

X-SOM: Ontology Mapping and Inconsistency Resolution

Carlo Curino, Giorgio Orsi, and Letizia Tanca

Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci, 32—20133 Milano (Italy)
{curino,orsi,tanca}@elet.polimi.it

Abstract. Data integration is an old but still open issue in the database research area, where Semantic Web technologies, such as ontologies, may be of great help. Aim of the Context-ADDICT project is to provide support for the integration and context-aware reshaping of data coming from heterogeneous data sources. Within this framework, we use ontology extraction, alignment and tailoring to find and solve conflicts due to data source heterogeneity. In this paper we present X-SOM: an ontology mapping tool developed within the Context-ADDICT project. The contribution of this high precision mapping tool is twofold: (i) a modular and extensible architecture that automatically combines several matching techniques by means of a neural network, and (ii) a subsystem for the (semi)-automatic resolution of semantic inconsistencies. Besides describing the tool components, we discuss the prototype implementation, which has been tested against the OAIE 2006 benchmark with promising results, and effectively exploited within the Context-ADDICT project.

1 Introduction

System interoperability is a well known issue, especially for information systems, and exploiting ontology-based knowledge representations may help make such interoperability automatic and transparent for the users. System interoperability is faced in the *Context-ADDICT* project [1] as a matter of data integration and personalization: the *Context-ADDICT* framework supports context-aware applications design, where several data sources must be integrated, and tailored depending on the user context.

In this project ontologies are used as a common representation paradigm to cope with data-source heterogeneity. The data-source's schemata are represented in an ontological format (i.e., a subset of OWL-DL) and integrated with a Domain Ontology referring to a specific application domain. The result is a coherent and homogeneous view of the schemata of the available data sources.

The user's *context* plays an important role in the personalization process. Following a formal methodology [2] such context is modeled [3] and the portions of the available data relevant in each individual context isolated. The upshot of this process is that the user or the application achieve a context-aware view over a set of heterogeneous data sources.

Within the framework of *Context-ADDICT*, we operate run-time ontology extraction, alignment and tailoring. The ontology alignment is performed by means of X-SOM, an eXtensible Smart Ontology Mapper used to (semi)automatically map the data source's ontologies to the Domain Ontology.

The ontology mapping problem has received a lot of attention and several techniques have been developed: [4] provides a rich survey of the existing approaches. Some of the most effective systems make use of syntactical and structural mappers like: PROMPT [5], Anchor-PROMPT [6] Chimaera [7], S-Match [8] and HMatch [9]. Other interesting approaches are: IF-MAP [10], a mapping tool based on the flow theory, AMON [11] which exploits logical inference, but also tools based on probabilistic approaches (bayesian networks) such as OMEN [12] and BayesOWL [13], or machine learning-based mapping tools such as GLUE [14]. Automatic ontology mapping is a really challenging task and we believe that the combination of several techniques can contribute to increase overall systems performance, as proven by COMA++ [15], Cupid [16] and oMAP [17], the most promising attempts, to the best of our knowledge, that have been made to combine matching techniques.

The novelty of the X-SOM ontology mapper is not in the set of matching techniques, which implements (mainly) existing, well-known algorithms, but rather in the combination of a neural network, which weights existing matching techniques, with reasoning and local heuristics aimed to both *discover* and *solve* semantic inconsistencies.

The paper is organized as follows: Section 2 presents the overall architecture of X-SOM, Section 3 discusses the Matching Subsystem while Section 4 is devoted to the Mapping Subsystem, Section 5 is dedicated to the most original contribution of this paper, the Inconsistency Resolution Subsystem; Section 6 reports the experimental results of our tool. Future development and conclusions are discussed in Section 7 and Section 8.

2 The X-SOM extensible Architecture

X-SOM operates on pairs of consistent¹ input ontologies and produces, as output, an ontology expressing the mapping among concepts and roles of the (imported) input ontologies; at the current level of development, due to its usage within *Context-ADDICT*, X-SOM does not map instances. The overall X-SOM architecture, shown in Figure 1, is composed by four subsystems: *Matching*, *Mapping*, *Inconsistency Resolution*, *Input-Output*.

The Matching Subsystem is constituted by a set of modules, sharing a uniform interface, each of which implements a known matching technique. The modules, invoked by the Mapping Subsystem according to a certain *mapping strategy*, receive as input two ontologies and return a similarity map (i.e., an estimation of the similarity of each pair of resources). The set of similarity maps is collected by the Mapping Subsystem and weighed by means of the neural network, in order to compute an aggregate matching value for each pair of resources. Given these matching values, the Mapping Subsystem applies thresholds and heuristics to compute a set of *candidate mappings*.

¹ Such assumption can be dropped, to the price of giving up the global consistency check. In such a case X-SOM will not guarantee the consistency of the mappings.

