Proceedings of the Third International Workshop on Debugging Ontologies and Ontology Mappings - WoDOOM14

Anissaras/Hersonissou, Greece
May 26, 2014.

Edited by:
Patrick Lambrix
Guilin Qi
Matthew Horridge
Bijan Parsia

Preface

Developing ontologies is not an easy task and, as the ontologies grow in size, they are likely to show a number of defects. Such ontologies, although often useful, also lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed. Defects in ontologies can take different forms. Syntactic defects are usually easy to find and to resolve. Defects regarding style include such things as unintended redundancy. More interesting and severe defects are the modeling defects which require domain knowledge to detect and resolve such as defects in the structure, and semantic defects such as unsatisfiable concepts and inconsistent ontologies. Further, during the recent years more and more mappings between ontologies with overlapping information have been generated, e.g. using ontology alignment systems, thereby connecting the ontologies in ontology networks. This has led to a new opportunity to deal with defects as the mappings and other ontologies in the network may be used in the debugging of a particular ontology in the network. It also has introduced a new difficulty as the mappings may not always be correct and need to be debugged themselves.

The WoDOOM series deals with these issues. This volume contains the proceedings of its third edition: WoDOOM14 - Third International Workshop on Debugging Ontologies and Ontology Mappings held on May 26, 2014 in Anissaras/Hersonissou, Greece. WoDOOM14 was an ESWC 2014 (11th Extended Semantic Web Conference) workshop.

In his excellent invited talk, Jérôme Euzenat considered the problem of revising a network of ontologies. Further, there were presentations of four research and two demonstration papers. The topics included work on defects regarding incorrectness and incompleteness and dealt with both detection and repair of defects.

The editors would like to thank the Program Committee for their work in enabling the timely selection of papers for inclusion in the proceedings. We also appreciate our cooperation with EasyChair as well as our publisher CEUR Workshop Proceedings.

May 2014

Patrick Lambrix
Guilin Qi
Matthew Horridge
Bijan Parsia

Workshop Organization

Workshop Organizers

Patrick Lambrix	Linköping University, Sweden
Guilin Qi	Southeast University, China
Matthew Horridge	Stanford University, USA
Bijan Parsia	University of Manchester, UK

Program Committee

Grigoris Antoniou	University of Huddersfield, UK
Samantha Bail	University of Manchester, UK
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Ronald Cornet	Universiteit van Amsterdam, The Netherlands and Linköping University, Sweden
Bernardo Cuenca Grau	University of Oxford, UK
Jérôme Euzenat	INRIA, France
Peter Haase	fluid Operations, Germany
Matthew Horridge	Stanford University, USA
Maria Keet	University of KwaZulu-Natal, South Africa
Patrick Lambrix	Linköping University, Sweden
Yue Ma	TU Dresden, Germany
Christian Meilicke	University of Mannheim, Germany
Tu Anh T. Nguyen	Open University, UK
Bijan Parsia	University of Manchester, UK
Rafael Peñaloza	TU Dresden, Germany
Guilin Qi	Southeast University, China
Uli Sattler	University of Manchester, UK
Stefan Schlobach	Vrije Universiteit Amsterdam, The Netherlands
Bariş Sertkaya	SAP Research Dresden, Germany
Kostyantyn Shchekotykhin	University Klagenfurt, Austria
Kewen Wang	Griffith University, Australia
Renata Wassermann	University of Sao Paulo, Brazil
Fang Wei-Kleiner	Linköping University, Sweden

Table of Contents

Foundations for revising networks of ontologies

Jérôme Euzenat

INRIA & LIG
Grenoble, France
Jerome.Euzenat@inria.fr
http://exmo.inria.fr

The framework of belief revision has been studied for years in the context of logic theories. It has been considered several times for description logics and more recently for aligned ontologies. We consider more generally the problem of revising a network of ontologies: given a set of ontologies connected by alignments, how to evolve them such that they account for new information, i.e., new formulas or correspondences. Revision is a typical problem of the semantic web due to its open nature.

There are two extreme ways to approach this problem: on the one hand, transforming the network of ontologies in a single logic theory and applying classical revision; on the other hand, applying revision locally to each ontology and to each alignment and communicating the changes to related elements. We keep a middle term between these two approaches: local revision alone is not sufficient to revise networks of ontologies but preserving the separation of ontologies and alignments can be exploited by revision.

We first use existing semantics of networks of ontologies for defining the notions of closure and consistency for networks of ontologies. Inconsistency can come from two different sources: local inconsistency in a particular ontology or alignment, and global inconsistency between them. Revision, in turn, can affect any of these components: retracting assertions from closed ontologies, like in classical belief revision, or correspondences from closed alignments, like in current alignment repair.

Then, we define revision postulates for networks of ontologies and we show that revision cannot be simply based on local revision operators on both ontologies and alignments: they may fail to reach a consistent network of ontologies although solutions exist. We define a global revision operator by adapting the partial meet revision framework to networks of ontologies. We show that it indeed satisfies the revision postulates.

Finally, we discuss strategies based on network characteristics for designing concrete revision operators.

First experiments in cultural alignment repair

Jérôme Euzenat

INRIA & LIG,
Grenoble, France
Jerome.Euzenat@inria.fr
http://exmo.inria.fr

Abstract. Alignments between ontologies may be established through agents holding such ontologies attempting at communicating and taking appropriate action when communication fails. This approach has the advantage of not assuming that everything should be set correctly before trying to communicate and of being able to overcome failures. We test here the adaptation of this approach to alignment repair, i.e., the improvement of incorrect alignments. For that purpose, we perform a series of experiments in which agents react to mistakes in alignments. The agents only know about their ontologies and alignments with others and they act in a fully decentralised way. We show that such a society of agents is able to converge towards successful communication through improving the objective correctness of alignments. The obtained results are on par with a baseline of a priori alignment repair algorithms.

Keywords: Ontology alignment; alignment repair; cultural knowkedge evolution; agent simulation; coherence; network of ontologies

1 Motivation

The work on cultural evolution applies, an idealised version of, the theory of evolution to culture. Culture is taken here as an intellectual artifact shared among a society. Cultural evolution experiments typically observe a society of agents evolving their culture through a precisely defined protocol. They perform repeatedly and randomly a task, called game, and their evolution is monitored. This protocol aims to experimentally discover the common state that agents may reach and its features. Luc Steels and colleagues have applied it convincingly to the particular artifact of natural language [9].

We aim at applying it to knowledge representation and at investigating some of its properties. A general motivation for this is that it is a plausible model of knowledge transmission. In ontology matching, it would help overcoming the limitations of current ontology matchers by having alignments evolving through their use, increasing the robustness of alignments by making them evolve if the environment evolves.

In this paper, we report our very first experiments in that direction. They consider alignments between ontologies as a cultural artifact that agents may repair while trying to communicate. We hypothesise that it is possible to perform meaningful ontology repair with agents acting locally. The experiments reported here aims at showing that, starting from a random set of ontology alignments, agents can, through a very simple and distributed mechanism, reach a state where (a) communication is always successful, (b) alignments are coherent, and (c) F-measure has been increased. We also compare the obtained result to those of state-of-the-art repair systems.

Related experiments have been made on emerging semantics (semantic gossiping [3, 2]). They involve tracking the communication path and the involved correspondences. By contrast, we use only minimal games with no global knowledge and no knowledge of alignment consistency and coherence from the agents. Our goal is to investigate how agents with relatively little common knowledge (here instances and the interface to their ontologies) can manage to revise networks of ontologies and at what quality.

2 Experimental framework

We present the experimental framework that is used in this paper. Its features have been driven by the wish that experiments be easily reproducible and as simple as possible. We first illustrate the proposed experiment through a simple example (§2.1), before defining precisely the experimental framework (§2.2) following [9].

2.1 Example

Consider an environment populated by objects characterised by three boolean features: color={white|black}, shape={triangle|square} and size={small|large}. This characterises $2^3 = 8$ types of individuals: ■, ▲, □, △, ■, ▲, □, △.

Three agents have their own ontology of what is in the environment. These ontologies, shown in Figure 1, identify the objects partially based on two of these features. Here they are a circular permutation of features: FC (shape, color), CS (color, size) and SF (size, shape).

In addition to their ontologies, agents have access to a set of shared alignments. These alignments comprise equivalence correspondences between their top (all) classes and other correspondences. Initially, these are randomly generated equivalence correspondences. For instance, they may contain the (incorrect) correspondence: SF:small $\equiv$ CS:black.

Agents play a very simple game: a pair of agents a and b are randomly drawn as well as an object of the environment o. Agent a asks agent b the class c (source) to which the object o belongs, then it uses an alignment to establish to which class c' (target) this corresponds in its own ontology. Depending on the respective relation between c and c', a may take the decision to change the alignment.

For instance, if agent CS draws the small-black-triangle (▲) and asks agent SF for its class, this one will answer: small-triangle. The correspondence SF:small $\equiv$ CS:black and the class of ▲ in CS is black-small which is a subclass of CS:black, the result is then a SUCCESS. The fact that the correspondence is not valid is not known to the agents, the only thing that counts is that the result is compatible with their own knowledge.

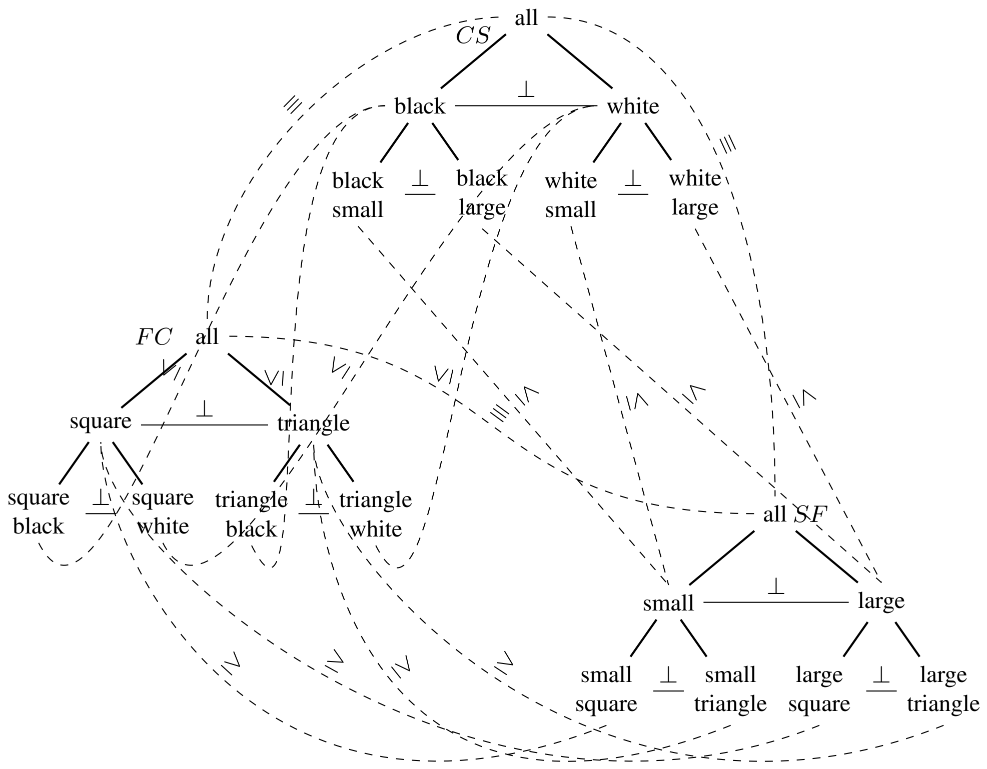


Fig. 1. Example of a generated network of ontologies with the exact reference alignments.

If, on the contrary, the drawn instance is small-white-triangle (△), SF would have made the same answer. This time, the result would be a FAILURE because △ belongs to class CS:white-small which is disjoint from CS:black-small.

How to deal with this failure is a matter of strategy:

delete SF:small $\equiv$ CS:black can be suppressed from the alignment;
replace SF:small $\equiv$ CS:black can be replaced by SF:small $\leq$ CS:black;
add in addition, the weaker correspondence SF:small $\geq$ CS:all can be added to the alignment (but this correspondence is subsumed by SF:all $\equiv$ CS:all).

In the end, it is expected that the shared alignments will improve and that communication will be increasingly successful over time. Successful communication can be observed directly. Alignment quality may be assessed through other indicators: Figure 1 shows (in dotted lines) the correct (or reference) alignments. Reference alignments are not known to the agents but can be automatically generated and used for measuring the quality of the resulting network of ontologies through F-measure.

2.2 Experimental set up

We systematically describe the different aspects of the carried out experiments in the style of [9].

Environment: The environment contains objects which are described by a set of n characteristics (we consider them ordered). Each characteristic can take two possible values which, in this experiment, are considered exclusive.

Population: The experiment uses n agents with as many ontologies. Each agent is assigned one different ontology. In this first setting, each agent will have an ontology based on $n-1$ of these characteristics (each agent will use the first $n-1$ characteristics starting at the agent's rank). The ontology is a simple decision trees of size 2^{n-1} in which each level corresponds to a characteristic and subclasses are disjoint.

Shared network of ontologies: A complete network of $\frac{n\times(n-1)}{2}$ alignments between the ontologies is shared among agents (public). The network is symmetric (the alignment between o and o' is the converse of the alignment between o' and o) and a class is in at most one correspondence per alignment.

Initialisation: In the initial state, each alignment contains equivalence correspondences between the most general classes of both ontologies, plus 2^{n-1} randomly generated equivalence ($\equiv$) correspondences.

Game: A pair of distinct agents $\langle a, b\rangle$ is randomly picked up as well as a set of characteristic values describing an individual (equiprobable). The first agent (a) asks the second one (b) the (most specific) class of its ontology to which the instance belongs ($source$). It uses the alignment between their respective ontologies for finding to which class this corresponds in its own ontology ($target$). This class is compared to the one the instance belongs to in the agent a ontology ($local$).

Success: Full success is obtained if the two classes ($target$ and $local$) are the same. But there are other cases of success:

- $target$ is a super-class of $local$: this is considered successful (this only means that the sets of alignments/ontologies are not precise enough);
- $target$ is a sub-class of $local$: this is not possible here because for each instance, $local$ will be a leaf.

Failure: Failure happens if the two classes are disjoint. In such a case, the agent a will proceed to repair.

Repair: Several types of actions (called modalities) may be undertaken in case of failure:

delete the correspondence is simply discarded from the alignment;

replace if the correspondence is an $\equiv$ correspondence it is replaced by the $\leq$ correspondence from the target class to the source class;

add in addition to the former a new $\leq$ correspondence from the source to a superclass of the target is added. This correspondence was entailed by the initial correspondence, but would not entail the failure.

Success measure: The classical success measure is the rate of successful communication, i.e., communication without failure.
Secondary success measure: Several measures may be used for evaluating the quality of the reached state: consistency, redundancy, discriminability. We use two different measures: the averaged degree of incoherence [7] and the semantic F-measure [4]. Indeed, this setting allows for computing automatically the reference alignment in the network, so we can compute F-measure.
External validation: The obtained result can be compared with that of other repair strategies. We compare the results obtained with those of two directly available repair algorithms: Alcomo [6] and LogMap repair [5].

3 Experiments

We report four series of experiments designed to illustrate how such techniques may work and what are their capabilities

The tests are carried out on societies of at least 4 agents because, in the setting with 3 agents, the delete modality drives the convergence towards trivial alignments (containing only all≡all) and the other modalities do it too often.

All experiments have been run in a dedicated framework that is available from http://lazylav.gforge.inria.fr.

3.1 Convergence

We first test that, in spite of mostly random modalities (random initial alignments, random agents and random instances in each games), the experiments converge towards a uniform success rate.

Four agents are used and the experiment is run 10 times over 2000 games. The evolution of the success rate is compared.

3.2 Modality comparison

The second experiment tests the behaviour of the three repair modalities: delete, replace, add.

Four agents are used and the experiment is run 10 times over 2000 games with each modalities. The results are collected in terms of average success rate and F-measure.

3.3 Baseline comparison

Then the results obtained by the best of these modalities are compared to baseline repairing algorithms in terms of F-measures, coherence and number of correspondences.

The baseline algorithms are Alcomo and LogMap repair. The comparison is made on the basis of success rate, F-measure and the number of correspondences.

LogMap and Alcomo are only taken as a baseline: on the one hand, such algorithms do not have the information that agents may use, on the other hand, agents have no global view of the ontologies and knowledge of consistency or coherence.

3.4 Scale dimension

Finally we observe settings of increasing difficulty by taking the modality providing the best F-measure and applying it to settings with 3, 4, 5 and 6 ontologies.

This still uses 10 runs with the add modality over 10000 games. Results are reported as number of correspondences, F-measure and success rate and compared with the best F-measure of Alcomo and LogMap.

4 Results

Results of the four presented experiments are reported and discussed.

4.1 Convergence

Figure 2 shows the result of our first experiment: 10 runs with a random network as defined above with 4 ontologies. Each curve corresponds to one of the 10 runs over 2000 iterations.

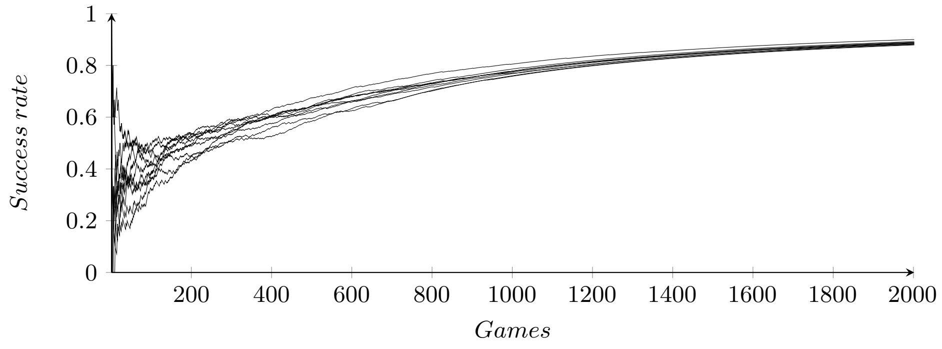


Fig. 2. Ten random runs and their overall success rate, i.e., the proportion of games which were successful so far [mod=add; #agents=4; #games=2000; #runs=1].

Figure 2 shows a remarkable convergence between the runs. After the first 200 games dominated by randomness, they converge assymptotically and at the same pace towards 100%. Indeed, as soon as the network of ontologies has been cleaned up (around 1200 iterations maximum), the rate only grows. It never reaches 1 because of the initial period which contains failures.

From now on, we will still consider 10 runs, but the results will be averaged over these runs.

4.2 Modality comparison

Figure 3 shows the evolution over 2000 iterations of the success rate and F-measure of the three presented modalities.

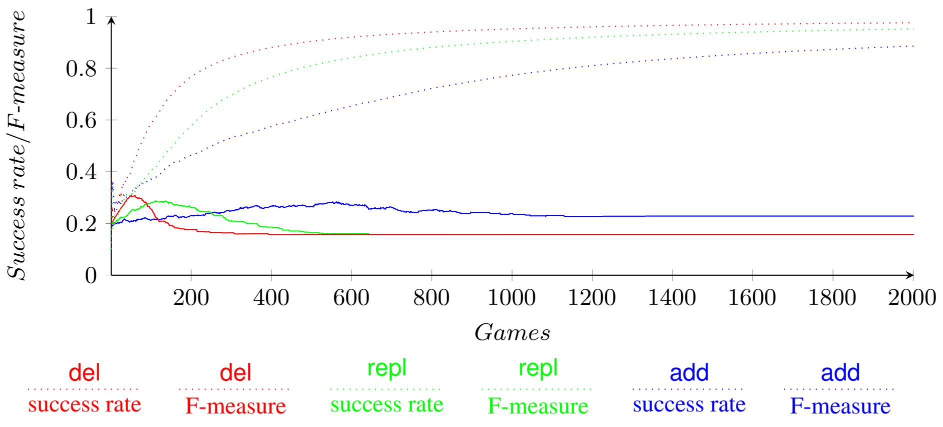


Fig. 3. The average F-measures (dashed) and success rate (plain) with the three different modalities: delete (red), replace (green) and add (blue) [mod=del,repl,add; #agents=4; #games=2000; #runs=10].

delete converges more quickly than replace which converges more quickly than add. This can easily be explained: delete suppresses a cause of problem, replace only suppresses half of it so it may need one further deletion for converging, while add replaces one incorrect correspondence by two correspondences which may be incorrect, so it requires more time to converge.

For the same reason, the success rate is consequently higher. Table 1 shows that for the delete modality, 97.6% success rate corresponds to 48 failure, i.e. 48 deleted correspondences over 54. The 6 remaining correspondences are all≡all correspondences. replace reaches the same result with a 95.2% rate, which corresponds to twice as many failures.

The results of delete and replace modalities are the same: in order to be correct, alignments are reduced to the all≡all correspondences. This is unavoidable for delete (because initial correspondences are equivalences, although, by construction, the correct correspondences are subsumption, so the initial correspondences are incorrect in at least one direction). This is by chance, and because of averaging, for replace.

