X-SOM: A Flexible Ontology Mapper*

Carlo Curino, Giorgio Orsi, and Letizia Tanca
Politecnico di Milano - P.zza L. da Vinci, 32 – 20133 Milano (Italy)
E-mail: {curino,orsi,tanca}@elet.polimi.it

Abstract

System interoperability is a well known issue, especially for heterogeneous information systems, where ontology-based representations may support automatic and user-transparent integration. In this paper we present X-SOM: an ontology mapping and integration tool. The contribution of our tool is a modular and extensible architecture that automatically combines several matching techniques by means of a neural network, performing also ontology debugging to avoid inconsistencies. Besides describing the tool components, we discuss the prototype implementation, which has been tested against the OAEI 2006 benchmark with promising results.

1 Introduction

Semantic heterogeneity is a relevant problem in modern information systems, where the ability to map different schemata related to the same domain, yet preserving model consistency and query answerability, is paramount.

In this paper we present X-SOM, the eXtensible Smart Ontology Mapper we have adopted within the Context-ADDICT project [3]. In Context-ADDICT, heterogeneous, dynamic data-sources are captured and automatically integrated, to generate context-based data views to be loaded on a portable device.

In the last few years several ontology mapping techniques have been developed [14]. Some of the most effective systems make use of syntactical and structural matchers like: PROMPT [13], Chimaera [11] and HMatch [4]. Other interesting approaches are: AMON [15] which exploits logical inference, OMEN [12], based on a probabilistic approach (bayesian networks), and machine learning-based mapping tools such as GLUE [6]. Automatic ontology mapping is a challenging task and we believe that the automatic combination of several techniques may improve performance, as proven by APFEL [7] and COMA++ [1]: the most promising attempt, to the best of our knowledge, of combining different matching techniques.

X-SOM automatically combines several matching techniques by means of a neural network and performs (semi)-automatic resolution of inconsistencies. Given two consistent ontologies, X-SOM produces a set of (equivalence or subsumption) mappings between their concepts and roles, ensuring also the consistency of the resulting model. The novelty of our ontology mapper is not in the set of implemented matching techniques, but rather in their combination based on a neural network, which estimates the impact of each technique on the overall similarity, and in the subsequent application of a consistency checking process.

The paper is organized as follows: In Section 2 we present the ontology mapping problem, Section 3 presents the overall architecture of X-SOM, Section 4 discusses the Matching Subsystem while Section 5 is devoted to the Mapping Subsystem; Section 6 introduces our consistency checking process and Section 7 presents the experimental results of our tool. Conclusions and future developments are discussed respectively in Section 8 and Section 9.

2 The Ontology Mapping Problem

Ontology mapping/alignment has been defined as *the process of bringing two or more ontologies into mutual agreement, by relating their similar concepts and roles by means of some kind of mappings, and making them consistent and coherent*. Throughout this paper we limit the definition to *homogeneous* mappings, where classes are mapped to classes, roles to roles and individuals to individuals. Moreover we consider two kinds of mapping relationships: equivalence and subsumption expressed through the OWL/RDFS primitives `owl:equivalentClass` and `rdfs:subclassOf` respectively.

3 The X-SOM extensible Architecture

The overall X-SOM architecture is composed by three subsytems: *Matching, Mapping* and *Inconsistency Resolution*.

*This research is partially supported by the Italian MIUR projects: ART-DECO (FIRB), and ESTEEM (PRIN).

The Matching Subsystem is constituted by a set of modules, each of which implements a matching technique and is invoked by the Matching Subsystem according to a configurable schedule. Each module receives as input two ontologies and returns a set of proposed mappings between pairs of resources with a similarity degree. This structure is called *similarity map*. Similarity maps are collected by the Mapping Subsystem and weighed by means of a neural network, in order to compute an aggregate matching value $\upsilon \in [0,1]$ for each pair of resources. Given these aggregate matching values, the Mapping Subsystem computes a set of *candidate mappings* by applying a threshold value.

An ontology mapping process can produce inconsistencies [9,10,16]; for this reason, the set of candidate mappings computed by the Mapping Subsystem is handed to the Inconsistency Resolution Subsystem, responsible for guaranteeing the global consistency of the final model.

