SEMANTIC MATCHING

Fausto Giunchiglia, Pavel Shvaiko and Mikalai Yatskevich

August 2007

Technical Report DIT-07-064

Also : to appear in the Encyclopedia of Database Systems, Springer, 2008

SEMANTIC MATCHING

Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich
Department of Information and Communication Technology
University of Trento, Povo, Trento, Italy
{fausto|pavel|yatskevi}@dit.unitn.it

SYNONYMS

None

DEFINITION

Semantic matching: given two graph representations of ontologies G1 and G2, compute N1 $\times$ N2 *mapping elements* $\langle ID_{i,j}, n1_i, n2_j, R' \rangle$, with $n1_i \in$ G1, i=1,...,N1, $n2_j \in$ G2, j=1,...,N2 and R' the strongest *semantic relation* which is supposed to hold between the *concepts at nodes* $n1_i$ and $n2_j$.

A *mapping element* is a 4-tuple $\langle ID_{ij}, n1_i, n2_j, R \rangle$, i=1,...,N1; j=1,...,N2; where ID_{ij} is a unique identifier of the given mapping element; $n1_i$ is the i-th node of the first graph, N1 is the number of nodes in the first graph; $n2_j$ is the j-th node of the second graph, N2 is the number of nodes in the second graph; and R specifies a semantic relation which is supposed to hold between the concepts at nodes $n1_i$ and $n2_j$.

The *semantic relations* are within *equivalence* ($=$), *more general* ($\sqsupseteq$), *less general* ($\sqsubseteq$), *disjointness* ($\perp$) and *overlapping* ($\sqcap$). When none of the above mentioned relations can be explicitly computed, the special *idk* (I don't know) relation is returned. The relations are ordered according to decreasing binding strength, i.e., from the strongest ($=$) to the weakest (*idk*), with more general and less general relations having equal binding power. The semantics of the above relations are the obvious set-theoretic semantics.

Concept of a label is the logical formula which stands for the set of data instances or documents that one would classify under a label it encodes. *Concept at a node* is the logical formula which represents the set of data instances or documents which one would classify under a node, given that it has a certain label and that it is in a certain position in a graph.

HISTORICAL BACKGROUND

An ontology typically provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. Depending on the precision of this specification, the notion of ontology encompasses several data and conceptual models, for example, classifications, database schemas, or fully axiomatized theories. In open or evolving systems, such as the semantic web, different parties would, in general, adopt different ontologies. Thus, just using ontologies, just like using XML, does not reduce heterogeneity: it raises heterogeneity problems to a higher level. Ontology matching is a plausible solution to the semantic heterogeneity problem faced by information management systems. Ontology matching aims at finding correspondences or mapping elements between semantically related entities of the input ontologies. These mapping elements can be used for various tasks, such as ontology merging, query answering, data translation, etc. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate [6].

Many diverse solutions of matching have been proposed so far, see [7, 18, 17] for recent surveys, which addressed the matching problem from different perspectives, including databases, artificial intelligence and information systems; while the major contributions of the last decades are provided in [14, 2, 19]. Some examples of individual approaches addressing the matching problem can be found in [4, 15, 16, 8, 5][1]. Finally, ontology matching has been given a book account in [6]. This work provided a uniform view on the topic with the help of several classifications of the available methods, discussed these methods in detail, etc. In particular, the matching methods are primarily classified and further detailed according to (*i*) the input of the algorithms, (*ii*) the characteristics of the matching process and (*iii*) the output of the algorithms.

The work in [10] mixed the process dimension of matching together with the output dimension and classified matching approaches into *syntactic* and *semantic*. Syntactic are those approaches that rely on purely syntactic matching methods, e.g., edit distance between strings, tree edit distance. The semantic category, in turn, represents methods that work with concepts and compare their meanings in order to compute mapping elements. However, these have been also constrained by a second condition dealing with the output dimension: syntactic techniques return coefficients in the [0 1] range, while semantic techniques return logical relations, such as equivalence, subsumption (and justified by deductive techniques for instance). The work in [4] provided a first implementation of semantic matching.