On the contrary, the add modality has a 88.6% success rate, i.e., 228 failures. This means that on average for each correspondence it has generated 4 alternative correspondences. This is only an average because after 2000 games (and even after 10000 games), there remain more than 12 correspondences.

Contrary to the other modalities, add improves over the initial F-measure.

Table 1 shows that all methods reach full consistency (incoherence rate=0.) from a network of ontologies with 50% incoherence, i.e., half of the correspondences are involved in an inconsistency (or incoherence).

Concerning F-measure, add converges towards a significantly higher value than the two other approaches. With four ontologies, it has a chance to find weaker but more

Modality	Size	Success rate	Incoherence degree	Semantic F-measure	Syntactic F-measure	Convergence
reference	70	-	0.0	1.0	1.0	-
initial	54	-	[0.46-0.49]	0.20	(0.20)	-
delete	6	0.98	0.0	0.16	(0.16)	400
replace	6	0.95	0.0	0.16	(0.16)	1000
add	12.7	0.89	0.0	0.23	(0.16)	1330
Alcomo	25.5	-	0.0	0.26	(0.14)	-
LogMap	36.5	-	0.0	0.26	(0.14)	-

Table 1. Results of the three different modalities compared with Alcomo and LogMap on 10 runs, 4 ontologies and 2000 iterations. Syntactic F-measure has been obtained in an independent but identical evaluation.

correct correspondences. The add strategy is more costly but more effective than the two other strategies.

4.3 Baseline comparison

This experiment exploits the same data as the previous one (§4.2); exploiting those of the next experiment (on 10000 iterations) provides similar results.

Table 1 shows that all three methods are able to restore full coherence and to slightly improve the initial F-measure. Their result is overall comparable but, as can be seen in Figure 4, the agents do not reach the F-measure of logical algorithms.

The agents find half of the correspondences of Alcomo and one third of those of LogMap. This is expected because Alcomo only discards the minimum number of correspondences which bring incoherence, while LogMap weaken them (like the add modality). The agents having more information on what is incorrect, discard more correspondences.

When looking at F-measures, it seems that logical repair strategies can find more than 6 new correspondences which are correct while the add strategy can only find more than 3. This is not true, as shown in Table 1, because we use *semantic* precision and recall [4]. These methods preserve correspondences which are not correct, but which entails correct correspondences. This increases semantic recall and F-measure.

There is a large variation on the results given by the different methods. Out of the same 10 runs, LogMap had the best F-measures 5 times, Alcomo 3 times, and the agents twice. But the largest variation is obtained by the agents with a F-measure ranging from 0.16 to 0.33. Its result is indeed highly dependent on the initial alignment.

4.4 Scale dimension

So far, we concentrated on 4 agents, what happens with a different number of agents? The number of agents does not only determine the number of ontologies. It also determines the number of alignments (quadratic in the number of ontologies), the number of correspondences per alignments and the number of features per instances. This means

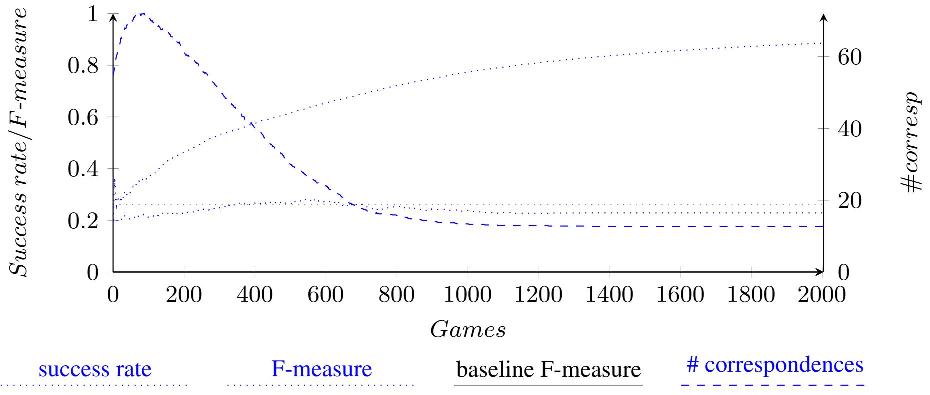


Fig. 4. Average success, F-measure and number of correspondences for the add modality compared to the Alcomo and LogMap F-measure as a baseline [mod=add; #agents=4; #games=2000; #runs=10].

that the more agents are used, the slower is the convergence. So, we played 10000 games in order to have a chance to reach a satisfying level of F-measure.

Figure 5 shows the regular pattern followed by agents: the first phase is random and increases the number of correspondences (due to the add modality). Then, this number slowly decreases. Agents are slower to converge as the problem size increases. This is easily explained: as the correction of the alignment converges, the number of failure-prone games diminishes. Since games are selected at random, the probability to pick up the last configurations (in the end there is only one) becomes lower and lower. The increased number of iterations to converge is directly tied to the largely increased difficulty of the task (number of agents, number of alignments, size of ontologies, characteristics of objects).

This increase is not a measure of the complexity of the approach itself. In fact, it is highly distributed, and it is supposed to be carried out while agents are achieving other tasks (trying to communicate). All the time spend between the two last failures are time of communicative *success*, i.e., agents never had to suffer from the wrong correspondences.

A very simple strategy for improving this would be that agents try to select themselves examples in order to verify the correspondences that they have not already tested.

Table 2 seems to show that, as the complexity of the problem increases, the F-measure of agents is better than that of logical repair mechanisms.

5 Discussion

The relatively low F-measure rate is tied to the type of experiments: agents do not invent any correspondences, they only repair them. Hence, they are constrained by the initial alignment. To this respect, they are on par with logical repair algorithms.

However, they have more information than these repair algorithms. It could then be expected that their results are higher. This is not the case because, when an initial

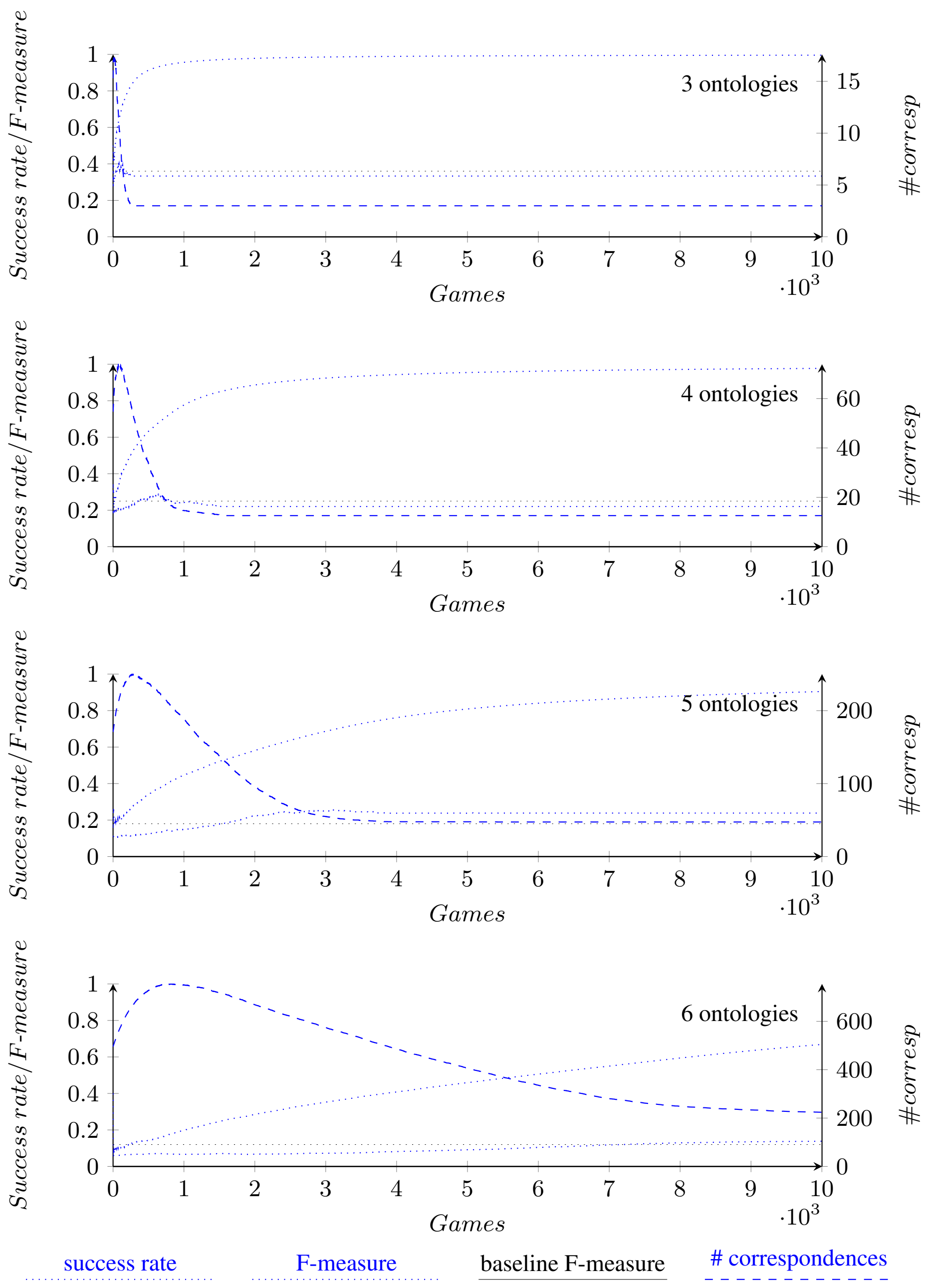


Fig. 5. 10.000 games with 3, 4, 5 and 6 ontologies [mod=add,#agents=3,4,5,6; #games=10000; #runs=10].

# agents	# correpondences					Incoherence				F-measure				Convergence
	Reference	Initial	LogMap	Alcomo	Final	Initial	LogMap	Alcomo	Final	Initial	LogMap	Alcomo	Final	
3	15	15	12	10.3	3	0.31	0.	0.	0.	0.32	0.35	0.36	0.33	300
4	70	54	36.7	28.4	12.4	0.47	0.	0.	0.	0.20	0.24	0.25	0.21	1670
5	250	170	94.7	71.7	47.4	0.58	0.	0.	0.	0.11	0.18	0.17	0.24	5400
6	783	495	234	182	224	0.63	0.	0.	0.	0.06	0.12	0.11	0.14	10.000+

Table 2. Number of correspondences, incoherence rate and F-measure over 10000 games.

correspondence is unrelated to the valid one, agents will simply discard them. They will thus end up with few correspondences with a high precision and low recall.

The state-of-the-art repair algorithms will preserve more correspondences because their only criterion is consistency and coherence: as soon as the alignment is coherent, such algorithms will stop. One could expect a lower precision, but not a higher recall since such algorithms are also tied to the initial alignment.

But because we use semantic precision and recall, it happens that among these erroneous correspondences, some of them entail some valid correspondences (and some invalid ones). This contributes to raise semantic recall.

6 Conclusion

We explored how mechanisms implemented as primitive cultural evolution can be applied to alignment repair. We measured:

– Converging success rate (towards 100% success);
– Coherent alignments (100% coherence);
– F-measures on par with logical repair systems;
– A number of games necessary to repair increasing very fast.

The advantage of this approach are:

– It is totally distributed: agents do not need to have the knowledge of what is an inconsistent or incoherent alignment (only an inconsistent ontology).
– The repair of the network of ontologies is not blind, i.e., restoring inconsistency without knowing if it is likely to be correct, so it also increases F-measure (which is not necessarily the case of other alignment repair strategies [8]).

Yet, this technique does not replace ontology matching nor alignment repair techniques.

7 Perspectives

We concentrated here on alignment repair. However, such a game can perfectly be adapted for matching (creating missing correspondences and revising them on the fly).

In the short term, we would like to adapt this technique in two directions:

– introducing probabilities and using such techniques in order to learn confidence on correspondences that may be used for reasoning [1],
– dealing with alignment composition by propagating instances across agents in the same perspective as the whispering games (propagating classes and see what comes back, setting weights to correspondences) [3].

In the longer term, such techniques do not have to be concentrated on one activity, such as alignment repair. Indeed, they are not problem solving techniques (solving the alignment repair problem). Instead, they are adaptive behaviours, not modifying anything as long as activities are carried out properly, and reacting to improper situations. So, cultural knowledge evolution has to be involved in broader activities, such as information gathering.

8 Acknowledgements

Thanks to Christian Meilicke (Universität Mannheim) and Ernesto Jimenez Ruiz (Oxford university) for making Alcomo and LogMap available and usable. Thanks to an anonymous reviewer for further suggestions.

References

1. Manuel Atencia, Alexander Borgida, Jérôme Euzenat, Chiara Ghidini, and Luciano Serafini. A formal semantics for weighted ontology mappings. In *Proc. 11th International Semantic Web Conference (ISWC)*, volume 7649 of *Lecture notes in computer science*, pages 17–33, Boston (MA US), 2012.
2. Thomas Cerqueus, Sylvie Cazalens, and Philippe Lamarre. Gossiping correspondences to reduce semantic heterogeneity of unstructured P2P systems. In *Proc. 4th International Conference on Data Management in Grid and Peer-to-Peer Systems, Toulouse (FR)*, pages 37–48, 2011.
3. Philippe Cudré-Mauroux. *Emergent Semantics: Interoperability in large-scale decentralized information systems*. EPFL Press, Lausanne (CH), 2008.
4. Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 348–353, Hyderabad (IN), 2007.
5. Ernesto Jiménez-Ruiz, Christian Meilicke, Bernardo Cuenca Grau, and Ian Horrocks. Evaluating mapping repair systems with large biomedical ontologies. In *Proc. 26th Description logics workshop*, 2013.
6. Christian Meilicke. *Alignment incoherence in ontology matching*. PhD thesis, Universität Mannheim, 2011.
7. Christian Meilicke and Heiner Stuckenschmidt. Incoherence as a basis for measuring the quality of ontology mappings. In *Proceedings of the 3rd ISWC international workshop on Ontology Matching*, pages 1–12, 2008.
8. Catia Pesquita, Daniel Faria, Emanuel Santos, and Francisco Couto. To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In *Proc. 8th ISWC ontology matching workshop (OM), Sydney (AU)*, pages 13–24, 2013.
9. Luc Steels, editor. *Experiments in cultural language evolution*. John Benjamins, Amsterdam (NL), 2012.

Repairing Learned Ontologies

Daniel Fleischhacker

Research Group Data and Web Science, University of Mannheim, Germany
daniel@informatik.uni-mannheim.de

Abstract. Though ontology learning can help ontology engineers in creating ontologies more rapidly, it also poses new challenges to them especially in creating coherent, high-quality ontologies. Given that there have been proposed ontology learning approaches which are even supporting the generation of highly expressive schemas, the need for pushing debugging methods for such schemas further also arises. Throughout the literature, several methods applicable for ontology debugging were presented which might be well-suited for being used on learned ontologies. Thus, in this work, we describe some of these approaches, and also present an approach based on Markov logic networks which is inspired by recent work in this direction and which we adapted to learned ontologies. Afterwards, we evaluate the approaches regarding their runtime performance on learned ontologies and their suitability for automatic and semi-automatic debugging of learned ontologies.

1 Introduction

Linked Data without schema information is already providing many chances for implementing new applications. Most Linked Data repositories are not accompanied by schemas giving additional knowledge about their structure and also new possibilities to deduce not yet explicitly contained information by means of inferencing mechanisms. This could, for example, be beneficial in a query answering scenario for inferring additional instances fulfilling an information need. The additional value of schemas gets even higher when not only light-weight schemas are provided, e.g., containing subsumption hierarchies, but more expressive schemas are accompanying the data. Even containing logically simple elements like disjointness would give many additional possibilities for using the data. Unfortunately, there are hardly any data repositories serving expressive schemas which is mostly caused by the effort required to create such schemas. The even greater expressivity of the OWL2 seems to be practically unused throughout the Linked Data cloud.

To reduce the impact of this so-called knowledge acquisition bottleneck, there has been much work in the direction of ontology and schema learning trying to generate expressive ontologies from a variety of data sources like texts or structured data [2]. Recently, methods also supporting large parts of the expressivity of OWL2 have been introduced [6]. However, since ontology learning approaches are considered to be error-prone and erroneous schemas are limited in their usage, ontology learning is typically used for supporting human ontology engineers with the creation of schemas. This also poses new challenges to the engineers since the approaches might generate ontologies containing a great number of complex axioms which could lead to a high degree of incoherence in the ontology whose causes are not as easy to grasp as for more basic axioms. Thus, additional support for humans in debugging learned ontologies or even automatic means for repairing them is desired. Lately, some new debugging methods for ontologies which are based on Markov logic networks [14] and specialized diagnosis tools trying to take advantage from the specific characteristics of learned ontologies [4] have been proposed.

In this paper, we perform experiments on learned ontologies to compare and evaluate common and also more recently proposed debugging methods on expressive learned ontologies. In particular, we focus on the runtime properties and the scalability of the approaches. Furthermore, we examine them regarding their applicability in automatized and semi-automatized scenarios.

The rest of the paper is structured as follows: After giving a short overview on the related work relevant for this paper (Section 2), we give required definitions regarding the incoherence of ontologies and the notion of explanations as well as a short overview on the basics of Markov Logic networks in Section 3. Afterwards, in Section 4, we describe the different approaches we considered for making the ontology coherent before presenting the comparison of these approaches in Section 5. Finally, we summarize our findings and draw conclusion regarding the usage of different approaches for debugging learned ontologies in Section 6.

2 Related Work

In this section, we describe other works which concentrate on the diagnosis and repair of ontologies.

Most approaches for repairing incoherent or inconsistent ontologies are based on finding the roots of discovered errors in the ontology which are sets of axioms leading to the specific logical error. Finding these so-called explanations is most often done employing diagnosis facilities integrated in reasoning tools (white-box approach) as implemented in the Pellet reasoner [19] or black-box approaches which work independently from the underlying reasoning system [8]. Since, as also argued by Horridge et al. [9], it is practically not possible to generate all explanations for a given unsatisfiability in many cases, these approaches concentrate on retrieving a limited number of explanations. There has also been some work regarding the computation of full explanation sets for learned ontologies by trying to exploit the specific characteristics of the learned ontologies [4].

A tool based on the explanation generated capabilities of Pellet is Swoop [11] which is an ontology editor able to present unsatisfiabilities in the ontology and the causes of these logical problems. The Protege ontology editor can [1] also use Pellet to generate explanations for incoherent classes or other inconsistencies found in the ontology. However, the main purpose of these tools is to provide the information about the causes of the unsatisfiability to the ontology engineers, but they neither provide automatic means to solve the problems nor help on choosing axioms to remove. Furthermore, though

[1] `http://protege.stanford.edu/`

these tools are applicable to both T-box and A-box, they are not specifically targeted at cleaning only the terminology.

Lehmann and Bühmann [12] proposed the ORE tool which combines learning and debugging processes into a common workflow in which a given ontology is progressively enriched with additional, possibly complex axioms and the resulting inconsistencies or incoherences are presented to the user along with possible solutions. As reported in the paper, ORE's enrichment part is currently limited to a very basic set of axioms and, e.g., not supporting disjointness axioms. In contrast to our work presented here, ORE is also combining both incoherence and inconsistency repair and needs user interaction.

There are cases in which it might be impossible for a human to check and solve all logical problems in the ontology manually without additional support, e.g., if the number of wrong explanations is overwhelmingly high and thus it is hard to find the actually problematic axiom. Thus, there are also approaches which propose methods of automatically determining fixes for logical errors. One of the earliest works in this direction has been done by Schlobach [18] who describes an approach for debugging incoherences, i.e., unsatisfiable classes in ontologies, based on computing and removing minimum sets of axioms causing the incoherence. In addition, he also addresses the problem that many real world ontologies are not expressive enough and are especially missing disjointness axioms which makes debugging almost impossible. Schlobach provides a possible solution by using the Strong Disjointness Assumption to consider all sibling classes to be disjoint.

Another work concentrating on debugging terminologies comes from Qi et al. [16] who propose and experimentally evaluate kernel revision operators for cleaning up the terminology. For finding the axioms which are removal candidates, these approaches use scoring, like the number of occurrences of an axiom in explanations, or confidence values as they could be assigned to the axioms by an ontology learning approach. Based on explanations generated by the Pellet reasoning tool, this also leads to an approach which can potentially be run fully automatically to generate coherent ontologies. The proposed approaches are evaluated on an ontology generated by an ontology learning tool and an ontology mapping scenario since both methods used in these areas commonly create logically incoherent schemas.

The RaDON tool [10] is a tool providing automatized support for repairing inconsistent or incoherent ontologies using different strategies which can be chosen by the user based on the characteristics of the ontology to repair. These strategies differ in the number of explanations which are computed per unsatisfiability and can thus especially be adapted when working on large ontologies containing great numbers of unsatisfiable concepts. In contrast to Ji et al., who put special emphasis on ontology mappings, we in this work concentrate on learned ontologies for which we evaluate different repairing strategies. We present an approach similar to the one proposed by Noessner and Niepert [14] for debugging $\mathcal{EL}$ ontologies containing uncertainties which showed promising results. Our approach uses a different set of inference rules specially targeted at learned, expressive and TBox-only ontologies.