Ontologies are often published on the web and not accessible for modifications. For this reason and to preserve the original representations, X-SOM mappings are stored in a separated ontology called *mapping ontology*.

4 Matching

The Matching Subsystem has been designed to be extensible, to allow easy integration of new matching modules. Since this architecture makes experimenting new modules very easy, X-SOM can also be used as a framework for evaluating matching techniques.

The implemented modules can be roughly classified into two families: *syntactical*, comparing resources by analyzing their names, labels and comments; and *structural*, comparing the structures of the resources' neighborhoods; we have currently implemented five modules: (1) the `Jaro` module (syntactical) implements an algorithm based on Jaro String Similarity [5]; (2) the `Levenshtein` module (syntactical) computes the Levenshtein string distance; (3) the `WordNetSimilarity` module (syntactical) uses the WordNet thesaurus, exploiting the knowledge of synsets; (4) the `QOM Similarity` module (structural) exploits the structural matching algorithms proposed in [8]; finally, (5) the `Walk` module (structural) is a bounded path matcher [13], which compares corresponding terms along two paths in the ontology graph. Some of the modules, typically the structural ones, require to operate on an "a-priori" similarity map, which is thus pre-computed by one of the other modules (called the *feeder*). Each technique has shown, in our tests, weaknesses and strengths; as an example, syntactical analyzers are not able to distinguish homonyms, while structural modules are in general very precise but may lack in recall. This is why X-SOM weighs the results of all the modules by means of a neural network, trained to optimize the combination of the various techniques.

5 Mapping

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

Once the neural network has computed the similarity map with the aggregate similarity values, X-SOM filters them by means of two configurable thresholds: *accept* and *discard*. These thresholds determine the level of automation of the tool, called *behavior*, which can be:

- *Fully-automatic*: The Mapping Subsystem distinguishes two sets of matchings: those with similarity degree greater than the accept threshold are designated as candidate mappings, while the others are discarded.
- *Conservative*: The Mapping Subsystem groups the matchings into three sets: the matchings with a similarity degree greater than the accept threshold are accepted, those with a similarity degree lower than the discard threshold are discarded. The remaining matchings are considered as uncertain and submitted to the user in order to be manually evaluated.
- *Human-intensive*: X-SOM produces two sets of matchings; those with a similarity value lower than the accept threshold are considered as uncertain and submitted to the user, while the others are automatically elected to candidate mappings.

5.1 The Neural Network

The most challenging issue is to assign the right weight to each matching algorithm. Formally, the problem is to estimate the optimal aggregation function $y = W(\mathbf{X})$ where each component $x_t \in \mathbf{X}$ is the matching degree given by the t^{th} module of the schedule with respect to a resource pair, and y is the aggregate similarity.

We have equipped X-SOM with a three-layer feed-forward neural network that is used to approximate the W function. Noticed that this task can be carried out also manually using some simple functions (linear, sigmoidal and geometrical means and max, min functions) but this solution implies that the expert knows in advance how reliable the various techniques are, in order to assign appropriate weights. Due to the nature of the implemented functions, this solution also assumes the problem to be linear.

The first layer (input) has a neuron for each matching module in the schedule, and implements a simple transfer function $y=\alpha x$ with α tuned by the training algorithm. The third layer (output) has one neuron with a sigmoidal transfer function and produces the aggregate similarity degree for candidate mappings. The interesting matter is the hidden layer structure: in our experiments we have tested three

different transfer functions (linear, sigmoid and arctan) and three error back-propagation algorithms: *on-line back-propagation* (OEBP), *batch back-propagation* (BEBP) and *resilient back-propagation* (REBP). The best results were obtained using a sigmoidal transfer function, chosen to amplify matching degrees grater than 0.5 and to dull the others, and using the BEBP algorithm with cross-validation. The number of neurons of this layer has been set to:

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

The learning rate and the momentum have been chosen by simulations (actually a learning rate of 0.25 and a momentum of 0.2). We use 85% of the samples to train the net, and the remaining 15% to perform cross-validation.

