SEMA: Results for the Ontology Alignment Contest OAEI 2007[1]

Vassilis Spiliopoulos[1, 2], Alexandros G. Valarakos[1], George A. Vouros[1], and Vangelis Karkaletsis[2]

[1] AI Lab, Information and Communication Systems Engineering Department, University of the Aegean, Samos, 83 200, Greece
{vspiliop, alexv, georgev}@aegean.gr
[2] Institution of Informatics and Telecommunications, NCSR "Demokritos", Greece
vangelis@iit.demokritos.gr

Abstract. In this paper we present SEMA tool for the automatic mapping of ontologies. The main purpose of SEMA is to locate one to one equivalence correspondences (mappings) between elements (i.e., classes and properties) of two input ontologies. Towards this goal, SEMA synthesizes lexical, semantic and structural matching algorithms through their iterative execution.

1 Presentation of the system

1.1 State, purpose, general statement

Ontologies have been realized as the key technology to shaping and exploiting information for the effective management of knowledge and for the evolution of the Semantic Web and its applications. In such a distributed setting, ontologies establish a common vocabulary for community members to interlink, combine, and communicate knowledge shaped through practice and interaction, binding the knowledge processes of creating, importing, capturing, retrieving, and using knowledge. However, it seems that there will always be more than one ontology even for the same domain. In such a setting, where different conceptualizations of the same domain exist, information services must effectively answer queries, bridging the gaps between conceptualizations of the same domain. Towards this target, networks of semantically related information must be created at-request. Therefore mapping of ontologies is a major challenge for bridging the gaps between agents (software and human) with different conceptualizations.

[1] This work is part of research project ONTOSUM (www.ontosum.org), implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%).

Tools for the automated mapping of ontologies have achieved remarkable results but still there is lot of space for improvements when dealing with real world ontologies. Building on our experience in participating in OAEI 2006 with AUTOMS [1], we intent to further increase the precision and recall of our matching methods, and further minimize the efficiency cost, by devising enhanced techniques and combinations of methods.

This paper presents the SEMA tool for the mapping of ontologies. SEMA is built on top of AUTOMS-F [2], which a framework implemented as a Java API, aiming to facilitate the rapid development of tools for the automatic mapping of ontologies. AUTOMS-F provides facilities for synthesizing individual ontology matching methods.

The main purpose of SEMA is to locate one to one equivalence correspondences (mappings) between the elements (i.e., classes and properties) of two input ontologies, by increasing the recall of the mapping process and achieving a fair balance between precision and recall. SEMA combines lexical, semantic and structural matching algorithms: A semantic matching method exploiting Latent Dirichlet Allocation model (LDA) [3], requiring no external resources, in combination with the lexical matcher COCLU (COmpression-based CLUstering) [4] and a matching method that exploits structural features of the ontologies by means of simple rules. This combination of approaches contributes towards automating the mapping process by exploiting lexical, structural and semantic features of the source ontologies, resulting to increased recall and precision. It must be emphasized that the aggregation of the mappings produced by the individual methods is performed through their iterative execution as described in [5, 6].

It must be pointed that the experience gained by participating in the OAEI contest helped us towards the following aspects: (i) We increased the precision and recall of SEMA by iteratively combining the individual matching methods, (ii) we improved AUTOMS-F framework by adding more facilities towards the synthesis of individual matching methods, (iii) we noticed the fact that tools such as SEMA tend to fail to notice subsumption relations between elements of distinct ontologies, since they assess only equivalences between them, and finally, (vi) we managed to improve the execution time of matching methods, such as the one based on LDA.

1.2 Specific techniques used

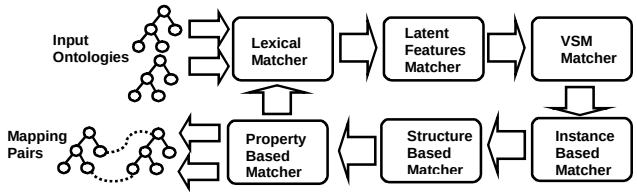

Fig. 1. Overview of SEMA.