For the area of ontology matching, there is work which recourses to disjointness axioms generated by ontology learning approaches for debugging ontology mappings [13].

3 Preliminaries

In the following, we first give a short overview on the most important notions regarding incoherence in ontologies. Since two approaches presented in Section 4 are based on Markov Logic Networks, we also give a short introduction into these.

3.1 Incoherences in Ontologies

Description logic is the foundation for ontologies on the Semantic Web since the mostly used ontology language OWL is based on it. A description of all its details is given by Baader et al. [1] for those parts required for OWL in its first version. The logical features added later in OWL2 are described by Grau et al. [7]. In the following, we focus on the definitions relevant for our use case.

Since in this work we are exploring approaches to repair ontologies, i.e., make incoherent ones coherent, we first have to define the notion of incoherence. As already done previously [4], we extend the notion of incoherence, which is usually only defined for classes, to object properties. This is especially important since, with the introduction of property disjointness in OWL2, properties can get unsatisfiable more easily.

Definition 1 (Incoherence). *Given an interpretation function $\mathcal{I}$, a class or property definition D in an ontology $\mathcal{O}$ is unsatisfiable iff for each model $\mathcal{I}$ of $\mathcal{O}$ $D^{\mathcal{I}} = \varnothing$ holds. An ontology is said to be* incoherent *iff it contains at least one unsatisfiable named class or property.*

According to this definition, unsatisfiable classes or properties are equivalent to the bottom class respectively to the bottom object property. Thus, a common way to check for the unsatisfiability of a class C is to check whether $\mathcal{O} \models C \sqsubseteq \bot$. Checking an object property P for unsatisfiability can be done similarly by checking $\mathcal{O} \models \exists P.\top \sqsubseteq \bot$. For debugging purposes, it is important to detect the roots of specific errors, so-called explanations or *minimal incoherence-preserving sub-TBoxes* (MIPS).

Definition 2 (Explanation). *Given an ontology $\mathcal{O}$ and an axiom α, a subset $\mathcal{O}' \subseteq \mathcal{O}$ is an explanation for α iff $\mathcal{O}' \models \alpha$ and there exists no $\mathcal{O}'' \subset \mathcal{O}'$ with $\mathcal{O}'' \models \alpha$.*

Thus, an explanation for an axiom α is a set of axioms which implies the validity of α and cannot be further minimized by removing contained axioms. Obviously, there might be a great number of explanations for a single axiom.

3.2 Markov Logic Networks

Markov logic networks as introduced by Richardson et al. [17] are a way of formulating uncertain logical knowledge based on Markov networks. For this purpose, they extend first order logic by allowing the annotation of formulas with weights. In contrast to pure description logic where all formulas represent hard constraints and a world (an assignment to all atomic variables) not satisfying all constraints is no valid world, in Markov logic a world violating a constraint is not an impossible world but only a less probable one. The higher the weight associated to a formula the less probable a world violating it. Because of this property, it is even possible to have formulas in the knowledge base which contradict each other. Furthermore, by adding infinite weights to formulas it is possible to set these formulas as hard constraints which are not allowed to be violated.

More formally, a Markov logic network (MLN) is given by a set of pairs (F_i, w_i) where each F_i is a first-order logic formula and each w_i a real number. Together with a set of constants C the logic network can be used to determine a ground Markov network. On this Markov network it is then possible to define the probability distribution over possible worlds x by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right)$$

with F being the number of formulas in the MLN and $n_i(x)$ the number of true groundings of F_i in x. Given a world, we are able to compute its probability based on this definition.

However, as for our use case, often the more interesting scenario is to find the most likely world y given evidences e, i.e.,

$$\arg\max_y P(y|e)$$

which is also called MAP inference and is a task commonly supported by Markov logic solving tools. One possibility to solve a MAP inference problem is based on Integer Linear Programming and employed by the MAP query engine RockIt [15].

4 Approaches

In this section, we present the different approaches we examined in this work for repairing incoherent ontologies. The first two approaches are common methods which start with a diagnosis step by computing explanations for incoherences and afterwards perform a repair step to create a coherent version of the ontology. The further two approaches belong to the new family of MLN-based methods. First, we describe an approach using MLN to compute coherent ontologies based on pre-computed explanations. Then, we present a fourth approach which avoids the computation of explanations by implementing inference rules directly in Markov logic.

For the first three approaches, we assume the set of all explanations of incoherences to be given. In this work, we implemented the following methods using the TRex system [4] for gathering explanations.

A1: Baseline Approach As a baseline approach, we use Algorithm 1. It iterates over all explanations and for each explanation the axiom with the lowest confidence is removed from the ontology. Since each explanation is the minimum set of axioms causing the specific incoherence, after removing it this incoherence does no longer exist. If the currently considered explanation contains an axiom already removed when fixing an earlier explanation the current incoherence is already resolved by this removal and no further axiom removal is required. Obviously, this approach is neither optimal with respect to the number of removed axioms, which is bounded by the number of incoherence explanations in the ontology, nor with respect to the confidence values.

Algorithm 1 Greedy ontology debugging

Precondition: $\mathcal{O}$ is a learned ontology, $expl_u(\mathcal{O})$ is the set of explanations for all incoherences

```
function GreedyDebug($O, expl_u(\mathcal{O})$)
    $H \leftarrow \{\}$                                   ▷ set for storing already removed axioms
    for all $e \in expl_u(\mathcal{O})$ do                ▷ $e$ is an explanation, i.e., a set of axioms
        if $e \cap H = \varnothing$ then
            $a \leftarrow$ axiom with lowest confidence value in $e$
            $O \leftarrow O \setminus \{a\}$
            $H \leftarrow H \cup \{a\}$
```

A2: Axiom Adding Approach Algorithm 2 also uses the set of all incoherence explanations. But instead of iterating over all explanations, it iterates over learned axioms and adds them one by one starting with the highest confidence axioms and continuing with the lower confidence ones. After each axiom addition, we check whether the resulting ontology fully contains one of the original explanations which would mean that the ontology is incoherent. If so, the last axiom addition is reverted and the process continues with the axiom having the next lower confidence. This guarantees that no explanation is fully contained in the ontology and thus the occurrence of all detected incoherences is prevented.

Algorithm 2 Hitting Set

Precondition: $\mathcal{O}$ is a learned ontology, $expl_u(\mathcal{O})$ is the set of explanations for all incoherences

```
function HittingSetDebug($\mathcal{O}, expl_u(\mathcal{O})$)
    $H \leftarrow \{\}$                                   ▷ set of removed axioms
    $L \leftarrow$ learned axioms contained in $\mathcal{O}$ sorted by descending confidence
    $\mathcal{O}' \leftarrow \mathcal{O} \setminus L$     ▷ $\mathcal{O}'$ is ontology without learned axioms
    for all $a \in L$ do
        $\mathcal{O}' \leftarrow \mathcal{O}' \cup \{a\}$
        if $\exists e \in expl_u(\mathcal{O}) : e \subseteq \mathcal{O}'$ then
            $\mathcal{O}' \leftarrow \mathcal{O}' \setminus \{a\}$
            $H \leftarrow H \cup \{a\}$
```

After the termination of this algorithm, H contains a hitting set for the set of explanations, i.e., H is a set of axioms so that $\forall e \in expl_u(\mathcal{O}) : e \cup H \neq \varnothing$. It is important to note that due to the greedy nature of the algorithm, H is a minimal but not a minimum-cardinality hitting set.

A3: MAP inference based approach In contrast to the two approaches presented before, this approach is not a greedy one. Instead it uses the Markov logic-based RockIt [15] system for finding a minimum confidence set of axioms which have to be removed in order to make the learned ontology coherent. For this purpose, we first define a model as shown in Figure 1. The presented model supports explanation sets with at maximum 2 axioms per explanation but can be adapted easily and automatically for larger explanations.

```
active(axiom)
*activeConf(axiom, _float)
*conflict1(axiom)
*conflict2(axiom, axiom)
conf: active(x) v !activeConf(x, conf)
!conflict1(x0) v !active(x0).
!conflict2(x0,x1) v !active(x0) v !active(x1).
```

Fig. 1. RockIt model for approach A3

In this model, we define a way of setting axioms to active, i.e., include them into the final ontology, and furthermore give the possibility to assign confidence values to axioms. The `conflict1` and `conflict2` predicates are defined to represent explanations containing one resp. two axioms. The last three lines define that active axioms contribute towards the total confidence value and that for each set of conflicting axioms at least one has to be set to inactive.

Using this model as base, we generate the evidence for the MAP inference step by setting unlearned axioms as `active` (hard constraints) while learned axioms get their confidence value assigned using `activeConf` (soft constraints). For each generated explanation, we create a `conflict` predicate containing the identifiers of all contained axioms. The RockIt system then determines a world with the highest sum of confidence values, i.e., it gives a list of all active axioms in the most probable world which we then include into the result ontology. Since the last two lines in Figure 1 guarantee that at least one axiom for each explanation is not set to active, the result is coherent.

A4: Pure Markov Logic Approach We also considered an approach which is based purely on Markov Logic and does not require a set of explanations to be computed. This approach is highly inspired by the work of Niepert and Noessner [14] but instead of using the inference rules for the logic $\mathcal{EL}$ we implement the rules also used in TRex to perform inference and generate explanations using Markov logic.[2] This model again defines activity and activity confidence predicates but this time for each supported axiom type separately. The axioms contained in the ontology are transformed into the representing predicates and set as active if they are unlearned axioms resp. get assigned a confidence for learned axioms. Again, after applying RockIt to this data, we get a list of active axioms which are then used to build the final coherent ontology.

The approaches differ in a number of characteristics which will be the focus of our evaluation. First, the runtime of the approaches and the complexity of ontologies the approaches can be applied to is an important factor for practical usage since learned ontologies can pose challenges in both, their size and their complexity. Since ontol-

[2] The file containing the MLN formulation of the TRex rules is available at `http://web.informatik.uni-mannheim.de/trex/trex-rules-model.txt`

ogy learning approaches are mostly used for assisting humans in creating ontologies, it is also an important factor how transparent their repair steps are. More transparent approaches are better suited for being used in an interactive scenario.

5 Evaluation

In the following, we first describe the setup used to evaluate the performance of the approaches shown above. Afterwards, we describe the results of our evaluation.

5.1 Experimental Setup

For the evaluation, we worked on different ontologies all generated by means of ontology learning approaches. The first ontology, which we refer to as $\mathcal{A}$, is based on the DBpedia ontology[3] and additionally contains class disjointness axioms as generated by the GoldMiner [5] tool. We have a high-quality gold standard of class disjointness axioms for the DBpedia ontology.[4] As a second data set, we employ the one already used to evaluate the TRex tool [4]. This data set also contains axioms learned by the GoldMiner tool but instead of only being enriched by class disjointness, the ontology was enriched with additional axiom types as property disjointness or domain and range restriction axioms. The data set consists of 11 ontologies where all axioms contained in the smaller ontologies are also contained in the larger ontologies. This enables us to assess the scalability of the approaches. Furthermore, the additional learned axioms make a more demanding use case since they lead to more possibilities for incoherences. In the following, we call the 11 ontologies $\mathcal{B}_0$ to $\mathcal{B}_{10}$. Finally, we performed the experiments on an ontology fully generated from a text corpus by the Text2Onto [3] tool. This dataset was already used for similar experiments by Qi et al. [16] and is interesting for our experiments since, in contrast to the enriched DBpedia ontologies, it is a fully learned ontology which might differ considerably regarding its basic characteristics. This ontology is called $\mathcal{C}$ in the following. Table 1 summarizes the most important characteristics of all ontologies. It is also worth noting, that the first two data sets do not contain instances at all while for the third ontology, we only considered the TBox.

On these three data sets, we run the different approaches described in Section 4 and compared them regarding their runtime and the number of axioms removed from the ontology. Based on our class disjointness gold standard for the first data set, we computed the correctness of the axiom removals. For this purpose, we define correctness as also used by Qi et al. [16] as $(\#\text{ correctly removed axioms}/\#\text{ removed axioms})$. It is important to note, that by "correctness" we mean the correctness regarding human assessment not regarding their influence on the logical satisfiability. For the second DBpedia data set, an ontology engineer inspected the list of axioms removed from one

[3] Both DBpedia ontology and data set were used in version 3.7. The enriched ontology is available at `http://web.informatik.uni-mannheim.de/trex/enriched-dbpedia-ontology.zip`

[4] This gold standard has been created by three ontology engineers and will be subject of a future publication.

Table 1. Statistics about ontologies used in experiments.

Ontology	Axioms	Classes	Properties	Unsat. Classes	Unsat. Properties
$\mathcal{A}$	48,186	394	855	8	8
$\mathcal{B}_0$	23,706	300	654	3	5
$\mathcal{B}_1$	32,814	304	673	6	7
$\mathcal{B}_2$	41,941	309	689	9	14
$\mathcal{B}_3$	51,056	316	702	15	29
$\mathcal{B}_4$	60,166	319	714	26	50
$\mathcal{B}_5$	69,271	321	724	32	82
$\mathcal{B}_6$	78,375	323	730	49	112
$\mathcal{B}_7$	87,468	324	736	63	162
$\mathcal{B}_8$	96,555	324	737	83	209
$\mathcal{B}_9$	105,642	324	742	132	336
$\mathcal{B}_{10}$	114,726	324	742	152	396
$\mathcal{C}$	22,416	9,827	548	3,992	455

of the incoherent ontologies regarding their correctness. Thus, we were able to compare the performance of the approaches regarding the actual correctness of the resulting ontology. For the third data set, we only did a runtime evaluation.

All experiments were performed on a system with an Intel Core i7 3.4GHz with 32GB of RAM. As mentioned, for the Markov logic-based approaches, we used the RockIt[5] MAP query engine which in turn uses the ILP solver Gurobi[6].

5.2 Results

Applied to the first ontology, the approaches using explanations for incoherences performed similar, all of them removing the same 10 axioms from the ontology and having similar runtimes of about 40 seconds. During the evaluation of the removed axioms, the correctness turned out to be only at 0.4. Approach A4 run 12 seconds and removed only 6 axioms with a correctness of 0.

We examined the low correctness value and the high overlap regarding removed axioms and discovered that it comes from the fact that the debugged ontology has one central point of incoherence which is centered around the disjointness of organization and educational institution classes. For instance, the class `Library` is a subclass of both `Organisation` and `Building` which are marked as disjoint in the disjointness gold standard. Since the subclass axiom, which is the actual cause of the overall problem, is contained in the base ontology, the approaches are not allowed to remove it and try to find a minimal set of axioms mitigating the problem. Seemingly, the approaches based on explanation generation are not able to find a minimum cardinality set of axioms to remove, in contrast to the purely Markov logic based method. The latter however does not remove any axioms whose removal is justified according to human assessment. During its evaluation, one disadvantage of not computing explanations was discovered. Fully based on Markov logic, there is almost no possibility to

[5] `https://code.google.com/p/rockit/`, Version 0.3.228

[6] `http://www.gurobi.com/`, Version 5.6.0

reconstruct the reasons for the removal of certain axioms which makes human intervention hardly possible whereas having access to explanations enables humans to better track and understand the reasons for certain removals. In particular, this is relevant in semi-automatized ontology debugging scenarios.

For the second data set, we manually assessed the removed axioms only for ontology $\mathcal{B}_5$. Since this ontology contains more different axioms and more potential incoherences than $\mathcal{A}$, there are much more variations in the number of removed axioms and their correctness than for the first ontology. The results are given in Table 2.

Table 2. Results for approaches on ontology $\mathcal{B}_5$

Approach	Runtime	# Removed axioms	# correct axioms	correctness
A1	12,502	54	43	0.80
A2	15,029	46	40	0.87
A3	19,006	106	73	0.67
A4	23,864	98	75	0.77

The greedy approaches performed better regarding the number of removed axioms and the correctness. They only removed about half of the axioms the MLN-based approaches remove. This is probably caused by some axioms with lower confidence being removed by the MLN methods but again hard to track down because of the black box characteristics of the MLN approaches. For this smaller ontology, the greedy approaches are even better with respect to the runtime. However, the MLN-only method is more capable of handling an increasing number of axioms as shown in Figure 2. The runtimes of the explanation-based approaches increase more significantly than the MLN-only approach caused by the increasing number and size of explanations and the time required for collecting them beforehand. Furthermore, the number of explanations has a more drastic influence on approach A2 since its runtime is not linear in the number of explanations in contrast to approach A1.

The performance advantage of the MLN-only approach was even more drastically shown by the experiments on the third ontology. Having nearly 10,000 classes the explanation generation for all incoherences was not possible in reasonable time[7]. Since only approach A4 does not depend on the explanation generation, it was the only one applicable to this dataset. With a total runtime of about 32 seconds and a total number of removed axioms of 3,097 it showed a performance suitable for most practical use cases, especially when considering the high number of incoherent entities in the ontology. This qualifies the approach especially for usage on large ontologies potentially containing many incoherences and for cases where no human intervention is desired. Additionally, compared to the original results of Qi et al. [16], the MLN-only approach was able to process the whole ontology at once instead of having to add additional axioms in chunks, then checking and repairing the ontology and add another chunk of axioms. Our approach also has a lower runtime than the one reported for the original approach. Interestingly, we remove more axioms for reaching a coherent ontology. Both aspects could also be influenced by the iterative addition of axioms.

[7] We aborted the computation after one hour.

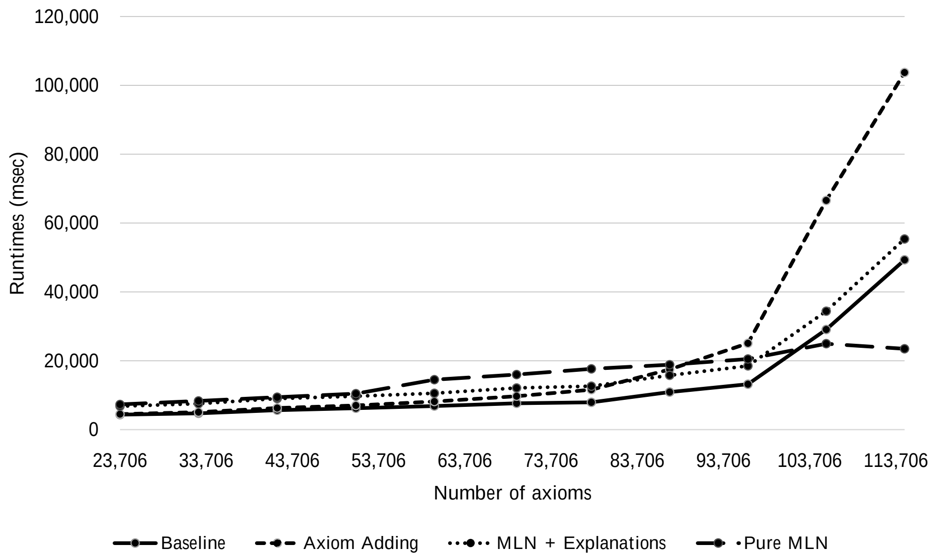


Fig. 2. Runtime behavior

6 Conclusion

In this paper, we compared four approaches regarding their performance and characteristics when used to repair learned ontologies. In particular, we concentrated on the TBox of learned ontologies. Besides traditional greedy repairing approaches, we also evaluated two approaches using Markov logic networks. The approach which did not rely on the computation of explanations but was fully MLN-based showed promising runtime and scalability characteristics. The main problem of approaches is the missing possibility to get further insights into the repair process since with circumventing the explicit diagnosis of incoherences the chances for human engineers to understand the axiom removals also decrease. This was also partly visible for the MLN-based approach which worked on the generated explanations. These discoveries seem to imply that the results of globally optimizing strategies like employed by the MLN approaches are harder to understand for humans than those of the more locally optimizing greedy approaches. However, this needs further examination and discussion. Based on the experiments presented here, we would choose approaches based on ontology diagnosis for smaller ontologies and when the full process should be interactive. Approaches like A4 seem to be qualified for larger ontologies in fully automatized scenarios.

In future work, we will integrate the most promising approaches presented here into a data debugging system which employs learned schemas for detecting data errors.