SCIENTIFIC FUNDAMENTALS

In order to motivate the matching problem two simple XML schemas are used. These are represented as trees in Figure 1 and exemplify one of the possible situations which arise, for example, when resolving a schema integration task. Suppose an e-commerce company A1 needs to finalize a corporate acquisition of another company A2. To complete the acquisition, databases of the two companies have to be integrated. The documents of both companies are stored according to XML schemas A1 and A2, respectively. A first step in integrating the schemas is to identify candidates to be merged or to have taxonomic relationships under an integrated schema. This step refers to a process of ontology (schema) matching. For example, the elements with labels *Personal_Computers* in A1 and *PC* in A2 are the candidates to be merged, while the element with label *Digital_Cameras* in A2 should be subsumed by the element with label *Photo_and_Cameras* in A1.

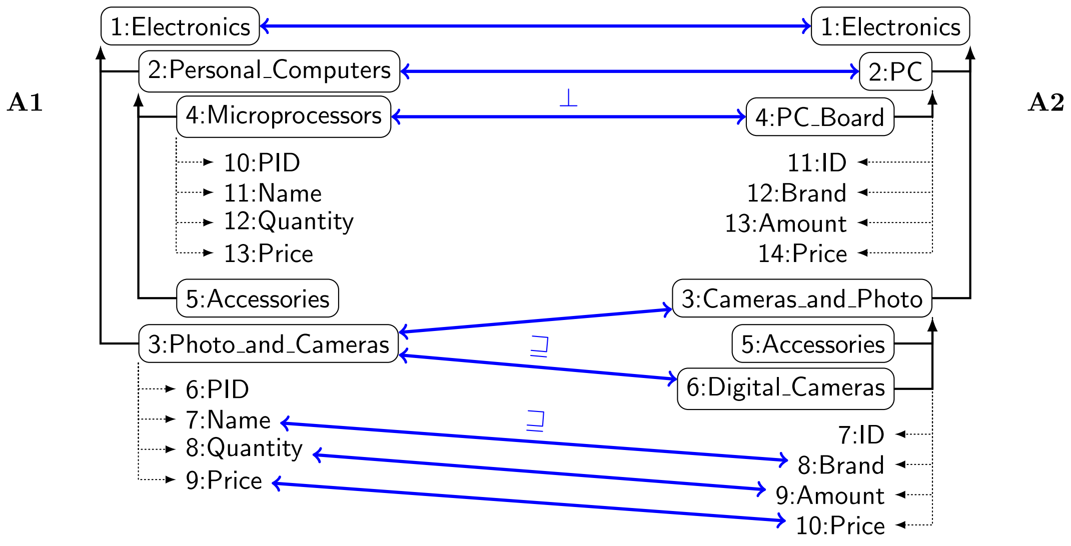

Figure 1: Two simple XML schemas. The XML elements are shown in rectangles with rounded corners, while attributes are shown without them. Numbers before the labels of tree nodes are the unique identifiers of the XML elements and attributes. In turn, the mapping elements are expressed by arrows. By default, their relation is =; otherwise, these are mentioned above the arrows.

Consider semantic matching as first motivated in [10] and implemented within the S-Match system [13]. Specifically, a schema-based solution is discussed, where only the schema information is exploited. It is assumed that all the data and conceptual models, e.g., classifications, database schemas, ontologies, can be generally

[1]See `http://www.ontologymatching.org` for a complete information on the topic.

represented as graphs. This allows for the statement and solution of a *generic (semantic) matching problem* independently of specific conceptual or data models, very much along the lines of what is done, for example, in Cupid [15].

The semantic matching takes as input two graph representations of ontologies and returns as output logical relations, e.g., equivalence, subsumption (instead of computing coefficients rating match quality in the [0 1] range, as it is the case with other approaches, e.g., [16, 15]), which are supposed to hold between the nodes in the graphs. The relations are determined by (i) expressing the entities of the ontologies as logical formulas, and (ii) reducing the matching problem to a logical validity problem. In particular, the entities are translated into logical formulas which explicitly express the concept descriptions as encoded in the ontology structure and in external resources, such as WordNet[2]. This allows for a translation of the matching problem into a logical validity problem, which can then be efficiently resolved using (sound and complete) state of the art satisfiability solvers.