SEMA combines six matching methods, executed in a predefined sequence, as depicted in Fig. 1. Each method in sequence exploits the results of the previous methods, aiming to find additional mapping element pairs. This policy is applied as

the competition is restricted to the discovery of one-to-one mappings, and in order to exploit the complementary nature of the combined methods. As already pointed, SEMA iteratively executes the overall mapping method, providing the mappings computed during an iteration as input to the next iteration, until no change in the matching pairs arises. In more detail, the ontology mapping problem is modeled as an iterative process that finds the most nearest reachable fixed point of a vector function, as presented in similar approaches in the literature ([5], [6] and [7]).

The following paragraphs present the matching methods in the order of their execution:

Lexical matcher: As said, SEMA uses a lexical matcher implementing the COCLU lexical similarity approach [4].

COCLU was originally proposed as a method for discovering typographic similarities between sequences of characters over an alphabet (ASCII or UTF character set), aiming to reveal the similarity of classes instances' lexicalizations during ontology population [4]. It is a partition-based clustering algorithm which divides data into clusters and searches the space of possible clusters using a greedy heuristic. Each cluster is represented by a model, rather than by the collection of data assigned to it. The cluster model is realized by a corresponding Huffman tree which is incrementally constructed as the algorithm dynamically generates and updates the clusters by processing one string (instance's surface appearance) at a time. The use of a model classifies the algorithm to the conceptual or model based learning algorithms. To decide whether a new string should be added in a cluster (and therefore, that it lexicalizes the same class/property as the other strings in the cluster do) the algorithm employs a score function that measures the compactness and homogeneity of a cluster. This score function, Cluster Code Difference (*CCDiff*), is defined as the difference of the summed length of the coded string tokens that are members of the cluster, and the length of the cluster when it is updated with the candidate string. This score function groups together strings that contain the same set of frequent characters according to the model of a cluster (e.g., Pentium III and PIII).

According to the above, COCLU takes as input two strings and returns their similarity. The local name, label or comment of an OWL class or property, considered as a string, belongs in a particular cluster when its *CCDiff* is below a specific threshold and it is the smallest between the *CCDiff's* of the given string and all existing clusters. Based on our experience with COCLU, the similarity threshold (ranging in [0,1]) was set to 0.986. A new cluster is created if the candidate string cannot be assigned to any of the existing clusters. As a result, it is possible to use the algorithm even when no initial clusters are available.

Matching pairs of ontology elements are generated according to the following rules:

1. A pair of ontology elements is a matching pair if the similarities of their local names, and labels, and comments are greater than the threshold value.
2. If two *ontology elements* do not match according to rule 1, then the elements are considered to match if the similarities of either their local names, or labels, or comments are greater than the threshold value.
3. If two *ontology elements* do not match according to rule 1 and rule 2, then their labels are substituted by their synonyms found in WordNet (in case *they have* WordNet entries) and rules 1 and 2 are repeated using the synonyms.

Latent Features Matcher: The second matching method applied is the semantic matching one detailed in [3]. This method aims at discovering and exploiting *latent features* that reveal the intended meaning of ontology elements. This is a contextual approach to the ontology mapping problem, where at first, ontology elements are transformed into vectors according to specific rules [3] that exploit elements' vicinity (e.g., labels, comments, instances, properties, super/sub elements, domain and range of properties etc.) with respect to the semantics of the specifications. The vectors' length corresponds to the number of the distinct words in both input ontologies and each entry holds the frequency of each word in the vicinity of the corresponding element. Exploiting this representation of ontology elements, the method computes *latent features* that express the intended meaning of ontology elements. This is done by applying the reverse generative process of the Latent Dirichlet Allocation (LDA) [8] model. Doing so, each element is represented as a distribution over *latent features*, and similarities between elements' pairs of the two ontologies is computed by means of the Kullback-Leibler divergence [9] measure. This measure estimates the divergence of distributions over latent features i.e., the divergence of elements' approximated intended meaning.