References

1. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 3–28. Intl. Handbooks on Information Systems, Springer (2004)

2. Cimiano, P., Mädche, A., Staab, S., Völker, J.: Ontology learning. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 245–267. Intl. Handbooks on Information Systems, Springer (2009)
3. Cimiano, P., Völker, J.: Text2Onto. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005, LNCS, vol. 3513, pp. 227–238. Springer (2005)
4. Fleischhacker, D., Meilicke, C., Völker, J., Niepert, M.: Computing incoherence explanations for learned ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013, LNCS, vol. 7994, pp. 80–94. Springer (2013)
5. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: Meersman, Robert et al. (ed.) OTM 2011, LNCS, vol. 7045, pp. 680–697. Springer (2011)
6. Fleischhacker, D., Völker, J., Stuckenschmidt, H.: Mining RDF data for property axioms. In: Meersman, Robert et al. (ed.) OTM 2012, LNCS, vol. 7566, pp. 718–735. Springer (2012)
7. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web 6(4), 309 – 322 (2008)
8. Horridge, M., Parsia, B., Sattler, U.: Laconic and precise justifications in OWL. In: Sheth, Amit et al. (ed.) ISWC 2008, LNCS, vol. 5318, pp. 323–338. Springer (2008)
9. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in OWL ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009, LNCS, vol. 5785, pp. 124–137. Springer (2009)
10. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmüller, S.: RaDON — repair and diagnosis in ontology networks. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009, LNCS, vol. 5554, pp. 863–867. Springer (2009)
11. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., Hendler, J.: Swoop: A web ontology editing browser. Web Semantics: Science, Services and Agents on the World Wide Web 4(2), 144 – 153 (2006)
12. Lehmann, J., Bühmann, L.: ORE - a tool for repairing and enriching knowledge bases. In: Patel-Schneider, Peter et al. (ed.) ISWC 2010, LNCS, vol. 6497, pp. 177–193. Springer (2010)
13. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning disjointness for debugging mappings between lightweight ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008, LNCS, vol. 5268, pp. 93–108. Springer (2008)
14. Noessner, J., Niepert, M.: ELOG: A probabilistic reasoner for OWL EL. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011, LNCS, vol. 6902, pp. 281–286. Springer (2011)
15. Noessner, J., Niepert, M., Stuckenschmidt, H.: Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In: des Jardins, M., Littman, M.L. (eds.) Proc. of the 27th AAAI Conference on Artificial Intelligence (2013)
16. Qi, G., Haase, P., Huang, Z., Ji, Q., Pan, J., Völker, J.: A kernel revision operator for terminologies — algorithms and evaluation. In: Sheth, Amit et al. (ed.) ISWC 2008, LNCS, vol. 5318, pp. 419–434. Springer (2008)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
18. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005, LNCS, vol. 3532, pp. 226–240. Springer (2005)
19. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51 – 53 (2007)

Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection

Heiko Paulheim

University of Mannheim, Germany
Research Group Data and Web Science
heiko@informatik.uni-mannheim.de

Abstract. Links between datasets are an essential ingredient of Linked Open Data. Since the manual creation of links is expensive at large-scale, link sets are often created using heuristics, which may lead to errors. In this paper, we propose an unsupervised approach for finding erroneous links. We represent each link as a feature vector in a higher dimensional vector space, and find wrong links by means of different multi-dimensional outlier detection methods. We show how the approach can be implemented in the RapidMiner platform using only off-the-shelf components, and present a first evaluation with real-world datasets from the Linked Open Data cloud showing promising results, with an F-measure of up to 0.54, and an area under the ROC curve of up to 0.86.

Keywords: Linked Open Data, Link Quality, Data Quality, Link Debugging, Outlier Detection

1 Introduction

Links between datasets are an essential ingredient for Linked Open Data [6]. For reasons of scalability, such interlinks are often not created manually, but generated (semi-)automatically by heuristics, which leads to occasional wrong links.

There are different reasons why link sets may contain errors. The first (and probably most frequent) reason is that the **heuristic mechanism** that creates the links does not work at an accuracy of 100%. Typical heuristic approaches for generating links combine different string metric of the entities' labels, sometimes combined with some filtering by type (e.g., only linking entities of type *Person*) [26]. Those heuristics can work well, but are not free from errors, e.g., linking two different persons which share the same name, or a river and a region with the same name. Moreover, with such heuristics, there is a trade-off between recall and precision, which leads to incorrect links. For example, [28] reports that around 20% of the links between DBpedia and Freebase are incorrect. A further problem is that the link generation heuristics are usually not re-created every time one of the linked data sources changes, thus, links may be outdated, e.g., pointing to to resources that do not exist anymore.

Another source of errors is that entities are linked which are **not exactly the same**. While in theory, entities linked by `owl:sameAs` should refer to the same real-world entity, this is not often the case, e.g., when linking a description of the company *Starbucks* to an actual Starbucks café. A study in 2010 has shown that only about half of all `owl:sameAs` actually denote two descriptions of the same real world entity [13].

In order to increase the quality of links between datasets, we propose an approach which uses multi-dimensional outlier techniques for detecting wrong links. To that end, features for each link are created, so that the link can be described as a point in a high dimensional feature space. We use outlier detection methods to find those links that are represented by points which are far from the overall distribution, assuming that those points represent wrong links.

The rest of this paper is structured as follows. In section 2, we show our approach for finding wrong links with outlier detection. In section 3, we introduce the experimental setup we used for validating our approach, and discuss the results. We conclude the paper with a review of related work in section 4, and an outlook on future work in section 5.

2 Approach

For finding wrong links with outlier detection, we first represent each link as a feature vector. Possible features are, e.g., the direct types of resources in the linked datasets, i.e., all objects of statements that have the linked resource as a subject and `rdf:type` as a predicate. A simplified example is shown in Fig. 1: two datasets contain links between artists and music works. Instances of `Song` and `Album` in dataset 1 are linked to instances of `Music Work` in dataset 2, and instances of `Artist` in dataset 1 are mapped to instances of `Music Artist` in dataset 2. It can be observed that in that feature space, there are relatively dense clusters, and single outliers (such as the one dot in the upper middle, which represents an album wrongly linked to an artist). Assuming that the majority of links between two datasets is correct, the clusters are likely to represent the correct links, while the singular outliers are likely to be wrong links.

Such singular outliers can be found by methods of *outlier* or *anomaly detection* [8, 15]. These methods automatically assign labels or scores to data points which significantly deviate from the majority of data points in the overall dataset. The outlier detection approach to be used has to be *multi-dimensional*, i.e., find data points that are abnormal w.r.t. the combination of their coordinates. In contrast, *single-dimensional* or *univariate* outlier detection methods (such as Grubbs' test or IQR) find suspicious data points in only one dimension, e.g., unusually large or small temperature values measured by a sensor. In Fig. 1, the outlying data point would not be an outlier if only considering one dimension, i.e., only the type in dataset 1 or the type in dataset 2.

To facilitate the detection of wrong links by outlier detection, our approach consists of three basic steps:

1. Read a link set, and create a feature vector representation for each link

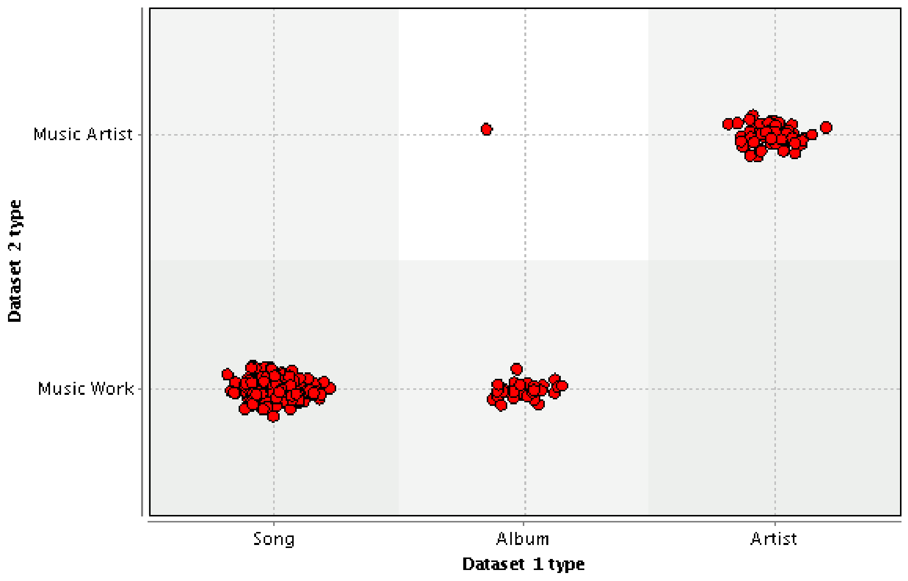


Fig. 1. A simplified example of links being represented in a vector space. The single dot in the upper middle quadrant represents a wrong link.

2. Perform outlier detection on the set of vectors, i.e., assign an outlier score to each link
3. Order the links by outlier score, and store them

In a semi-automatic setting, a user would work through the list from top to bottom until the false positive rate begins to rise above a certain limit. For fully automatic link correction, all links with an outlier score above a threshold τ would be regarded as outliers.

3 Experiments

To evaluate our approach, we have set up a process in the RapidMiner[1] platform for data mining, combining operators from the Linked Open Data extension [23] and the Anomaly Detection extension [11]. The basic pipeline is shown in Fig. 2: first, a set of links is read, e.g., from a SPARQL endpoint, and for both resources linked, features are added to the feature vector representation using the Linked Open Data extension. The resulting vector is then passed to an outlier detection algorithm, which assigns outlier scores. The output is written to a file containing pairs of resources, augmented with scores.[2]

[1] `http://www.rapidminer.com`
[2] A step-by-step explanation of how to set up such a process is shown at `http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension/rapidminer-lod-extension-example-discovering-wrong-links-between-datasets/`

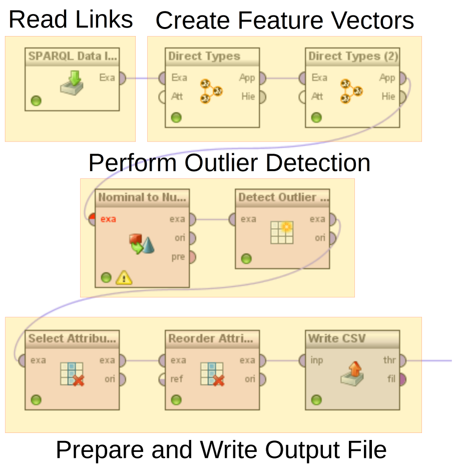


Fig. 2. The implementation of our approach in the RapidMiner platform

3.1 Feature Vector Creation

We examine two different strategies of creating feature vectors:

– *Using all direct types.* A binary feature is created for each schema class, which is set to `true` for a link if the linked resource has the class defined as its `rdf:type`.
– *Using all ingoing and outgoing properties.* Two binary features are created for each data and object property, which are set to `true` if the linked resource is the subject resp. the object of a triple which uses the property as its predicate.

The same feature creation technique is applied to each of the two linked resources, where distinct features are created for both resources. Furthermore, we examine the union of both feature sets.

3.2 Datasets

We evaluate our approach on two link sets between three datasets of the Linked Open Data cloud. The three datasets are:

– *DBpedia*, a cross-domain dataset created from Wikipedia infoboxes [20].
– *Peel Sessions*, a dataset describing the John Peel Sessions at BBC, the artists involved, and the songs performed [25].
– *DBTropes*, a dataset collecting information about movies, TV shows, computer games, and books, among others, as well as tropes used in those [17].

For DBpedia, we use the mapping-based types and the mapping-based properties datasets of the 3.9 release[3]. For the Peel Sessions dataset, we use the dump

[3] `http://wiki.dbpedia.org/Downloads39`

Table 1. Sizes of feature vectors for the different link sets

Dataset	Peel Session	DBpedia	DBTropes	DBpedia
# Links	2,087		4,229	
# Types	3	31	2	79
# Properties	4	56	18	124

available at the web site[4]. For the DBTropes dataset, which provides daily snapshots, we use a snapshot obtained on November 8th, 2013.[5]

Both the Peel Sessions and the DBTropes data set are linked to DBpedia. The Peel Sessions dataset contains 2,087 `owl:sameAs` links to DBpedia, the DBTropes dataset contains 4,229 `owl:sameAs` links to DBpedia. While the Peel Sessions dataset is rather restricted to the type of entities it links (in particular: artists and songs), DBTropes contains a larger variety of entities, including general concepts such as *Celtic Mythology*.

Besides random links two homonymous resources (e.g., the TV series *Material Girl* and the Madonna song), one typical source of errors is the linking of instances derived from disambiguation pages (both DBpedia and DBTropes, which is also derived from a Wiki, have such instances). A typical source of errors for the Peel Session dataset is the linking of songs to albums with the same name. Furthermore, the Peel Sessions dataset links different persons of the same name – e.g., a blues musician named *Jimmy Carter* to the U.S. president.

Table 1 depicts the sizes of the feature vectors for both link sets, i.e., the number of classes and properties used for the elements that are mapped. The counts of DBpedia classes and properties, show that the variety of objects linked from DBTropes is higher. Furthermore, it is noteworthy that although DBTropes uses two classes, one of those is only used for two objects, while the remaining 4,219 instances have the class `TVTItem`, which is only a generic class comparable to `owl:Thing`. The properties used in the dataset are similarly generic.

For our experiment, we have randomly sampled 100 links from both link sets, and manually evaluated them for correctness, thus creating small partial gold standards. From the Peel Session link set, 90 out of the 100 links are correct, for the DBTropes link set, 76 out of the 100 links are correct. For the gold standard, we use a strict definition of `owl:sameAs`, e.g., a book and its protagonist are not considered the same, neither are a book and a movie based on that book.

3.3 Outlier Detection Methods

To detect outliers, we compare six different multi-dimensional outlier detection methods. For all outlier detection methods, we use the implementation in the RapidMiner Anomaly Detection extension, and used the default parameters unless specified otherwise.

- The *k-NN global anomaly score (GAS)* is the average distance to the k nearest neighbors [5], following the intuition that outliers are located in rather

[4] `http://dbtune.org/bbc/peel/`, downloaded on November 6th, 2013

[5] `http://skipforward.opendfki.de/wiki/DBTropes`

sparsely populated areas of the vector space (cf. Fig. 1). Since values for k between 10 and 50 are recommended [11], we compute a GAS with $k = 10$, $k = 25$, and $k = 50$.

– The *Local Outlier Factor (LOF)* is computed from the density of data points around the point under inspection, which in turn is computed from the distances to the k nearest neighbors [7]. Since the algorithm allows the setting of a minimum and a maximum k, we use $k_{min} = 10$ and $k_{max} = 50$ following the recommendation above.
– The *Local Outlier Probability (LoOP)* follows a similar idea as LOF, but maps the outlier scores to probabilities in a $[0; 1]$ interval (the scores assigned by other methods are usually unbound) [19]. Like for GAS, we compute LoOP with $k = 10$, $k = 25$, and $k = 50$.
– The *Cluster-based Local Outlier Factor (CBLOF)* uses the output of a clustering algorithm. It follows the intuition that outliers are located outside of larger clusters, and thus assigns an outlier score based on the size of the cluster in which a data point is located, and the distance to the next large cluster [14]. According to the recommendation in [11], we set the α value to the expected percentage of correct instances, i.e., 0.90 for the Peel dataset, and 0.76 for the DBtropes dataset.[6] As a clustering algorithm, we use the *X-means* algorithm, which restarts k-means with different values for k, in order to find an optimal one [24]. For the X-means clustering, we set $k_{min} = 2$ and $k_{max} = 60$.
– The *Local Density Cluster-based Outlier Factor (LDCOF)* works similar to CBLOF, but also takes the local density of the cluster into account [3]. We again use it together with X-means in the same configuration as above.
– *One-Class Support Vector Machines* aim at training a support vector machine covering only positive examples, so that the majority of data points is separated from the rest. In our experiment, we use one-class SVMs with a robust kernel defined particularly for outlier detection [4].

Most of the above methods (including the clustering algorithm) require the definition of a distance function. Here, we use *cosine similarity*, since we want two links to be more similar if they share a feature (both are of type `Person`), but not if they share the absence of a feature (e.g., both are *not* of type `City`). In contrast, other distance functions, such as Euclidean distance, would weigh the shared presence and absence of a feature equally.

3.4 Results

We have tested each of the above outlier detection methods with three different feature groups – direct types, properties, and the combination of both – on both datasets, performing a total of 60 runs of the approach. The results are depicted in table 2. We report the area under the ROC curve (AUC), the best F1-measure

[6] Strictly speaking, setting these values according to observations on a labeled sample of the data makes the approach using CBLOF no longer fully supervised.

Table 2. Results on both datasets using different feature sets and methods. For each dataset, the top three AUC and F-measure values are marked in bold.

Dataset	Peel				DBTropes			
Features / Method	AUC	F1	τ	total	AUC	F1	τ	total
types								
GAS (k=10)	0.353	0.185	1.414	2,049	0.404	0.390	0.000	4,088
GAS (k=25)	0.341	0.182	0.476	2,071	0.424	0.390	0.000	4,009
GAS (k=50)	0.341	0.182	0.478	2,071	0.422	0.390	0.000	3,943
LOF	0.753	0.454	0.953	1,843	**0.619**	**0.500**	1.084	3,025
LoOP (k=10)	0.749	0.454	0.311	1,834	0.413	0.412	0.000	1,636
LoOP (k=25)	**0.803**	**0.500**	0.378	1,181	**0.581**	**0.488**	0.143	2,978
LoOP (k=50)	**0.803**	**0.500**	0.378	1,181	**0.581**	**0.488**	0.920	2,969
CBLOF	0.754	**0.537**	245.423	1,051	0.413	0.404	0.000	1,498
LDCOF	0.696	0.432	0.953	1,352	0.410	0.404	0.000	1,498
1-class SVM	**0.857**	0.471	2.689	1,514	0.456	0.421	3.712	1,795
properties								
GAS (k=10)	0.341	0.182	0.955	2,059	0.411	0.387	0.000	786
GAS (k=25)	0.344	0.182	0.969	2,046	0.405	0.387	0.000	563
GAS (k=50)	0.381	0.182	0.000	663	0.391	0.387	0.000	461
LOF	0.516	0.217	1.102	1,225	0.529	0.424	0.984	1,006
LoOP (k=10)	0.364	0.222	0.156	1,810	0.510	0.425	0.076	2,037
LoOP (k=25)	0.438	0.250	0.706	1,992	0.422	0.387	0.000	1,060
LoOP (k=50)	0.452	0.235	0.531	1,966	0.489	0.411	0.000	1,012
CBLOF	0.402	0.189	68.426	426	0.496	0.400	197.739	254
LDCOF	0.516	0.208	1.013	1,509	0.428	0.390	0.619	276
1-class SVM	0.360	0.189	2.000	426	0.378	0.387	2.000	200
all								
GAS (k=10)	0.331	0.200	0.553	1,942	0.412	0.387	0.000	785
GAS (k=25)	0.349	0.200	0.591	1,927	0.407	0.387	0.000	562
GAS (k=50)	0.440	0.222	0.520	1,529	0.390	0.387	0.000	460
LOF	0.638	0.280	1.105	1,002	0.481	0.400	1.010	567
LoOP (k=10)	0.454	0.333	0.802	2,063	0.547	0.420	0.064	1,881
LoOP (k=25)	0.430	0.250	0.478	2,004	0.445	0.388	0.000	1,065
LoOP (k=50)	0.378	0.235	0.473	1,980	0.502	0.420	0.008	1,253
CBLOF	0.313	0.189	25.302	235	0.366	0.403	223.036	240
LDCOF	0.530	0.250	1.326	1,876	0.467	0.390	0.632	272
1-class SVM	0.303	0.180	2.000	237	0.353	0.387	2.000	199

that can be achieved, the threshold τ that has to be set on the outlier score in order to achieve that F1-measure, and the total number of outliers that are identified at that threshold.

Multiple observations can be made from the table. First, in particular in terms of AUC, the results on the Peel dataset are much better than those on the DBTropes dataset. There are two main reasons for that: on the one hand, the schema used in the Peel dataset is more fine-grained than that of the DBTropes dataset, where the latter essentially has only major class, which is `TVTItem`. Second, with around 24%, the fraction of outliers on the DBTropes dataset is rather large, and larger than the amount of outliers many outlier detection methods are built for. This can be observed very well on the results for the 1-class SVM method, which reaches the best AUC on the Peel dataset, but performs only average on the DBTropes dataset.

Second, using only the type features works best, and the results do not improve when combining both feature sets. As shown in table 1, the number of features created from direct types is much smaller than that created from relations, i.e., the outlier detection problem to be solved has a much lower dimensionality. A large number of dimensions, however, is a problem for many outlier detection methods, in particular those based on nearest neighbor methods. The combination of type features and LoOP yields good results, with an AUC of 0.803 and 0.581, respectively, while the optimal results are achieved by the 1-class SVM (AUC=0.857) and CBLOF (F1=0.537) on the Peel dataset, and by LOF (AUC=0.619, F1=0.5) on the DBTropes dataset. The absolute numbers of identified outliers for the optimal F1 show that in those cases, the F1 is optimized mainly because of a high recall value, flagging up to three quarters of all links as outliers. This shows that selecting an optimal configuration is difficult.

In order to obtain a more fine-grained picture of the differences between the approaches, figures 3 and 4 show the ROC curves of all approaches, using only type features. It shows that in particular the LoOP approaches show very good results on both datasets. The steep ascend of the respective ROC curves show that there are five actually wrong links among the top 10 identified outliers.

The approach runs very fast in most cases. While the creation of feature vectors strongly depends on the data access method (e.g., working with a public SPARQL endpoint over the internet is much slower than using a local dump), the outlier detection itself takes less than 10 seconds on both datasets for all the methods used in our experiments. The only exceptions are the clustering-based methods, where the clustering can take up to 30 seconds, and most dominantly the One-Class SVM method, which can take up to 15 minutes.