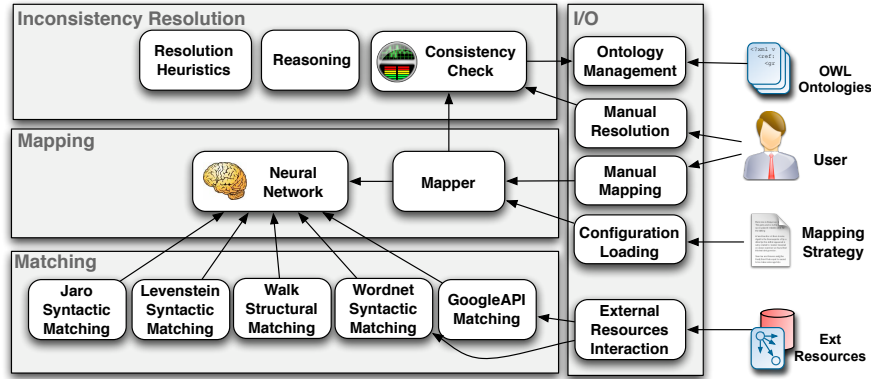


Fig. 1. X-SOM overall architecture

It is known that mapping two ontologies can produce inconsistencies [18–21]; for this reason, the set of candidate mappings computed by the Mapping Subsystem is handed over to the Inconsistency Resolution Subsystem, responsible for guaranteeing the global consistency of the final model. This subsystem uses reasoning and heuristics to discover and solve semantic inconsistencies.

X-SOM is designed to work with independent data sources and the resolution of some kind of inconsistencies may require the use of merging techniques. These techniques may produce new concepts and roles that will be stored in the mapping ontology to preserve the independence of the ontologies of the original data sources. The result of this process is a mapping ontology, written in OWL-DL, which stores all the mappings (and, if needed, resources added to solve some of the inconsistencies), expressed by means of OWL-DL constructs.

The Input-Output Subsystem is in charge of managing communication with the users and external systems: input and output ontologies' loading and writing, system configuration and parameters' loading, interaction with the user, and use of other external resources (e.g., Wordnet and Google are needed by some of the matching modules). The next Sections are devoted to a detailed discussion of the Matching, Mapping and Inconsistency Resolution Subsystems.

3 Matching

The Matching Subsystem of X-SOM has been designed to be extensible in order to easily integrate new matching modules. All the modules follow the same pattern, and implement a shared interface. Since this architecture makes experimenting new modules really easy, X-SOM can also be used as a simple framework for testing matching techniques.

Each module operates on two ontologies, received in input as Jena [22] *Ontology-Models*, and produces as output a similarity map, containing, for each pair of resources (concepts or roles), the similarity value computed by a matching technique.

Some of the modules, typically the structural ones, requires to operate on an "a-priori" similarity map, which is thus pre-computed by one of the modules (called in this case the *feeder*). It is worth noting that within the *Context-ADDICT* project there is the need to compare and match resources *at the schema level*, thus X-SOM, at the current stage of development, operates only on the T-BOX elements of the ontologies by performing a *homogeneous* analysis of the ontologies' similarities (i.e., it compares concepts with concepts and roles with roles). The implemented modules can be roughly classified into two families: the *syntactical* modules, comparing resources by analyzing their names; and the *structural* modules, comparing the structures of the two resources' neighborhoods; The modules currently implemented are:

- Jaro module (syntactical): finds the similarity of two terms using an algorithm based on Jaro String Similarity [23].
- Levenshtein module (syntactical): computes the Levenshtein distance between two terms and evaluates their degree of similarity [24].
- WordNetSimilarity module (syntactical): uses the WordNet thesaurus [25], builds a path of hypernym relations between two terms, and calculates their similarity with respect to this path.
- Googleapi module (syntactical): queries the Google search engine and, based on the number of results, computes the similarity of two terms; this technique is a variant of the one presented in [26]. This module is highly experimental and still under development.
- QOM Similarity module (structural): this module is a structural analyzer that exploits the structural matching algorithms proposed in [27, 28].
- Walk module (structural): this module is a bounded path matcher [6] which takes two paths in the input ontologies and compares corresponding terms, computing a similarity value.

Each technique has shown, in our tests, weaknesses and strengths; As an example, syntactical analyzers are not able to deal with homonyms, while structural modules are in general really precise but may have a low recall. This is why in X-SOM we weigh the results of all the modules by means of a neural network, trained to optimize the combination of the various techniques. Other modules are currently under development, in particular we are developing a bayesian network module and a module inspired by simulated annealing algorithms. Correlation analysis will later be used to discard redundant modules.

4 Mapping

The Mapping Subsystem receives as input the set of matching similarities computed by all the modules of the Matching Subsystem and produces, as output, a set of candidate mappings to be verified by the Inconsistency Resolution Subsystem.

To fulfill this goal, the Mapping Subsystem has to follow precise policies with respect to: matching modules' schedule, results aggregation, relationships between modules, greedy policies, cut thresholds and overall system behavior (automatic versus semi-automatic). Such policies are stored in a set of XML files containing the system configuration, loaded by the Input-Output Subsystem. The system configuration establishes:

- The *mapping strategy*: module schedule, modules’ parameters, greedy policies.
- The *average function*: the choice (type and parameterization) of the function used to combine the matching similarities computed by the matching modules.
- The *system behavior*: whether the system operates automatically or not and the thresholds to accept or discard candidate mappings.

