Adaptive Semantic Integration

Marc Ehrig and York Sure

Institute AIFB, University of Karlsruhe, Germany
{ehrig,sure}@aifb.uni-karlsruhe.de

Abstract

Schema integration has become a task of increasing importance. In this paper we give a guide on how to automatically optimally set the parameters of semantic alignment algorithms respectively tools for a wide range of use cases. We basically define a utility function to rate the results of alignment algorithms. Then we backtrack from the utility function first to concrete requirements for the results and then to the parameters of alignment algorithms. The gains of our methodology are confirmed by its implementation and evaluation.

1 Introduction

Semantic integration has become a task of increasing importance. It is a necessary precondition to establish interoperation between agents or services using different schemas. Thus, many approaches have been brought up to align, map, integrate and merge data schemas, ontologies, etc. [1]. In this paper we will concentrate on semantically rich schemas as represented through ontologies. Thus, we can make use of more information than simple schemas would generally allow. Similarly as for the schema integration domain for ontology alignment special tools such as GLUE, PROMPT, etc. [5, 11] have been created but normally focus on solving exactly one task, thus making it rather difficult to apply the tools for already slightly different use cases. However, in bigger projects such as SEKT we face exactly the problem that we have different use cases to address and at the same time aim for one integrated alignment framework [3].

In this paper we will present a methodology for deriving a strategy for ontology alignment so that it can be optimally applied for a wide range of alignment problems. In specific, we will give a guide on how to automatically set the parameters of ontology alignment algorithms respectively tools based on underlying use cases such as ontology merging or query rewriting. This will be done by defining a utility function, which we then are going to maximize. Through this methodology it will be possible to reuse very good specialized alignment algorithms not only for one use case, but for a variety of them, thus increasing the impact of the ontology alignment technology. As our methodology will focus on a general alignment process rather than a specific tool, we expect the results of this paper to be applied to other existing schema integration approaches as well.

We start this paper with an overview of our adaptive methodology in Section 2. Its main steps will then be presented separately thereafter: creation of a utility function (Section 3), deriving the requirements for result dimensions (Section 4), and finally deriving the algorithm parameters (Section 5). The methodology of dynamically setting the parameters of ontology alignment algorithms has been prototypical implemented and evaluated in our tool FOAM (framework for ontology alignment and mapping) in Section 6. The paper closes with related work, an outlook, and the conclusion.

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

2 Overview

With this paper we aim to present a methodology for optimally adapting ontology alignment algorithms to different use cases. Identifying the actual goal of the alignment

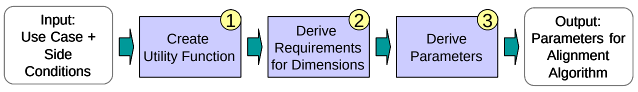
Input: Use Case + Side Conditions
Create Utility Function
1
Derive Requirements for Dimensions
2
Derive Parameters
3
Output: Parameters for Alignment Algorithm

Figure 1: Overview of Methodology

process is one input we need. Further we require the specification of different side conditions such as the ontologies to align as well as the computational infrastructure. In the understanding of this paper an ontology consists of both schema and instantiating data. Common languages to represent ontologies are RDF(S)[1] or OWL[2]. Semantically an alignment returns two entities linked by an identity relation. In a first step we create a utility function to rate retrieved alignments. Obviously we want to maximize the

[1]http://www.w3.org/TR/rdf-schema/
[2]http://www.w3.org/TR/owl-ref/

utility function. Through this second step, the optimization, we derive requirements for core dimensions of alignments such as quality or efficiency. This also means considering the use case, as well as the side conditions. And in a third step we derive the actual parameters based on the just mentioned requirements. The final output consists of the parameters which have to be assigned to an alignment algorithm such as our framework for ontology alignment and mapping (FOAM)[3]. Only then the alignment process is executed leading to the presumably best possible results for the given use case.

