measures.

## 4. External element similarity measures

In contrast to internal element measures that exploit the element features without considering the position (context) of the element in the schema tree, the external element similarity measures consider the impact of other surrounding elements to the element while comparing it with elements of another schema. To consider the context of the element, the external measures make use of the element relationships instead of its features.

### 4.1. Element context

The external element measures rely heavily on the element context, as shown in Fig. 5, which is reflected by its descendants, ancestors, and siblings. The descendants of an element include both its immediate children and the leaves of the subtrees rooted at the element. The immediate children reflect its basic structure, while the leaves reflect the element's content. In general, the element context comprises child, leaf, ancestor, and sibling contexts. In a schema matching system, one or more of these contexts can be exploited. We hence utilize these four element contexts in our implementation.

- *The child context* of an element is defined as the set of its immediate children nodes including attributes and subelements. The child context of an atomic element (leaf node) is an empty set.
- *The leaf context* of an element is defined as the set of leaf nodes of subtrees rooted at the element. The leaf context of an atomic element (leaf node) is an empty set.

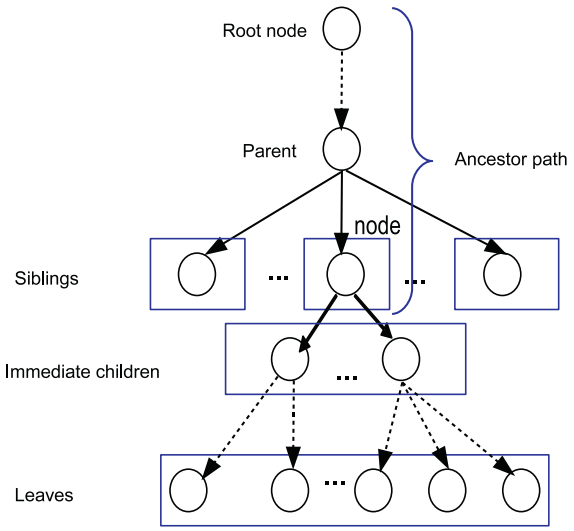



**Fig. 5.** The context of an XML tree element.

- *The ancestor context* of an element is defined as the path extending from the root element to the element. The ancestor context of the root element is a NULL path.
- *The sibling context* of an element contains both the preceding siblings and the following siblings of the element.

Given a schema tree, to extract the context of an element, $\mathcal{E}\ell$, we respectively define the following four functions:

- $child(\mathcal{E}\ell) = \{\mathcal{E}\ell_i | (\mathcal{E}\ell, \mathcal{E}\ell_i) \in E_T\}$; is a function that returns the child context set of the element.
- $leaf(\mathcal{E}\ell) = \{\mathcal{E}\ell_i | \mathcal{E}\ell_i \text{ is an atomic element } \wedge \text{ there is a path between } \mathcal{E}\ell_i \text{ and } \mathcal{E}\ell\}$; is a function that determines the leaf context set of the element.
- $ancestor(\mathcal{E}\ell) = \mathcal{E}\ell_{root}/\ldots/\mathcal{E}\ell$; is a function that identifies the ancestor path for the element. Since the XML schema is represented as a tree, this function returns only one path extended from the root of the tree to the element.
- $sibling(\mathcal{E}\ell) = \{\mathcal{E}\ell_i | \exists \mathcal{E}\ell_j \; s.t. (\mathcal{E}\ell_j, \mathcal{E}\ell) \in E_T \wedge (\mathcal{E}\ell_j, \mathcal{E}\ell_i) \in E_T\}$; is a function that returns the sibling context set of the element.

**Example 7.** Consider schema trees shown in Fig. 1, child ($ST1.n_1$) = $\{n_2, n_3, n_4\}$, leaf ($ST2.n_4$) = $\{n_6, n_7, n_8, n_9, n_{10}\}$, ancestor ($ST1.n_9$) = $n_1/n_4/n_5/n_9$, and sibling ($ST2.n_7$) = $\{n_6, n_8\}$

### 4.2. Element context measure

The context of an element is the combination of its *child*, *leaf*, *ancestor*, and *sibling* contexts. Two elements are structurally similar if they have similar contexts. The structural node context defined above relies on the notions of *path* and *set*. In order to compare two ancestor contexts, we essentially compare their corresponding paths. On the other hand, in order to compare two child, leaf, and/or sibling contexts, we need to compare their corresponding sets.

Although path comparison has been widely used in XML query answering frameworks, it relies on strong matching following the two crisp constraints: node constraint and edge constraint [36]. Under such constraints paths that are semantically similar may be considered as unmatched, or paths that are not semantically similar may be considered as matched. Hence, these constraints should be relaxed. Several relaxation approaches have been proposed to approximate answering of queries [7]. Examples of such relaxations allow matching paths even if nodes are not embedded in the same manner or in the same order and allowing two elements within each path to be matched even if they are not identical but their linguistic similarity exceeds a fixed threshold [14].

To determine the context (structural) similarity between two elements $\mathcal{E}\ell_1 \in ST_1$ and $\mathcal{E}\ell_2 \in ST_2$, the similarity of their child, leaf, sibling, and ancestor contexts should be computed. We present a few similarity measures that are commonly used in measuring the context similarity between elements.

1. *Child context similarity.* The child context set is first extracted for each element, say $C\_set_1 = child(\mathcal{E}\ell_1) = \{\mathcal{E}\ell_{11}, \mathcal{E}\ell_{12}, \ldots, \mathcal{E}\ell_{1k}\}$ and $C\_set_2 = child(\mathcal{E}\ell_2) = \{\mathcal{E}\ell_{21}, \mathcal{E}\ell_{22}, \ldots, \mathcal{E}\ell_{2k'}\}$. The internal similarity between each pair of children in the two sets is determined, the matching pairs with maximum similarity values are selected, and finally the average of best similarity values is computed [6]. The child context similarity, *ChSim*, can be computed using

$$ChSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = \frac{\sum_{i=1}^{i=k} \left[ \max_{j=i}^{j=k'} InterSim(\mathcal{E}\ell_{1i}, \mathcal{E}\ell_{2j}) \right]}{max(|k|, |k'|)} \tag{11}$$

   where $InterSim(\mathcal{E}\ell_{1i}, \mathcal{E}\ell_{2j})$ is the internal similarity computed using Definition 5. Alternatively, having obtained the child context set for each element, a suitable set comparison function such as Cosine or Euclidean Distance can be applied to find similarity between two sets.
2. *Leaf context similarity.* First, the leaf context set is extracted for each element, say $L\_set_1 = leaf(\mathcal{E}\ell_1) = \{\mathcal{E}\ell_{11}, \mathcal{E}\ell_{12}, \ldots, \mathcal{E}\ell_{1k}\}$ and $L\_set_2 = leaf(\mathcal{E}\ell_2) = \{\mathcal{E}\ell_{21}, \mathcal{E}\ell_{22}, \ldots, \mathcal{E}\ell_{2k'}\}$. Then, a suitable set comparison function can be applied. The authors in [6] convert the leaf context sets into numerical vectors and then make use of the cosine measure (CM) to determine the similarity between vectors. CM is in the interval [0,1]. A result close to 1 indicates that the vectors tend in the same direction, and a value close to 0 denotes a total dissimilarity between the two vectors. The leaf context similarity value can be obtained similarly using the formula given in Eq. (11) with the leaf context sets.

**Example 8.** Considering the two schema trees shown in Example 1, the leaf context set of $ST1.n_1$ is $L\_set1 = leaf(n_1(11)) = \{n_2(1), n_3(2), n_7(3), n_8(4), n_9(6), n_{10}(7), n_{11}(9)\}$ and the leaf context set of $ST2.n_1$ is $L\_set2 = leaf(n_1(11)) = \{n_2(1), n_6(2), n_7(3), n_8(4), n_9(6), n_{10}(7), n_{11}(9)\}$. The corresponding numerical vector of $ST1.n_1$ is $v_1 = \{10,9,8,7,5,4,2\}$ (the differences between the postorder number of the element and the postorder numbers of its leaf set elements) and the corresponding numerical vector of $ST2.n_1$ is $v_2 = \{10,9,8,7,5,4,2\}$. The cosine measure CM of the two vectors gives CM $(v_1, v_2) = 1.0$. The leaf context similarity between nodes $ST1.n_1$ and $ST2.n_1$ can be considered 1.0.

3. *Sibling context similarity.* To compute the sibling context similarity between two elements, we compare their sibling context sets. For this, first, the sibling context set is extracted for each element, say $S\_set_1 = sibling(\mathcal{E}\ell_1) = \{\mathcal{E}\ell_{11}, \mathcal{E}\ell_{12}, \ldots, \mathcal{E}\ell_{1k}\}$ and $S\_set_2 = sibling(\mathcal{E}\ell_2) = \{\mathcal{E}\ell_{21}, \mathcal{E}\ell_{22}, \ldots, \mathcal{E}\ell_{2k'}\}$. The internal similarity between each pair of siblings in the two sets is then determined, the matching pairs with maximum similarity values are selected, and finally the average of best similarity values is computed. The sibling context similarity, *SibSim*, can be computed either using Eq. (12) or using a suitable set comparison function.

$$SibSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = \frac{\sum_{i=1}^{i=k}\left[\max_{j=i}^{j=k'} InterSim(\mathcal{E}\ell_{1i}, \mathcal{E}\ell_{2j})\right]}{max(|k|, |k'|)} \tag{12}$$

where $InterSim(\mathcal{E}\ell_{1i}, \mathcal{E}\ell_{2j})$ is the internal similarity computed using Definition 5.