Consider tree-like structures, e.g., classifications, and XML schemas. Real-world ontologies are seldom trees, however, there are (optimized) techniques, transforming a graph representation of an ontology into a tree representation, e.g., the graph-to-tree operator of Protoplasm [3]. From now on it is assumed that a graph-to-tree transformation can be done by using existing systems, and therefore, the focus is on other issues instead.

Consider Figure 1. "C" is used to denote concepts of labels and concepts at nodes. Also "$C1$" and "$C2$" are used to distinguish between concepts of labels and concepts at nodes in tree 1 and tree 2, respectively. Thus, in A1, $C1_{Photo\_and\_Cameras}$ and $C1_3$ are, respectively, the concept of the label *Photo_and_Cameras* and the concept at node 3. Finally, in order to simplify the presentation whenever it is clear from the context, it is assumed that the formula encoding the concept of label is the label itself. Thus, for example in A2, $Cameras\_and\_Photo_2$ is a notational equivalent of $C2_{Cameras\_and\_Photo}$.

The algorithm inputs two ontologies and outputs a set of mapping elements in four macro steps. The first two steps represent the pre-processing phase. The third and the fourth steps are the element level and structure level matching, respectively[3].

Step 1. For all labels L in the two trees, compute *concepts of labels*. The labels at nodes are viewed as concise descriptions of the data that is stored under the nodes. The meaning of a label at a node is computed by taking as input a *label*, analyzing its real-world semantics, and returning as output a *concept of the label*, C_L. Thus, for example, $C_{Cameras\_and\_Photo}$ indicates a shift from the natural language ambiguous label *Cameras_and_Photo* to the concept $C_{Cameras\_and\_Photo}$, which codifies explicitly its intended meaning, namely the data which is about cameras and photo. Technically, concepts of labels are codified as propositional logical formulas [9]. First, labels are chuncked into *tokens*, e.g., $Photo\_and\_Cameras \rightarrow \langle photo, and, cameras\rangle$; and then, *lemmas* are extracted from the tokens, e.g., $cameras \rightarrow camera$. Atomic formulas are WordNet *senses* of lemmas obtained from single words (e.g., cameras) or multiwords (e.g., digital cameras). Complex formulas are built by combining atomic formulas using the connectives of set theory. For example, $C2_{Cameras\_and\_Photo} = \langle Cameras, senses_{WN\#2}\rangle \sqcup \langle Photo, senses_{WN\#1}\rangle$, where $senses_{WN\#2}$ is taken to be disjunction of the two senses that WordNet attaches to *Cameras*, and similarly for *Photo*. The natural language conjunction "and" has been translated into the logical disjunction "$\sqcup$".

Step 2. For all nodes N in the two trees, compute *concepts at nodes*. During this step the meaning of the positions that the labels at nodes have in a tree is analyzed. By doing this, concepts of labels are *extended* to *concepts at nodes*, C_N. This is required to capture the knowledge residing in the structure of a tree, namely the context in which the given concept at label occurs. For example, in A2, by writing C_6 it is meant the concept describing all the data instances of the electronic photography products which are digital cameras. Technically, concepts of nodes are written in the same propositional logical language as concepts of labels. XML schemas are hierarchical structures where the path from the root to a node uniquely identifies that node (and also its meaning). Thus, following an *access criterion* semantics, the logical formula for a concept at node is defined as a conjunction of concepts of labels located in the path from the given node to the root. For example, $C2_6 = Electronics_2 \sqcap Cameras\_and\_Photo_2 \sqcap Digital\_Cameras_2$.

[2]`http://wordnet.princeton.edu/`

[3]Element level matching techniques compute mapping elements by analyzing entities in isolation, ignoring their relations with other entities. Structure level techniques compute mapping elements by analyzing how entities are related together.

Table 1: Element level semantic matchers.

Matcher name	**Execution order**	**Approximation level**	**Matcher type**	**Schema info**
WordNet	1	1	Sense-based	WordNet senses
Prefix	2	2	String-based	Labels
Suffix	3	2	String-based	Labels
Edit distance	4	2	String-based	Labels
Ngram	5	2	String-based	Labels

Step 3. For all pairs of labels in the two trees, compute *relations* among atomic *concepts of labels*. Relations between concepts of labels are computed with the help of a library of element level semantic matchers. These matchers take as input two atomic concepts of labels and produce as output a semantic relation between them. Some of them are re-implementations of the well-known matchers used, e.g., in Cupid. The most important difference is that these matchers return a semantic relation (e.g., $=$, $\sqsupseteq$, $\sqsubseteq$), rather than an affinity level in the [0 1] range, although sometimes using customizable thresholds.