The following subsections are devoted to these issues.

4.1 Mapping Strategy

The Mapping Subsystem execution flow is determined by the information provided by the mapping strategy, an XML file which contains:

- The *modules’ schedule*, storing, for each matching module, all the information that has to be used during the matching phase: the module’s name, execution priority and weight (or trust degree), the path to the parameters’ file, and the feeder module (where needed).
- The *internal parameters of the modules*: such as internal thresholds. Optimal parameter settings have been computed in the current X-SOM implementation by means of an extensive² simulation campaign.
- The *greedy policies*: operating both inter- and intra-module, they are currently under test. The goal is to increase tool scalability by reducing the number of similarity evaluations.

4.2 Average Functions: the Neural Network

The *average function* is used to combine matching modules’ opinions about the matching degree of a given resource pair. X-SOM provides standard functions (linear, sigmoidal and geometrical means and max, min functions) together with a three-layer neural network.

The most challenging issue is to assign the right weight to each matching algorithm. This task can be carried out:

- Manually, by a human expert, if a standard average function is chosen.
- Automatically, by a neural network.

The first solution implies that the operator know in advance how reliable the various techniques are, in order to assign appropriate weights. Due to the nature of the implemented functions, this solution assumes the classification problem to be linear.

To solve these problems we have equipped X-SOM with a three-layer neural network (based on the Joone API [29]). The network automatically learns, on a domain-independent training set, how to weigh each matching algorithm in order to maximize overall precision and recall. Note that, differently from other ontology matching/aligning tools, this neural network is not used to “find” the matchings, but to approximate the optimal weighting function.

² More than 54.000 runs of the tool.

The first layer (input layer) has a neuron for each matching modules in the schedule. This layer implements a simple transfer function $y=\alpha x$ where α is the bias tuned by the training algorithm.

The hidden layer's neurons have a sigmoidal transfer function, chosen to amplify matching degrees grater than 0.5 and to dull the others. The number of neurons of this layer has been chosen following the literature guidelines [30]:

$$\|hiddenNeurons\| = \lceil \log_2(\|inputs\| \times \|outputs\|) \rceil \quad (1)$$

The third layer (output layer) has one neuron with a sigmoidal transfer function and produces, for each pair of resources evaluated, the network output: the overall degree of similarity.

In our experiments we have tested three back-propagation algorithms: *on-line back-propagation*, *batch back-propagation* and *resilient back-propagation* (RPROP) [31] using cross-validation techniques to avoid net overfitting. The learning rate and the momentum have been chosen by simulations. We use 85% of the samples to train the net, and the remaining 15% to perform validation.

Our experiments have proven that the neural network training is almost domain independent, so the network training can be done once and stored as part of the system configuration. When the system runs semi-automatically the user interaction is used to perform further steps of training of the network.

Training Set: The training set for the X-SOM's neural network is constituted by a set of t -uples, where t is equal to the number of matching modules we have to weigh. Each t -uple contains the similarity values given by each module which refers to the same pair of resources, as shown in this table:

Resource1	Resource2	Jaro	WordNet	Levenshtein	Walk	Bayesian
O1:HighPerformanceVehicle	O2:SportCar	none	0.795	0.000	0.950	0.500
...	...	...	...	...	...	...

The desiderata set is built using a hand-written mapping ontology which contains all the "correct alignments", thus, when the neural network's output differs from a given correct alignment, the weights of the synapses are adjusted by using the specified back-propagation algorithm.

To build the training set we have to clean the values [30] coming from the schedule's run. During this phase the *cleaning process* removes: duplicate samples (i.e., same inputs and same desiderata), conflicting samples (i.e., same inputs but different desiderata) and adds, if not already in the set, an all-zeros sample (i.e., a sample which refers to a pair of totally different resources) and an all-ones sample, (i.e., a sample which refers to a pair of identical resources: 100% of similarity).

When one of the matching modules produces no similarity values for a pair of resources, as shown above for the Jaro module, X-SOM repairs the missing information through an average of the values produced by modules of the same family.

4.3 System Behavior

The output of the averaging function is a similarity map which aggregates the results of the various matching techniques. The next step is the decision about which of the matchings found should become a mapping. The system behavior depends on how *accept* and *discard* thresholds are set. These two thresholds determine also the level of automation of the tool:

- *Fully-automatic behavior*: the accept and discard thresholds are set to the same value γ . The Mapping Subsystem distinguishes two sets of matchings: those with similarity degree greater than γ are designated as candidate mappings, while those with a similarity degree lower than γ are discarded.
- *Conservative behavior*: the accept and discard thresholds are set to two distinct values, γ and δ . The Mapping Subsystem groups the matchings into three sets: the matchings with a similarity degree greater than γ are accepted, those with a similarity degree lower than δ are discarded. The matchings with a similarity degree between γ and δ are considered uncertain and are passed to the user in order to be manually evaluated.
- *Human-intensive behavior*: the accept threshold is set to a suitable value γ while the discard threshold is set to zero. X-SOM produces two sets of matchings as in the fully-automatic behavior, but there are no discarded matchings; the matchings with a similarity value lower than γ are considered as uncertain and submitted to the user, while the others are automatically elected to candidate mappings.