4. *Ancestor context similarity.* The ancestor context similarity captures the similarity between two elements based on their ancestor contexts. As mentioned before, the ancestor context for a given element $\mathcal{E}\ell_i$ is the path extending from the root node to $\mathcal{E}\ell_i$. To compare between paths, authors in [16] established four scores, which then have been used in [14,6]. Other different methods have been proposed to measure the similarity between paths [52,18]. The proposed method in [18] depends on the use of the edit distance algorithm by representing element names in each path as a sequence of strings. Hence, the path similarity, PathSim, can be computed as follows:

$$PSim(P_1, P_2) = 1 - \frac{editDistance(P_1, P_2)}{max(|P_1|, |P_2|)} \tag{13}$$

*Putting all together.* To get precise structural similarities between elements, it is desirable to consider all element contexts. In this situation, the arising conflict is how to combine these context measures. The external element similarity measure can be defined as follows:

**Definition 6.** The external similarity of two elements $\mathcal{E}\ell_1 \in ST_1$ and $\mathcal{E}\ell_2 \in ST_2$ is the combination of child context similarity (ChSim), leaf context similarity (LeafSim), sibling context similarity (SibSim), and path similarity (PSim):

$$\begin{aligned} ExterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = Combine_E(&ChSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2),\\ &LeafSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2),\\ &SibSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2),\\ &PSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)) \end{aligned}$$

where $Combine_E$ is a function used to combine the various external similarity measures.

## 5. Combining similarity measures

It has been shown that each individual element similarity measure exploits a specific feature of the schema element, e.g. the name measure only makes use of the element name. Assessing the similarity between schema elements using an individual measure is not sufficient. Therefore, there is a need to consider a variety of element features along with a variety of measures to assess the similarity. This multi-measure nature is potent in that it makes a matching system highly flexible and adaptable to a particular application domain. However, it results in considerable challenges on how to combine these measures. Without a proper means of combining, the element similarity measures fail to produce correct correspondences across schema elements. In this section, we first present various scenarios in which combining similarity measures is needed. We then discuss the different strategies to combine element similarity measures introducing the nonlinear strategy.

Consider two schema trees *ST*1 and *ST*2 having $n$ and $m$ elements, respectively. To get a total similarity value between pairs of schema elements, we should combine their individual similarity values resulted from different similarity measures. Following are the three scenarios that explain the need of combining various similarity values as shown in Fig. 6:

1. *Within the individual measures.* The name measure encompasses two measures: syntactic-based and semantic-based. Furthermore, the syntactic measure includes Edit distance and n-gram for determining the degree of syntactic similarity. To get the name similarity value, these individual measures should be combined. This combining scenario is also needed in the constraint measure when the values of minOccurs and maxOccurs do not subject to conditions illustrated in the table of Fig. 4.
2. *Within either internal or external measure.* The internal measure comprises four element similarity measures (name, data type, cardinality, and annotation). Consequently, to obtain a total similarity value of the internal measure, the similarity values produced by these element measures should be combined. The function $Combine_I$ has the input of $k_1 \times n \times m$ similarity matrices to produce an $n \times m$ internal similarity matrix, as shown in Fig. 6, where $k_1$ is the number of internal similarity measures. Similarly, the external measure includes four individual element measures. The function $Combine_E$ is used to combine $k_2 \times n \times m$ similarity matrices to produce an $n \times m$ external similarity matrix, as shown in Fig. 6, where $k_2$ is the number of external similarity measures.

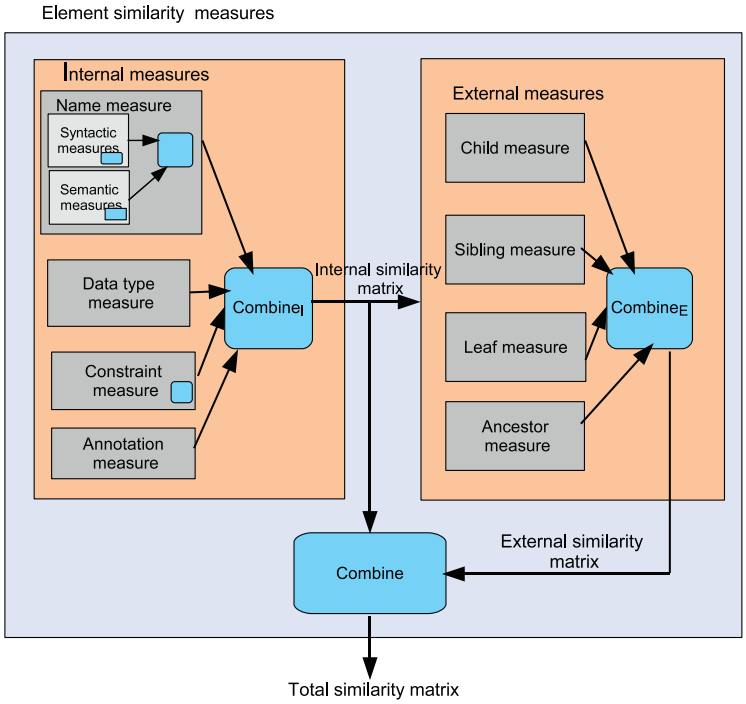



**Fig. 6.** Similarity combining scenarios.

3. *Within the element measure*. The overall similarity between a pair of elements is calculated by internal and external similarity measures. The internal and external similarity matrices are then combined using the *Combine* function to output the $n \times m$ total similarity matrix.

Due to the need of considering a wide variety of combining techniques, similarity combining is far from being trivial. Furthermore, the number of element similarity measures is continuously growing, and this diversity by itself complicates the choice of the most appropriate combining strategy for a given application domain. Commonly, similarity combining can be done using an aggregation function that can be defined as follows [11]:

**Definition 7.** An aggregation function $F$ is a function of $\mathcal{N} > 1$ arguments that maps the (n-dimensional) unit cube onto the unit interval $F : [0,1]^{\mathcal{N}} \to [0,1]$, with the following properties

1. $F(\underbrace{0,0,\ldots,0}_{\mathcal{N}-times}) = 0$ and $F(\underbrace{1,1,\ldots,1}_{\mathcal{N}-times}) = 1$.
2. $x \preceq y$ implies $F(x) \leqslant F(y)$ for all $x, y \in [0,1]^{\mathcal{N}}$

The input value of 0 is interpreted as no membership, or no evidence, and naturally, an aggregation of $\mathcal{N}$ 0s yields 0. Similarly, the value of 1 is interpreted as full membership, and an aggregation of $\mathcal{N}$ 1s yields 1. This is the first fundamental property of aggregation functions, *the preservation of bounds*. The second property is *the monotonicity condition*. The condition can be stated that for every $x_1, x_2, \ldots, x_{\mathcal{N}} \in [0,1]$ and $y_1, y_2, \ldots, y_{\mathcal{N}} \in [0,1]$ such that $x_i \leqslant y_i$, $F(x_1, x_2, \ldots, x_{\mathcal{N}}) \leqslant F(y_1, y_2, \ldots, y_{\mathcal{N}})$.

Many aggregation functions have been proposed [11]. They can be grouped in various families, such as means, triangular norms and conorms, and many others. To choose the most suitable aggregation function, it should be consistent with the semantics of the aggregation procedure and fit with the desired application. In the context of schema matching, the existing aggregation methods focus on linear combination functions, which cannot sufficiently explore the interdependencies between different element similarities [25,28,46]. The linear combining strategy ignores the fact that the relation between individual element similarity measures and may not be necessarily linear. Another case in which the linear combining strategy is not adequate is that some individual similarity measures are conditionally dependent on other individual measures. These conditional dependencies must be captured because they represent significant semantic interrelationships. However, linear combinations cannot capture these relationships. Moreover, a single (fixed) aggregation function is always applied to combine similarities from different schemas without considering the special features of each schema pair.

In the following, we present various combining strategies of element similarity measures.

- *Linear fixed methods*; These methods only consider the effect of individual similarities without considering the interdependencies between them. One of the following methods can be used [24,25]:
  - *Max*. This method selects the largest similarity value of any measure.
  - *Min*. This method selects the smallest similarity value of any measure.
  - *Weighted sum*. This method determines a weighted sum of similarity values of the individual measures and needs relative weights which should correspond to the expected importance of the measures. The weighted-sum function can be defined as follows:

**Definition 8.** Given a weighting vector $w$, such that $\sum_{i=1}^{\mathcal{N}} w_i = 1$, the weighted-sum function is the function $F_w(X) = w_1x_1 + w_2x_2 \ldots w_{\mathcal{N}}x_{\mathcal{N}} = \sum_{i=1}^{\mathcal{N}} w_ix_i$

  - *Average*. This method represents a special case of Weighted sum and returns the average similarity over all measures, i.e. considers them equally important.