The element level semantic matchers are briefly summarized in Table 1. The first column contains the names of the matchers. The second column lists the order in which they are executed. The third column introduces the matcher's approximation level. The relations produced by a matcher with the first approximation level are always correct. For example, *name* $\sqsupseteq$ *brand* returned by the *WordNet* matcher. In fact, according to WordNet *name* is a hypernym (superordinate word) of *brand.* In WordNet *name* has 15 senses and *brand* has 9 senses. Some sense filtering techniques are used to discard the irrelevant senses for the given context, see [13] for details. Notice that matchers are executed following the order of increasing approximation. The fourth column reports the matcher's type, while the fifth column describes the matcher's input. As from Table 1, there are two main categories of matchers. *String-based* matchers have two labels as input. These compute only equivalence relations (e.g., equivalence holds if the weighted distance between the input strings is lower than a threshold). *Sense-based* matchers have two WordNet senses as input. The *WordNet* matcher computes equivalence, more/less general, and disjointness relations. The result of step 3 is a matrix of the relations holding between atomic concepts of labels. A part of this matrix for the example of Figure 1 is shown in Table 2.

Table 2: The matrix of semantic relations holding between atomic concepts of labels.

	$Cameras_2$	$Photo_2$	$Digital\_Cameras_2$
$Photo_1$	idk	$=$	idk
$Cameras_1$	$=$	idk	$\sqsupseteq$

Step 4. For all pairs of nodes in the two trees, compute *relations* among *concepts at nodes*. During this step, initially the tree matching problem is reformulated into a set of node matching problems (one problem for each pair of nodes). Then, each node matching problem is translated into a propositional validity problem. Semantic relations are translated into propositional connectives in an obvious way, namely: equivalence ($=$) into equivalence ($\leftrightarrow$), more general ($\sqsupseteq$) and less general ($\sqsubseteq$) into implication ($\leftarrow$ and $\rightarrow$, respectively) and disjointness ($\perp$) into negation ($\neg$) of the conjunction ($\wedge$). The criterion for determining whether a relation holds between concepts at nodes is the fact that it is entailed by the premises. Thus, it is necessary to prove that the following formula:

$$axioms \longrightarrow rel(context_1, context_2) \qquad (1)$$

is valid, namely that it is *true* for all the truth assignments of all the propositional variables occurring in it. $context_1$ is the concept at node under consideration in tree 1, while $context_2$ is the concept at node under consideration in tree 2. *rel* (within $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) is the semantic relation (suitably translated into a propositional connective) to be proved to hold between $context_1$ and $context_2$. The *axioms* part is the conjunction of all the relations (suitably translated) between atomic concepts of labels mentioned in $context_1$ and $context_2$. The validity of formula (1) is checked by proving that its negation is unsatisfiable. Specifically, it is done, depending on a matching task, either by using ad hoc reasoning techniques or standard propositional satisfiability solvers. From the example in Figure 1, trying to prove that $C2_6$ is less general than $C1_3$, requires constructing formula (2), which turns out to be unsatisfiable, and therefore, the less general relation holds.

$$\begin{aligned}(2)\quad &((Electronics_1 \leftrightarrow Electronics_2) \wedge (Photo_1 \leftrightarrow Photo_2)\wedge \\ &(Cameras_1 \leftrightarrow Cameras_2) \wedge (Digital\_Cameras_2 \rightarrow Cameras_1))\wedge \\ &(Electronics_2 \wedge (Cameras_2 \vee Photo_2) \wedge Digital\_Cameras_2)\wedge \\ &\neg(Electronics_1 \wedge (Photo_1 \vee Cameras_1))\end{aligned}$$

A part of this matrix for the example of Figure 1 is shown in Table 3.

Table 3: The matrix of semantic relations holding between concepts at nodes (the matching result).