3 Create Utility Function

3.1 Example

Throughout the paper we will illustrate our methodology with a practical running example. In each section we will thereafter extend the example by more general considerations. If the reader only would like to get the basic idea, he may skip the detailed descriptions and continue with the example in the subsequent section.

Let us assume we have two OWL-ontologies with about 300 entities each describing the biological domain of animals. Our user wants to merge the two ontologies to receive one integrated view of the animal world. For this he has a powerful tool for ontology alignment at hand, but he is not an ontology technology expert himself. He wants to use the optimal strategy given the ontologies and his use case, but does not know how to set the individual parameters of the algorithm. He wants to maximize the overall *utility value*.

3.2 Utility of Ontology Alignment

Ontology alignment is done with a specific goal in mind. Whether this goal has been reached can be quantified through a utility function. An ontology alignment algorithm has certain parameters p which have to be set in beforehand. When using the parameterized algorithm on two ontologies we receive concrete alignment results. From these we can calculate the evaluation values r for different dimensions such as quality or efficiency. Further, there are side constraints originating from the involved ontologies, such as their size or format, and computing infrastructure s. And finally the ontologies o_1 and o_2 themselves effect the results. To overall evaluate the results we calculate a utility u. We argue that the utility is based on preferences for each of the result dimensions and that the preferences themselves are dependent on the underlying use cases c extended by a subjective human preference m. The utility u may also be written as function h.

$$r_i = f_i(p_1, \dots, p_m, s_1, \dots, s_k, o_1, o_2)$$

$$u = g(c, m, r_1, \dots, r_m)$$

$$u = h(c, m, p_1, \dots, p_m, s_1, \dots, s_k, o_1, o_2)$$

[3]http://www.aifb.uni-karlsruhe.de/WBS/meh/foam

3.3 Methodology

Goal of our adaptive methodology is to maximize the utility $\hat{u}$ by setting the algorithm parameters correctly $\hat{p}$. We assume that the use case is given and fixed, just like the ontologies and computing equipment.

$$\hat{u} = h(\hat{p_1}, \dots, \hat{p_m})$$

$$\forall u : \hat{u} \geq u, with\, c, s_1, \dots, s_k\, fixed; m, o_1, o_2\, abstracted$$

In the next sections we backtrack the utility calculation. As we want to do the optimization before doing the actual alignment, we rely on estimate functions for f_i and g. For the prediction we abstract from specific variables and head for a generic approach. We are looking for optimal results for typical ontologies and an average user, and may therefore ignore the unknown and subjective parameters o_1, o_2 and m.

4 Derive Requirements

4.1 Example

We will now have a look at the *use case*: ontology merging is a common use case in business applications. Ontologies of individual persons or departments may be merged to create a larger potentially company wide understanding [11]. This is one promising approach for actually engineering ontologies in a distributed manner [12]. In our example ontologies are merely schema information. If the original ontologies e.g. represent database schemas, the merged ontology can be the basis for further integration steps such as data integration. In general the source ontologies would disappear and only the merged ontology remains. To ensure a good final ontology, quality requirements are high, whereas time resources tend to be less critical, as the merging process can be run in the background. Further, ontology merging will normally include a human in the loop, who finally checks the merged ontologies for correctness.

From the use case description we have identified four main *result dimensions* r influencing the utility u. Quality i.e. precision and recall have to be high, time should be of medium duration, and semi-automation is the ideal level of user interaction.

Unfortunately, there are further *side conditions* s which also effect the alignment results. Therefore, to ensure the utility still reaches a maximum certain constraints are implied to the requirements. In the running example we face the following critical side conditions: the ontologies do not contain a lot of instances. Further, the ontologies have a rather small overlap. Sparse alignments will require a more thorough search. Fortunately, these critical side conditions do not further constrain the requirements.

From the example we have derived the final *requirements* in Table 1.

4.2 Dimensions

We identified the following main dimensions.

