CODI: Combinatorial Optimization for Data Integration – Results for OAEI 2011

Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke

KR & KM Research Group
University of Mannheim, Germany
{jahuber, tsztyler}@mail.uni-mannheim.de
{jan, christian}@informatik.uni-mannheim.de

Abstract. In this paper, we describe our probabilistic-logical alignment system CODI (Combinatorial Optimization for Data Integration). The system provides a declarative framework for the alignment of individuals, concepts, and properties of two heterogeneous ontologies. CODI leverages both logical schema information and lexical similarity measures with a well-defined semantics for A-Box and T-Box matching. The alignments are computed by solving corresponding combinatorial optimization problems.

1 Presentation of the system

1.1 State, purpose, general statement

CODI (**C**ombinatorial **O**ptimization for **D**ata **I**ntegration) leverages terminological structure for ontology matching. The current implementation produces mappings between concepts, properties, and individuals. The system combines lexical similarity measures with schema information to completely avoid *incoherence* and *inconsistency* during the alignment process. CODI participates in 2011 for the second time in an OAEI campaign. Thus, we put a special focus on differences compared to the previous 2010 version of CODI.

1.2 Specific techniques used

CODI is based on the syntax and semantics of Markov logic [2] and transforms the alignment problem to a maximum-a-posteriori optimization problem. This problem needs a-priori confidence values for each matching hypotheses as input. Therefore, we implemented an aggregation method of different similarity measures. Another new feature of CODI is the recognition of ontology pairs belonging to different versions of the same ontology. In instance matching CODI does not compute lexical similarities for all existing pairs of instances but utilizes object-property assertions for reducing the necessary comparisons.

Markov Logic Framework Markov logic combines first-order logic and undirected probabilistic graphical models [11]. A Markov logic network (MLN) is a set of first-order formulae with weights. Intuitively, the more evidence there is that a formula is true the higher the weight of this formula. It has been proposed as a possible approach to several problems occurring in the context of the semantic web [2]. We have shown that Markov logic provides a suitable framework for ontology matching as it captures both *hard* logical axioms and *soft* uncertain statements about potential correspondences between entities. The probabilistic-logical framework we propose for ontology matching essentially adapts the syntax and semantics of Markov logic. However, we always *type* predicates and we require a strict distinction between *hard* and *soft* formulae as well as *hidden* and *observable* predicates. Given a set of constants (the classes and object properties of the ontologies), a set of formulae (the axioms holding between the objects and classes), and confidence values for correspondences, a Markov logic network defines a probability distribution over possible alignments. We refer the reader to [8, 7] for an in-depth discussion of the approach and some computational challenges. For generating the Marcov logic networks we used the approach described in [12]. Our OAEI paper from last year contains a more technical description of the framework [9].

Cardinality Constraints A method often applied in real-world scenarios is the selection of a functional one-to-one alignment [1]. Within the ML framework, we can include a set of hard cardinality constraints, restricting the alignment to be functional and one-to-one.

Coherence Constraints Incoherence occurs when axioms in ontologies lead to logical contradictions. Clearly, it is desirable to avoid incoherence during the alignment process. All existing approaches that put a focus on alignment coherence remove correspondences after computing the alignment. Within the ML framework we can incorporate incoherence reducing constraints *during* the alignment process.

Stability Constraints Several approaches to ontology matching propagate alignment evidence derived from structural relationships between concepts and properties. These methods leverage the fact that existing evidence for the equivalence of concepts C and D also makes it more likely that, for example, child concepts of C and D are equivalent. One such approach to evidence propagation is *similarity flooding* [6]. As a reciprocal idea, the general notion of stability was introduced, expressing that an alignment should not introduce new structural knowledge [5].

Combination of Different Similarity Measures Compared to last year we improved our lexical string similarity measures significantly. In a first step we collect and standardize all string information like ids, labels and annotations from the entities. During the standardization process we split tokens into separate words if necessary (e.g. *hasAuthor* is transformed to *has Author*), replace special characters with spaces, and remove few words like *a* or *the* according to a stop-words list.

Furthermore, the functionality of computing string similarities has been improved. CODI is able to combine several string similarity measures by taking the average, the maximum or by weighting each measure with a specific predefined weight. These weights could be learned with machine learning algorithms. In the standard configuration CODI combines the Cosine, Levenshtein, Jaro Winkler, Simth Waterman Goto, Overlap coefficient, and Jaccard similarity measures[1] with specific weights.

Matching different Ontology Versions A specific task in ontology matching is the alignment of different versions of the same ontology. The test cases of the benchmark track can be seen as an example for this kind of task. In the following we argue that (a) matching versions requires a different approach compared to a standard matching task, and (b) that, therefore, it is required to detect automatically that two ontologies are different versions of the same ontology.