4 Related Work

In this paper, we have analyzed the use of multi-dimensional outlier detection for finding erroneous links. This work is orthogonal to the approach sketched in [27], where we use outlier detection in a one-dimensional setting to find wrong numeric literals in DBpedia.

While a larger body of work is concerned with automatically *creating* links, there are not too many approaches that try to automatically find errors in links between datasets. Moreover, most approaches discussed so far assume some prior knowledge about the datasets, e.g., links on the schema level.

[12] use a set of five network metrics, such as degree and centrality, to predict typical properties of nodes in two interlinked datasets, as well as try to find wrongly linked resources. They report a recall of 0.68 and a precision of 0.49 (although on a different dataset), i.e., a result quality comparable to the approach discussed in this paper. In [9], links between more than two datasets are exploited to find the set of `owl:sameAs` that minimize the contradictions. The authors

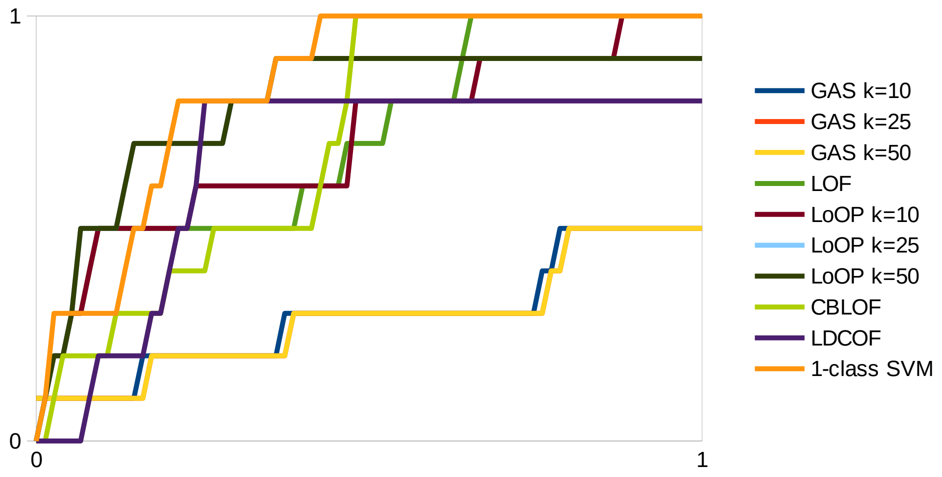


Fig. 3. ROC curve of the results on the Peel dataset, using only type features. The three curves for GAS are mostly identical; so are LoOP for $k = 25$ and $k = 50$.

show that they are capable of identifying a significant amount of contradictions, however, they do not state the precision of their approach. A similar problem is addressed in [10], where the authors aim at finding the most coherent set of links from a set of possible link candidates.

An approach using statistical distributions of properties, such as average degrees, is discussed in [16]. Like our approach, the authors compute confidence scores for `owl:sameAs` links. However, there is a fundamental difference: the authors expect the same schema to be used by both linked resources. In contrast, our approach can cope with entities using different schemas. The two link sets used in this paper could not have been processed with such an approach expecting the same schema for both linked datasets.

The *Databugger* framework allows for finding typical patterns of wrong and/or incomplete data, formulated as SPARQL queries [18]. The key difference is that, while Databugger relies on schema information (e.g., `owl:equivalentClass` definitions), our approach is agnostic with respect to the schemas used in the datasets at hand. In [1], a crowd sourcing approach is introduced for evaluating the quality of interlinks between datasets. While a considerable precision of 0.94 is achieved using majority voting over Amazon MTurk tasks, the results are not directly comparable, since the approach discussed in this paper works fully automatically and unsupervised, while the authors exploit the wisdom of the crowd. In [2], an approach is discussed for assessing the completeness of link sets, based on manually defined schema mappings. This is complementary to our work, which is concerned with *correctness*, not *completeness*.

The approaches in this paper focus on debugging link sets between individuals, i.e., links on the A-box level. A related problem is the debugging of schema mappings, i.e., links on the T-box level. Here, reasoning based approaches are frequently used [21]. While reasoning would also be a possible approach for A-box level link set debugging, the problems here are scalability and missing expressiv-

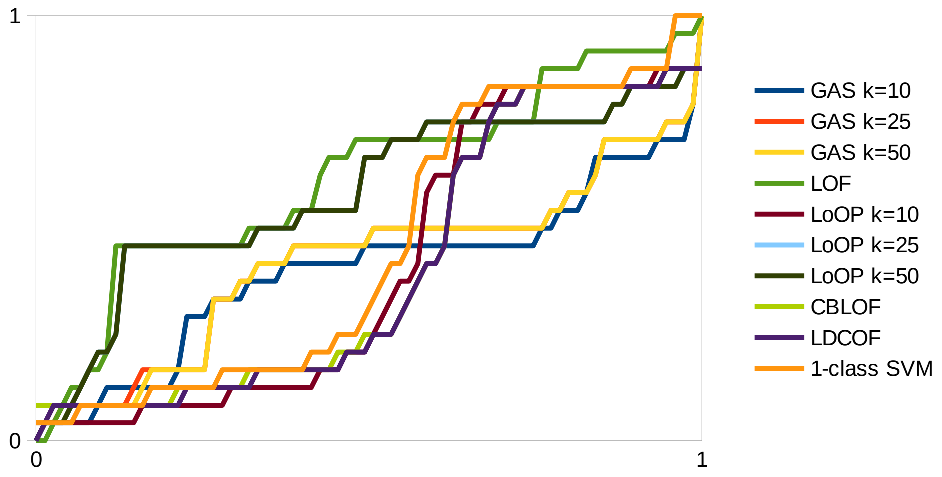


Fig. 4. ROC curve of the results on the DBTropes dataset, using only type features. The curves for GAS $k = 25$ and $k = 50$ are mostly identical; so are the curves for LoOP with $k = 25$ and $k = 50$, and the curves for CBLOF and LDCOF.

ity of the schemas used for Linked Open Data, and the A-box data often being too noisy for reasoning to yield useful results [22].

5 Conclusion and Outlook

In this paper, we have presented an approach for finding wrong links between datasets, which uses multi-dimensional outlier detection techniques. An evaluation on two datasets has shown promising results, with an area under the ROC curve up to 0.86 (i.e., wrong links get lower scores than correct links with a probability of 86%), and an F-measure up to 0.54. The approach is scalable, as it processes link sets between real datasets from the LOD cloud in a few seconds to a few minutes, depending on the configuration used.

Although the datasets used for evaluation only use `owl:sameAs` links, it can be applied to all sorts of datasets interlinks, the approach is not limited to a particular type of links. It may also be used, e.g., on a dataset of persons linked to a dataset of locations using `foaf:basedNear` links, or even for finding wrong instantiations of any property *within* a single dataset.

Given the amount of work that has been done in supervised or active learning of dataset interlinks, a link validation method such as the one introduced in this paper could be an interesting counterpart to be used in such learning systems. Given that the features used for learning and for validating the links are different, our method could provide a direct feedback loop for refining the learned links.

In essence, there are two basic degrees of freedom in our approach: the strategy for creating feature vectors, and the outlier detection algorithm (and its parametrization). With respect to feature vectors, we have experimented with direct types and properties so far. A further option are qualified relations, as discussed in [23], which, however, may impose scalability issues. Network measures, as discussed in some related works, are an interesting option for generating possible features, and domain or dataset specific features, such as Wikipedia categories for DBpedia, may also be csonidered. Furthermore, since many outlier detection algorithms experience problems in higher dimensional spaces, applying feature selection might be a useful preprocessing step, which, however, has to be taken with great care, since particularly in our setting, the very sparse features (which are likely to be eliminated by many feature selection approaches) are often those which are well suited for finding outliers.

As far as the selection of outlier detection methods is concerned, we have observed some trends, in particular that Local Outlier Factor, Local Outlier Probabilities, and 1-class SVMs perform quite well, however, especially the latter two need to be carefully parametrized. Since many automatic parameter tuning methods rely on a supervised rather than an unsupervised setting, it might be an interesting option to wrap our approach in a semi-supervised setting, using a small set of labeled links for automatic parameter tuning.

References

1. Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *International Semantic Web Conference (ISWC)*, 2013.
2. Riccardo Albertoni and Asunción Gómez Pérez. Assessing linkset quality for complementing third-party datasets. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 52–59. ACM, 2013.
3. Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*, pages 1–12, 2012.
4. Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15. ACM, 2013.
5. Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *Principles of Data Mining and Knowledge Discovery*, pages 15–27, 2002.
6. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
7. Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. 29(2):93–104, 2000.
8. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), 2009.
9. Gerard de Melo. Not quite the same: Identity constraints for the web of linked data. In *Proceedings of the American Association for Artificial Intelligence*, 2013.
10. Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *Extended Semantic Web Conference*, 2014.
11. Markus Goldstein. Anomaly detection. In *RapidMiner – Data Mining Use Cases and Business Analytics Applications.* 2014.

12. Christophe Guéret, Paul Groth, Claus Stadler, and Jens Lehmann. Assessing linked data mappings using network measures. In *Extended Semantic Web Conference (ESWC)*, pages 87–102. Springer, 2012.
13. Harry Halpin, PatrickJ. Hayes, JamesP. McCusker, DeborahL. McGuinness, and HenryS. Thompson. When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In *The Semantic Web – ISWC 2010*, pages 305–320. Springer Berlin Heidelberg, 2010.
14. Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.
15. Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
16. Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. Some entities are more equal than others: statistical methods to consolidate linked data. In *4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010)*, 2010.
17. Malte Kiesel and Gunnar Aastrand Grimnes. Dbtropes—a linked data wrapper approach incorporating community feedback. In *EKAW 2010 Demo & Poster Abstracts. International Conference on Knowledge Engineering and Knowledge Management (EKAW-10), 17th International Conference on Knowledge Engineering and Knowledge Management, October 11-15, Lisbon, Portugal*, 2010.
18. Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, and Roland Cornelissen. Test-driven evaluation of linked data quality. 2014.
19. Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652. ACM, 2009.
20. Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.
21. Christian Meilicke. *Alignment Incoherence in Ontology Matching*. University of Mannheim, 2011.
22. Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *12th International Semantic Web Conference (ISWC)*, 2013.
23. Heiko Paulheim and Johannes Fürnkranz. Unsupervised Generation of Data Mining Features from Linked Open Data. In *International Conference on Web Intelligence, Mining, and Semantics (WIMS'12)*, 2012.
24. Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.
25. Yves Raimond, Christopher Sutton, and Mark B Sandler. Automatic interlinking of music datasets on the semantic web. In *Linked Data on the Web*, 2008.
26. Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *The Semantic Web - ISWC 2009*, pages 650–665. Springer Berlin Heidelberg, 2009.
27. Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *Extended Semantic Web Conference*, 2014.
28. Amrapali Zaveri, Dimitris Kontokostas, Mohamed A Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *9th International Conference on Semantic Systems (I-SEMANTICS '13)*, 2013.

Interactive Ontology Debugging using Direct Diagnosis

Kostyantyn Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss

Alpen-Adria Universität, Klagenfurt, 9020 Austria, email: firstname.lastname@aau.at

Abstract. Sequential diagnosis methods compute a series of queries for discriminating between diagnoses. Queries are answered by some oracle such that eventually the set of faults is identified. The computation of queries is based on the generation of a set of *most probable* diagnoses. However, in diagnosis problem instances where the number of minimal diagnoses and their cardinality is high, even the generation of a set of minimum cardinality diagnoses is unfeasible with the standard conflict-based approach. In this paper we propose to base sequential diagnosis on the computation of *some* set of minimal diagnoses using the direct diagnosis method, which requires less consistency checks to find a minimal diagnosis than the standard approach. We study the application of this direct method to high cardinality faults in ontologies. In particular, our evaluation shows that the direct method results in almost the same number of queries for cases when the standard approach is applicable. However, for the cases when the standard approach is not applicable, sequential diagnosis based on the direct method is able to locate the faults correctly.

1 Introduction

Standard sequential model-based diagnosis (MBD) methods [18, 15] acquire additional information in order to discriminate between diagnoses. Queries are generated and answered either by automatic probing or by asking humans for additional observations about the system to be diagnosed. As various applications show, the standard methods work very satisfactorily for cases where the number of faults is low (single digit number), consistency checking is fast (single digit number of seconds), and sufficient possibilities for observations are available for discriminating between diagnoses.

MBD is a general method which can be used to find errors in hardware, software, knowledge-bases, orchestrated web-services, configurations, etc. In particular, OWL ontology debugging tools [14, 7, 10] can localize a (potential) fault by computing sets of axioms $\mathcal{D} \subseteq \mathcal{O}$ called *diagnosis* for an ontology $\mathcal{O}$. At least all axioms of a diagnosis must be modified or deleted in order to formulate a fault-free ontology $\mathcal{O}^*$. The latter is faulty if some requirements, such as consistency of $\mathcal{O}$, presence or absence of specific entailments, are violated.

All the discrimination and diagnosis approaches listed above follow the standard MBD technique [18] and compute diagnoses using minimal conflict sets, i.e. irreducible sets of axioms $CS \subseteq \mathcal{O}$ that violate some requirements, by using a consistency checker (black-box approach). Furthermore, diagnoses are ordered and filtered by some preference criteria, e.g. probability or cardinality, in order to focus debugging on the most likely cases.

In the common ontology development scenario where a user develops an ontology manually, the changes between validation steps, e.g. consistency checking, are rather small. Therefore, the number of faulty axioms is in a range where standard sequential MBD methods are applicable [20]. However, there are cases when the changes are substantial. For example, in ontology matching two ontologies with several thousands of axioms are merged into a single one. High quality matchers (e.g. [12]) require the diagnosis of such merged ontologies, but often cannot apply standard MBD methods because of the large number of minimum cardinality diagnoses and their high cardinality (e.g. greater than 20). This observation is supported by analysis of justifications [11], which is a dual problem to computation of diagnoses. Moreover, most of the diagnostic problems are NP-complete even if reasoning is done in polytime [4, 17].

In order to deal with hard diagnosis instances, we propose to relax the requirement for sequential diagnosis to compute a set of *preferred* diagnoses, such as a set of most probable diagnoses. Instead, we compute *some* set of diagnoses which can be employed for query generation. This allows to use the direct computation of diagnoses [19] *without* computing conflict sets. The direct approach was applied for non-interactive diagnosis of ontologies [3, 2] and constraints [6]. The computation of a diagnosis $\mathcal{D}$ by a variant of QUICKXPLAIN [13] requires $O(|\mathcal{D}| \log(\frac{|\mathcal{O}|}{|\mathcal{D}|}))$ consistency checks, where $|\mathcal{D}|$ is the cardinality of the diagnosis and $|\mathcal{O}|$ the size of the knowledge base. If m diagnoses are required for query generation, then only m calls to a direct diagnosis generator are needed. A recent approach [21] does not generate the standard HS-TREE, but still depends on the minimization of conflict sets, i.e. $|\mathcal{D}|$ minimized conflicts have to be discovered. Consequently, if $|\mathcal{D}| \gg m$, substantially more consistency checks are required.

Since we are replacing the set of most probable diagnoses by just a set of diagnoses, some important practical questions have to be addressed. (1) Is a substantial number of additional queries needed, (2) is this approach able to locate the faults, and (3) how efficient is this approach?

In order to answer these questions we have exploited the most difficult diagnoses problems of the ontology alignment competition [5]. Our evaluation shows that sequential diagnosis by direct diagnosis generation needs approximately the same number of queries (± 1) in order to identify the faults. This evaluation was carried out for cases where the standard sequential diagnosis method was applicable. Furthermore, the evaluation shows that our proposed direct method is capable of locating faults in all cases correctly. Moreover, for the hardest instance the computation costs which are introduced in addition to the computational costs of theorem proving are less than 7%.

The remainder of the paper is organized as follows: Section 2 gives a brief introduction to the main notions of sequential ontology diagnosis. The details of the suggested algorithms and their applications are presented in Section 3. In Section 4 we provide evaluation results.

2 Basic concepts

In the following we present (1) the fundamental concepts regarding the diagnosis of ontologies and (2) the interactive localization of axioms that must be changed.

Diagnosis of ontologies. Given an ontology $\mathcal{O}$ which is a set of logical sentences (axioms), the user can specify particular requirements during the knowledge-engineering process. The most basic requirement is consistency, i.e. a logical model exists. A further frequently employed requirement is coherence. In addition, as it is common practice in software engineering, the knowledge-engineer (user for short) may specify test cases, which are axioms which must (not) be entailed by a valid ontology.

Given a set of axioms P (positive test cases) and a set of axioms N (negative test cases), an ontology $\mathcal{O}^*$ is valid iff $\mathcal{O}^*$ is consistent (and coherent if required) and

1. $\mathcal{O}^* \models p$ *for all* $p \in P$
2. $\mathcal{O}^* \not\models n$ *for all* $n \in N$

Let us assume that there is a non-valid ontology $\mathcal{O}$, then a set of axioms $\mathcal{D} \subseteq \mathcal{O}$ must be removed and possibly some axioms EX must be added by a user s.t. an updated $\mathcal{O}^*$ becomes valid, i.e. $\mathcal{O}^* := (\mathcal{O} \setminus \mathcal{D}) \cup EX$. The goal of diagnosis is to provide information which sets of axioms $\mathcal{D}$ should be revised in order to formulate a valid ontology. Furthermore, we allow the user to define a set of axioms $\mathcal{B}$ (called the background theory) which must not be changed (i.e. the correct axioms).

Definition 1. *Let $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ be a diagnosis problem instance (DPI) where $\mathcal{O}$ is a ontology, $\mathcal{B}$ a background theory, P a set of axioms which must be implied by a valid ontology $\mathcal{O}^*$, and N a set of axioms which must* not *be implied by $\mathcal{O}^*$.*

A set of axioms $\mathcal{D} \subseteq \mathcal{O}$ is a candidate diagnosis iff the set of axioms $\mathcal{O} \setminus \mathcal{D}$ can be extended by a set of logical sentences EX such that:

1. *$(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup EX$ is consistent (and coherent if required)*
2. *$(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup EX \models p$ for all $p \in P$*
3. *$(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup EX \not\models n$ for all $n \in N$*

$\mathcal{D}$ is a diagnosis *iff there is no $\mathcal{D}' \subset \mathcal{D}$ such that $\mathcal{D}'$ is a candidate diagnosis. $\mathcal{D}$ is a* minimum cardinality diagnosis *iff there is no diagnosis $\mathcal{D}'$ such that $|\mathcal{D}'| < |\mathcal{D}|$.*

The following proposition of [20] characterizes diagnoses by replacing EX with the positive test cases.

Corollary 1. *Given a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$, a set of axioms $\mathcal{D} \subseteq \mathcal{O}$ is a diagnosis iff $(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup \{\bigwedge_{p \in P} p\}$ is consistent (coherent) and $\forall n \in N \; : \; (\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup \{\bigwedge_{p \in P} p\} \not\models n$*

In the following we assume that there is always a diagnosis.

Proposition 1. *A diagnosis $\mathcal{D}$ for a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ exists iff $\mathcal{B} \cup \{\bigwedge_{p \in P} p\}$ is consistent (coherent) and $\forall n \in N \; : \; \mathcal{B} \cup \{\bigwedge_{p \in P} p\} \not\models n$*

For the computation of diagnoses *conflict sets* are usually employed to constrain the search space. A conflict set is the part of the ontology that preserves the inconsistency/incoherency.

Definition 2. *Given a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$, a set of axioms $CS \subseteq \mathcal{O}$ is a conflict set iff $CS \cup \mathcal{B} \cup \{\bigwedge_{p \in P} p\}$ is inconsistent (incoherent) or there is an $n \in N$ s.t. $CS \cup \mathcal{B} \cup \{\bigwedge_{p \in P} p\} \models n$. CS is a minimal conflict set iff there is no $CS' \subset CS$ such that CS' is a conflict set.*

Minimal conflict sets can be used to compute the set of diagnoses as it is shown in [18]. The idea is that each diagnosis should include at least one element of each minimal conflict set.

Proposition 2. *$\mathcal{D}$ is a diagnosis for the DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ iff $\mathcal{D}$ is a minimal hitting set for the set of all minimal conflict sets of the DPI.*

Generation of a minimal conflict set is done by specific algorithms such as the divide-and-conquer method QUICKXPLAIN (QX) [13]. In the worst case, QX requires $O(|CS| \log(\frac{|\mathcal{O}|}{|CS|}))$ calls to the reasoner, where CS is the returned minimal conflict set.

The computation of diagnoses in ontology debugging systems is implemented using Reiter's Hitting Set HS-TREE algorithm [18]. The algorithm constructs a directed tree from the root to the leaves, where each non-leave node is labeled with a minimal conflict set and leave nodes are labeled by ✓ (*no conflicts*) or $\times$ (*pruned*).

Each (✓) node corresponds to a diagnosis. The subset minimality of the diagnoses is guaranteed by the minimality of conflict sets used for labeling the nodes, the pruning rule and the breadth-first strategy for tree generation [18]. Moreover, because of the breadth-first strategy the diagnoses are generated in increasing order of their cardinality. Under the assumption that diagnoses with lower cardinality are more probable than those with higher cardinality, HS-TREE generates most probable diagnoses first.