The major advantages of this approach are as follows: The use of latent features helps to deal with problems of imprecise and vague ontology elements' descriptions, as well as with cases of polysemy and synonymy. Also, the proposed approach does not presuppose the existence of any external resource, as it exploits words in the vicinity of ontology elements.

Vector Space Model (VSM) Matcher: The third method is a standard Vector Space Model [10] based technique where ontology elements are represented as vectors of weights. Each weight corresponds to a word and is being calculated using the TF/IDF measure. The similarity between two vectors is being computed by means of the cosine similarity measure. Element pairs with cosine similarity above a predefined threshold (0.2 in our experiments) are returned as matched pairs. The rules used for the extraction of words from ontology elements are the same as in the *latent features'* based matcher.

Instance Based Matcher: The fourth mapping method of SEMA is a lexical matching method exploiting the instances of classes. Specifically, two classes are considered to match if the percentage of their mapped instances is above a predefined threshold (10% in our experiments). Two instances match if the percentage of their matched properties is above a predefined threshold (10% in our experiments). Two properties of two distinct classes' instances match if their values are assessed to match by the COCLU lexical matcher.

Structural Based Matcher: The fifth method of SEMA is a structural matching method, which utilizes the mappings produced by the above described matching methods. According to this method, if two classes have at least a pair of matched super classes and a pair of matched sub class, then they are also considered to match.

Property Based Matcher: Similarly to the previous structural method, this one utilizes the properties' mappings produced by the other methods in order to locate new matching pairs of classes. Specifically, two classes are considered to match, if the percentage of their mapped properties is above a predefined threshold (90% in our experiments).

Iterative Execution: The above mentioned matching methods, performed in the specified sequence, compute matching pairs by taking into account the vicinity of each ontology element, as well as its features (local name, label and comments). The vicinity includes the elements directly related to this element, together with their features. However, performing iteratively, the vicinity of each element can be extended to include its matching element in the target ontology.

This introduces a recursive dependency, which as it is pointed in [6], requires non-standard computational means. This problem has been approached by Bisson [5] and Euzenat et al [6].

As it has been proposed in [6], given the recursive nature of these computations and aiming to compute the intended meaning of classes, we can still find the intended meaning of each class through an iterative process that finds the most nearest reachable fixed point of a vector function. Based on this work, SEMA, aiming to compute matching pairs of the input ontologies, performs an iterative computation as follows:

Repeat the following process until there is no change in the mapping pairs between the input ontologies.
1. For each element *E* do the following:
1.1. For each element in the vicinity of *E*
In case there is no mapping element associated to this element
compute the initial mapping based on its features.
1.2. Repeat the following until there is no change in the mapping computed for the element *E*
    1.2.1 Compute the mapping of *E* using the mappings of
        elements in its vicinity
    1.2.2 Re-compute the mappings of elements in its vicinity
        changing only the mapping for *E*

Performing iteratively, SEMA improves its precision, as it manages to propagate mappings to elements' vicinity, while also achieving a high degree of individual methods' combination: The results of each method feed the input of the other methods in the next iteration. In the current SEMA version the methods that contribute to the iterative computation are the VSM matcher, the property based matcher and the structure based matcher. The latent features matcher is executed only in the first iteration due to its requirements of high computational resources.

1.3 Adaptations made for the evaluation

Since SEMA is built on top of AUTOMS-F framework [2], as already mentioned above, there was no need for particular adaptations in order to run SEMA on the benchmark test cases and output the resulting mappings in the requested format. AUTOMS-F provides specific classes and methods for this purpose.