It has been shown that the linear fixed methods make use of constant weight values, which need to be specified manually in most cases. To make weight values adaptable, the linear adaptive methods can be used.

- *Linear adaptive methods*; These methods do not consider the interdependencies between similarity measures. However, they deem the adaptive behavior of weights. The values of weights should be determined through a learning process using a learning technique. The similarity combining function, $F_w(X) = \sum_{i=1}^{\mathcal{N}} w_ix_i$, is considered as an optimization problem, where the weights should be chosen to maximize $F_w(X)$. The Hopfield Neural Network [40] or the machine-learning techniques [26,46] can solve these optimization problems.
  It is noted that both linear fixed and adaptive methods rely on using linear combining functions, which cannot sufficiently explore the interdependencies between element measures. Furthermore, the linear combining strategy is not adequate to solve the problems mentioned above. To address (some of) these problems, we propose using the nonlinear methods.
- *Nonlinear methods*; These methods consider the interdependencies between element measures by using nonlinear techniques, such as nonlinear networks. Since the similarity values are ranging between 0 and 1, the similarity combining function should be restricted to the second order. In this direction, the similarity combining function can be defined as follows:

$$F_w(X) = \lambda \sum_{i=1}^{\mathcal{N}} w_ix_i \pm (1-\lambda) \sum_{j=1}^{\mathcal{N}} \sum_{k=j}^{\mathcal{N}} x_jx_k \tag{14}$$

  The first term in the equation presents similarity combining without considering the interdependencies of similarity measures, while the second part presents these interdependencies. The equation also shows that the two values are either added or subtracted depending on the linear similarity value. The intuition behind this is that elements are likely to be similar for the higher (linear) similarity between elements and the two parts should be added. In contrast, in the case of low similarity, elements are not likely to be similar and the two parts should be subtracted. Finally, the constant $\lambda$ is used to ensure the total similarity value in the normalized range, i.e. $[0,1]$.

In the following section, we make use of the first and the third methods and experimentally compare them, while the second method is outside the scope of the paper.

## 6. Experimental evaluation

In this section, we introduce the experimental evaluation to validate the performance of selected element similarity measures in the context of schema matching. We first present the evaluation measures and criteria used during the evaluation process. We then show several evaluation scenarios reporting on the experimental results.

### 6.1. Evaluation measures

In the context of schema matching, there are two performance aspects that should be considered during the matching process: *matching effectiveness (quality)* and *matching efficiency*.

*Effectiveness measures.* Ideally, the match task should be manually solved to get the real correspondences (matches) $R_m$. Then, the matching system solves the same problem to obtain automatic correspondences (matches) $A_m$. After identifying both real and automatic matches, it is possible to define some terms that will be used in computing match effectiveness, as shown in Fig. 7. *False negatives*, $A = R_m - A_m$, are the needed matches but not identified by the system. *True positives*, $B = R_m \cap A_m$, are the correct matches and correctly identified by the system. *False positives*, $C = A_m - R_m$,

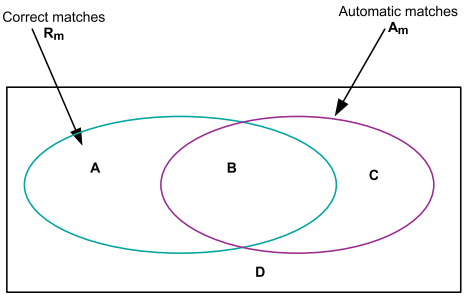



**Fig. 7.** Complete set of correspondences.

are the false matches but identified by the system. *True negatives*, D, are the false matches and correctly discarded by the system. To measure the effectiveness of the matching result, we use the same measures used in the literature, including the following:

- *Precision* & *Recall.* Precision and recall are the most prominent criteria used to evaluate matching effectiveness [9,57]. Precision, $P$, determines the degree of correctness of the match result. It measures the ratio of correctly identified matches (true positives, $B$) over the total number of identified matches (automatic matches, $A_m$). It can be computed as $P = \frac{|B|}{|B|+|C|}$. Recall, $R$, assesses the degree of completeness of the matching system. It measures the ratio of correctly identified matches (true positives, $B$) over the total number of correct matches (correct matches, $R_m$). It can be computed as: $R = \frac{|B|}{|B|+|A|}$.
  However, neither precision nor recall alone can accurately assess the matching quality [23]. Precision evaluates the post-match effort that is needed to remove false positives, while recall evaluates the post-match effort that is needed to add true negatives from the final match result. Hence, it is necessary to consider a trade-off between them. There are several methods to handle such a trade-off, one of them is to combine both measures. The mostly used combined measure is:
- *F-measure. F-measure* is the weighted harmonic mean of precision and recall. The traditional F-measure or balanced F-score is:

$$F - measure = \frac{2 * |B|}{(|B| + |A|) + (|B| + |C|)} = 2 * \frac{P * R}{P + R} \tag{15}$$

  If an application requires stringent matches, it is necessary to obtain higher F-measure values, in hence, a higher similarity threshold value should be set in an attempt to avoid false matches.

*Efficiency measures.* Efficiency of a matching system is usually determined by using several aspects, two of them are: *response time* (the time it takes for an operation to be completed), and *space* (the memory or non-volatile storage used up by its construct). In this study, we make use of the response time ($\mathbb{T}$) as an indicator for the schema matching efficiency.

### 6.2. Evaluation scenarios

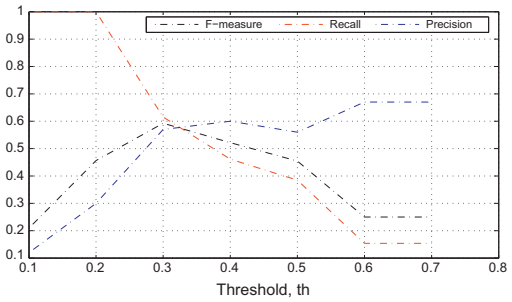
#### 6.2.1. First scenario

In this scenario we aim to evaluate element similarity measures using different combining settings and strategies. First, we conducted a set of experiments to validate internal similarity measures by using each individual measure and their different combinations. This set of experiments gave us an overview of the quality of matching results using each of the internal measures and their best combination. Second, we conducted another set of experiments to assess the performance of different combinations of internal measures with external measures. Lastly, we conducted a set of experiments to evaluate and assess the effect of using external information sources in calculating the semantic measure of name similarity. All the test cases have been performed using the weighted sum as a linear combining strategy as well as the combining function in Eq. (14) as a nonlinear combining strategy. Various weighting schemes were used and the best values were derived empirically after a set of experiments. These details are provided with the results. Furthermore, these sets of experiments are conducted using several threshold values in the range of 0.1 to 0.8. It should be noted that higher threshold values ( e.g. 0.9 and 1) have not been used due to the fact that at such a higher threshold value, it was very hard to find right matches. We aim to draw conclusions from this scenario that can be used as guides through the second scenario. The quality of element similarity measures is verified using *precision (P)*, *recall (R)*, and *F-measure*. The data set used in the first scenario include the two schemas described in Fig. 1. The properties of schemas are illustrated in Table 3, including the object identifier (OID), names, type/data types, and cardinality constraints of schema elements. Each element of $ST1$ will be compared with the elements of $ST2$ yielding $n \times m$ complexity, where $n$ is the element size of $ST1$ and $m$ is the element size of the second schema.