Diagnoses discrimination. For many real-world DPIs, an ontology debugger can return a large number of diagnoses. Each diagnosis corresponds to a different set of axioms that must be changed in order to formulate a valid ontology. The user may extend the test cases P and N s.t. diagnoses are eliminated, thus identifying exactly those axioms that must be changed. That is, we assume that the user (oracle) is equipped with sufficient knowledge about the valid ontology $\mathcal{O}^*$ such that axiom Q can be classified either as entailed by $\mathcal{O}^*$ or not. If a user finds that Q must be entailed by $\mathcal{O}^*$, then it is added to the set P yielding the new DPI $\langle \mathcal{O}, \mathcal{B}, P \cup \{Q\}, N \rangle$, and to N, i.e. $\langle \mathcal{O}, \mathcal{B}, P, N \cup \{Q\} \rangle$, otherwise. According to Definition 1, any diagnosis of the original DPI is not a diagnosis of an updated DPI if it violates any of its test cases. Moreover, in case $Q \in \mathcal{O}$, each diagnosis of an updated DPI must comprise Q if $Q \in N$ and not comprise Q if $Q \in P$.

Property 1. Given set of diagnoses $\mathbf{D}$ for a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ and an axiom Q representing the oracle query $\mathcal{O}^* \models Q$. If the oracle gives the answer *yes* then every diagnosis $\mathcal{D}_i \in \mathbf{D}$ is a diagnosis for $\langle \mathcal{O}, \mathcal{B}, P \cup \{Q\}, N \rangle$ iff both conditions hold:

$$(\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup \{ \bigwedge_{p \in P} p \} \cup \{Q\} \textit{ is consistent (coherent)}$$

$$\forall n \in N \ : \ (\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup \{ \bigwedge_{p \in P} p \} \cup \{Q\} \not\models n$$

If the oracle gives the answer *no* then every diagnosis $\mathcal{D}_i \in \mathbf{D}$ is a diagnosis for $\langle \mathcal{O}, \mathcal{B}, P, N \cup \{Q\} \rangle$ iff both conditions hold:

$$(\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup \{ \bigwedge_{p \in P} p \} \textit{ is consistent (coherent)}$$

$$\forall n \in (N \cup \{Q\}) \ : \ (\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup \{ \bigwedge_{p \in P} p \} \not\models n$$

However, many different queries might exist for some set of diagnoses $|\mathbf{D}| > 2$, in the extreme case exponentially many (in $|\mathbf{D}|$). To select the best query, the authors in [20] suggest two query selection strategies: SPLIT-IN-HALF (SPL) and ENTROPY (ENT). The first strategy is a greedy approach preferring queries which allow to remove half of the diagnoses in $\mathbf{D}$, for both answers to the query. The second is an information-theoretic measure, which estimates the information gain for both outcomes of each query and returns the query that maximizes the expected information gain. The *prior fault probabilities* required for evaluating the ENTROPY measure can be obtained from statistics of previous diagnosis sessions. For instance, if the user has problems to apply "$\exists$", then the diagnosis logs are likely to contain more repairs of axioms including this quantifier. Consequently, the prior fault probabilities of axioms including "$\exists$" should be higher. Given the fault probabilities of axioms, one can calculate prior fault probabilities of diagnoses as well as evaluate ENTROPY (see [20] for more details). The queries for both strategies are constructed by exploiting so called classification and realization services provided by description logic reasoners. Given a ontology $\mathcal{O}$ the classification generates the subsumption hierarchy, i.e. the entailments $\mathcal{O} \models A \sqsubseteq B$, where B is the most specific concept that subsumes A. Realization computes, for each individual name t occurring in an ontology $\mathcal{O}$, a set of most specific concepts A s.t. $\mathcal{O} \models A(t)$ (see [1] for details).

Due to the number of diagnoses and the complexity of diagnosis computation, not all diagnoses are exploited for generating queries but a set of (most probable) diagnoses of size less or equal to some (small) predefined number m [20]. We call this set the *leading diagnoses* and denote it by $\mathbf{D}$ from now on. The set of leading diagnoses $\mathbf{D}$ acts as a representative of the set of all diagnoses.

The *standard* sequential ontology debugging process can be sketched as follows. As input a DPI and some meta information, i.e. prior fault estimates $\mathcal{F}$, a query selection strategy s_Q (SPL or ENT) and a stop criterion σ, are given. As output a diagnosis is returned that has a posterior probability of at least $1 - \sigma$. For sufficiently small σ this means that the returned diagnosis is highly probable whereas all other leading diagnoses are highly improbable.

1. Using QX and HS-TREE (re-)calculate a set of leading diagnoses $\mathbf{D}$ of cardinality $\min(m, a)$, where a is the number of all diagnoses for the DPI and m is the number of leading diagnoses predefined by a user.
2. Use the prior fault probabilities $\mathcal{F}$ and the already specified test cases to compute (posterior) probabilities of diagnoses in $\mathbf{D}$ by the Bayesian Rule (cf. [20]).
3. If some diagnosis $\mathcal{D} \in \mathbf{D}$ has probability greater or equal to $1 - \sigma$ or the user accepts $\mathcal{D}$ as the axioms to be changed then stop and return $\mathcal{D}$.
4. Use $\mathbf{D}$ to generate a set of queries and select the best query Q according to s_Q.
5. Ask the user $\mathcal{O}^* \models Q$ and, depending on the answer, add Q either to P or to N.
6. Remove elements from $\mathbf{D}$ violating the newly acquired test case.
7. Repeat at Step 1.

3 Interactive Direct Diagnosis of Ontologies

The novelty of our approach is the interactivity combined with the direct calculation of diagnoses. To this end, we provide modifications to Step 1 of the diagnosis process given above. Namely, we utilize an "inverse" version of the QX algorithm [13] called INV-QX and an associated "inverse" version of HS-TREE termed INV-HS-TREE.

This combination of algorithms was first used in the earlier version of [6]. However, we introduced two modifications: (i) a depth-first search strategy instead of breadth-first and (ii) a new pruning rule which moves axioms from $\mathcal{O}$ to $\mathcal{B}$ instead of just removing them from $\mathcal{O}$, since not adding them to $\mathcal{B}$ might result in losing some of the diagnoses.

INV-QX – Key Idea. INV-QX relies on the monotonic semantics of the used knowledge representation language. The algorithm takes a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ and a ranking heuristic as input and outputs either one diagnosis or *no-diagnosis-exists*. The ranking heuristic assigns a fault probability to each axiom in $\mathcal{O}$, if this information is available; otherwise every axiom has the same rank. In the first step INV-QX verifies if a diagnosis exists, next whether $\mathcal{O}$ is faulty and, if so, sorts all axioms in descending order. Ordering of axioms according to their fault probabilities allows the algorithm to compute an approximation of a most probable diagnosis. Next, INV-QX enters the recursion in which $\mathcal{O}$ is partitioned into two subsets S_1 and S_2 such that S_1 comprises axioms with higher fault probabilities and S_2 with lower. In our implementation $\mathcal{O}$ is split in half. Then the algorithm verifies whether S_1 is a candidate diagnosis of the input DPI according to Definition 1. The algorithm continues to operate in a divide-and-conquer strategy until a diagnosis is found. INV-QX requires $O(|\mathcal{D}| \log(\frac{|\mathcal{O}|}{|\mathcal{D}|}))$ calls to a reasoner to find a diagnosis $\mathcal{D}$.

INV-QX is a deterministic algorithm. In order to obtain a different next diagnosis, the DPI used as input for INV-QX must be modified accordingly. To this end we employ INV-HS-TREE.

INV-HS-TREE – Construction. The algorithm is inverse to the HS-TREE algorithm in the sense that nodes are now labeled by diagnoses (instead of minimal conflict sets) and a path from the root to an open node is a partial conflict set (instead of a partial diagnosis). The algorithm constructs a directed tree from the root to the leaves, where each node nd is labeled either with a diagnosis $\mathcal{D}$ or $\times$ (*pruned*) which indicates that the node is closed. For each $s \in \mathcal{D}$ there is an outgoing edge labeled by s. Let $H(nd)$ be the set of edge labels on the path from the root to the node nd. Initially the algorithm generates an empty root node and adds it to a LIFO-queue, thereby implementing a *depth-first search* strategy. Until the required number m of diagnoses is reached or the queue is empty, the algorithm removes the first node nd from the queue and labels the node by applying the following steps.

1. (*reuse*): if $\mathcal{D} \cap H(nd) = \emptyset$ for some $\mathcal{D} \in \mathbf{D}$, then label the node with $\mathcal{D}$; add for each $s \in \mathcal{D}$ a node to the LIFO-queue and return
2. Call INV-QX$(\mathcal{O} \setminus H(nd), \mathcal{B} \cup H(nd), P, N) = Value$
3. (*prune*): if $Value =$ *no-diagnosis-exists*, then label the node with $\times$ (see Proposition 1) and return
4. (*assign*): Otherwise $Value$ is a diagnosis, label the node with $\mathcal{D} = Value$; add $\mathcal{D}$ to $\mathbf{D}$ and add for each $s \in \mathcal{D}$ a node to the LIFO-queue.

Reuse of known diagnoses in Step 1 and the addition of $H(nd)$ to the background theory $\mathcal{B}$ in Step 2 allows the algorithm to force INV-QX to search for a diagnosis that is different to all diagnoses in $\mathbf{D}$. In case INV-QX returns *no-diagnosis-exists* the node is pruned. Otherwise, a new diagnosis $\mathcal{D}$ is added to the set $\mathbf{D}$ and is used to label the node. The depth-first search strategy maintains only a set of diagnoses comprising at most m elements. No conflicts are stored. This allows a significant reduction of memory usage by INV-HS-TREE compared to HS-TREE. The worst case space complexity of INV-HS-TREE computing m diagnoses is *linear* and amounts to $O(m)$, whereas the worst case space complexity of HS-TREE is $O(|CS_{\max}|^d)$ where $|CS_{\max}|$ is the maximal cardinality minimal conflict set (i.e. there is no minimal conflict set with larger cardinality) and d is the depth were m diagnoses have been generated w.r.t. a DPI.

The disadvantage of INV-HS-TREE is that it cannot guarantee the computation of diagnoses in a special order, e.g. minimum cardinality or maximum probability first.

INV-HS-TREE – Update Procedure for Interactivity. Since paths in INV-HS-TREE are (1) irrelevant and need not be maintained, and (2) only a small (linear) number of nodes/paths is in memory due to the application of a depth-first search, the update procedure after a query Q has been answered involves a reconstruction of the tree. In particular, by answering Q, $m-k$ of (maximally) m leading diagnoses are invalidated and deleted from memory. The k still valid diagnoses are used to build a new tree. To this end, the root is labeled by any of these k diagnoses and a tree is constructed as described above where the k diagnoses are incorporated for the *reuse* check. Note, the recalculation of a diagnosis that has been invalidated by a query is impossible as in subsequent iterations a new DPI is considered which includes the answered query as a test case.

Example. Consider a DPI with $\mathcal{O} = \{ax_1 : C \sqsubseteq A \quad ax_2 : C \sqsubseteq E \quad ax_3 : A \sqsubseteq \neg(C \sqcup \neg B) \quad ax_4 : B \sqsubseteq C \quad ax_5 : B \sqsubseteq \neg D\}$ the background knowledge $\mathcal{B} = \{A(v), B(w), C(s)\}$, one positive $P = \{D(v)\}$ and one negative $N = \{E(w)\}$ test case. For the sample DPI the set of minimal conflict sets comprises four elements $\{CS_1 : \langle ax_1, ax_3\rangle, CS_2 : \langle ax_2, ax_4\rangle, CS_3 : \langle ax_3, ax_5\rangle, CS_4 : \langle ax_3, ax_4\rangle\}$, as well as the set of diagnoses $\{\mathcal{D}_1 : [ax_2, ax_3], \mathcal{D}_2 : [ax_3, ax_4], \mathcal{D}_3 : [ax_1, ax_4, ax_5]\}$. Assume also that the number of leading diagnoses required for query generation is set to $m = 2$. Applied to the sample DPI, INV-HS-TREE computes a diagnosis $\mathcal{D}_1 := [ax_2, ax_3]$ returned by INV-QX$(\mathcal{O}, \mathcal{B}, P, N)$ to label the root node, see Figure 1. Next, it generates one successor node that is linked with the root by an edge labeled with ax_2. For this node INV-QX$(\mathcal{O} \setminus \{ax_2\}, \mathcal{B} \cup \{ax_2\}, P, N)$ yields a diagnosis $\mathcal{D}_2 := [ax_3, ax_4]$ disjoint with $\{ax_2\}$. Now $|\mathbf{D}| = 2$ and a query is generated and answered as in Figure 1. Adding $C(w)$ to the negative test cases invalidates $\mathcal{D}_1$ because of $ax_4 \in (\mathcal{O} \setminus \mathcal{D}_1)$ and $B(w) \in \mathcal{B}$, that is $(\mathcal{O} \setminus \mathcal{D}_1) \cup \mathcal{B} \cup \{\bigwedge_{p \in P} p\} \models C(w)$. In the course of the update, $\mathcal{D}_1$ is deleted and $\mathcal{D}_2$ used as the root of a new tree. An edge labeled with ax_3 is created and diagnosis $\mathcal{D}_3 := [ax_1, ax_4, ax_5]$ is generated. After the answer to the second query is added to the positive test cases, $\mathcal{D}_3$ is invalidated and all outgoing edge labels ax_3, ax_4 of the root $\mathcal{D}_2$ of the new tree are conflict sets for the current DPI $\langle \mathcal{O}, \mathcal{B}, \{D(v), A \sqsubseteq C\}, \{E(w), C(w)\}\rangle$, i.e. all leaf nodes are labeled by $\times$ and the tree construction is complete. So, $\mathcal{D}_2$ is returned as its probability is 1.

4 Evaluation

We evaluated our approach DIR (based on INV-QX and INV-HS-TREE) versus the standard technique STD [20] (based on QX and HS-TREE) using a set of ontologies

$$\begin{tikzcd} {[ax_2, ax_3]} \arrow[d, "ax_2"'] \\ {[ax_3, ax_4]} \end{tikzcd} \Bigg\rangle \begin{array}{l} \mathbf{D}: \\ \mathcal{D}_1 = [ax_2, ax_3] \\ \mathcal{D}_2 = [ax_3, ax_4] \\ \text{Query: } C(w) \\ \text{Answer: } \textit{no} \end{array} \Bigg\rangle \begin{tikzcd} {[ax_3, ax_4]} \arrow[d, "ax_3"] \\ {[ax_1, ax_4, ax_5]} \end{tikzcd} \Bigg\rangle \begin{array}{l} \mathbf{D}: \\ \mathcal{D}_2 = [ax_3, ax_4] \\ \mathcal{D}_3 = [ax_1, ax_4, ax_5] \\ \text{Query: } A \sqsubseteq C \\ \text{Answer: } \textit{yes} \end{array}$$

$\rightarrow$ No further minimal diagnoses, return $[ax_3, ax_4]$

Fig. 1. Identification of the target diagnosis $[ax_3, ax_4]$ using interactive direct diagnosis.

created by automatic matching systems. Given two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$, a matching system outputs an *alignment* M_{ij} which is a set of *mappings* (correspondences) between semantically related entities of $\mathcal{O}_i$ and $\mathcal{O}_j$. Let $E(\mathcal{O})$ denote the set of all elements of $\mathcal{O}$ for which mappings can be produced, i.e. names of concepts. Each mapping is a tuple $\langle x_i, x_j, r, v\rangle$, where $x_i \in E(\mathcal{O}_i)$, $x_j \in E(\mathcal{O}_j)$ and x_i, x_j are either two concepts or two roles, $r \in \{\sqsubseteq, \equiv, \sqsupseteq\}$ and $v \in [0, 1]$ is a confidence value. The latter expresses the probability of a mapping to be correct. Each $\langle x_i, x_j, r, v\rangle \in M_{ij}$ can be translated to the axiom of the form $x_i\ r\ x_j$. Let $\mathcal{O}(M_{ij})$ be the set of axioms for the alignment M_{ij}, then the result of the matching process is an aligned ontology $\mathcal{O}_{ij} = \mathcal{O}_i \cup \mathcal{O}(M_{ij}) \cup \mathcal{O}_j$. The ontologies considered in this section were created by ontology matching systems participating in the Ontology Alignment Evaluation Initiative (OAEI) 2011 [5]. Each matching experiment in the framework of OAEI represents a scenario in which a user obtains an alignment M_{ij} by means of some (semi)automatic tool for two real-world *ontologies* $\mathcal{O}_i$ and $\mathcal{O}_j$.

The goal of the first experiment was to compare the performance of sequential STD and sequential DIR on a set of large, but diagnostically uncomplicated ontologies, generated for the Anatomy experiment of OAEI[1]. In this experiment the matching systems had to find mappings between two ontologies describing the human and the mouse anatomy. $\mathcal{O}_1$ (`Human`) and $\mathcal{O}_2$ (`Mouse`) include 11545 and 4838 axioms respectively, whereas the size of the alignment M_{12} produced by different matchers varies between 1147 and 1461 mappings. Seven matching systems produced a consistent but incoherent output. One system generated a consistent and coherent aligned ontology. However, this system employs a built-in heuristic diagnosis engine which does not guarantee to produce diagnoses. I.e. some axioms are removed without reason. Four systems produced ontologies which could not be processed by current reasoning systems (e.g. HermiT [16]) since consistency of these ontologies could not be checked within 2 hours.

For testing the performance of our system we have to define the correct output of sequential diagnosis which we call the target diagnosis $\mathcal{D}_t$. We assume that the only available knowledge is M_{ij} together with $\mathcal{O}_i$ and $\mathcal{O}_j$. In order to measure the performance of the matching systems the organizers of OAEI provided a *golden standard* alignment M_t considered as correct. Similarly to OAEI evaluation, in our experiments M_t was unavailable explicitly, e.g. in form of test cases, during a debugging session, just as none of matching systems has any knowledge of M_t during the competition. However, we assumed that an oracle answers debugging queries using its knowledge

[1] All ontologies and source code of programs used in the evaluation can be downloaded from http://code.google.com/p/rmbd/wiki/DirectDiagnosis. The tests were performed on Core i7, 64GB RAM running Ubuntu, Java 7 and HermiT as DL reasoner.

of M_t. Therefore, for every alignment $\mathcal{O}_{ij}$ we selected a diagnosis as target diagnosis $\mathcal{D}_t$ which is outside the golden standard, i.e. $M_t \cap \mathcal{D}_t = \emptyset$. Moreover, we used $\mathcal{O}_{ij} \setminus \mathcal{D}_t$ to answer the queries instead of M_t to ensure that the system finds exactly $\mathcal{D}_t$ and not some other diagnosis. By this procedure we mimic cases where additional information can be acquired such that no mapping of the golden standard is removed in order to establish coherence. We stress that this setting is unfavorable for diagnosis, since providing more information as test cases using the golden standard would reduce the number of queries to ask.

In particular, the selection of a target diagnosis for each $\mathcal{O}_{ij}$ output by a matching system was done in two steps: (i) compute the set of all diagnoses $\mathbf{AD}$ w.r.t. the mappings which are not in the golden standard, i.e. $\mathcal{O}(M_{ij} \setminus M_t)$, and use $\mathcal{O}_i \cup \mathcal{O}_j \cup \mathcal{O}(M_{ij} \cap M_t)$ as background theory. The set of test cases are empty. That is, the DPI is $\langle \mathcal{O}(M_{ij} \setminus M_t), \mathcal{O}_i \cup \mathcal{O}_j \cup \mathcal{O}(M_{ij} \cap M_t), \emptyset, \emptyset \rangle$. (ii) select $\mathcal{D}_t$ randomly from $\mathbf{AD}$. The prior fault probabilities of mapping axioms $ax \in \mathcal{O}(M_{ij})$ were set to $1 - v_{ax}$ where v_{ax} is the confidence value provided by the matching system.

The tests were performed for the mentioned seven incoherent alignments where the input DPI is $\langle \mathcal{O}(M_{ij}), \mathcal{O}_i \cup \mathcal{O}_j, \emptyset, \emptyset \rangle$ and the output is a diagnosis. We tested DIR and STD with both query selection strategies SPLIT-IN-HALF (SPL) and ENTROPY (ENT) in order to evaluate the quality of fault probabilities based on confidence values. Moreover, for generating a query the number of leading diagnoses was limited to $m = 9$.

The results of the first experiment are presented in Table 1. DIR computed $\mathcal{D}_t$ within 36 sec. on average and slightly outperformed STD which required 36.7 sec. The number of asked queries was equal for both methods in all but two cases resulting from ontologies produced by the `MapSSS` system. For these ontologies DIR required one query more using ENT and one query less using SPL. In general, the results obtained for the Anatomy case show that DIR and STD have similar performance in both runtime and number of queries. Both DIR and STD identified the target diagnosis. Moreover, the confidence values provided by the matching systems appeared to be a good estimate for fault probabilities. Thus, in many cases ENT was able to find $\mathcal{D}_t$ using one query only, whereas SPL used 4 queries on average. In the first experiment the identification of the target diagnosis by sequential STD required the computation of 19 minimal conflicts on average. Moreover, the average size of a minimum cardinality diagnosis over all ontologies in this experiment was 7. In the second experiment (see below), where STD is not applicable, the cardinality of the target diagnosis is significantly higher.