All the mappings are processed by the Inconsistency Resolution Subsystem, discussed in the next Section, to guarantee semantic consistency of the overall set of mappings.

5 Inconsistency Resolution

The problem of inconsistency check and resolution is faced in X-SOM at two different levels: *standard consistency check* and what we have called *semantic consistency check*. The standard consistency check is the process of testing if there exist contradictions in the formal representation of an application domain (unsatisfiable T-BOX, $T \models \perp$), while by *semantic consistency check* we mean the process of verifying not only the formal consistency of the T-BOX but also whether the inferences enabled by the creation of the mappings are meaningful in the application domain. While we can guarantee standard consistency by the use of a reasoner, semantic consistency cannot be formally guaranteed, and we can only apply a set of heuristics solving classes of semantic inconsistencies.

Referring to the classification of mismatches discussed in [18–21] and summarized in Figure 2, X-SOM has to deal only with ontology-level mismatches, since we assume only one natural language (English), and only one formal language (OWL-DL) and, as a consequence, the same data types for all ontologies (XSD/XML data types). The Inconsistency Resolution Subsystem architecture is a modular one, supporting seamless integration and testing of new checking algorithms. In Section 5.1 and 5.2 we present the two levels of consistency check performed by X-SOM and discuss our inconsistency resolution methods.

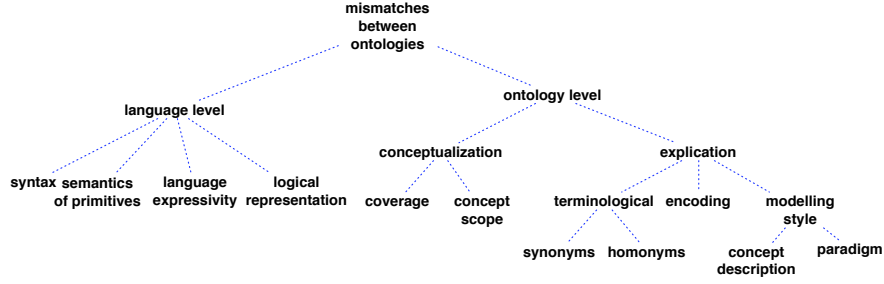


Fig. 2. Ontology mismatches classification

5.1 Standard Consistency Check

Since not input and mapping ontologies are written in OWL-DL, that is built upon the description logic $SHOIN(D_n)$, the standard consistency checking problem is equivalent to the problem of checking if there exists an interpretation I that is a model for the T-BOX. This task is easily performed by almost every reasoner. Note that, in any case, we assume that the T-Boxes of the input ontologies be satisfiable. To solve inconsistencies X-SOM applies a binomial search algorithm isolating the set of mappings responsible for the inconsistencies. In semi-automatic mode, the tool presents the inconsistent mappings to the user, while in automatic mode a heuristic choice is taken and a minimal set of mappings is discarded. The process is iterated until the T-BOX is satisfiable. Since this process requires global T-BOX satisfiability check, which is a quite expensive reasoning task, we introduced, to increase X-SOM scalability, local consistency analysis and inconsistency resolution, in order to reduce the load of the global satisfiability check. In the remainder of this section we discuss these local procedures.

Mapping two partitions A very common problem is the mapping of partitions: if two partitions (each in one of the input ontologies) partially overlap, some of the involved classes may be inferred as unsatisfiable. In Figure 3, the classes $O1:A$ and $O2:D$, due to the mappings, denoted as dashed lines, become unsatisfiable. A local search is used

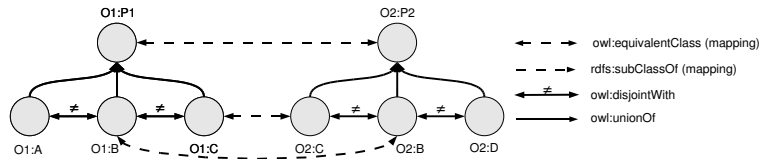


Fig. 3. Mapping two partitions

to find this kind of mismatches, which are solved by creating a third partition, stored in the mapping ontology, which represent a super-partition used to merge the initial ones. Figure 4 shows the results of this process as applied to the problem of Figure 3. The initial partitions are mapped as subclasses of the merged partition.