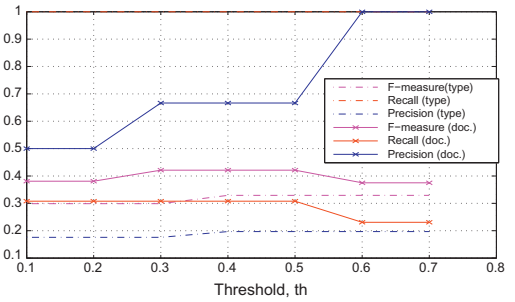
**Table 3**
Schema tree characteristics.

| OID | ST1 | | | | ST2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Name | Type | Cardinality | | Name | Type | Cardinality | |
| | | | minOccurs | maxOccurs | | | minOccurs | maxOccurs |
| $n_1$ | CSDeptUs | Complex | 0 | Unbounded | CSDeptAust | Complex | 0 | Unbounded |
| $n_2$ | UnderGradCourses | string | 0 | 1 | Courses | String | 0 | 1 |
| $n_3$ | GradCourse | String | 0 | 1 | Staff | Complex | 0 | Unbounded |
| $n_4$ | People | Complex | 0 | Unbounded | AcademicStaff | Complex | 0 | Unbounded |
| $n_5$ | Faculty | Complex | 0 | Unbounded | Lecturer | Complex | 0 | Unbounded |
| $n_6$ | AssistantProfessor | Complex | 0 | Unbounded | FirstName | String | 0 | 1 |
| $n_7$ | Name | String | 0 | 1 | LastName | String | 0 | 1 |
| $n_8$ | Degree | String | 0 | 1 | Education | String | 0 | 1 |
| $n_9$ | AssociateProfessor | String | 0 | 1 | SeniorLecturer | String | 0 | 1 |
| $n_{10}$ | Professor | String | 0 | 1 | Professor | String | 0 | 1 |
| $n_{11}$ | Staff | String | 0 | 1 | TechnicalStaff | String | 0 | 1 |

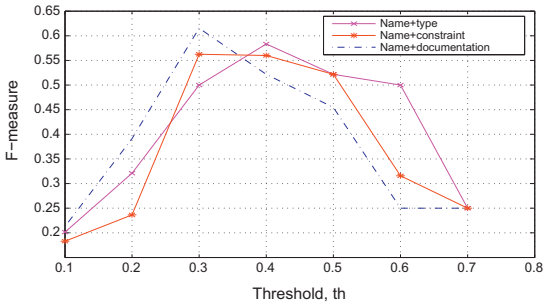
1. *Internal measures without external information sources.*
   - Using the linear combination strategy.
     The first set of experiments was implemented to observe the quality (effectiveness) of internal element similarity measures without external information sources. The quality of each individual internal measure (name, data type, annotation, and cardinality constraint) is first evaluated alone and then different combinations between them are evaluated using the linear combining strategy. The similarity value between every pair of schema tree elements is first computed using a specified element measure. The values of all pairs are then ranked and the ones that are higher than a predefined *threshold* (*th*) are selected. F-measure, Recall and Precision values are determined based on those values. The results of these evaluations are reported in Fig. 8, which represents an anticipated finding regarding to the used threshold. Small threshold values result in a large number of false positives (small precision values) and a small number of false negatives (large recall values). Increasing the value of threshold causes an opposite situation.Fig. 8(a), (b) indicates that no single element measure is able to discover the qualitative correspondences. The name measure achieves F-measures ranging between 20% and 58%, the data type measure produces F-measures between 29% and 32%, while the annotation measure gives F-measures between 37% and 42% at various threshold values. It should be noted that the cardinality constraint is also evaluated. However, its matching quality seems to be similar to the matching quality of the type measures. Hence, it is not included in Fig. 8. It ascertains that different combinations should be used to get a better matching quality. As the element name is considered the most effective feature of schema tree elements among other features—as verified by the results in Fig. 8(a), (b)—we base the combining process on the name measure. First, the name measure is combined with one of the other internal measures. The results reported in Fig. 8(c) show that combining name and documentation measures performed better than the other two combinations. It should be noted that these results have been obtained after several experiments using different weighting schemes for combining two element measures. Best results are obtained when the weight of the name measure, ($w_n$), is set as a value of 0.8 while combining it with either the type measure or the annotation measure, on the other hand are obtained when $w_n$ is set to a value of 0.9 when combining it with the constraint measure.
     The name measure is then combined with two of the other internal measures. Fig. 8(d) illustrates that F-measure improves and its value reaches 67% when combining name, type, and annotation/cardinality constraint measures. The weighting scheme producing these results is illustrated in the figure ($w_n = 0.7, w_t = 0.2, w_{doc} = 0.1$ or ($w_c = 0.1$)). Using all internal measures improves F-measure to 72%, as shown in Fig. 8(e). We obtained this result using two weighting schemes. The first scheme uses the type similarity weight ($w_t = 0.2$) to be higher than the value of annotation similarity weight ($w_{doc} = 0.1$), while the second scheme uses the opposite combination. Since the element data type is easily available than the element annotation, we decided to select the first weighting scheme ($w_n = 0.6, w_t = 0.2, w_{doc} = 0.1, w_c = 0.1$) in the further experiments.
   - Using the nonlinear combination method.
     The above set of experiments has been repeated using the nonlinear combining strategy. Further, these experiments were directed and guided by the results obtained from the first set. This means that the first term in Eq. (14) $\left(\sum_{i=1}^{\mathcal{N}} w_i x_i\right)$ is directly obtained from the first set of experiments. The constant $\lambda$ is set to 0.5 to ensure that the total similarity is in the range [0,1]. The results of these experiments are reported in Fig. 9. Fig. 9(a) shows the results after combining the name measure with other internal measures using the nonlinear strategy. Fig. 9(b) presents a comparison between combining internal measures using both linear and nonlinear strategies. The figures present two interesting findings. (1) The largest F-measure occurs at lower threshold values, as shown in Fig. 9(a), compared to the results obtained using the linear strategy, as shown in Fig. 8(c). (2) Using the nonlinear combination strategy to aggregate a small number of element measures is not as effective as using it to combine a large number of element



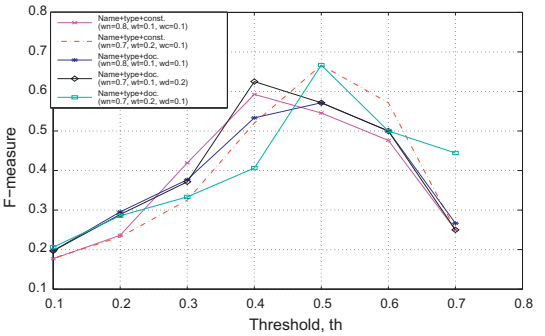

(a) Quality of name measure.



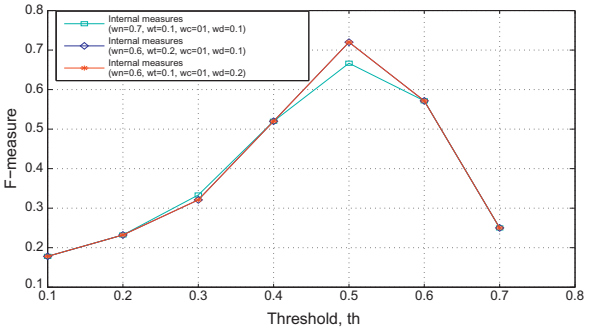

(b) Quality of type/annotation measure.




(c) Quality of name with one of internal measures.




(d) Quality of name with two of internal measures.



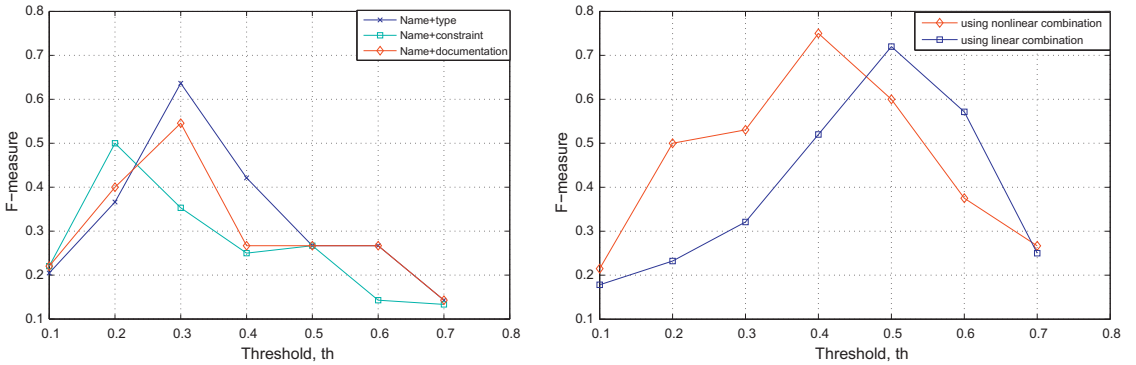

(e) Quality of all internal measures.

**Fig. 8.** Quality of internal measures using the linear combining strategy.

measures. Fig. 9(b) shows that using the nonlinear combining strategy achieves a higher F-measure than the linear strategy. However, using the nonlinear strategy, as shown in Fig. 9(a), to aggregate the name measure with another element measure does not achieve more quality improvement.