1.4 Link to the system and parameters file

http://iit.demokritos.gr/~vspiliop/SEMA.zip

1.5 Link to the set of provided alignments (in align format)

http://iit.demokritos.gr/~vspiliop/SEMA_results.zip

2 Results

Results produced by SEMA are grouped and discussed below. SEMA is implemented as a stand-alone Java programme and was executed on a dual core Ubuntu Linux PC (2 x 2.44 GHz cores, 1.5GB memory). Although, two cores where available no multi-threading mechanism was exploited.

2.1 Benchmark

2.1.1 Tests 101 to 104

In these test cases the target ontologies have no major differences (except test 102) from the common source ontology. All mapping pairs are produced by the COCLU lexical matching method. In 102, where the target ontology is irrelevant to the source ontology, no mappings are returned.

Tests	H-mean Precision	H-mean Recall
101-104	1.0	1.0

2.1.2 Tests 201 to 210

In these test cases ontology elements' features such as local names, labels and comments in the target ontologies have been changed in various ways (e.g., using uppercase letters, underscores, translating in foreign language, using synonyms, random strings or being suppressed). It is evident from the table below that the recall harmonic mean value drops significantly, comparing to the previous test category. This is due to tests 202, 209 and 210, where except from the fact that many source ontology elements' labels have been replaced by elements having completely different lexicalizations (foreign language, synonyms or random strings) in the target ontology, the comments are suppressed, limiting the common features of the input ontologies. However, even in test 202, where comments are suppressed and ontologies do not share much of lexical information, SEMA manages to achieve 74% precision and 30% recall. This is achieved by exploiting a small fragment of instances' common lexical information and by propagating similarity through the iterative execution of methods. However, as we will see in test cases 248-266 the propagation is not so effective and improvements should be considered.

It must be noticed that the exploitation of WordNet by the lexical matcher helps SEMA to perform 82% in terms of precision and 61% in terms of recall, even when no comments are available and labels are replaced by synonyms (test 209).

Tests	H-mean Precision	H-mean Recall
201-210	0.91	0.80

2.1.3 Tests 221 to 247

In these tests the changes made in the target ontology concern the hierarchy, the properties, the instances and the number of properties defined in classes. As we can see from the overall results, there is a minor decrease in terms of both precision and recall (comparing to test case 101). This performance is due mainly to test case 230 where SEMA performs 75% in terms of precision and 100% in terms of recall. The methods that introduce false mappings in this test (in order of negative influence) are the property based matcher, the VSM matcher and the latent features matcher. The first is due to the introduction of much more properties in the target ontology. The second is due to the noise introduced in the feature vectors representing classes and properties. More specifically, if elements different in meaning contain common words (resulting to similar representation vectors), then the VSM matcher may introduce false positives. Although, the latent features matcher is more tolerant to such noise, as it statistically generates latent features that focus on the significant statistical correlations between the elements of the two input ontologies, still it generates some false positives. It must be also pointed that as the latent features matcher is executed before the VSM matcher, it acts as a filter against such phenomena (since elements matched by the latent features matcher are not tested by the VSM matcher).

Tests	H-mean Precision	H-mean Recall
221-247	0.96	0.99

2.1.4 Tests 248 to 266

These are the most difficult tests of the benchmark track since local names, labels, and comments have been removed or replaced by random strings and only in some cases a fragment of the lexical features of instances is not altered from the target ontologies. SEMA relies totally on its instance based matcher to locate mappings between classes and properties. The iterative execution of SEMA does not manage to propagate similarity efficiently and restrain itself mainly in correcting some false positive mappings. By examining the raw results we observe that SEMA performs indifferently to the (non) existence of a class hierarchy. It must be pointed that the lexical matching method contributes only one mapping, i.e., the “lastName = lastName”. Concerning the recall values, they range in [25, 31], while the precision values range in [65, 100].