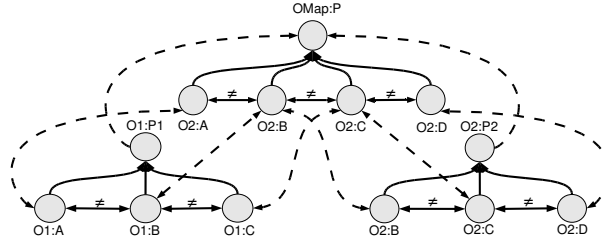


Fig. 4. Solved partitions inconsistency

Mappings between Opposites When a concept of the first input ontology is mapped (equivalence mapping) to two *disjoint* concepts of the second input ontology, an inconsistency emerges: a reasoner would infer both equivalence and disjointness of the two concepts of the second ontology. A local heuristic, which topologically looks for this kind of symptomatic structures, is applied. To solve this inconsistency one of the two mappings must be discarded; this is done either by the user, or automatically by preserving the mapping with greater matching value. Of course we have no guarantees that the automatic procedure leads to the correct mapping removal, however it seems the best we can do in a fully automatic way.

5.2 Semantic Consistency Check

By Semantic Consistency Check we mean the process of verifying whether there are mappings that induce the reasoner to infer some statement which is meaningless in the application domain, but *do not introduce contradictions in the logical theory*; Since no formal contradictions can be discovered, X-SOM has to exploit a set of procedures looking for local “symptoms” of semantic inconsistency. We have grouped such symptoms into four categories: *Bowties*, *Cycles*, *Multiple mappings with information*, *Multiple mappings without information*. In the following subsections we discuss procedures used to discover and solve these inconsistencies.

Bowties Different ontology designers may represent a portion of the real world in structurally different ways. An example is when the two input ontologies contain two “inverse” hierarchies. In such a case an automatic mapper would most probably create a “bowtie” topology as the one exemplified in the left-hand-side of Figure 5. We have a bowtie when two mapped concepts are in a subclass relation in one ontology and in a superclass relation in the other one. The four concepts directly involved in the bowtie collapse, since for each pair of concepts it is possible to deduce both direct and inverse subClassOf relations, and thus equivalence. An even worse side effect is that the reasoner can wrongly infer subclass relations among concepts. With respect to the left-hand-side of Figure 5, we can infer the following:

And, similarly, $O1 : Reader \sqsubseteq O2 : Author$. Such deductions are logically correct, although evidently false in the modeled world. To solve such inconsistencies X-SOM maps as equivalent classes the leaves of the bowtie³: in the example, $O1:Author$ and

³ It is easy to prove that this works for chains of bowties as well.

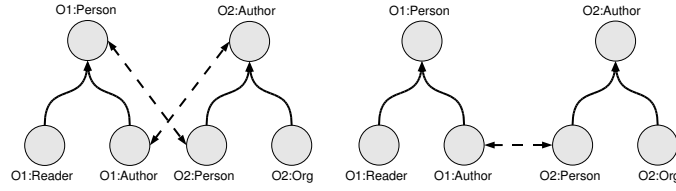


Fig. 5. An example of a bowtie inconsistency and its solution

$$\begin{array}{l}
 \text{O2:Organization} \sqsubseteq \text{O2:Author} \\
 \text{O2:Author} \equiv \text{O1:Author} \quad \Rightarrow \quad \text{O2:Organization} \sqsubseteq \text{O1:Person} \\
 \text{O1:Author} \sqsubseteq \text{O1:Person}
 \end{array}$$

O2:Person, as shown in the right-hand-side of Figure 5. This solution deems the two classes as having the same instances: in fact, *O1:Author* contains all the individuals that are persons and authors, while *O2:Person*, contains all the individuals that are authors and persons. In conclusion, the mapped classes represents the same set of individuals. This solution, although not intuitive, leads to correct reasoning over the instances and proper instance retrieval.

Cycles Since mappings may introduce `rdfs:subClassOf` relations among concepts it is possible that cyclic structures arise when adding mappings. The result will be the deduction of equivalence of all the nodes included in the cycle. This is logically correct although introduces new, often wrong, knowledge. In fact, if the concepts were equivalent, the ontology designers should have declared it explicitly in the original ontologies. To solve this symptom of inconsistency the cycle must be interrupted in some point; we choose to discard the mapping with lowest matching value.

Multiple Mappings with Information If the Mapping Subsystem returns multiple mappings among the same set of concepts [32], it may happen that new inferences are enabled about concepts of the same ontology. The left-hand-side of Figure 6 shows an example where a `subClassOf` relations between *O1:A* and *O1:B* becomes an equivalent-Class due to inference over the mappings. We consider this, as already said for bowties and cycles, a symptom of inconsistency, in the sense that this brings no contradictions in the logical theories, but modifies the semantics of the input ontology. If *X-SOM* is operating in semi-automatic mode, these mappings are considered as uncertain mappings and the user is asked to solve the inconsistency. Otherwise, we can exploit the prior information of the relationship present in the input ontologies among the concepts involved in the multiple mapping, to automatically solve such inconsistencies, this is done by means of procedures which, depending on equivalence and subclass relationship among concepts and the matching values estimate the most probable mapping. The right-hand-side of Figure 6 shows a conservative solution of the problem.