The second experiment was performed on ontologies of the OAEI Conference benchmark which turned out to be problematic for STD. For these ontologies we observed that the minimum cardinality diagnoses comprise 18 elements on average. In 11 of the 13 ontologies of the second experiment (see Table 2) STD was unable to find any diagnosis within 2 hours. In the other two cases STD succeeded to find one diagnosis for `csa-conference-ekaw` and nine for `ldoa-conference-confof`. However, DIR even succeeded to find 30 diagnoses for each ontology within time acceptable for interactive diagnosis settings. Moreover, on average DIR was able to find 1 diagnosis in 8.9 sec., 9 diagnoses in 40.83 sec. and 30 diagnoses in 107.61 sec. (see Column 2 of Table 2). This result shows that DIR is a stable and practically applicable method even in cases where an ontology comprises high-cardinality faults.

		HS-TREE			INV-HS-TREE		
System	Scoring	Time	#Queries	Reaction	Time	#Queries	Reaction
AgrMaker	ENT	19.62	1	19.10	20.83	1	18.23
AgrMaker	SPL	36.04	4	8.76	36.03	4	8.28
GOMMA-bk	ENT	18.34	1	18.07	14.47	1	12.68
GOMMA-bk	SPL	18.95	3	6.15	19.51	3	5.91
GOMMA-nobk	ENT	18.26	1	17.98	14.26	1	12.49
GOMMA-nobk	SPL	18.74	3	6.08	19.47	3	5.89
Lily	ENT	78.54	1	77.71	82.52	1	72.83
Lily	SPL	82.94	4	20.23	115.24	4	26.93
LogMap	ENT	6.60	1	6.30	13.41	1	11.36
LogMap	SPL	6.61	2	3.17	15.13	2	6.82
LogMapLt	ENT	14.85	1	14.54	12.89	1	11.34
LogMapLt	SPL	15.59	3	5.05	17.45	3	5.29
MapSSS	ENT	81.06	4	19.86	56.17	3	17.32
MapSSS	SPL	88.32	5	17.26	77.59	6	12.43

Table 1. HS-TREE and INV-HS-TREE applied to Anatomy benchmark. Time is given in sec, **Scoring** stands for query selection strategy, **Reaction** is the average reaction time between queries.

In the Conference experiment we first selected the target diagnosis $\mathcal{D}_t$ for each $\mathcal{O}_{ij}$ just as it was done in the described Anatomy case. Next, we evaluated the performance of sequential DIR using both query selection methods. The results of the experiment presented in Table 2 show that DIR found $\mathcal{D}_t$ for each ontology. On average DIR solved the problems more efficiently using ENT than SPL because also in the Conference case the confidence values provided a reasonable estimation of axiom fault probabilities. Only in three cases ENT required more queries than SPL. Moreover, the experiments show that the efficiency of debugging methods depends highly on the runtime of the underlying reasoner. For instance, in the hardest case consistency checking took 93.4% of the total time whereas all other operations – including construction of the search tree, generation and selection of queries – took only 6.6% of time. Consequently, sequential DIR requires only a small fraction of computation effort. Runtime improvements can be achieved by advances in reasoning algorithms or the reduction of the number of consistency checks. Currently DIR requires $O(m * |\mathcal{D}| \log(\frac{|\mathcal{O}|}{|\mathcal{D}|}))$ checks to find m leading diagnoses. A further source for improvements can be observed for the `ldoa-ekaw-iasted` ontology where both methods asked the same number of queries. In this case, ENT required only half of the consistency checks SPL did, but an average consistency check of ENT took almost twice as long as an average one for SPL. The analysis of this ontology showed that there is a small subset of axioms (hot spot) which made reasoning considerably harder. Identification of such hot spots [8] could result in a significant improvement of diagnosis runtime, since a hot spot can be resolved by suitable queries. This can be observed in the `ldoa-ekaw-iasted` case where SPL acquired appropriate test cases early and thereby found $\mathcal{D}_t$ faster.

In order to speed up the computation of conflicts and diagnoses we tested two popular module extraction methods [9, 3]. In ontology debugging these methods are used as preprocessors allowing to generate a smallest possible ontology $\mathcal{O}' \subseteq \mathcal{O}$ for a faulty

Ontology (Expressivity)	30 Diag	min $\|\mathbf{D}\|$	Scoring	Time	#Queries	Reaction	#CC	CC
ldoa-conference-confof	48.06	16	ENT	11.6	6	1.5	430	0.003
$\mathcal{SHIN}(\mathcal{D})$			SPL	11.3	7	1.6	365	0.004
ldoa-cmt-ekaw	42.28	12	ENT	48.6	21	2.2	603	0.016
$\mathcal{SHIN}(\mathcal{D})$			SPL	139.1	49	2.8	609	0.054
mappso-confof-ekaw	55.66	10	ENT	10	5	1.9	341	0.007
$\mathcal{SHIN}(\mathcal{D})$			SPL	31.6	13	2.3	392	0.021
optima-conference-ekaw	62.13	19	ENT	16.8	5	2.6	553	0.008
$\mathcal{SHIN}(\mathcal{D})$			SPL	16.1	8	1.9	343	0.012
optima-confof-ekaw	44.52	16	ENT	24	20	1.1	313	0.014
$\mathcal{SHIN}(\mathcal{D})$			SPL	17.6	10	1.7	501	0.006
ldoa-conference-ekaw	56.98	16	ENT	56.7	35	1.5	253	0.053
$\mathcal{SHIN}(\mathcal{D})$			SPL	25.5	9	2.7	411	0.016
csa-conference-ekaw	62.82	17	ENT	6.7	2	2.8	499	0.003
$\mathcal{SHIN}(\mathcal{D})$			SPL	22.7	8	2.7	345	0.02
mappso-conference-ekaw	70.46	19	ENT	27.5	13	1.9	274	0.028
$\mathcal{SHIN}(\mathcal{D})$			SPL	71	16	4.2	519	0.041
ldoa-cmt-edas	15.47	16	ENT	24.7	22	1	303	0.008
$\mathcal{ALCOIN}(\mathcal{D})$			SPL	11.2	7	1.4	455	0.002
csa-conference-edas	39.74	26	ENT	18.4	6	2.7	419	0.005
$\mathcal{ALCHOIN}(\mathcal{D})$			SPL	240.8	37	6.3	859	0.036
csa-edas-iasted	377.36	20	ENT	1744.6	3	349.2	1021	1.3
$\mathcal{ALCOIN}(\mathcal{D})$			SPL	7751.9	8	795.5	577	11.5
ldoa-ekaw-iasted	229.72	13	ENT	23871.5	9	1886	287	72.6
$\mathcal{SHIN}(\mathcal{D})$			SPL	20449	9	2100.1	517	37.2
mappso-edas-iasted	293.74	27	ENT	18400.3	5	2028.3	723	17.8
$\mathcal{ALCOIN}(\mathcal{D})$			SPL	159299	11	13116.6	698	213.2

Table 2. Interactive debugging with direct computation of diagnoses. **30 Diag** the time required to find 30 diagnoses, **min** $|\mathcal{D}|$ the cardinality of a minimum cardinality diagnosis, **Scoring** query selection strategy, **Reaction** average system reaction time between queries, **#CC** number of consistency checks, **CC** gives average time needed for one consistency check. Time is given in sec.

ontology $\mathcal{O}$ such that $\mathcal{O}'$ comprises all axioms relevant to a fault and often $|\mathcal{O}'| \ll |\mathcal{O}|$. The algorithm based on syntactic locality [9] was used in all STD tests since it improved performance of conflict set computation. The second module extraction algorithm [3] was not as effective as [9] when applied to compute conflict sets for most of our test cases. In fact, [3] tended to generate large modules including all axioms relevant to top classes of the hierarchy since all these classes are declared to be pairwise disjoint in all but two Conf ontologies. The same was observed for Anat where all top classes of the `Human` ontology are defined to be disjoint as well.

5 Conclusions

In this paper we presented a sequential diagnosis method for faulty ontologies which is based on the direct computation of diagnoses. We reduce the number of consistency checks by avoiding the computation of minimized conflict sets and by computing *some* set of diagnoses instead of a set of most probable diagnoses or a set of minimum cardinality diagnoses. The evaluation results presented in the paper indicate that the performance of the suggested sequential diagnosis system is either comparable with or outperforms the existing approach in terms of runtime and the number of queries in case a ontology includes a large number of faults. The scalability of the algorithms was demonstrated on a set of large ontologies including thousands of axioms.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications (2007)
2. Baader, F., Knechtel, M., Peñaloza, R.: Context-dependent views to axioms and consequences of Semantic Web ontologies. J. Web Semant. 12-13, 22–40
3. Du, J., Qi, G., Pan, J.Z., Shen, Y.D.: A Decomposition-Based Approach to OWL DL Ontology Diagnosis. In: ICTAI. pp. 659–664 (2011)
4. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. JACM 42(1), 1–49 (1995)
5. Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., dos Santos, C.T.: Final results of the Ontology Alignment Evaluation Initiative 2011. In: OM Workshop. pp. 1–29. (2011)
6. Felfernig, A., Schubert, M., Zehentner, C.: An efficient diagnosis algorithm for inconsistent constraint sets. AI EDAM 26(1), 53–62 (2012)
7. Friedrich, G., Shchekotykhin, K.: A General Diagnosis Method for Ontologies. In: ISWC. pp. 232–246. (2005)
8. Goncalves, R.S., Parsia, B., Sattler, U.: Performance Heterogeneity and Approximate Reasoning in Description Logic Ontologies. In: ISWC. pp. 82–98 (2012)
9. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular Reuse of Ontologies : Theory and Practice. J. Artif. Intell. Res. 31, 273–318 (2008)
10. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: ISWC. pp. 323–338 (2008)
11. Horridge, M., Parsia, B., Sattler, U.: Extracting justifications from BioPortal ontologies. In: ISWC 2012. pp. 287–299 (2012)
12. Jiménez-Ruiz, E., Grau, B.C.: LogMap: Logic-based and scalable ontology matching. In: ISWC. pp. 273–288 (2011)
13. Junker, U.: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In: AAAI. pp. 167–172 (2004)
14. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all Justifications of OWL DL Entailments. In: ISWC. pp. 267–280 (2007)
15. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artif. Intell. 32(1), 97–130 (1987)
16. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. J. Artif. Intell. Res. 36(1), 165–228 (2009)
17. Peñaloza, R., Sertkaya, B.: On the complexity of axiom pinpointing in the EL family of description logics. In: KR'10. pp. 280–289 (2010)
18. Reiter, R.: A Theory of Diagnosis from First Principles. Artif. Intell. 32(1), 57–95 (1987)
19. Satoh, K., Uno, T.: Enumerating Minimally Revised Specifications Using Dualization. In: JSAI Workshops. pp. 182–189 (2005)
20. Shchekotykhin, K., Friedrich, G., Fleiss, P., Rodler, P.: Interactive ontology debugging : two query strategies for efficient fault localization. J. Web Semant. 12-13, 88–103 (2012)
21. Stern, R., Kalech, M., Feldman, A., Provan, G.: Exploring the Duality in Conflict-Directed Model-Based Diagnosis. In: AAAI. pp. 828–834 (2012)

A System for Debugging Missing Is-a Structure in $\mathcal{EL}$ Ontologies

Zlatan Dragisic[1,2], Patrick Lambrix[1,2], Fang Wei-Kleiner[1]

(1) Department of Computer and Information Science, (2) Swedish e-Science Research Centre
Linköping University, 581 83 Linköping, Sweden

Abstract. With the increased use of ontologies in semantically-enabled applications, the issue of debugging defects in ontologies has become increasingly important. These defects can lead to wrong or incomplete results for the semantically-enabled applications. Debugging consists of the phases of detection and repairing. In this paper we introduce a system for repairing a particular kind of defects, i.e. missing relations in the is-a hierarchy of $\mathcal{EL}$ ontologies.

1 Introduction

Developing ontologies is a difficult task and it is often the case that the ontologies are incomplete or incorrect. More and more ontologies are used in semantically-enabled applications. Defects in these ontologies can cause incomplete or incorrect results so ontology debugging is a crucial step for acquiring high-quality results in these applications.

In this demonstration paper, we focus on missing is-a relations which are a type of modelling defects. This type of defects requires domain knowledge to detect and resolve. We consider ontologies that are represented by description logics (DLs), more specifically represented by TBoxes in $\mathcal{EL}$. $\mathcal{EL}$ is highly relevant for the representation of lightweight ontologies. For instance, several of the major ontologies in the biomedical domain, e.g., SNOMED[1] and Gene Ontology [1], can be represented in $\mathcal{EL}$ or small extensions thereof [2].

In this demonstation paper we briefly introduce the system introduced in [4]. We describe the system (Section 2) and an example run (Section 3). For the theory, the algorithm as well as more detailed discussion of the experiments we refer to [4]. In Section 4 we introduce the demonstration.

2 Approach

Debugging missing is-a structure consists of two phases, detection and repair. In the detection phase, missing is-a relations are identified while in the repair phase the idea is to make these identified missing is-a relations derivable in the ontology. If all missing is-a relations were identified in the detection phase, the repair phase would be straightforward as only adding these is-a relations is required. However, in general, detection

[1] http://www.ihtsdo.org/snomed-ct/

algorithms do not detect all missing is-a relations and in most cases only few. In cases when only some missing is-a relations are detected there are different approaches for repairing missing is-a structure.

In our setting we assume that our ontology is represented using a TBox T. Further, a detection algorithm or a domain expert has provided a set M of missing is-a relations (but not necessarily all) for the ontology. Then we want to identify a set of is-a relations S such that $T \cup S \models M$. We require that relations in S and M are is-a relations between named concepts as well as that the is-a relations in S should be correct according to the domain. In general, the set of all is-a relations using concepts in T that are correct according to the domain is not known beforehand. If this set was given then we would only have to add this to the ontology. The common case is that we do not have this set, but instead can rely on a domain expert that can decide whether an is-a relation is correct according to the domain. The role of the domain expert can be formalized by an oracle function that returns true or false given an is-a relation. The formal definitions of the problem can be found in [4].

While our earlier work focused on taxonomies [7, 5], in this work we focus on repairing missing is-a relations in $\mathcal{EL}$ ontologies. A TBox in $\mathcal{EL}$ ontologies is a finite set of general concept inclusions of the form $C \sqsubseteq D$ where C and D represent concept descriptions. Concept descriptions in $\mathcal{EL}$ are inductively formed using concept names, role names and concepts constructors which include the top concept, conjuction and existential restriction. In our approach for repairing missing is-a relations we require that the TBox is normalized as described in [2]. A normalized TBox T contains only axioms of the forms $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, $A \sqsubseteq \exists r.B$, and $\exists r.A \sqsubseteq B$, where A, $A_1, \ldots, A_n$ and B are concept names and r is a role.

Given that we are dealing with normalized $\mathcal{EL}$ ontologies the algorithm for repairing missing is-a relations uses the following intuitions. Given missing is-a relation $A \sqsubseteq B$:

1. if $A \sqsubseteq C$ and $D \sqsubseteq B$ are derivable from the ontology, then adding $C \sqsubseteq D$ would make the missing is-a relation derivable. Therefore, to acquire possible logical solutions we form two sets, Source and Target, containing the superconcepts of A and the subconcepts of B, respectively. Any is-a relation $C \sqsubseteq D$ such that $C \in$ Source and $D \in$ Target would be a logical solution for repairing $A \sqsubseteq B$.
2. if the ontology contains axioms $A \sqsubseteq \exists r.C$ and $\exists r.D \sqsubseteq B$ then adding is-a relation $C \sqsubseteq D$ would make $A \sqsubseteq B$ derivable.
3. if the ontology contains axioms $A \sqsubseteq \exists r.C$, $\exists r.D \sqsubseteq B$ and is-a relations $C \sqsubseteq F$ and $G \sqsubseteq D$ are derivable in the ontology then $F \sqsubseteq G$ would be a logical solution for the missing is-a relation $A \sqsubseteq B$. This intuition corresponds to generating Source and Target sets for the identified logical solution in the second intuition.

Following the above intuitions we identify logical solutions but not necessarily solutions that are correct according to the domain. Therefore, it is necessary to validate logical solutions with respect to the domain. The repair for the complete set of missing is-a relations is formed by taking the union of repairs for individual missing is-a relations. Any element in the repair for the complete set of missing is-a relations which is not in the initial set of missing is-a relations can be considered as a new missing is-a relation (which was not detected earlier). These new missing is-a relations can then be used as input for a new iteration of the process, thus possibly finding additional solutions.

We have implemented a system for repairing missing is-a structure in $\mathcal{EL}$ ontologies based on the described approach. The input to the system is a set of missing is-a relations which have been validated to be correct according to the domain. The repairing process is semi-automatic and requires interaction with the user who acts as an oracle and decides whether an is-a relation is correct according to the domain. The system has been implemented in Java and uses the ELK reasoner (version 0.4.1) [6] to calculate implicit entailments in the ontology.

3 Use

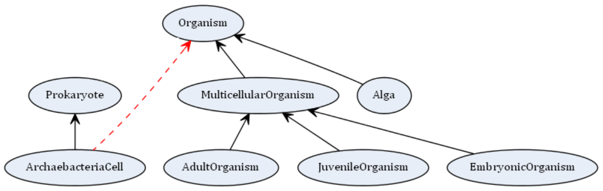
In order to demonstrate the use of the system, let us consider the process of repairing the BioTop ontology from the 2013 OWL Reasoner Evaluation Workshop [4]. The ontology contains 280 concepts and 42 object properties. The set of missing is-a relations consists of 47 is-a relations which were randomly selected in the ontology. Then the ontology was modified by removing relations from the ontology which would make the selected is-a relations derivable. The unmodified ontology has been used as domain knowledge.

The repairing process starts with the user loading the ontology and missing is-a relations into the system and pressing the button `Generate Repairing Actions`. The system then generates Source and Target sets according to intuition 1 and intuitions 2/3. The loaded missing is-a relations are shown in a drop down list allowing the user to easily switch between missing is-a relations. After selecting one of the missing is-a relations, the system shows Source and Target sets for that is-a relation. To repair the missing is-a relation the user needs to choose is-a relations which are correct according to the domain for that is-a relation. This is done by selecting one element from the Source set and one element from the Target set and pressing the `Validate` button thus validating the is-a relation as a repairing action. The system allows multiple repairing actions for each missing is-a relation.

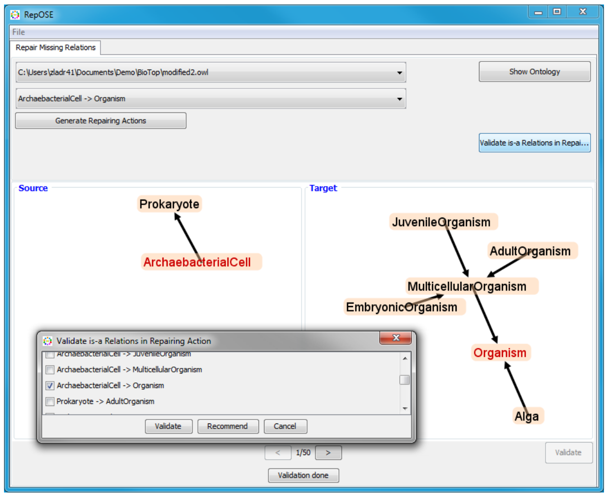
In the BioTop use case the system generates Source and Target sets for 50 is-a relations, 47 according to intuition 1 and 3 according to intuitions 2/3. An example of a Source and Target set generated according to intuition 1 is given in Figure 1(b). Given that is-a relation ArchaebacteriaCell $\sqsubseteq$ Organism is in the input set of missing is-a relations the is-a relation is automatically validated to be correct according to the domain. In this case, the domain expert will also validate is-a relation Prokaryote $\sqsubseteq$ Organism as correct thus introducing new knowledge to the ontology.

An example of Source and Target sets acquired following the intuitions 2/3 is shown in Figure 2(b). In this case, we have the Source and Target set for the is-a relation SpeciesHomoSapiensQuality $\sqsubseteq$ FamilyHominidaeQuality which is a logical solution for the missing is-a relation Human $\sqsubseteq$ GreatApe given that the ontology contains axioms Human $\sqsubseteq$ $\exists$hasInherence.SpeciesHomoSapiensQuality and $\exists$hasInherence.FamilyHominidaeQuality $\sqsubseteq$ GreatApe (Figure 2(a)). Unlike the previous example, the is-a relation SpeciesHomoSapiensQuality $\sqsubseteq$ FamilyHominidaeQuality has to be validated explicitly by the domain expert as it was found using intuitions

2/3. In addition, is-a relation GenusHomoQuality $\sqsubseteq$ FamilyHominidaeQuality will also be validated by the domain expert as it is correct according to the domain.