Input	Value	Prec/Rec	Time	Autom.
Use Case	Merging	high/high	medium	semi-
Content	more schema	high/med.		semi-
Overlap	small	high/high	medium	semi-
Final		high/high	medium	semi-

Table 1: Requirements

Quality: There are two values measuring quality of the alignment process. The precision measures the fraction of found alignments which are actually correct: $precision = \frac{\#correct\_found\_mapping}{\#found\_mappings}$. Typically precision is balanced against another measure from information retrieval. Recall measures the fraction of alignments found in comparison with the total number of existing alignments: $recall = \frac{\#correct\_found\_mappings}{\#existing\_mappings}$. Obviously a very high recall will entail also many false alignments (a lower precision) and vice versa. Often precision and recall are balanced against each other with the so called f-measure.

Time: Time may be a critical factor for many applications. The requirements might range from instantaneous response to overnight batch processing. Most existing tools do not deal with this factor explicitly.

Level of Automation: Most algorithms require interaction of the human knowledge engineer. However, for some use cases user interaction is not desirable or even possible. Different levels of automation range from manual alignment, proposal generation, conflict resolution, to fully automatic.

4.3 Use Cases and Corresponding Requirements

Semantic integration has been a topic of research in many projects such as SWAP [13][4], SEKT [3][5], KnowledgeWeb[6], or DIP[7]. Based on the requirements analysis from these projects we derived a set of more general use cases which need ontology alignment. We also mention typical preferences of each use case with respect to the just described dimensions. The use cases were carefully selected to represent a wide range of applications, but we do not claim to be complete. Real world applications will be a combination of these extreme cases.

Query and Answer Translation: An operation occurring very frequently in knowledge management applications is querying of information sources. For this task users formulate a query in a certain query language, based on one ontology, normally the individual's or community's ontology. This query is sent to a reasoner, possibly it is also passed across the network to other computers [13]. This reasoner then returns an answer fitting the query. We want to allow an application to query different heterogeneous information sources without actually caring about all the different ontological representations. In order to achieve this, a query written in terms of the application's ontology, needs to be rewritten using the terms in the target source's ontology. Rewriting means to translate the schema information as well as the instance information included in the query during runtime. A user being used to internet search-engines will presumably be tolerant towards wrong results, as long as the correct results are returned.

Ontology Evolution / Versioning: Ontologies are not static objects, they evolve, they develop over time. Keeping track of these different versions is a key challenge. Business processes should seamlessly continue with new versions. Ontology alignment is necessary to link the different versions of the evolving ontologies. The new schema needs to replace the old schema. This also means that the classification of instances has to be changed, further attributes may also have to be altered. Quality should generally be high.

Data Integration: Data Integration is concerned with the use of data from different sources within one application. The data from the different sources needs to be presented to the user in a unified way [8]. For applications equal instances should not be distinguishable, even if they originated from different underlying repositories. Most times this can be done in the background without strict time requirements, but, due to the assumed high amount of instance information it will have to be done more or less automatically.

Reasoning: Another use case heavily dependent on ontology alignment is reasoning in a global setting such as envisioned for the Semantic Web. New information is inferred from distributed and heterogeneous ontologies. In a distributed web of ontologies errors and inconsistencies cannot be eliminated completely. This will certainly also have effects on the alignment of different ontologies. The probably biggest tasks are requirements on quality. Through inferencing wrong alignments can quickly result in further wrong results, which again trigger additional wrong results in a cascading manner.

Based on the use cases we determined the requirements r for a maximum utility (see Table 2).

Use Case	Prec/Rec	Time	Automation
Query Translation	med./high	fast	full-
Ontology Merging	high/high	medium	semi-
Ontology Evolution	high/high	medium	semi-
Data Integration	med./med.	medium	semi-
Reasoning	high/med.	slow	full-

Table 2: Result Requirements based on Use Cases

4.4 Constraints on Requirements based on Side Conditions

The next step was to check whether side conditions further restrict these requirements. They are based on two areas: the ontologies to be aligned, and the computational infrastructure. Again, we do not claim this is complete, but try to cover various aspects. Only the constraining side conditions are listed in Table 3.