Note that, differently from other ontology matching/aligning tools, this neural network is not used to identify the matchings, but to approximate the optimal aggregation function.

One interesting problem is the construction of the training set needed to perform the estimation of the W function. It is constituted by a set of tuples, each containing the similarity values given by each module with respect to the same pair of resources. To perform the regression we need also the desired outcome of the aggregation function that is generated from a manually-aligned reference ontology. Two examples of tuples are shown in Table 1.

Res1	Res2	Jaro	WordNet	Struct.	Walk	**Desiderata**
Institution	Institute	0.86	0.67	1.00	0.67	**1.00**
collection	booktitle	0.55	0.00	0.00	none	**0.00**

Table 1. Example of Training Set Tuples

The so generated training set cannot be used "as-is", it has to be pre-processed [2]. The *cleaning process* removes: duplicate samples (i.e., same inputs and same desiderata), conflicting samples (i.e., same inputs but different desiderata), linearly dependent samples, 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).

Another problem is the numerical gap between positive and negative samples, respectively those with desired outcome equal to 1.0 and 0.0. Given two non-trivial ontologies, with n and m resources respectively, and r correct alignments expressed with k mapping relations, we can have $k \times (n \times m) - r$ negative samples with $r \ll k \times (n \times m)$. This gap may lead the regression algorithm to overfit the positive samples. A possible solution is adding to the training set all the positive samples and to sample the negative ones; we have seen that training sets with a positive/negative sample rate from 1:2 to 1:4 are balanced enough for the regression task.

When one of the matching modules does not produce a similarity value for a pair of resources, as shown in Table 1 for the Walk module, X-SOM repairs the missing information through an average of the values produced by modules of the same family. When acting with the conservative behavior, X-SOM can use the neural network also to learn from the user interaction. When the user corrects a tool proposal, the network trainer performs additional training steps until the result of the network agrees with the user. This behavior can be seen as a manual fine tuning of the aggregation function.

6 Inconsistency Resolution

The Inconsistency Resolution Subsystem has a modular architecture, supporting seamless integration of new checking algorithms. It takes the candidate mappings from the Mapping Subsystem and produces a set of mappings, in which the tool or the user have solved all the inconsistencies. Since the input ontologies are supposed to be consistent, consistency resolution is reduced to finding those mappings that introduce a contradiction into the final model. This problem is faced in X-SOM at two different levels: *standard consistency check* and what we have called *semantic consistency check*. Since X-SOM is designed to handle ontologies with an expressive power contained in that of the SHOIN($\mathbf{D}_n$) description logic, the standard consistency check is the process of testing if there exists an interpretation I that is a model for the T-BOX obtained once the mappings are applied.

To find the contradictions X-SOM applies a binomial search algorithm isolating the set of mappings responsible for the inconsistencies. When the tool acts with the semi-automatic behavior, the inconsistent mappings are submitted to the user who selects the correct ones. When acting with the fully-automatic behavior, X-SOM discards the conflicting mappings with lowest similarity degree until a non-conflicting set of mappings is found.
Since this process requires global T-BOX satisfiability check, which is a quite expensive task, we introduced some checks directly into the similarity map to discard evidently contradictory mappings (e.g. between declared disjoint concepts) to increase X-SOM scalability.

By Semantic Consistency Check we mean the process of verifying whether there are mappings that introduce into the model a *semantic contradiction* without introducing a logical contradiction into the T-BOX. To better explain the concept of semantic contradiction let us introduce the notion of *local entailment*: an entailment A $\sqsubseteq$ B is said to be local to an ontology O if it involves only resources of O. By semantic contradiction we mean the presence of one or more local entailments that were not enabled before applying the mappings.

An example of semantic contradiction is the cycle of subsumptions of Fig. 1. In this situation all the concepts belonging to the cycle are collapsed into a unique concept because, for any two concepts, it is always possible to infer that one is subclass of the other and vice-versa. The semantic inconsistency that arises is the local entailment O2:C $\sqsubseteq$ O2:B that was not possible in the original ontology.