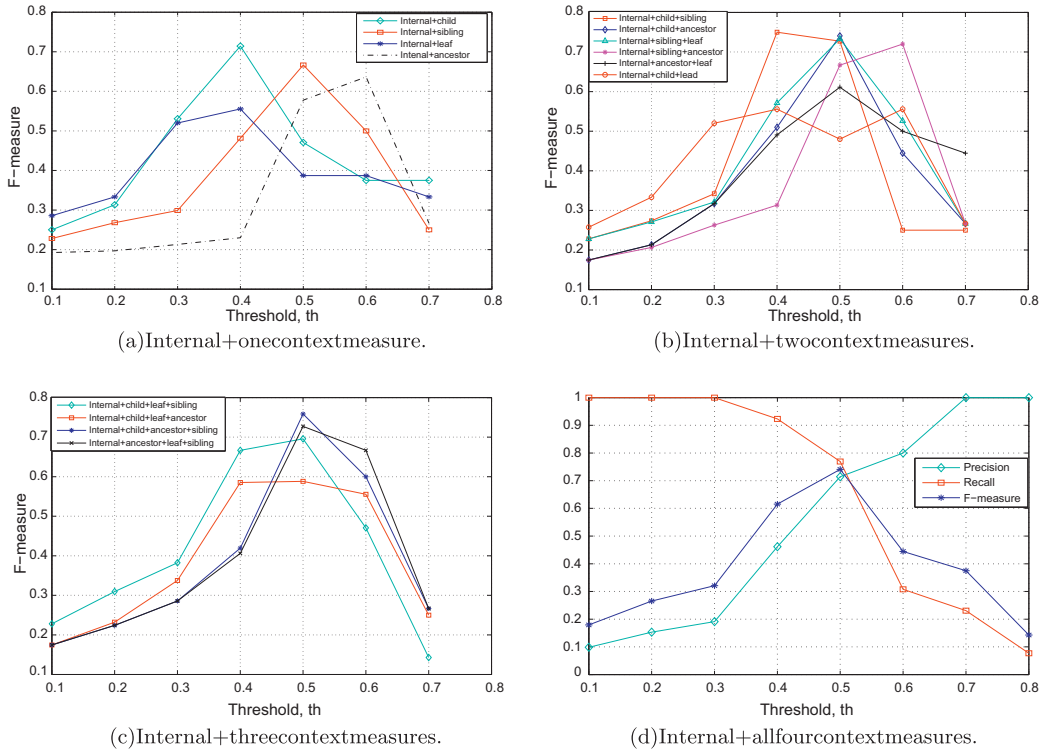
2. *Quality of internal & external measures.*
   - Using the linear combining method

   After settling on the weighting scheme used in combining internal element similarity measures, we conducted a second set of experiments to observe the matching quality of various combinations of external measures with the combined internal element measures (Name with $w_n = 0.6$ + Type with $w_t = 0.2$ + Annotation with $w_{doc} = 0.1$ + Constraint with $w_c = 0.1$). In the case of combining the internal similarity value with more than one external measures, we first combined the individual external measures, and then we combined the two internal and external similarity values. To combine individual external measures, we should consider the type of involved elements. For example, combining an atomic element from a schema tree with an atomic/complex element from another schema tree is only based on sibling and ancestor context similarities, otherwise, all content measures should be considered. The weight values for combining the internal and external similarity measures heavily depend on the type of elements: atomic or complex. When combining two atomic elements, the experimental results show that a higher weight value should be set for the internal similarity measure; when combining two complex elements, a higher values should be set for the external measure. Otherwise, we equally set values for the weighting scheme.




(a) Name with one of internal measures (b) Internal measure with nonlinear/linear combination

**Fig. 9.** Quality of internal measures using the nonlinear combining strategy.

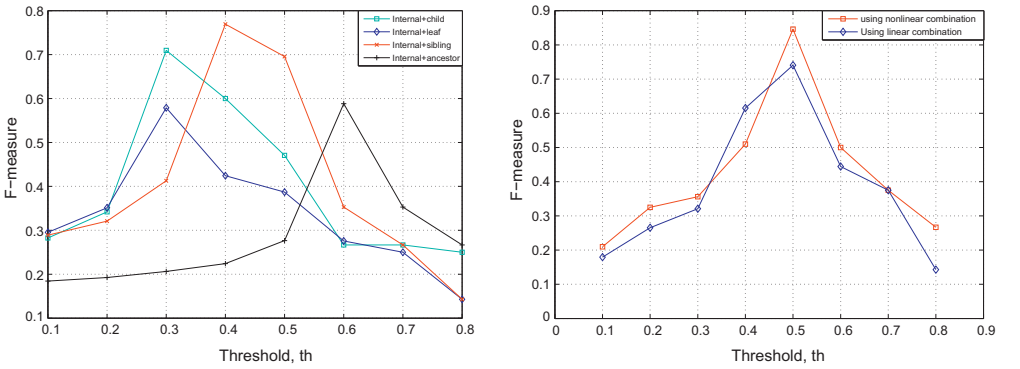



(a)Internal+onecontextmeasure. (b)Internal+twocontextmeasures.

(c)Internal+threecontextmeasures. (d)Internal+allfourcontextmeasures.

**Fig. 10.** Quality of internal & external measures using the linear combining strategy.

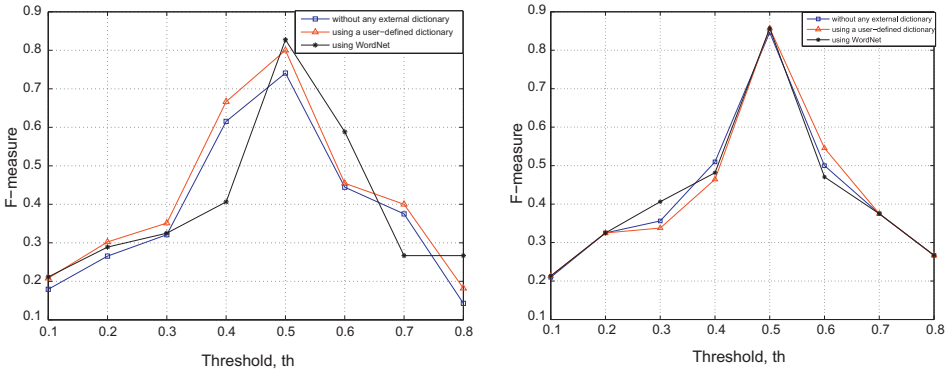
This set of experiments has the same procedure as the first one: similarity values of combining each external measure with the combined internal measure for all pairs of schema elements are first computed, ranked, and ones that are higher than the predefined *threshold* (*th*) are selected. Results of these evaluations are reported in Fig. 10. The results indicate that combining the leaf context with the internal measure deteriorates the matching quality, while the child context outperformed the other combinations, as shown in Fig. 10(a). We then evaluated the quality of combining two external similarity measures with the internal measures. Fig. 10(b) shows that combining the child context measure with another context measure other than the leaf measure surpasses the other combinations. The highest F-measure (75%) is achieved when using the child/sibling combination. The quality of combining three external measures with the combined internal measure was then evaluated and results are reported in Fig. 10(c). The matching quality is slightly improved and F-measure reaches 76%. However, the figure represents an interesting finding: combining the child measure with the leaf measure often declines the matching quality. This can be explained due to the fact that both child and leaf measures only deal with complex elements and ignore atomic elements in the schema tree. Fig. 10(d) outlines the results produced by combining the internal and all four external measures. It is noted that the matching quality has not been improved (F-measure = 0.76 and obtained at a threshold of 0.5).




(a) Internal with one context measure.

(b) Internal & external measures with nonlinear/linear combination

**Fig. 11.** Quality of internal & external measures using the nonlinear combining strategy.




(a) Using linear combining strategy.

(b) Using nonlinear combining strategy.

**Fig. 12.** Effect of using external dictionaries.

- Using the nonlinear combining method. We conducted another set of experiments to validate the matching quality of different combinations of external measures with the combined internal element measures using the nonlinear combining strategy. This set of experiments is also directed and guided by the results obtained from the set using the linear strategy. The results of these experiments are reported in Fig. 11. Fig. 11(a) represents the effect of using the nonlinear combining strategy on coalescing the internal measure with one of the external measures, while Fig. 11(b) shows a comparison between combining internal and external measures using both linear and nonlinear strategies. Fig. 11 outlines two interesting findings. (1) The largest F-measure does not occur at lower threshold values, as shown in Fig. 11(a), (b). Compared to the results using the linear strategy, the largest F-measure occurs at the same threshold value. Indeed, combining the internal measure with external measures increases the element similarity that results in higher F-measure values at higher threshold values. (2) Using the nonlinear combination strategy achieves a higher F-measure than the linear strategy, as shown in Fig. 11.

3. *Effect of external information sources.*

Although the used test schemas are small, the matching quality is not high due to the existence of different heterogeneities in the tested schemas. In the case of using the linear combining strategy, F-measure values range between 17% and 76% depending on the used element measures and the selected threshold. The best F-measure is 72% when using all four internal measures and its value is increased to 76% when the internal measures are combined with external context measures. In this section, we study the effect of exploiting external information sources (dictionaries) on the matching quality. We make use of both user-defined and generic dictionaries. We first built a domain-specific dictionary based on our experience and evaluated the effect of using it. Then, we considered the semantic measures (Eqs. (8) and (9)). In these cases the name measure becomes a combination of syntactic and semantic measures. We conducted another set of experiments to observe the effect of external information sources on the matching quality using both combining strategies. In this set of experiments, we evaluated the combined internal and external measures in the presence of WordNet as an external dictionary. Results of these evaluations are reported in Fig. 12.