Multiple Mappings without Information An even worse situation is the one in which no prior knowledge is available about the relationships among the concepts involved in

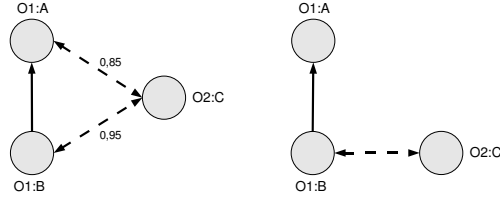


Fig. 6. A multiple mapping mismatch with information

the multiple mapping. To solve this kind of mismatches X-SOM needs to isolate the *clusters of concepts* involved in the mapping, as shown in Figure 7, i.e., sets of resources bound to each other by the mappings. Formally we can write:

Let $\mathbf{M}$ be a set of mappings between two ontologies O_1 and O_2 and, for each i and j , let m_{ij} be a mapping in $\mathbf{M}$ that maps a concept C_i of O_1 to a concept C_j of O_2 .

$\mathbf{M}$ is a cluster of concepts if and only if at least one of the following properties holds:

1. $|\mathbf{M}| = 1$.
2. $\forall m_{ij} \in \mathbf{M} \exists m_{ik} \in \mathbf{M} \mid C_k \text{ belong to } O_2$.
3. $\forall m_{ij} \in \mathbf{M} \exists m_{kj} \in \mathbf{M} \mid C_k \text{ belong to } O_1$.

After all clusters have been identified, X-SOM looks for a consistent set of mappings defined as follows:

Let $\mathbf{M}$ be a set of mappings between two ontologies O_1 and O_2 , and let m_{ij} be a mapping in $\mathbf{M}$ that maps a concept C_i of O_1 to a concept C_j of O_2 .

$\mathbf{M}$ is a consistent set of mappings (CSM) if and only if $\forall m_{ij} \in \mathbf{M} \nexists m_{ik}, m_{kj}, i \neq k, j \neq k \mid m_{ik}, m_{kj} \in \mathbf{M}$.

We define an optimal set of mapping (OSM) a CSM that maximizes a *performance measure*

$$\Psi = f(sim(C_i, C_j)) \quad (2)$$

where C_i and C_j are two resources mapped with $m_{ij} \in \mathbf{M}$, and $sim(C_i, C_j)$ is the similarity value between C_i and C_j . Possible performance measures we have considered are:

$$\Psi(M) = \sum_{m_{ij} \in M} sim(C_i, C_j) \quad (3)$$

$$\Psi(M) = \frac{1}{|\mathbf{M}|} \sum_{m_{ij} \in M} sim(C_i, C_j) \quad (4)$$

$$\Psi(M) = max(|\mathbf{M}|) \quad (5)$$

The problem of finding the OSM is solved with the support of a Branch&Bound tree by using a variant of the A* algorithm [33].

For each cluster, X-SOM builds a mapping tree and computes an optimal set of mappings (OSM). Each branch of the tree represents a possible OSM as shown in Figure 7, X-SOM computes the performance function Ψ on each branch and takes the one with the greatest Ψ . It is easy to see that the maximum depth of the tree is known in advance, and we can apply pruning technique to reduce the number of explored paths, as shown in Figure 7. In this example the OSM, using the average of similarity value performance measure (4), is:

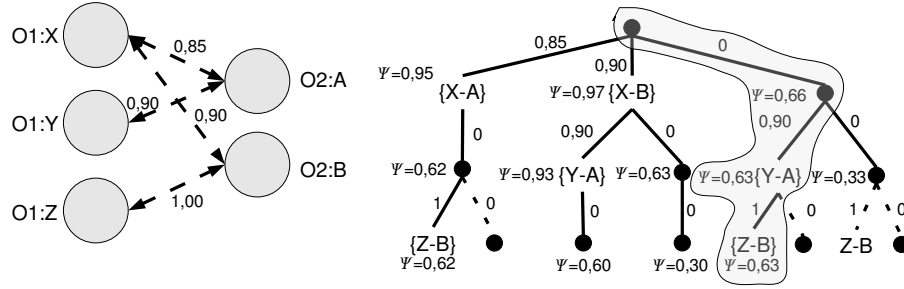


Fig. 7. Clusters of concepts and correspondent mapping tree

O1:Y equivalentClass O2:A
O1:Z equivalentClass O2:B
with $\psi(M) = 0.63$

The result of the Inconsistency Resolution Subsystem is a set of mappings, in which the tool or a human expert have solved all the syntactic and semantic inconsistencies.

6 Experimental Results

X-SOM's performance has been evaluated in terms of the (traditional) precision and recall measures. The tests have been made on the systematic benchmark series proposed by the Ontology Alignment Evaluation Initiative Campaign (OAEI 2006) [34], which refers, this year, to the bibliography domain. It is important to recall that X-SOM does not compute mappings between individuals and that the tool is designed to handle OWL-DL ontologies using reasoning to improve mapping quality. As a consequence, in the OAEI benchmark the tool has performed worse than in the real life situations for which it has been designed. However, the overall results are definitely encouraging: on the average, X-SOM reaches a **recall** of **58%** and a **precision** of **95%**. Such performances are shown, aggregated by test family, in Figure 8.