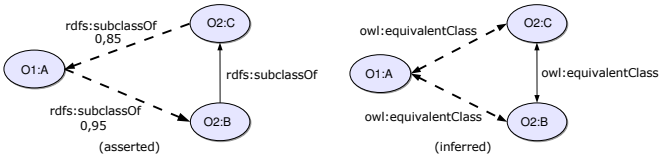

Figure 1. Cycle Semantic Contradiction

Notice that, while we can guarantee standard consistency using a reasoner SAT service, semantic consistency cannot be easily guaranteed. Notice also that a semantic inconsistency is not always an error: it may arise because of a partial knowledge of the domain by the designers of the input ontologies. In these situations X-SOM resorts to a set of heuristics or asks for the user intervention. In the situation described above, X-SOM cuts the cycle discarding the involved mapping with lowest similarity degree or, as in conservative behavior, submits the mappings involved into the cycle to the user that choses the correct one.

7 Experimental Results

X-SOM performance has been evaluated in terms of the precision and recall. The tests have been made on the systematic benchmark series proposed by the Ontology Alignment Evaluation Initiative campaign (OAEI 2006), which refers to the bibliography domain. The test campaign has been made with a non-competitive frame of mind: our purpose is to show X-SOM performance with a standard strategy and without any re-parameterization between different test cases. Such performance is shown, aggregated by test family, in Fig. 2.

Test family *L* includes two tests that map ontologies with different expressive powers (OWL-DL, OWL-Lite and OWL-Full[1]). It can be seen that a different expressive power does not affect the tool capabilities.

Family *RN* contains ontology pairs in which the names of resources and metadata are replaced by random ones. These are the most critical tests for X-SOM because the syntactical modules cannot help find the similarities, and the structural modules are fed with the results coming from the syntactical ones. The results show that the tool is not able to find any similarity between the ontologies. Pure structural mapping techniques, currently under development, will be added to X-SOM to overcome such limitation.

The *SY* family involves ontology pairs whose resources have names with syntactic variants like synonyms, different naming conventions, acronyms, different languages or simply misspelled words. XSOM's behavior is quite good: the most relevant difficulty is the identification of similarities between words and their acronyms due to different acronym construction strategies (e.g. dotted notation or upper-case characters) that mislead string-distance algorithms like Jaro and Levenshtein.

The last test family, named *R*, refers to real ontologies to be aligned. These are the MIT, UMBC, Karlshrue and INRIA bibliographic ontologies. For our tool this is the most relevant family, because it involves real ontologies that were not built to stress a particular matching algorithm, and reflects real ontology design patterns.

8 Conclusions

One of the most relevant results of the test campaign is the improvement in performance emerged when using a neural network instead of a linear or sigmoidal average function to aggregate the results of the Matching Subsystem. Fig. 2 shows that the recall measure increases by a 21% in average against the sigmoidal average and by a 37% against the linear average; the precision measure increases by a 10% and 16.5% respectively. The weights used with the linear and sigmoidal average have been established by linear regression on the same set of data used to train the neural network.

Another conclusion is that the confidence degree assigned to each matching technique by the neural network seems to be *not dependent of the particular application domain*. As said above, X-SOM has been tested using the OAEI benchmark test suite (bibliographic domain); For the test campaign we have chosen to train the neural network with datasets generated from the Networks, Animals and Computer Science ontologies pairs used by the I3CON Ontology Alignment Contest (I3CON 2004). Notice that none of them contains concepts or roles belonging to the bibliographic domain.

Tests showed a maximum variance of 6.8% in recall and of 3.6% in precision using the three different training sets. We believe that the neural network learns that a given combination of matching algorithms is better than another and mixes the matching proposals to maximize precision and recall. Some problems in establishing the optimal aggregation function emerge when we use techniques which results are statistically dependent (e.g. feeded modules). For this reason we performed a *stepwise regression* with Matlab to remove a-priori statistically irrelevant modules.

Obviously, the neural network can only approximate the

[1]Please notice that for OWL-Full ontologies the standard consistency check must be turned off, and only semantic consistency check is performed.