Tests	H-mean Precision	H-mean Recall
248-266	0.75	0.27

2.1.5 Tests 301 to 304

In the real world tests SEMA performs relatively satisfactory in terms of recall, but its precision definitely needs improvement. The main reason for the low precision is the low threshold value (0.2) of the VSM matcher. The threshold has been tuned in order to maximize the overall (101-304) precision and recall values. But this threshold is not the optimum if the evaluation is narrowed to tests 301-304. On the other hand, as

it has been explained, the latent features matcher has a better discriminating ability and introduces very few false positives.

Tests	H-mean Precision	H-mean Recall
301-304	0.67	0.79

2.2 Anatomy, Directory and Food

We were not able to run these tests due to technical problems in parsing these ontologies.

2.3 Conference

Several experiments were performed using various ontologies provided in the Conference track. The main lesson learned is that SEMA tends to "confuse" subsumption relations between elements of different ontologies with equivalence ones. For example, when mapping ekaw.owl to Conference.owl there where such cases, such as "Research_Topic=Topic" or "Assigned_Paper=Paper". This is mainly due to the low threshold value (0.2) of the Vector Space Model mapping method. On the other hand, in other ontology pairs (e.g. ekaw.owl – crs_dr.owl) there is no such a phenomenon: all mappings assessed to be equivalences are indeed equivalences.

3 General comments

3.1 Comments on the results

The major advantages of SEMA are:
(i) The extensive exploitation of all linguistic features of the input ontologies through different methods.
(ii) The aggregation of results and methods' combination via iteration, leading to the correction of false positive mappings of previous iterations and simultaneously performing propagation of similarities.
(iii) The use of latent features as a way to overcome the problems introduced by the phenomena of synonymy and polysemy, and finally,
(vi) The efficient and effective lexical similarity assessment performed by the COCLU lexical matcher.

The major weaknesses of SEMA are:
(i) Its dependence on threshold values. This leads SEMA to "confuse" subsumption relations with equivalence mappings (as noticed in the real world cases of the Consensus Workshop Track). This phenomenon has been also reported by others in the OAEI 2006 contest [11].
(ii) The inability of SEMA to parse large ontologies.

3.2 Discussions on the way to improve the proposed system

The ways to improve SEMA directly derive from its weaknesses, as follows:
(i) SEMA's parsing abilities must be enhanced in order to be tested in all tracks of the OAEI contest.
(ii) Advanced mapping policies must be specified, so as to exploit input ontologies' characteristics, deciding on the mapping methods to be applied. We believe that this will lead to an improvement of the precision values of SEMA.
(iii) We plan to introduce a matcher that locates subsumption relations [12] as a way to filter subsumption relations that are "confused" as equivalence relations.

3.3 Proposed new measures

We believe that systems should be able to locate not only equivalences between elements of distinct ontologies, but also other types of relations such as subsumption (inclusion) ($\sqsupseteq$ or $\sqsubseteq$), mismatch ($\perp$) and overlapping ($\sqcap$) and still be able to discriminate among them. This is of particular importance in the case of real world ontologies, where different conceptualizations make the complete alignment of ontologies very difficult. As a result, a new measure could be the number of false positive pairs assessed to belong in the equivalence relation, while belonging to an other relation (e.g., to the subsumption relation).

4 Conclusion

Our participation in the OAEI 2007 contest with SEMA tool has been a significant experience. We have actually been able to identify pros and cons of our tool, and improve several of its aspects. The organizers' feedback and the comparison with the other tools will also contribute to future improvements of the tool.