Note that, in the tests 101 to 104, ontologies with different expressive powers have to be mapped: due to the type of mappings X-SOM computes, our tool performs quite well. The aggregated results we show do not include test 102, consisting in mapping totally different ontologies (and precision is not computable). It is important, however, to point out that, when running test 102, we got two false-positives, due to “almost-homonyms” terms in the input ontologies. Tests 201-247 stress different matching techniques by: using syntactic variants in resources' names like different languages and naming conventions, or replacing original names with random ones, or modifying the structure of hierarchies exploding or flattening them. The most difficult situation for X-SOM is that of random name substitution, since our structural tools are not purely structural but operate over the results of syntactic techniques, which in case of random naming are non-effective. Pure structural mapping techniques, currently under development, will be added to X-SOM to improve performance in this test.

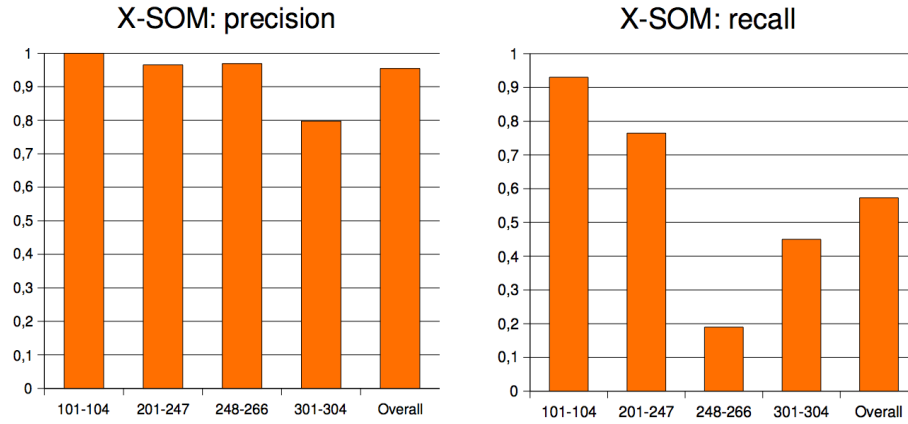


Fig. 8. X-SOM's aggregated performances

Tests 248 to 266 mix the previous tests, so random names combined with flattened or exploded hierarchies; the same kind of problems arises due to resource random naming. Tests 301 to 304 refer to real ontologies (BIBTex/Mit, BibTex/UMBC, Karlsruhe, INRIA) to be aligned with the reference ontology of test 101.

We believe that the high precision of X-SOM is due to the selectivity of the structural matching techniques combined with the consistency checking process, which turn out to be even more effective in real life OWL-DL ontology mapping, as the one we have in the Context-ADDICT project.

7 Future Developments

X-SOM is still a prototype and much work is still in order, about:

- Scalability: a set of ongoing activities is aimed at the optimization of X-SOM's implementation, in order to improve its scalability. Some of the ongoing activities concern greedy mapping strategies and early pruning of bad candidate matchings.
- New Matching Techniques: more matching techniques are currently under development, aimed to increase recall; they are based on bayesian networks, metadata inspection, instance comparison, pure structural similarity, simulated annealing structure modifications.
- Mapping Class improvement: currently, X-SOM manages homogeneous mappings, while we are currently working to extend the class of mappings to richer ones, as for instance GLAV mappings [35].
- Consistency Check Formalization: the semantic consistency check is currently implemented as a set of heuristics; a formalization effort is ongoing to provide a more systematic framework for semantic consistency analysis.
- Graphical User Interface: Considering the semi-automatic execution of X-SOM, we are applying ontology visualization techniques to build an effective interface to the tool.

8 Conclusions

We have presented a high-precision, extensible ontology mapping tool which performs consistency check and resolution.

Several matching techniques are weighted by means of a neural network, trained in a domain-independent way. Candidate mappings are analyzed to discover inconsistencies, which are solved (semi-)automatically. The extensibility of the architecture makes X-SOM a good framework for testing both matching techniques and consistency check heuristics. The preliminary results we obtained in terms of precision and recall are definitely promising and we believe the next enhancements do not present excessive difficulties.