(a) Suppose that $\mathcal{O}$ and $\mathcal{O}'$ are versions of the same ontology. Further, let $\mathcal{O}$ contain less concepts and properties than $\mathcal{O}'$. Then it is highly probable that many or nearly all entities in $\mathcal{O}$ have a counterpart in $\mathcal{O}'$. A good one-to-one alignment will have, thus, as many correspondences as there are entities in $\mathcal{O}$. Based on this assumption it makes sense to lower the threshold or to use a structural measure in addition to the computation of string-based similarities. In particular, we apply the following measure.

We first calculate the number of subclasses $\#sub$, superclasses $\#sup$, disjoint classes $\#dis$, and domain- and range-restrictions ($\#dom$ and $\#ran$) for a specific concept C. These results are then used to calculate a similarity. For example, given $C \in \mathcal{O}$ and $D \in \mathcal{O}'$ we have $sim_{\#sub}(C, D) = (1 + min(\#sub(C), \#sub(D)))/(1 + max(\#sub(C), \#sub(D)))$. The overall similarity $sim(C, D)$ is then computed as weighted average over all different similarity values for each of $\#sub$, $\#sup$, $\#dis$, $\#dom$, $\#ran$.

The resulting similarity measure is highly imprecise, but has a high recall if we apply it to two ontologies with high structural similarity. Whenever there is a high probability that the two input ontologies are versions of the same ontology, we add for each concept C the top-k counterparts D with respect to $sim(C, D)$ as matching hypotheses with low confidence to the optimization problem (same for properties). This approach sounds quite drastic, but keep in mind that there are anchor-correspondences generated by our string-based measures and constraints that interact and result in a meaningful final alignment.

(b) In order to determine whether two ontologies are versions of each other, we apply the Hungarian method on the input generated by our structural measure. The Hungarian method finds an optimal one-to-one alignment $\mathcal{A}_{opt}$. Now suppose that we match an ontology on itself. The number of correspondences in $\mathcal{A}_{opt}$ is then equal to the number of entities in the ontology, i.e., $\mathcal{A}_{opt}$ has a full coverage. Moreover, the total of confidences $\sum_{c \in \mathcal{A}_{opt}} conf(c)$ will be $|\mathcal{A}_{opt}|$. In general, we assume that $\sum_{c \in \mathcal{A}_{opt}} conf(c)$ divided by the size of the smaller ontology is close to 1 for versions of the same ontology. In particular, we treat each pair of ontologies as versions if the measured value is above 0.9.

[1] Implemented in http://sourceforge.net/projects/simmetrics/.

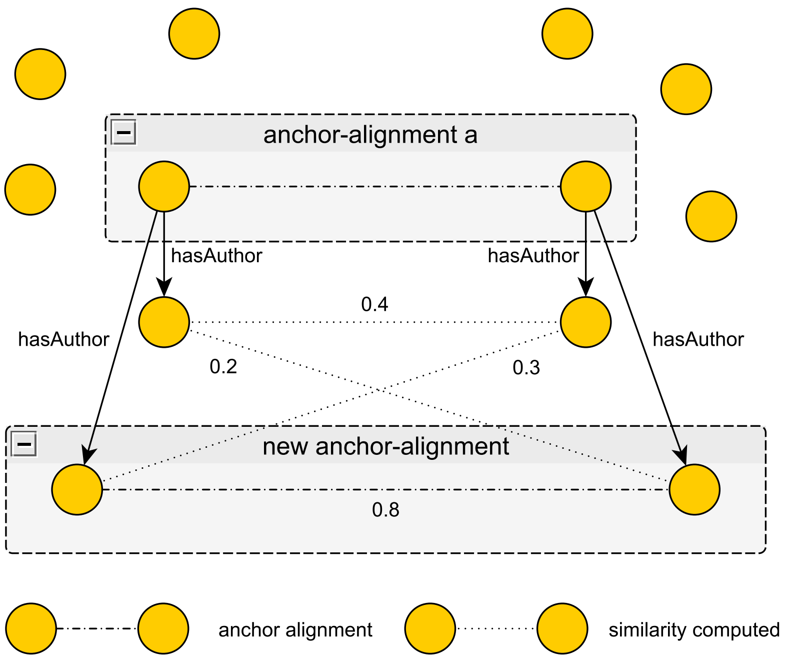

Fig. 1. Process of Selecting Individuals for Computing their Lexical Similarities with $thres = 0.7$.