In the case of using the linear combining strategy, it can be seen from the results in Fig. 12(a) that exploiting either domain-specific or generic (semantic measures) dictionaries improves the matching quality. The figure shows that F-measure has nearly the same value with/without the external dictionary at a threshold value of 0.1. Using the user-defined dictionary, F-measure improved gradually. The best F-measure obtained is 80% at a threshold value of 0.5. While exploiting WordNet achieves the highest F-measure 83% at the same threshold value.

To study the trade-off between matching quality and matching efficiency due to adding the domain-specific dictionary on the matching performance, we calculated the quality improvement ratio (QIR) of using both the user-defined domain-specific dictionary and WordNet as:

$$QIR_{user\text{-}defined} = \frac{quality\ increment}{quality\ without\ external\ source} = \frac{4}{76} = 5.26\% \tag{16}$$

$$QIR_{WordNet} = \frac{7}{76} = 9.21\% \tag{17}$$

This quality improvement normally causes a decline in matching efficiency, computed as the time needed to perform the matching task. We calculate the performance decline ratio (PDR) as:

$$PDR_{user\text{-}defined} = \frac{performance\ decrease}{performance\ without\ external\ source} = \frac{35}{200} = 17.5\% \tag{18}$$

$$PDR_{WordNet} = \frac{600}{200} = 300\% \tag{19}$$

This means that in order to improve the matching quality by a ratio of 5.26% (9.21%), we must pay a performance cost ratio of 17.5% (300%). Therefore, a trade-off between matching quality and matching efficiency should be considered, especially in the large-scale context.

In the case of utilizing the nonlinear combining strategy, as shown in Fig. 12(b), the effect of using the external dictionary on the matching quality is not significant. It increases F-measure from 0.846 to 0.857 at a threshold of 0.5, which results in a quality improvement ratio of 1.3% ($\frac{85.7-84.6}{84.6} \times 100$) at the same performance decline ratio. It should be noted that exploiting WordNet achieves higher quality improvement than exploiting the domain-specific dictionary, while it consumes more time. This can be explained as follows. WordNet is a large dictionary that gives several opportunities to assess the semantic similarity between element names. In contrast, the domain-specific dictionary only contains the specified relations between a set of element names.

#### 6.2.2. Second scenario

Our strategy in this scenario is to evaluate the quality of the "best" combinations of element measures directed by the results obtained from the first scenario, and to validate the efficiency of different element measures and their combinations using real-world data sets.

1. *Quality evaluation*

   *Data set* In this scenario, we experimented with data sets from 8 different domains. From each domain, we collected four schemas.[5] We chose the data sets because they show different characteristics in the number of schema elements (schema size) and their depth (the number of node nesting), and they represent different application domains, as shown in Table 4.

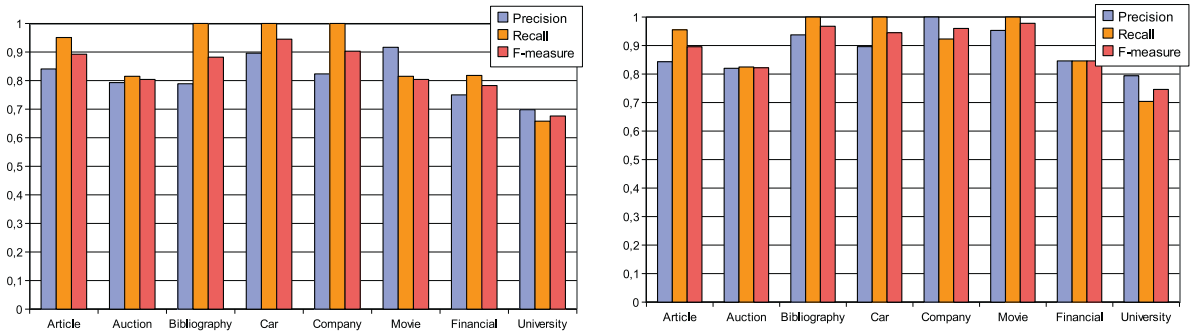
   *Evaluation methodology* Every schema pair inside the corresponding domain has been matched at a time. Hence, we have a total of 48 ($\frac{S\times(S-1)}{2} \times d$, where $S$ is the number of schemas in each domain and $d$ is the number of domains) matching tasks. The required parameters, such as weighting schemes and the threshold value, are selected guided by the findings obtained from the first scenario. Furthermore, each matching task has been conducted utilizing both linear and nonlinear combining strategies. The performance for each matching task has been first evaluated and then tasks within the same domain have been averaged.

   *Experimental results* For each schema pair in the same domain, we conducted two sets of experiments: one using the linear combining strategy and the other using the nonlinear strategy. Element similarity measures (all matchers) discover candidate matchings that exceed the predefined threshold. The matching quality criteria are then computed for the schema pair. The quality criteria for all schema pairs in the same domain are then averaged to obtain the final quality criteria for the domain. Results are summarized in Fig. 13, and present several interesting findings. (1) The nonlinear combining strategy outperforms the linear strategy across all the tested domains. Using the nonlinear strategy, F-measure ranges between 76% (the university domain) and 98% (the movie domain), while the linear strategy achieves F-measures between 68% (the university domain) and 95% (the car domain). This is due to the fact that nonlinear strategy considers the interdependencies between element similarities. (2) Both combining strategies produce matching quality over the Article, Bibliography,

[5] Schemas from domains Auction, Financial, and University are collected from http://www.cs.washington.edu/research/xmldatasets/www/repository.html, while other schemas are generated from their corresponding DTDs.

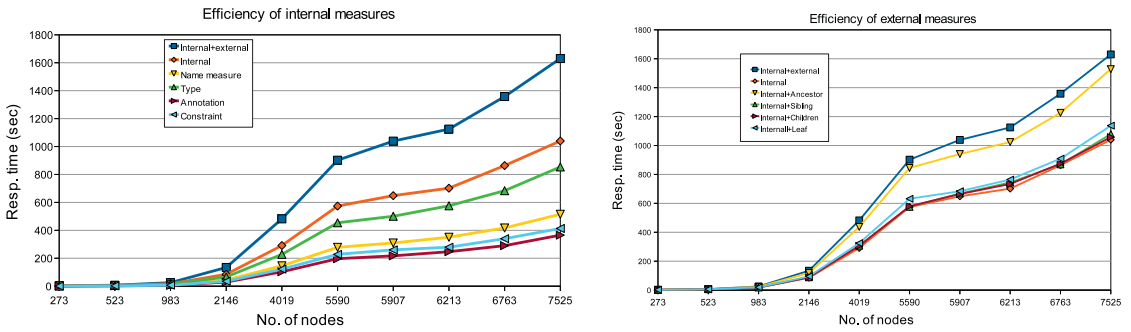
**Table 4**
Data set details.

| Domain | No. of schemas/elements | Avg. no. elements | Min./max. depth | Total size (KB) |
|---|---|---|---|---|
| Article | 4/530 | 135 | 5/10 | 100 |
| Auction | 4/140 | 35 | 5/6 | 12 |
| Bibliography | 4/60 | 15 | 6/6 | 8 |
| Car | 4/344 | 83 | 5/6 | 30 |
| Company | 4/60 | 15 | 6/6 | 8 |
| Movie | 4/40 | 10 | 5/6 | 8 |
| Financial | 4/60 | 15 | 5/6 | 8 |
| University | 4/38 | 10 | 4/5 | 8 |




(a) Using the linear combination. (b) Using the nonlinear combination.

**Fig. 13.** Matching quality for real-world data sets.




(a) Efficiency of internal measures. (b) Efficiency of external measures.

**Fig. 14.** Response time for real-world data sets.

Car, Company, and Movie domains higher than the matching quality over the Auction, Financial, and University domains. This results due to the fact that XML schemas in the first set of domains are more homogeneous than XML schemas in the second set of domains.

2. *Efficiency evaluation* We conducted another set of experiments to validate and study the effect of individual element measures on the matching efficiency. Results reported in the first scenario indicate that both linear and nonlinear combining strategies nearly require the same response time. Therefore, this set of experiments have been carried out only using the linear combining strategy.
*Data set* To validate the efficiency of element measures, we collected more schemas from the 8 used domains. The number of schemas used in this set of experiments reaches 250 with a total size of 1.2 MB.
*Evaluation methodology* Every schema pair in the data set (from different domains) is matched at a time. We determined the response time required to perform the matching task as a function of the number of schema tree elements (nodes).
*Experimental results* We first evaluated individual internal/external similarity measures and their combinations to study the effect of each measure on the matching efficiency. The results are reported in Fig. 14, which shows that the time complexity of the matching process is quadratic ($O(n^2)$, where $n$ is the number of schema elements). Fig. 14(a) illustrates that the annotation measure performs better than the other internal measures. This is due to the fact that the annotation measure is based on an IR-based measure. The figure also shows that while the type and constraint measures are based on

similarity tables, however, the constraint measure is faster than the data type measure. This is due to the fact that the number of built-in XML data types is greater than the number of cardinality constraints.