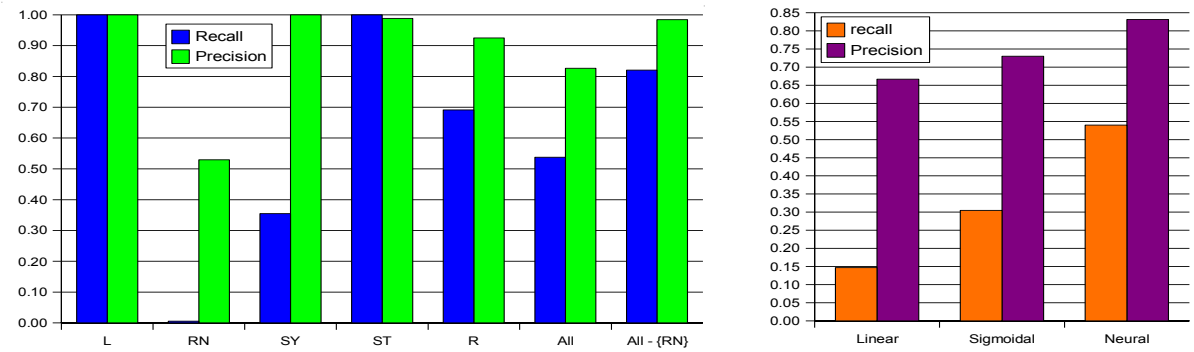

Figure 2. X-SOM: Precision and Recall Measures

ideal aggregation function and some wrong mappings survive even after the threshold application. The consistency checking process has been introduced to address this problem. It identifies and, in simple cases, automatically removes errors in the final mappings set thus improving the performances measures.

9 Future Developments

X-SOM is still a prototype and much work is still in order, in particular about its efficiency. We are studying to introduce greedy matching algorithms and early pruning techniques of bad matchings along with new matching modules based on bayesian networks, metadata inspection and pure structural analysis to increase the recall.

An interesting issue is the mappings expressiveness; we are working to increase the complexity of the mappings between resources, e.g. to associate a concept of one ontology with a SPARQL query over another ontology also known as GAV (Global as View) mappings.

The semantic consistency check is currently implemented as a set of heuristics; a formalization effort is ongoing to provide a more systematic and formal framework for semantic consistency analysis.

References

[1] D. Aumueller, H. 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*, pages 906–908, 2005.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1994.

[3] 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.

[4] 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.

[5] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. *Proc. of the IJCAI-2003 Workshop on Information on the Web*, 2003.

[6] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach. *Handbook on Ontologies in Information Systems*, pages 397–416, 2004.

[7] M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with apfel. *In Proc. of the 4th Int. Semantic Web Conf. (ISWC-2005).*, pages 1148–1149, 2005.

[8] M. Ehrig and Y. Sure. Ontology mapping: an integrated approach. *Proc. of the 1st European Semantic Web Symposium*, pages 76–91, 2004.

[9] M. Klein. Combining and relating ontologies: an analysis of problems and solutions. *Workshop on Ontologies and Information Sharing, IJCAI*, 2001.

[10] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. *Proc. of the 18th Nat. Conf. on AI*, 2002.

[11] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. *Proc. of the 17th Nat. Conf. on AI*, 2000.

[12] P. Mitra, N. F. Noy, and A. R. Jaiswal. Omen: A probabilistic ontology mapping tool. *Workshop on Meaning Coordination and Negotiation at ISWC-04*.

[13] N. F. Noy and M. A. Musen. The prompt suite: Interactive tools for ontology merging and mapping. *Int. Journal of Human-Computer Studies*, pages 983–1024, 2004.

[14] L. Predoiu, C. Feier, F. Scharffe, J. D. Bruijn, F. Martın-Recuerda, D. Manov, and M. Ehrig. State-of-the-art survey on ontology merging and aligning. *Institut AIFB, Universitat Karlsruhe, Tech. Rep*, 2005.

[15] A. Sánchez-Alberca, R. García-García, C. Sorzano, C. Gutiérrez-Cossío, M. Chagoyen, and M. F. López. Amon: A software system for automatic generation of ontology mappings.

[16] P. Visser, D. Jones, Bench-Capon, and Shave. An analysis of ontological mismatches: Heterogeneity versus interoperability. *AAAI Spring Symp. on Ontological Engineering, Stanford*, 1997.