References

1. K. Kotis, et al. AUTOMS: Automating Ontology Mapping through Synthesis of Methods, OAEI 2006 contest, Ontology Matching International Workshop, USA, 2006
2. A. Valarakos, V. Spiliopoulos, K. Kotis, G. Vouros. AUTOMS-F: A Java Framework for Synthesizing Ontology Mapping Methods. I-KNOW 2007 Special Track on Knowledge Organization and Semantic Technologies 2007 (KOST '07) September 5, 2007, Graz
3. V. Spiliopoulos, G. Vouros, V. Karkaletsis. Mapping Ontologies Elements using Features in a Latent Space. The 2007 IEEE / WIC / ACM International Conference on Web Intelligence (WI 07), Silicon Valley, USA
4. Valarakos A., Paliouras A., Karkaletsis G., Vouros G. A name-Matching Algorithm for supporting Ontology Enrichment. In Proceedings of SETN'04, 3rd Hellenic Conference on Artificial Intelligence, Samos, Greece (2004)

5. Gilles Bisson. Learning in FOLwith similarity measure. In Proc. 10th American AAAI conference, San-jose (CA US), 1992
6. Jérôme Euzenat, Petko Valtchev, Similarity-based ontology alignment in OWL-Lite, Ramon López de Mantaras, Lorenza Saitta (eds), Proc. 16th european conference on artificial intelligence (ECAI), Valencia (ES), pp333-337, 2004
7. Vouros A. and Kotis K.: Extending HCONE-merge by approximating the intended interpretations of concepts iteratively. 2nd European Semantic Web Conference, Heraklion, Crete May 29 – June 1, 2005. Proceedings, Series: Lecture Notes in Computer Science, vol. 3532, Asunción Gómez-Pérez, Jérôme Euzenat (Eds.), Springer-Verlag
8. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993-1022
9. Lin, J. (1991). Divergence measures based on Shannon entropy. IEEE Transactions on Information Theory, 37(14), 145–51
10. Raghavan, V.V. and Wong, S.K.M. A Critical Analysis of Vector Space Model for Information Retrieval. JASIS 37(5) 1986, 279-287
11. O. Svab, V. Svatek, H. Stuckenschmidt: A Study in Empirical and 'Casuistic' Analysis of Ontology Mapping Results. In Proceedings of ESWC, 2007
12. V. Spiliopoulos, A. Valarakos, G. Vouros, V. Karkaletsis: Learning Subsumption Relations with *CSR*: A Classification-based Method for the Alignment of Ontologies. The Second International Workshop on Ontology Matching, Korea, 2007 (accepted as poster)

Appendix: Raw results

Matrix of results

#	Name	Prec.	Rec.	Time (sec)
101	Reference alignment	1	1	10
102	Irrelevat ontology	NaN	NaN	36
103	Language generalization	1	1	9
104	Language restriction	1	1	8
201	No names	0,92	0,98	9
202	No names, no comments	0,74	0,3	11
203	No comments	1	1	5
204	Naming conventions	0,95	0,96	8
205	Synonyms	0,93	0,96	9
206	Translation	0,94	0,97	11
207		0,92	0,97	11
208		0,89	0,8	6
209		0,82	0,61	9
210		0,81	0,47	10
221	No specialisation	1	1	8
222	Flatenned hierachy	0,96	1	9
223	Expanded hierarchy	0,97	0,98	10
224	No instance	1	1	7
225	No restrictions	1	1	8

228	No properties	1	1	16
230	Flatenned classes	0,75	1	8
231		1	1	8
232		1	1	7
233		1	1	16
236		1	1	12
237		0,96	1	6
238		0,97	0,97	9
239		0,94	1	16
240		1	1	22
241		1	1	12
246		0,94	1	12
247		0,94	0,94	12
248		0,73	0,3	11
249		0,73	0,28	12
250		1	0,27	14
251		0,65	0,26	11
252		0,65	0,25	11
253		0,71	0,28	11
254		1	0,27	15
257		1	0,27	14
258		0,66	0,25	12
259		0,68	0,26	14
260		1	0,31	14
261		1	0,27	18
262		1	0,27	14
265		1	0,31	14
266		1	0,27	18
301	BibTeX/MIT	0,7	0,75	8
302	BibTeX/UMBC	0,62	0,6	5
303	Karlsruhe	0,55	0,8	9
304	INRIA	0,76	0,93	6