[4]http://swap.semanticweb.org/
[5]http://sekt-project.org/
[6]http://knowledgeweb.semanticweb.org/
[7]http://dip.semanticweb.org/

Side Condition	Value	Prec/Rec	Time	Autom.
Ontology Size	large		fast	full-
Ontology Complexity	low	high/high		
	high		fast	
Content	more schema	high/med.		semi-
	more instances	med./high	fast	full-
Overlap	small	high/high		semi-
Computational Resources	low		fast	

Table 3: Constraints on Result Requirements based on Side Conditions

5 Derive Parameters

5.1 Ontology Alignment Process with Parameters

We have observed that alignment methods like QOM [6] or PROMPT [11] may be mapped onto a generic alignment process (Figure 2). We refer to [6] for a detailed description. Here we will focus on mentioning the six major steps to clarify which parameters actually have to be set.

1. *Feature Engineering*, i.e. select excerpts of the overall ontology definition to describe a specific entity (e.g. label).

2. *Search Step Selection*, i.e. choose two entities from the two ontologies to compare. Either every entity pair is compared or only an efficient subset.

3. *Similarity Assessment*, i.e. indicate a similarity for a given description of two entities (e.g., $\text{sim}_{\text{superConcept}}(e_1,e_2)$=1.0).

4. *Similarity Aggregation*, i.e. aggregate multiple similarity assessment for one pair of entities into a single measure. The feature and similarity selection, together with the weighting scheme represents the underlying general alignment strategy. This may either be done in an semantically simple or complex way.

5. *Interpretation*, i.e. use all aggregated numbers, a threshold and interpretation strategy to propose the alignment (align(e_1)=‘e_2’). This may also include a user validation.

6. *Iteration*, i.e. as the similarity of one entity pair influences the similarity of neighboring entity pairs, the equality is propagated through the ontologies.

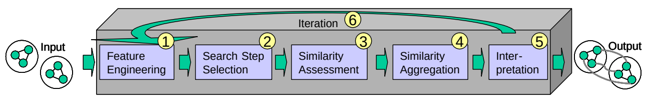

Figure 2: General Alignment Process

The correlations between parameters and results have been identified in various experiments with our framework [6, 7]. Most of them are intuitive: semantical more complex features lead to higher quality, just as longer run times of the alignment algorithm do.

5.2 Example

We continue with the running example and now *assign the parameters* accordingly. High precision and high recall represents the highest quality requirement. For this we have to focus on using semantically complex features and heuristics. Further, it is necessary to take a complete look at the alignment candidates. Again to increase quality the user is included in the loop and can provide feedback during runtime. And finally we have to ensure that alignments have flooded the ontology graphs in a complete way, which can only be achieved through many iteration steps. To ensure medium fast processing, we can still rely on complex features and heuristics. However, we need an efficient strategy only including the most promising alignment candidates for comparison and rely only on a low number of iterations. Compared to a very fast approach we can use slightly more iterations. Finally, the required level of automation can be directly translated into the corresponding parameter. Semi-automation requirements trigger semi-automation of the algorithm. The final parameters are also shown in Table 4.

Requirement	Value	Feature/ Similarity	Search Steps	Threshold	Interaction	Iterations
Precision	high	complex	complete		semi-	many
Recall	high					
Time	med.	complex	efficient		auto	few
Auto.	semi-				semi-	
Final		complex	efficient	med.	semi-	med.

Table 4: Parameters

From this last table we can extract the final parameters: use of complex feature and heuristic as strategy, an efficient search step selection, application of a medium threshold value, a semi-automatic approach with user interaction, and a medium number of iterations.