Fig. 14(b) presents the effect of the external element measures on the matching efficiency. This figure indicates that the ancestor context measure performs the worst among the other external measures. It can also be seen that child, sibling, and leaf context measures add insignificant response times to the matching efficiency. The reason behind this can be explained as follows: the ancestor measure is based on the path comparison, which requires the identification of paths to be compared. This identification process consumes much time.

### 6.3. Discussion and lessons learned

The experimental results conducted in this study present several interesting findings that can be used as guidelines during the development of a new XML element similarity measure, the implementation of a new matching component, or the comparison of existing matching systems. Furthermore, these findings can be used to frame the obtained results from the two scenarios. These findings can be summarized as follows:

- The results from the first scenario, Fig. 8(a), (b), indicate that exploiting one feature of schema tree elements is not sufficient to assess the similarity between XML schema elements. The results specify that even if the name measure outperforms other internal measures, it is not adequate to correctly quantify the element similarity. This necessitates the need to utilize several element measures exploiting both internal features and external relationships of the elements.
- Utilizing several element measures provides the advantage of matching algorithms to be more flexible and adaptable for various application domains as well as it improves the matching quality, as shown in Fig. 8(c), (d), (e). However, it also embeds a challenge of how to combine these similarity measures. In this study, we selected the aggregation function (weighted sum) as a combining methodology making use of two combining strategies: linear and nonlinear.
  - According to the linear strategy, equations in Definitions 5 and 8 can be written as follows

$$InterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = w_n \times Nsim(\mathcal{E}\ell_1.name, \mathcal{E}\ell_2.name) + w_t \times Tsim(\mathcal{E}\ell_1.type, \mathcal{E}\ell_2.type) + w_c \times Csim(\mathcal{E}\ell_1.card, \mathcal{E}\ell_2.card) + w_d \times Asim(\mathcal{E}\ell_1.annotation, \mathcal{E}\ell_2.annotation)$$

$$ExterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = w_{ch} \times ChSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) + w_l \times LeafSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) + w_{sib} \times SibSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) + w_{an} \times PSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)$$

    The reported results demonstrate that the name measure is the dominant measure among the other internal similarity measures. As a result, we set $w_n \gg (w_t \cong w_d \cong w_c)$ and combining the other individual internal measures is based on the name measure. However, given that combining individual external measures is heavily based on the type of involved elements (atomic or complex), a general rule to set its weighting scheme cannot be defined. Once obtained the internal and external similarity values, a total similarity value between a pair of elements can be determined, and Eq. (1) can also be rewritten

$$Sim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) = w_I \times InterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2) + w_E \times ExterSim(\mathcal{E}\ell_1, \mathcal{E}\ell_2)$$

    where $w_I$ and $w_E$, the weights to quantify the importance of *InterSim* measure and *ExterSim* measure, are heavily relied on the type of the elements.
  - Using the nonlinear strategy provides the possibility to consider the interdependencies between element similarities, and, thus improves the matching quality compared to the linear strategy, as shown in Figs. 9 and 11.
  - The strategy used to combine element similarity measures affects the matching quality, but it has no effect on the matching efficiency. According to results in the introduced scenarios, using the nonlinear combining strategy improves the matching quality.

    It is worth noting that this finding promotes developing of a new matching component, which comprises of a set of element measures and a combining strategy. The finding presents the outline of how to combine this set of similarity measures.
- While comparing name and type similarity measures, the name measure contributes a much larger proportion in comparison to the type measure in determining the matched elements in lesser response time. Therefore, the type measure can be ignored in large-scale applications.
- The external measures are more important when one of the matching elements is of complex type. It is important to take the surroundings into consideration while matching the elements. When both the elements are of same types, the internal measures are important.
- Concerning matching efficiency, results from the second scenario illustrate that the type measure is the most costly internal measure, while the ancestor measure is the most expensive external measure, as shown in Fig. 14. They also confirm the fact that utilizing more measures results in better matching quality, however, with increased response times.
- Selecting the candidate correspondences is largely based on the value of *threshold*. Low values of threshold result in a large number of false positives (very low precision) and a small number of false negatives (high recall), while high values of threshold causes an inverse situation, as shown in Fig. 12. Therefore, it is recommended that the matching systems should settle on using medium values for threshold ranging between 0.4 to 0.6.

- It is obvious that exploiting external information sources, such as WordNet or domain-specific dictionaries, can improve the matching quality. Reported results point out that using WordNet achieves more quality than domain-specific dictionaries. However, to get this improvement, the matching efficiency declines. In the large-scale context, a trade-off between matching effectiveness and matching efficiency should be considered.

## 7. Summary and future directions

As XML has become the standard for information representation and exchange over the Web, the number of XML-based applications has been proliferated. There has been a growing need to identify and discover the semantic correspondences among these applications. To this end, a myriad of element similarity measures have been proposed and developed.

In this paper, we presented a survey that reviews, classifies, and experimentally compares different XML element similarity measures, especially in the context of XML schema matching. Based on the exploited information in element similarity computation, we categorized element measures into internal element measures that capture the element features and external element measures that exploit element relationships. Within each category, we advised several measures that contribute to the computation of element similarity. To combine similarity values produced by different measures, we made use of both linear and nonlinear strategies. To evaluate these measures, we conducted intensive sets of experiments using two different scenarios. The first scenario was used to draw general remarks that can be used during the second one, while the experiments in the second scenario were used to evaluate the element measures against real-world data sets.

While substantial work has been conducted around the schema matching problem, further issues still deserve attention from the research community [56]. We classify these issues into the following directions:

1. *Data representation*; To perform its matching task, schema matching systems use tree structure (or tree-like) to represent XML schema. Is the tree representation sufficient for this task? In the large-scale context, another representation or an extension to the tree, such as the sequence representation or XML structural summaries, is necessary to face the performance requirements.
2. *Element similarity measures*; Specific measures should be used based on the new data representation. The arguments behind this statement are as follows: First, so far, all proposed and developed element measures are based on the internal data representation, either tree-based or graph-based. Element measures based on tree (graph) representation are derived from other domains, such as IR, and are highly expensive. Second, the combining of individual element measures is not a simple task. Therefore, we need a new set of element measures that exploit semantic and structure features of schema elements and effectively assess the element similarity.
3. *Element measures combination*; As the experimental results illustrate that no single measure is sufficient to quantify the similarity between XML schema elements, the need to combine multiple measures becomes essential. However, the arising question is how to combine these measures. Two different strategies have been presented in this study. Other methods should also be analyzed, especially the adaptive methods.
4. *Matching evaluation*; Introducing the schema matching problem to the large-scale context necessitates the need for new criteria to evaluate the matching result. Matching quality alone is not sufficient to judge the match result, matching performance should be considered. Moreover, should the two criteria be evaluated separately or together? [3].
5. *Scalability*; XML applications are becoming more and more popular. Consequently, a large number of schemas become available to match. There exists a need to develop matching methods more efficiently. In the large-scale context it becomes infeasible to match each element in a schema to another element in another schema. Solutions to group the elements and then to compare of different group of elements is one research area that can be exploited.

## Acknowledgements

We are very grateful to the referees for their constructive comments and suggestions that have led to an improved version of this paper. This work is a revised and extended version of the paper presented in [2]. The work of A. Algergawy is supported by the BMBF, grant 03FO2152.

## References

[1] S. Abiteboul, D. Suciu, P. Buneman, Data on the Web: From Relations to Semistructed Data and XML, Morgan Kaufmann, 2000.
[2] A. Algergawy, R. Nayak, G. Saake, XML schema element similarity measures: a schema matching context, in: Eighth International Conference on Ontologies, DataBases, and Applications of Semantics, ODBASE'09, Portugal, 2009, pp. 1246–1253.
[3] A. Algergawy, E. Schallehn, G. Saake, Combining effectiveness and efficiency for schema matching evaluation, in: First International Workshop, MBSDI'08, Berlin, Germany, 2008, pp. 19–30.
[4] A. Algergawy, E. Schallehn, G. Saake, A schema matching-based approach to XML schema clustering, in: 10th International Conference on Information Integration and Web-based Applications & Services, iiWAS'08, Linz, Austria, 2008, pp. 131–136.
[5] A. Algergawy, E. Schallehn, G. Saake, Efficiently locating web services using a sequence-based schema matching approach, in: 11th International Conference on Enterprise Information Systems, ICEIS'09, 2009, pp. 287–290.
[6] A. Algergawy, E. Schallehn, G. Saake, Improving XML schema matching using Prufer sequences, Data and Knowledge Engineering 68 (8) (2009) 724–747.