References

1. C. Bolchini, C. Curino, F. A. Schreiber, and L. Tanca, "Context integration for mobile data tailoring," in *Proc. IEEE/ACM of Int. Conf. on Mobile Data Management*. IEEE, ACM, May 2006.
2. C. Bolchini and E. Quintarelli, "Filtering mobile data by means of context: a methodology," *Springer-Verlag, LNCS 4278*, pp. pp. 1986–1995, 2006.
3. C. Curino, E. Quintarelli, and L. Tanca, "Ontology-based information tailoring," in *Proc. IEEE of 2nd Int. Workshop on Database Interoperability (InterDB 2006)*, April 2006, pp. 5–5.
4. L. Predoiu, C. Feier, F. Scharffe, J. de Bruijn, F. Martin-Recuerda, D. Manov, and M. Ehrig, "D4. 2.1 state-of-the-art survey on ontology merging and aligning," *Institut AIFB, Universität Karlsruhe, Tech. Rep.*, 2005.
5. N. F. Noy and M. A. Musen, "The prompt suite: interactive tools for ontology merging and mapping," *Int. J. Hum.-Comput. Stud.*, vol. 59, no. 6, pp. 983–1024, 2003.
6. N. Noy and M. Musen, "Anchor-prompt: Using non-local context for semantic matching," *Proc. of the Workshop on Ontologies and Information Sharing at the Int. Joint Conf. on AI (IJCAI)*, 2001.
7. D. McGuinness, R. Fikes, J. Rice, and S. Wilder, "The Chimaera Ontology Environment," *Proc. of the 17th Nat. Conf. on AI*, 2000.
8. F. Giunchiglia, P. Shvaiko, and M. Yatskevich, "S-match: an algorithm and an implementation of semantic match," in *Proc. of the ESWC2004 Symposium*, pp. 61–75, 2004.
9. S. Castano, A. Ferrara, and S. Montanelli, "H-match: an algorithm for dynamically matching ontologies in peer-based systems," in *Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003)*, Berlin, Germany, September 2003.
10. Y. Kalfoglou and M. Schorlemmer, "IF-Map: An ontology-mapping method based on information-flow theory," *Journal on Data Semantics*, vol. 1, no. 1, pp. 98–127, 2003.
11. A. Sánchez-Alberca, R. García-García, C. Sorzano, C. Gutiérrez-Cossío, M. Chagoyen, and M. López, "AMON: A Software System for Automatic Generation of Ontology Mappings."
12. P. Mitra, N. Noy, and A. Jaiswal, "OMEN: A Probabilistic Ontology Mapping Tool," *Workshop on Meaning Coordination and Negotiation at ISWC2004*, 2004.
13. Z. Ding, Y. Peng, R. Pan, and Y. Yu, "A Bayesian Methodology towards Automatic Ontology Mapping," *Proc. of the 1st Int. Workshop on Contexts and Ontologies: Theory, Practice and Apps at AAAI*, vol. 5.
14. P. D. A. Doan, J. Madhavan and A. Halevy, "Ontology matching: A machine learning approach," *Handbook on Ontologies in Information Systems*, pp. 397–416, 2004.

15. D. Aumüller, H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," *Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of data*, pp. 906–908, 2005.
16. J. Madhavan, P. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid," *The VLDB Journal*, pp. 49–58, 2001.
17. U. Straccia and R. Troncy, "oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies," *6th Int. Conf. on Web Information Systems Engineering, New York City, New York, USA*, pp. 133–147, 2005.
18. M. Klein, "Combining and relating ontologies: an analysis of problems and solutions," *Workshop on Ontologies and Information Sharing, IJCAI*, vol. 1, 2001.
19. P. D. A. Y. H. Jayant Madhavan, Philip A. Bernstein, "Representing and reasoning about mappings between domain models," *Proc. of the 18th Nat. Conf. on AI*, 2002.
20. J. a. M. Sofia Pinto, Assunción Gómez-Perèz, "Some issues on ontology integration," *Proc. of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99*, Aug 1999.
21. B.-C. S. Pepijn Visser, Dean Jones, "An analysis of ontological mismatches: Heterogeneity versus interoperability," *AAAI Spring Symp. on Ontological Engineering, Stanford*, 1997.
22. "Jena - a semantic web framework for java," <http://jena.sourceforge.net/>.
23. S. E. F. William W. Cohen, Pradeep Ravikumar, "A comparison of string distance metrics for name-matching tasks," *Proc. of the IJCAI-2003 Workshop on Information on the Web*, 2003.
24. D. Sankoff and J. Kruskal, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley Reading, MA, 1983.
25. G. A. Miller, "Wordnet: a lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
26. P. Cimiano and S. Staab, "Learning by googling," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 2, pp. 24–33, 2004.
27. S. S. Marc Ehrig, "Qom quick ontology mapping," *Proc. of the 3rd Int. Semantic Web Conference (ISWC2004)*, Nov 2004.
28. M. Ehrig and Y. Sure, "Ontology mapping-an integrated approach," *Proc of the 1st European Semantic Web Symposium*, vol. 3053, pp. 76–91, 2004.
29. "The joone project," <http://www.jooneworld.com/>.
30. C. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 1994.
31. M. Hagan, H. Demuth, and M. Beale, *Neural network design*. PWS Publishing Co. Boston, MA, USA, 1997.
32. J. Stuller, "Data integration, data inconsistencies and data semantics," *Proc. of VLDB 2003 Workshop*, 2002.
33. S. Russel, P. Norvig, *et al.*, "Artificial intelligence: A modern approach," 1995.
34. "Oaei 2006 campaign," <http://oaei.ontologymatching.org/2006/>.
35. "Data integration: a theoretical perspective."