Instance Matching In real-world instance matching tasks we are often faced with data sources containing a large amount of instances. Hence, it is obvious that computing the lexical similarity for every pair of these instances is not suitable. We implemented an approach which utilizes object-properties to determine the instances for which the similarity should be computed. Our approach assumes that we have one common TBox and two different ABoxes. Consequently, we assume that both TBoxes have been integrated beforehand.

In a first step we compute *anchor*-alignments. Therefore, we compare a small subset of all individuals with each other (e.g. all individuals which are asserted to a specific concept like $Film$), compute their lexical similarities $lexSim$, and add those to the anchor-alignments if their respective similarities are above a threshold $thres$. Then, we take the first anchor-alignment a. For all individuals which are connected with an object-property-assertion with one of the individuals in the alignment a we again compute the lexical similarity $lexSim$. We add them to the *end* of the anchor-alignments if $lexSim$ is higher than the threshold $thres$. Figure 1 visualizes this process. The anchor-alignments is a unique set, which means that only new alignments are added. We repeat this procedure for the second, third, and all following anchor-alignments until we went through the whole set.

The lexical similarity $lexSim$ is computed as described in [9]. However, we integrated coherence checks as proposed by [10] in order to avoid inconsistent alignments. Comparisons can be further reduced, by omitting those individual pairs which have no asserted inferred concept in common.

This basic idea is extended by some post-processing steps. For catching correspondences which are not connected with an object-property-assertion, we compare all remaining individuals which do not yet occur in the anchor-alignment and add them if their lexical similarity $lexSim$ is above $thres$. At the end, a greedy algorithm for computing a one-to-one alignment is applied.

These techniques reduce the runtime significantly on large instance-matching benchmarks.

1.3 Adaptations made for the evaluation

Prior to each matching task, CODI automatically analyzes the input ontologies and adapts itself to the matching task. The first distinction is based on the use of OBO constructs. If this is the case CODI automatically switches to a setting optimized for matching biomedical terms. The main difference in this setting is the use of a different similarity measure which exploits the fact that in medical domains the order of words is often transposed. The measure basically splits the two strings in two sets of words and computes the largest common subset of these sets relative to the smaller one.

If this is not the case CODI checks if the ontologies might be versions of the same ontology. This test does not always correctly discriminate and we sometimes do not detect that two ontologies are different version of the same ontology resulting in poor performance for some of the benchmark test cases.

1.4 Link to the System

CODI can be downloaded from the SEALS portal via `http://www.seals-project.eu/tool-services/browse-tools`.

1.5 Link to the Set of Provided Alignments

The alignments for the tracks *Benchmark*, *Conference*, and *Anatomy* has been created on top of the SEALS platform. For *IIMB* the alignments can be found at `http://code.google.com/p/codi-matcher/downloads/list`

2 Results

Benchmark Track The benchmark track is constructed by applying controlled transformations on one source ontology. Thus, all test-cases consist of different versions of the same ontology. However, our *adaptive* method for detecting these ontologies only categorize about 50 % beeing different versions of each other. Especially if their semantic structure is heavily changed (e.g. deleting class hierarchy, etc.) our algorithm fails. Nevertheless, with our adaptive method we were able to improve our F_1 score from 0.51 to 0.75 compared to last year. If all test-cases would have been *correctly* categorized as different versions CODI's F_1 score would have been 0.83 which is 32 % higher than last year. For the newly introduced dataset 2 our adaptive setting even produces a slightly higher F_1 score of 0.70 compared to the correct assignments. Thus, the structure of some test cases differs so much that it is beneficial to consider them *not* as ontologies of the same version (even if they are). The results are shown in Table 1.

Table 1. Benchmark results

	Dataset 1			Dataset 2	
	2011		2010	2011	
	adaptive	correct		adaptive	correct
Precision	0.88	0.90	0.72	0.86	0.80
Recall	0.65	0.77	0.44	0.59	0.61
F_1 score	0.75	0.83	0.51	0.70	0.69

Conference Track Since the conference dataset contains many trivial correspondences matchers can easily reach a high precision. The challenge of this dataset consists in finding the non-trivial correspondences. Concentrating on these non-trivial correspondences we were able to increase our recall from 0.51 to 0.61 compared to the results of last year and gained 2 % additional F_1 score. In the conference track CODI was able to detect that all ontology pairs are not versions of the same ontology. Consequently, the adaptive and the correctly assigned results are similar (see Table 2). We also made some experiments where we matched the Conference ontologies with the fixed version-setting. We observed a significant loss in precision. This illustrates the importance of an adaptive approach.