With our methodology we have found the algorithm parameters leading to the required dimensions and thus maximizing the utility value for a certain use case. The user himself doesn’t have to worry about the optimized parameters. The system can determine them on the fly from the input use case and the side conditions. The parameters can now be applied to the alignment algorithm, thus resulting in the presumably highest utility for the user.

5.3 Parameters based on Requirements

In Table 5 we give an overview on how parameters have to be set based on the requirements. To determine the overall

Requirement	Value	Feature/ Similarity	Search Steps	Threshold	Interaction	Iterations
Precision/ Recall	high/high	complex	complete	med.	semi-	many
	high/med.	med.		high		med.
	med./high	med.		low		med.
	med./med.	simple	efficient	med.	auto	few
Time	fast	simple	efficient		auto	few
	med.	complex	efficient			med.
	slow	complex	complete		semi-	many
Automation	automatic				auto	
	semi-				semi-	

Table 5: Parameters based on Requirements

parameters for the alignment algorithm we further have to combine the found individual parameters from each dimension. By averaging we can propose an optimal strategy for the alignment algorithm.

6 Implementation and Evaluation

6.1 Implementation

The presented adaptive strategy has been implemented in the FOAM tool. After providing the corresponding use case and the underlying ontologies the system determines the best algorithm parameters according to the methodology presented in this paper. Unless the user wants to explicitly change the parameters they are applied as proposed.

6.2 Evaluation Set-Up

Evaluation is difficult as only humans can finally decide whether the maximum utility has been reached. Apart from this, it is impossible to try every combination of parameters. We therefore set-up five scenarios with different use cases (ontology evolution, data integration, ontology merging, reasoning, and query rewriting), ontologies (RDF vs. OWL, 50 to 300 entities each) and computing equipment (very powerful vs. less powerful). For each scenario we had four sets of alignment algorithm parameters: an optimal adaptive set according to our methodology, a fixed presumably optimal set, and two arbitrary fixed sets. Ten users were asked to carry out each of the five scenarios with each of the sets of algorithm parameters. Urged to keep the use case in mind the users afterwards had to decide which runs were best, effectively ranking the four runs.

6.3 Evaluation Results and Discussion

After averaging over the different users' opinions and normalizing we gain the results in Table 6.

	Case1	Case2	Case3	Case4	Case5	Avg.
Adaptive	1	1	2	3	2	*1*
Fixed	3	2	3	2	3	3
Other 1	2	4	4	1	1	2
Other 2	4	3	1	4	4	4

Table 6: Ranking

The users confirmed that the dynamically chosen strategy in average yielded the best results. This is a clear indicator, that dynamically parameterizing alignment methods are valuable.

However, the adaptive approach did not always appear to be the absolute best one. To understand the reason for this we had a closer look at Case 4 (two sports ontologies; reasoning use case). These ontologies were modeled very exact and clean. As an effect even though the adaptive approach took considerably longer in its search for further alignments, it didn't identify any additional correct ones. Due to this users preferred the fast only label-based approach. We conclude that there are things that effect the outcome of the ontology alignment process, but cannot be explicitly detected in beforehand. In the end these are the subjective parameters we could not measure and had to ignore for our methodology. Therefore, one should rather evaluate on an overall average basis, than claiming an optimal output for every individual case.

Surprisingly, the presumably generally "best" strategy only was ranked third, which shows how difficult it is to actually determine a best strategy.

In general the results support the assumption that ontology alignment can be done much more effective when the use case is kept in mind. Algorithms can actually be reused for different scenarios. Finally, it showed that our theoretic considerations could be brought to a practical implementation, despite the often only fuzzily defined coherences of parameters, goal dimensions, and overall utility.

7 Related Work

To the best of our knowledge there has not been any work on dynamically changing the parameters of an ontology alignment algorithm based on the underlying use case or the involved ontologies.