[7] S. Amer-Yahia, S. Cho, D. Srivastava, Tree pattern relaxation, in: Eighth International Conference on Extending Database Technology, EDBT'02, Prague, Czech Republic, 2002, pp. 496–513.
[8] P. Avesani, F. Giunchiglia, M. Yatskevich, A large-scale taxonomy mapping evaluation, in: International Semantic Web Conference, 2005, pp. 67–81.
[9] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, Addison-Wesley, 1999.
[10] M. Banek, B. Vrdoljak, A.M. Tjoa, Z. Skocir, Automating the schema matching process for heterogeneous data warehouses, in: Ninth International Conference of Data Warehousing and Knowledge Discovery, DaWaK'07, Regensburg, Germany, 2007, pp. 45–54.
[11] G. Beliakov, A. Pradera, T. Calvo, Aggregation Functions: A Guide for Practitioners, Studies in Fuzziness and Soft Computing, vol. 221, Springer, 2007.
[12] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, The spicy project: a new approach to data matching, in: 14th Italian Symposium on Advanced Database Systems, SEBD, Turkey, 2006, pp. 40–47.
[13] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, G. Summa, Schema mapping verification: the spicy way, in: 11th International Conference on Extending Database Technology, EDBT'08, France, 2008, pp. 85–96.
[14] A. Boukottaya, C. Vanoirbeek, Schema matching for transforming structured documents, in: The 2005 ACM Symposium on Document Engineering, DocEng'05, 2005, pp. 101–110.
[15] A. Budanitsky, G. Hirst, Evaluating WordNet-based measures of lexical semantic relatedness, Computational Linguistics 32 (1) (2006) 13–47.
[16] D. Carmel, N. Efraty, G.M. Landau, Y.S. Maarek, Y. Mass, An extension of the vector space model for querying XML documents via XML fragments, SIGIR Forum 36 (2) (2002).
[17] W. Cheng, Y. Sun, GSMA: a structural matching algorithm for schema matching in data warehousing, in: Fuzzy Systems and Knowledge Discovery, FSKD'05, 2005, pp. 408–411.
[18] I. Choi, B. Moon, H.-J. Kim, A clustering method based on path similarities of XML data, Data and Knowledge Engineering 60 (2007) 361–376.
[19] W.W. Cohen, P. Ravikumar, S.E. Fienberg, A comparison of string distance metrics for name-matching tasks, in: IIWeb, 2003, pp. 73–78.
[20] T. Dalamagas, A. Meliou, T. Sellis, Modeling and manipulating the structure of hierarchical schemas for the web, Information Sciences 178 (4) (2008) 985–1010.
[21] P.F. Dietz, Maintaining order in a linked list, in: 14th ACM Symposium on Theory of Computing, 1982, pp. 122–127.
[22] H.H. Do, Schema matching and mapping-based data integration, Ph.D. Thesis, Leipzig University, 2006.
[23] H.H. Do, S. Melnik, E. Rahm, Comparison of schema matching evaluations, in: The Second International Workshop on Web Databases, 2002, pp. 221–237.
[24] H.H. Do, E. Rahm, COMA- a system for flexible combination of schema matching approaches, in: 28th International Conference on Very Large Databases, VLDB'2, China, 2002, pp. 610–621.
[25] H.H. Do, E. Rahm, Matching large schemas: approaches and evaluation, Information Systems 32 (6) (2007) 857–885.
[26] A. Doan, P. Domingos, A. Halevy, Learning to match the schemas of data sources: a multistrategy approach, Machine Learning 50 (3) (2003) 279–301.
[27] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Ontology matching: a machine learning approach, Handbook on Ontologies, International Handbooks on Information Systems, 2004.
[28] C. Domshlak, A. Gal, H. Roitman, Rank aggregation for automatic schema matching, IEEE Transactions on Knowledge and Data Engineering 19 (4) (2007) 538–553.
[29] C. Drumm, M. Schmitt, H.-H. Do, E. Rahm, Quickmig – Automatic schema matching for data migration projects, in: 16th ACM Conference on Information and Knowledge Management, CIKM'07, Portugal, 2007, pp. 107–116.
[30] F. Duchateau, Z. Bellahsene, M. Roche, An indexing structure for automatic schema matching, in: ICDE Workshop, Turkey, 2007, pp. 485–491.
[31] J. Euzenat, et al., State of the art on ontology alignment, in Art of Research Project funded by the IST Program, Project number IST-2004-507482, Knowledge Web Consortium, 2004.
[32] A. Formica, Similarity of XML-schema elements: a structural and information content approach, The Computer Journal 51 (2) (2008) 240–254.
[33] A. Gal, A. Tavor, A. Trombetta, D. Montesi, A framework for modeling and evaluating automatic semantic reconciliation, VLDB Journal 14 (1) (2005) 50–67.
[34] F. Giunchiglia, F. Giunchiglia, M. Yatskevich, M. Yatskevich, Element level semantic matching, in: ISWC Workshops, 2004.
[35] F. Giunchiglia, M. Yatskevich, P. Shvaiko, Semantic matching: algorithms and implementation, Journal on Data Semantics 9 (2007) 1–38.
[36] G. Gou, R. Chirkova, Efficiently querying large XML data repositories: a survey, IEEE Transactions on Knowledge and Data Engineering 19 (10) (2007) 1381–1403.
[37] G. Guerrini, M. Mesiti, I. Sanz, An Overview of Similarity Measures for Clustering XML Documents, Web Data Management Practices: Emerging Techniques and Technologies, IDEA GROUP, 2007.
[38] Y. Hao, Y. Zhang, Web services discovery based on schema matching, in: 13th Australasian Computer Science Conference, ACSC'07, Australia, 2007pp. 107–113.
[39] A. Hliaoutakis, G. Varelas, E. Voutsakis, E.G.M. Petrakis, E.E. Milios, Information retrieval by semantic similarity, International Journal on Semantic Web and Information Systems 2 (3) (2006) 55–73.
[40] B. Jeong, D. Lee, H. Cho, J. Lee, A novel method for measuring semantic similarity for XML schema matching, Expert Systems with Applications 34 (3) (2008) 1651–1658.
[41] D. Lee, W.W. Chu, Comparative analysis of six XML schema languages, SIGMOD Record 9 (3) (2000) 76–87.
[42] M.L. Lee, L.H. Yang, W. Hsu, X. Yang, Xclust: clustering XML schemas for effective integration, in: International Conference on Information and Knowledge Management, CIKM'02, 2002, pp. 63–74.
[43] Y. Lee, M. Sayyadian, A. Doan, A. Rosenthal, eTuner: tuning schema matching software using synthetic scenarios, VLDB Journal 16 (1) (2007) 97–132.
[44] D. Lin, An information-theoretic definition of similarity, in: 5th International Conference on Machine Learning, ICML'98, 1998, pp. 296–304.
[45] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, in: 27th International Conference on Very Large Databases, VLDB'1, Roma, Italy, 2001, pp. 49–58.
[46] A. Marie, A. Gal, Boosting schema matchers, in: Confederated International Conference on the Move to Meaningful Internet Systems, OTM'08, 2008, pp. 283–300.
[47] B.D. Martino, Semantic web services discovery based on structural ontology matching, International Journal of Web and Grid Services 5 (1) (2009) 46–65.
[48] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: 18th International Conference on Data Engineering, ICDE'02, 2002, pp. 117–128.
[49] G. Navarro, A guided tour to approximate string matching, ACM Computing Surveys 33 (1) (2001) 31–88.
[50] R. Nayak, Fast and effective clustering of XML data using structural information, Knowledge and Information Systems 14 (2) (2008) 197–215.
[51] R. Nayak, W. Iryadi, XML schema clustering with semantic and hierarchical similarity measures, Knowledge-based Systems 20 (4) (2007) 336–349.
[52] R. Nayak, T. Tran, A progressive clustering algorithm to group the XML data by structural and semantic similarity, Pattern Recognition and Artificial Intelligence 21 (4) (2007) 723–743.
[53] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB Journal 10 (4) (2001) 334–350.
[54] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Fourth International Joint Conference on Artificial Intelligence, IJCAI'95, 1995, pp. 448–453.
[55] K. Saleem, Z. Bellahsene, E. Hunt, PORSCHE: performance oriented schema mediation, Information Systems 33 (7–8) (2008) 637–657.
[56] P. Shvaiko, J. Euzenat, Ten challenges for ontology matching, in: Confederated International Conference on the Move to Meaningful Internet Systems, OTM'08, 2008, pp. 1164–1182.
[57] A. Singhal, Modern information retrieval: a brief overview, IEEE Data Engineering Bulletin 24 (4) (2001) 35–43.

[58] J. Tekli, R. Chbeir, K. Yetongnon, An overview on XML similarity: background, current trends and future directions, Computer Science Review 3 (3) (2009) 151–173.
[59] J. Tekli, R. Chbeir, K. Yetongnon, Extensible user-based XML grammar matching, in: 28th International Conference on Conceptual Modeling, ER'09, 2009, pp. 294–314.
[60] A. Wojnara, I. Mlnkov, J. Dokulila, Structural and semantic aspects of similarity of document type definitions and XML schemas, Information Sciences 180 (10) (2010) 1817–1836.
[61] Z. Wu, M. Palmer, Verb semantics and lexical selection, in: 32nd Annual Meeting of the Association for Computational Linguistics, 1994pp. 133–138.
[62] S. Yi, B. Huang, W.T. Chan, XML application schema matching using similarity measure and relaxation labeling, Information Sciences 169 (1–2) (2005) 27–46.