Table 2. Conference results

	2011		2010
	adaptive	correct	
Precision	0.75	0.75	0.87
Recall	0.61	0.61	0.51
F_1 score	0.66	0.66	0.64

Anatomy Track Due to our special lexical similarity measure for medical ontologies, we were able to improve our F_1 score of last year from 0.794 to 0.879. Currently, our results are better than the best participating system of the OAEI 2010. CODI requires approximately 35min to finish this matching task on a 2.3GHz dual core machine with 8G RAM.

Table 3. Anatomy results

	2011	2010
Precision	0.955	0.954
Recall	0.815	0.680
F_1 score	0.879	0.794

IIMB Track The IIMB benchmark is created by applying lexical, semantical, and structural transformation techniques on real data extracted from freebase [3]. The transformations are divided into four transformation categories containing 20 transformations each. The size of the IIMB track heavily increased compared to last year. Each of the 80 existing transformations consist of ontology files with sizes larger than 20 MB. For computing a very basic string similarity for every pair of individuals the runtime explodes to over one hour per test case. With our new instance matching method which only compares related individuals we were able to reduce the runtime to 34 minutes per test-case in average. This runtime includes the time for consistency checking, for computing a functional one-to-one alignment, and for calculating a more sophisticated lexical similarity.

Beside the increase in size, the transformations have been made much harder. Thus, comparisons to last year results are not expedient. Table 4 summarizes the different results of the CODI system for each of the 4 transformation categories[2].

Table 4. IIMB results

Transformations	0-20	21-40	41-60	61-80	overall
Precision	0.93	0.83	0.73	0.66	0.79
Recall	0.78	0.59	0.67	0.28	0.63
F_1 score	0.84	0.68	0.64	0.36	0.66

3 General comments

3.1 Discussions on the way to improve the proposed system

Improvements in usability by designing a suitable user interface are future steps that have to be taken. Although we focussed this year on the implementation and evaluation of a combination of more sophisticated lexical similarity measures, we think that we still have not exploit CODIs full potential regarding this issue. Last but not least improvements in matching different ontology versions will be subject of next years participation.

3.2 Comments on the OAEI 2011 procedure

The SEALS evaluation campaign is very beneficial since it is the first time that the matchers are publically available for download implementing a common interface.

3.3 Comments on the OAEI 2011 measures

We encourage the organizers to use semantic precision and recall measures as described in [4].

[2] In several test cases every supplementary information for individuals has been deleted. These test cases will not be considered in the official OAEI evaluation and, thus, are omitted here.

4 Conclusion

This year we improved the lexical similarity measures and developed a methodology for automatically choosing between different settings. Combining these improvements with our Markov logic system from last year, we were able to improve our results for the anatomy, conference, and benchmark track significantly. Furthermore, we developed a new instance matching algorithm, which only computes the similarity of promising instances. With this technique we were able to reduce the runtime of the large instance matching benchmark.

The strength of the CODI system is the combination of lexical and structural information and the declarative nature that allows easy experimentation. We will continue the development of the CODI system and hope that our approach inspires other researchers to leverage terminological structure and logical reasoning for ontology matching.

References

1. I. Cruz, F. Palandri, Antonelli, and C. Stroe. Efficient selection of mappings and automatic quality-driven combination of matching methods. In *Proceedings of the ISWC 2009 Workshop on Ontology Matching*, 2009.
2. P. Domingos, D. Lowd, S. Kok, H. Poon, M. Richardson, and P. Singla. Just add weights: Markov logic for the semantic web. In *Proceedings of the Workshop on Uncertain Reasoning for the Semantic Web*, pages 1–25, 2008.
3. A. Ferrara, S. Montanelli, J. Noessner, and H. Stuckenschmidt. Benchmarking matching applications on the semantic web. In *The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011*, Lecture Notes in Computer Science, pages 108–122, Heraklion, Crete, Greece, 2011. Springer.
4. D. Fleischhacker and H. Stuckenschmidt. A Practical Implementation of Semantic Precision and Recall. In *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 986–991. IEEE, 2010.
5. C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *Proceedings of the Workshop on Ontology Matching*, Busan, Korea, 2007.
6. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of ICDE*, pages 117–128, 2002.
7. M. Niepert. A Delayed Column Generation Strategy for Exact k-Bounded MAP Inference in Markov Logic Networks. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2010.
8. M. Niepert, C. Meilicke, and H. Stuckenschmidt. A Probabilistic-Logical Framework for Ontology Matching. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.
9. J. Noessner and M. Niepert. Codi: Combinatorial optimization for data integration–results for oaei 2010. *Ontology Matching*, page 142, 2010.
10. J. Noessner, M. Niepert, C. Meilicke, and H. Stuckenschmidt. Leveraging Terminological Structure for Object Reconciliation. *The Semantic Web: Research and Applications*, pages 334–348, 2010.
11. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
12. S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 468–475, 2008.