However, as the basic ontology alignment problem has been around for some years, first tools have already been developed.The tools PROMPT and AnchorPROMPT [11] use labels and to a certain extent the structure of ontologies. However, their focus lies on ontology merging i.e. how to create one ontology out of two. Potential matches are presented to the user for confirmation. [5] use a general approach of relaxation labeling in their tool GLUE. Most of their work is based on the similarity of instances only, thus requiring ontologies with a fair amount of instances. As this paper focused on ontologies rather than general schema matching we will only briefly mention work from this related area. [2, 9] have proposed ways on combining different schema matching approaches. A foundation for matching graph-like structures was given by [10] in their similarity flooding approach.

Heading in the direction of finding an optimal algorithm are [4]. The focus of their work was to generally investigate which matcher performs best, mainly for XML documents. They also mentioned that this depends on e.g. the schema characteristics. However, they don't take the next step in thus including use cases for dynamically choosing and adjusting the best matcher.

8 Conclusion

Ontology alignment should not be considered as an independent topic. It will always occur in conjunction with specific use cases within real applications. To manage the step from research to application it is important to understand the detailed requirements of the use cases for an alignment algorithm. Further research in this direction is necessary including field studies and user questionnaires.

In this paper we have presented a methodology for deriving an optimal strategy for ontology alignment based on underlying use cases. We started with a utility function for ontology alignment taking into account the results of the alignment process and the preferences of the different use cases. We then backtracked from the utility value to requirements (e.g. quality) which had to be fulfilled. And finally we determine the parameters of a general alignment algorithm to actually achieve the requirements, also considering possible side conditions. This methodology of dynamically adapting the parameters of ontology alignment algorithms has been prototypical implemented in our tool FOAM. The evaluation thereof has shown that the dynamic parameters actually outperform fixed strategies.

Semantic integration has been a task of increasing importance. With the work presented in this paper we have given ideas on how algorithms can generally be adapted to different alignment use cases as required in the projects we are involved. Thus, it will be possible to technically reuse existing good alignment approaches for a variety of alignment problems. As the focus of this paper laid on a general alignment process rather than a specific tool, we expect the results of this paper to be transferred to other schema integration approaches as well. This is a key step for bringing Semantic Web technology from specialized domains to application in real world scenarios.

References

[1] R. Agrawal and R. Srikant. On integrating catalogs. In *Proceedings of the Tenth International Conference on the World Wide Web (WWW-10)*, pages 603–612. ACM Press, 2001.

[2] P. A. Bernstein, S. Melnik, M. Petropoulos, and C. Quix. Industrial-strength schema matching. *SIGMOD Rec.*, 33(4):38–43, 2004.

[3] P. Casanovas et al. SEKT legal use case components: Ontology and architectural design. In *Proceedings of ICAIL 05*, 2005.

[4] H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[5] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings to the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.

[6] M. Ehrig and S. Staab. QOM - quick ontology mapping. In F. van Harmelen, S. McIlraith, and D. Plexousakis, editors, *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, LNCS, pages 683–696, Hiroshima, Japan, 2004. Springer.

[7] M. Ehrig, S. Staab, and Y. Sure. Supervised learning of an ontology alignment process. In *Workshop on IT Tools for Knowledge Management Systems: Applicability, Usability, and Benefits (KMTOOLS) at 3. Konferenz Professionelles Wissensmanagement*, Kaiserslautern, April 2005.

[8] P. Haase et al. Bibster - a semantics-based bibliographic peer-to-peer system. In F. van Harmelen, S. McIlraith, and D. Plexousakis, editors, *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, LNCS, pages 122–136, Hiroshima, Japan, 2004. Springer.

[9] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with CUPID. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[10] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, page 117. IEEE Computer Society, 2002.

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

[12] H. S. Pinto, C. Tempich, and S. Staab. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In R. L. de Mantaras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397, Valencia, Spain, August 2004. IOS Press.

[13] C. Tempich et al. XAROP: A midterm report in introducing a decentralized semantics-based knowledge sharing application. In D. Karagiannis and U. Reimer, editors, *Proceedings of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004)*, LNCS, Vienna, Austria, December 2004. Springer.