	$C2_1$	$C2_2$	$C2_3$	$C2_4$	$C2_5$	$C2_6$
$C1_3$	$\sqsubseteq$	*idk*	$=$	*idk*	$\sqsupseteq$	$\sqsupseteq$

Finally, notice that the algorithm returns N1 $\times$ N2 correspondences, therefore the cardinality of mapping elements is *one-to-many*. Also, these, if necessary, can be decomposed straightforwardly into mapping elements with the *one-to-one* cardinality.

KEY APPLICATIONS

Semantic matching is an important operation in traditional metadata intensive applications, such as *ontology integration*, *schema integration*, or *data warehouses*. Typically, these applications are characterized by heterogeneous structural models that are analyzed and matched either manually or semi-automatically at design time. In such applications matching is a prerequisite of running the actual system. A line of applications that can be characterized by their dynamics, e.g., *agent communication*, *peer-to-peer information sharing*, *web service composition*, is emerging. Such applications, contrary to traditional ones, require (ultimately) a run time matching operation and take advantage of more explicit conceptual models [18].

FUTURE DIRECTIONS

Future work includes development of a fully-fledged *iterative* and *interactive* semantic matching system. It will improve the quality of the mapping elements by iterating and by focusing user's attention on the critical points where his/her input is maximally useful. Initial steps have already been done in this direction by discovering automatically *missing background knowledge* in ontology matching tasks [11]. Also, an *evaluation methodology* is needed, capable of estimating quality of the mapping elements between ontologies with hundreds and thousands of nodes. Initial steps have already been done as well; see for details [1, 12]. Here, the key issue is that in these cases, specifying reference mapping elements manually is neither desirable nor feasible task, thus a semi-automatic approach is needed.

EXPERIMENTAL RESULTS

In general, for the semantic matching approach, there is an accompanying experimental evaluation in the corresponding references. Also, there is the Ontology Alignment Evaluation Initiative[4] (OAEI), which is a coordinated international initiative that organizes the evaluation of the increasing number of ontology matching systems. The main goal of the Ontology Alignment Evaluation Initiative is to be able to compare systems and algorithms on the same basis and to allow anyone for drawing conclusions about the best matching strategies. From such evaluations, matching system developers can learn and improve their systems.

[4] `http://oaei.ontologymatching.org/`

DATA SETS

A large collection of datasets commonly used for experiments can be found at:
`http://oaei.ontologymatching.org/`

URL TO CODE

The OntologyMatching.org contains links to a number of ontology matching projects which provide code for their implementations of the matching operation: `http://www.ontologymatching.org/`

CROSS REFERENCE

VIII. DATA INTEGRATION
V.c. Peer-to-Peer data management
V.d. Mobile and ubiquitous data management
I.a. Data models (including semantic data models)

RECOMMENDED READING

[1] Paolo Avesani, Fausto Giunchiglia, and Mikalai Yatskevich. A large scale taxonomy mapping evaluation. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*, pages 67–81, Galway (IE), 2005.

[2] Carlo Batini, Maurizio Lenzerini, and Shamkant Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[3] Philip Bernstein, Sergei Melnik, Michalis Petropoulos, and Christoph Quix. Industrial-strength schema matching. *ACM SIGMOD Record*, 33(4):38–43, 2004.

[4] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, pages 130–145, Sanibel Island (FL US), 2003.

[5] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Y. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[6] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2007.

[7] Avigdor Gal. Why is schema matching tough and what can we do about it? *SIGMOD Record*, 35(4):2–5, 2006.

[8] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1):50–67, 2005.

[9] Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. *Journal on Data Semantics*, VIII:57–81, 2007.

[10] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.

[11] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI)*, pages 382–386, Riva del Garda (IT), 2006.

[12] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large scale dataset for the evaluation of ontology matching systems. *The Knowledge Engineering Review*, 2008, (to appear).

[13] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: algorithms and implementation. *Journal on Data Semantics*, IX:1–38, 2007.

[14] James Larson, Shamkant Navathe, and Ramez Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[15] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pages 48–58, Roma (IT), 2001.

[16] Natalya Noy and Mark Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[17] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[18] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV:146–171, 2005.

[19] Stefano Spaccapietra and Christine Parent. Conflicts and correspondence assertions in interoperable databases. *SIGMOD Record*, 20(4):49–54, 1991.