(a) Relevant part of the BioTop ontology.



C:\Users\zladr41\Documents\Demo\BioTop\modified2.owl

ArchaebacterialCell -> Organism

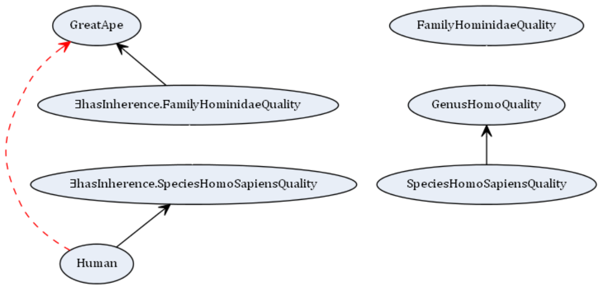
- [ ] ArchaebacterialCell -> JuvenileOrganism
- [ ] ArchaebacterialCell -> MulticellularOrganism
- [✓] ArchaebacterialCell -> Organism
- [ ] Prokaryote -> AdultOrganism

(b) Screenshot from the system

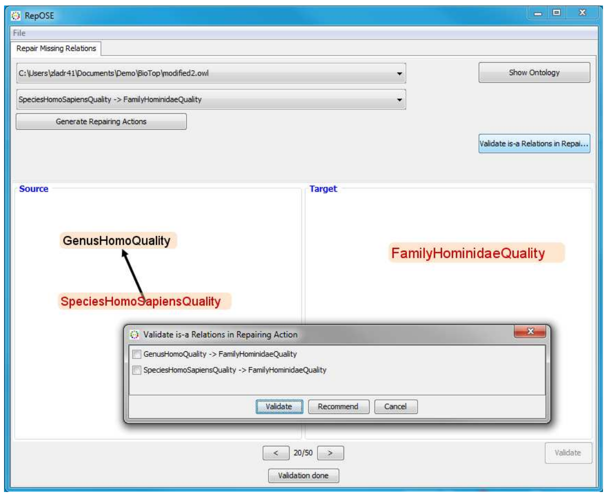
Fig. 1: Repairing ArchaebacteriaCell $\sqsubseteq$ Organism.

Clicking the `Validate Is-a Relations in Repairing Action` button opens a pop-up window where the user has a possibility to check validated is-a relations or see which relations can be validated. On this screen the user can also do the actual validation or remove already validated relations. By clicking on the `Recommend` button the system will recommend correct is-a relations by querying external sources. Currently the recommendations are acquired from WordNet, UMLS Methathesaurus and Uberon. In Figures 1(b) and 2(b) the validation panel for the is-a relations ArchaebacteriaCell $\sqsubseteq$ Organism and SpeciesHomoSapiensQuality $\sqsubseteq$ FamilyHominidaeQuality, respectively, is given.

The validation phase is ended by clicking on the `Validation Done` button. The user has a possibility to end validation phase at any point. If the user has not dealt with some missing is-a relation then the repairing for that is-a relation would be the missing is-a relation itself (as the missing is-a relations are automatically validated to be correct). The system then calculates a repair for the complete set of missing is-a


GreatApe
FamilyHominidaeQuality
∃hasInherence.FamilyHominidaeQuality
GenusHomoQuality
∃hasInherence.SpeciesHomoSapiensQuality
SpeciesHomoSapiensQuality
Human

(a) Relevant part of the BioTop ontology.



C:\Users\zladr41\Documents\Demo\BioTop\modified2.owl

SpeciesHomoSapiensQuality -> FamilyHominidaeQuality

Validate is-a Relations in Repairing Action

- [ ] GenusHomoQuality -> FamilyHominidaeQuality
- [ ] SpeciesHomoSapiensQuality -> FamilyHominidaeQuality

(b) Screenshot from the system

Fig. 2: Repairing Human $\sqsubseteq$ GreatApe.

relations. This repair is then used as input in the next iteration of the repairing process. If the repairing did not change between iterations, the system outputs the final solution.

In our example, 28 relations are repaired by adding a total of 26 new relations out of which 3 are acquired using intuitions 2/3. The remaining 19 missing is-a relations are repaired by adding the missing is-a relation itself. Before the start of the second iteration the system calculates a new set of non-redundant is-a relations from the union of repairing actions from the first iteration. In total 41 new non-redundant is-a relations are identified (4 redundant is-a relations are removed from the solution in iteration 1). In the next iteration the user is presented with Source and Target sets for a total of 64 is-a relations out of which 23 correspond to repairing actions acquired using intuitions 2/3. However, none of these 23 is-a relations are identified to be correct according to the domain. In this iteration 10 is-a relations are repaired by adding new is-a relations. Four out of these 10 is-a relations are from the initial set of missing is-a relations while others were added in the first iteration. For example, relation Virus $\sqsubseteq$ StructuredBiologicalEntity is repaired by adding relation Virus $\sqsubseteq$ Organism given that the relation Organism $\sqsubseteq$ StructuredBiologicalEntity was added in the first iteration. Figure 3(a) shows the relevant part of the BioTop ontology for the missing is-a relation Virus $\sqsubseteq$ StructuredBiologicalEntity with is-a relations added in the previous iteration marked in green. A screenshot from the system with Source and Target set for the missing is-a relation is given in Figure 3(b).

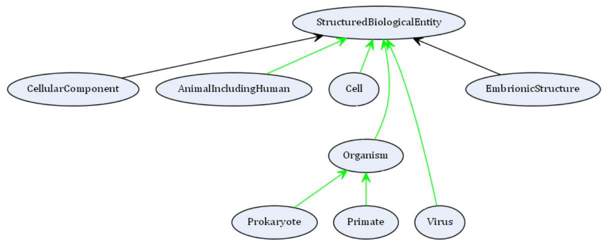
In the third iteration, the user is presented with Source and Target sets for 65 is-a relations out of which 42 are non-redundant is-a relations from the union of repairing actions in the second iteration and 23 are is-a relations which represent repairing actions acquired using intuitions 2/3. Out of these 23 is-a relations only one is validated to be correct according to the domain. Additionally, 2 relations are added in this iteration repairing a total of 4 is-a relations. Out of these 4 repaired is-a relations 3 are from the initial set of missing is-a relations while 1 is from the first iteration.

Finally, in the fourth iteration no new relations are added and the system outputs the solution.

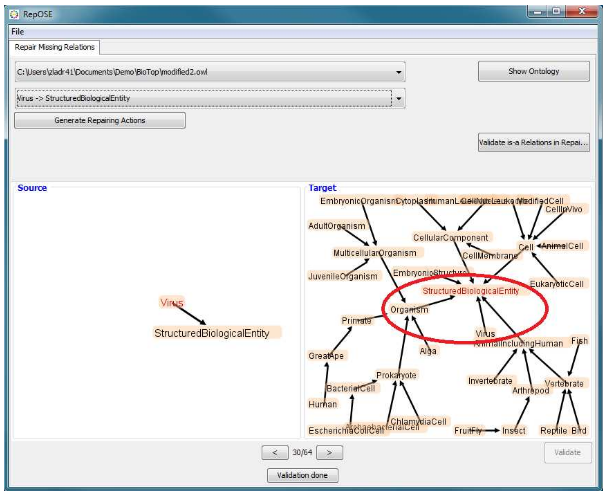
Given that validation can be a time consuming task for large ontologies, the system also implements sessions thus allowing the user to repair ontologies across multiple sessions. To accommodate this, the system implements mechanisms for saving currently validated relations as well as loading previously stored validated relations.

4 Demonstration

In the demonstration we will show two use cases from [4]. The first use case is the one described in Section 3. For the second use case we use Mouse anatomy (AMA) and a fragment of NCI human anatomy ontology (NCI-A) from the Anatomy track of the 2013 Ontology Alignment Evaluation Initiative [3]. The set of missing is-a relations for these two experiments were obtained using a logic-based approach presented in [7] which uses an alignment between these two ontologies to generate possible missing is-a relations which are then validated by a domain expert. The set of missing is-a relations consists of 94 is-a relations for the AMA ontology and 58 for the NCI-A ontology. The missing is-a relations were repaired by adding 101 is-a relations to AMA and 54 is-a



(a) Relevant part of the BioTop ontology.



C:\Users\zladr41\Documents\Demo\BioTop\modified2.owl

Virus -> StructuredBiologicalEntity

(b) Screenshot from the system

Fig. 3: Repairing Virus ⊑ StructuredBiologicalEntity.

relations to NCI-A. Out of 101 is-a relations in the repair for AMA 47 represent new is-a relations which do not appear in the initial set of missing is-a relations. In the case of NCI-A 10 new is-a relations were added.

Acknowledgments. We thank the Swedish Research Council (Vetenskapsrådet), the Swedish e-Science Research Centre (SeRC) and the Swedish National Graduate School in Computer Science for financial support.

References

1. M Ashburner, CA Ball, JA Blake, D Botstein, H Butler, JM Cherry, AP Davis, K Dolinski, SS Dwight, JT Eppig, MA Harris, DP Hill, L Issel-Tarver, A Kasarskis, S Lewis, JC Matese, JE Richardson, M Ringwald, GM Rubin, and G Sherlock. Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29, 2000.
2. F Baader, S Brandt, and C Lutz. Pushing the $\mathcal{EL}$ envelope. In *19th Int Joint Conf on Artificial Intelligence*, pages 364–369, 2005.
3. B Cuenca Grau, Z Dragisic, K Eckert, J Euzenat, A Ferrara, R Granada, V Ivanova, E Jiménez-Ruiz, AO Kempf, P Lambrix, A Nikolov, H Paulheim, D Ritze, F Scharffe, P Shvaiko, C Trojahn, and O Zamazal. Results of the ontology alignment evaluation initiative 2013. In *Proc. 8th workshop on ontology matching (OM)*, pages 61–100, 2013.
4. Z Dragisic, P Lambrix, and F Wei-Kleiner. Completing the is-a structure of biomedical ontologies. In *10th International Conference on Data Integration in the Life Sciences*, 2014.
5. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conf*, pages 1–15, 2013.
6. Y Kazakov, M Krötzsch, and F Simančík. Concurrent classification of $\mathcal{EL}$ ontologies. In *10th Int Semantic Web Conf*, pages 305–320. 2011.
7. P Lambrix and Q Liu. Debugging the missing is-a structure within taxonomies networked by partial reference alignments. *Data & Knowledge Engineering*, 86:179–205, 2013.

B-Annot: Supplying Background Model Annotations for Ontology Coherence Testing

Vojtěch Svátek[1], Simone Serra[1], Miroslav Vacura[1], Martin Homola[2] and Ján Kľuka[2]

[1] Univ. of Economics, Prague, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{svatek,vacuram}@vse.cz, serrazimone@gmail.com
[2] Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia
{homola,kluka}@fmph.uniba.sk

Abstract. The demo paper presents *B-Annot*, a Protégé plugin for annotation of ontologies and linked data vocabularies by background model distinctions. In addition, it briefly demonstrates the subsequent use of the annotations created by *B-Annot*, for verifying the ontological coherence of the ontologies/vocabularies at the level of meta-models. Finally, possible further extensions of the tool and its role in the (background-model-driven) ontological engineering workflow are briefly discussed.

1 Introduction and Motivation

With the growing popularity of the semantic web, a large portion of new ontologies, such as Linked Data (LD) vocabularies, has been directly authored in OWL and thus influenced from the beginning by its inventory of constructs, and also by particular application needs. Let us call such an operational artefacts *ontological foreground model* (OFM). On the other hand, by giving priority to mimicking as much as possible (at least, in some aspects) what is observed in the real-world, we arrive at an *ontological background model* (OBM). For instance:

– OWL classes may sometimes represent permanent *types* of objects and sometimes just *roles* played by these objects in a certain phase of their existence;
– OWL individuals may represent true *individual objects* ('particulars'), but also *universal* entities (types), or even *relationships* whose existence fully depends on the participating objects.

When the OFMs are, e.g., visualized, reused, matched or transformed, such 'hooding' may cause troubles. For example, in an OFN it can happen that a class (i.e. role) *Student* becomes superclass of classes *Human* and *Robot*; an object then may stop being member of the superclass while remaining member of the subclass. For another example see the upper part of the diagram (adapted from [8]) in Fig. 1, depicting the complex fact of a business entity (resource 3) offering exemplars (i.e., 'some items') of a certain musical album (resource 1) as product for sale, in a certain region. The fact refers to two LD vocabularies: the e-commerce ontology GoodRelations (GR)[3] and the Music Ontology (MO).[4] The

[3] http://purl.org/goodrelations/v1
[4] http://purl.org/ontology/mo/

remaining two instance-level resources in the diagram (2 and 4) are the 'offering' itself and the value '90' (minutes) understood as 'typical' and thus modeled as a resource rather than literal.[5] In the lower part of the diagram we approximate the *ontological background* of this fragment (omitting the entities that would be *types* in both diagrams, for easier readability). Among other things we see that notion of 'album', originally being the value of the object property `mo:release_type`, now becomes an additional type of the product offered, and that the 'Offering' object becomes absorbed by the 'offers' relationship (now with arity >2).

Obviously, modelling the ontological background for each individual data fragment is infeasible. The mapping between the 'foreground view' of the domain (as contained in the vocabulary) and the corresponding 'background view' thus has to be established at the level of entity *types*, which means, indirectly (note that especially less expressive vocabularies are just collections of unlinked entities whose connection is only established at the level of instance data). On the one side of the mapping is an *ontological foreground model* (OFM), i.e., the structure of an RDFS/OWL ontology; on the other side is an analogous *ontological background model* (OBM). OBM models should be represented in a suitable *OBM language* of modelling primitives (OBML). Two such languages are

- *OntoClean* [3], which labels OFM classes with the ontological notions of *essentiality*, *rigidity* (e.g., in the first example mentioned, 'permanent' classes *Human* and *Robot* would be rigid while the 'temporary' class *Student* would be anti-rigid), *identity* and *unity*.
- The recently designed *PURO* OBML [9],[6] aiming to capture the background distinctions of OFM entities as in the bottom part of Fig. 1: that between *objects* ('particulars') and their *types* ('universals') and that between *relationships* (or 'valuations' by a quantitative value) and self-standing objects.

OntoClean has proven useful for taxonomy-centric ontologies that dominate, e.g., in bioinformatics. On the other hand, PURO has been specifically designed for 'relation-centric' ontologies/vocabularies [8], which are prominent in LD. Another important phenomenon in LD is that an existing entity might be systematically used with a different background distinction than foreseen in the vocabulary specification; for example, a property that is assumed to have categories of objects in its range might refer to individual objects in some dataset. Therefore, 'generic' annotation of vocabularies might not be sufficient; we should also be able to annotate vocabularies 'as they are used' in a specific dataset.

By their capacity of underlying the entities from various operational (typically, domain-restricted) knowledge models with background ontological distinctions, OBMLs are analogous to *foundational ontologies*. The difference is in the way the 'surface' and 'deep' model are interconnected. A foundational ontology provides root concepts upon which the 'surface model' concepts are grafted; both models thus share the same space. In contrast, OBMs reside in their own 'layer';

[5] Such kind of modeling is not common in MO, but, rather, in GR-compliant ontologies, cf. `http://www.ebusiness-unibw.org/ontologies/opdm/#ontologies`.

[6] A more extensive description is in [7].

when connecting an OFM with an OBM, we thus need to 'inject' a 'proxy' of one model to the other model, in order not to let the one interfere with the formal semantics of the other. Two alternatives for creating such a 'proxy', assuming both layers are to be expressed in OWL-DL, are as follows:

– OBM entities could become values of specific *OWL annotation properties*, and be saved as unobtrusive part of (a copy of) the OFM.
– OFM entities (classes, properties and individuals) could be uniformly *meta-modelled* as syntactical instances to be inserted as an A-Box into a meta-modelling ontology, where their mapping to OBM can be captured.

The first alternative is favourable for visibility of the OBM distinctions to a human when working with the OFM. The second alternative, in turn, allows to carry out conceptual coherence checking according to constraints defined in the meta-modelling ontology, via a generic OWL DL reasoning mechanism. This approach has been previously tested for the OntoClean OBML in [2, 10], and later for the PURO OBML by us [9].

In this system/demonstration paper we present *B-Annot*: a Protégé plugin[7] that allows to create and save meta-models of a selected vocabulary with respect to either OntoClean or PURO, and (especially for the latter) in two modalities, 'generic' and 'dataset-specific'. (Storage of OBM distinctions in annotation property values, as well as other enhancements, is forthcoming.) We also briefly demonstrate how the annotations can be used for conceptual coherence checking; in contrast to PURO-only coherence checking described in [9], we nowadays rely on a modular set of ontologies that also includes an OntoClean module.

2 *B-Annot* Functionality

In summary, the tool allows the user, for the vocabulary to be annotated already loaded into the Protégé editor,

– to select the meta-ontology (either OntoClean or PURO) and decide whether generic or dataset-level annotation is going to take place;
– *for dataset-specific annotation*, to inspect the statistics of presence of entities from the given vocabulary in different datasets, fetched online from LOD-Stats [1], to select an appropriate dataset, and to view the list of entities from the vocabulary that occur in this dataset;
– *for dataset-specific annotation*, to browse a pre-computed summary of the dataset (inspired by [4]), with entities from the vocabulary highlighted;
– select an entity (in one of the Protégé tabs) and annotate it with a background model distinction;
– save the whole annotation set to an RDF file, and load it back.

[7] Available from `http://patomat.vse.cz/cz.vse.bannotation.plugin.view.jar`.

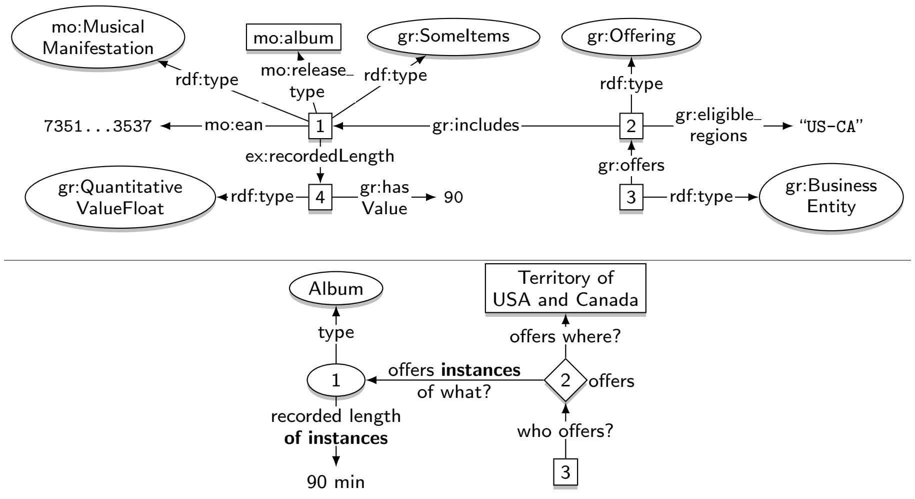


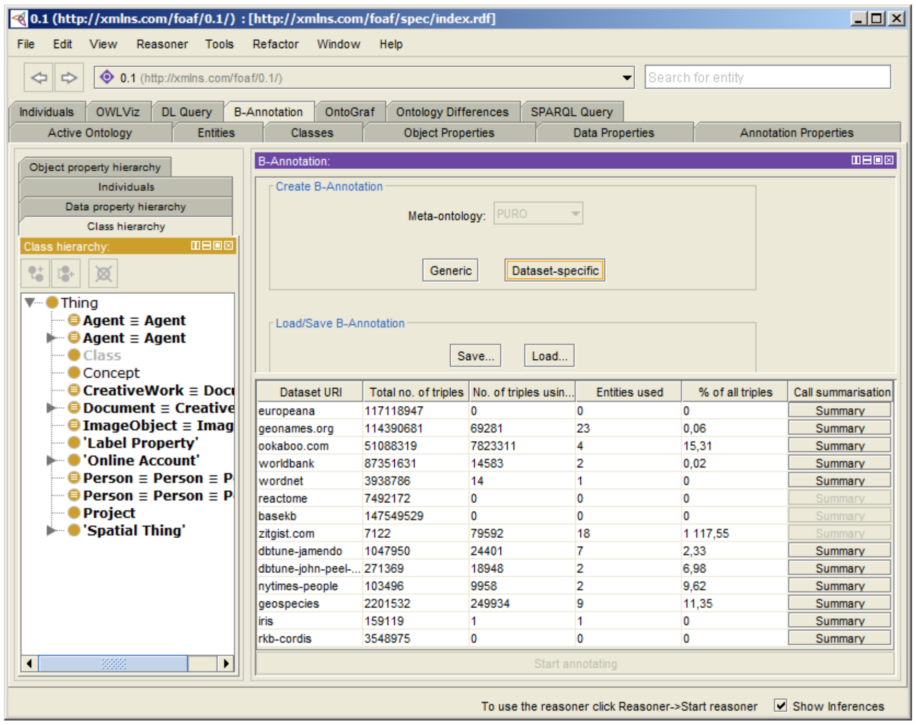
Fig. 1. RDF data fragment and its ontological background

We will now describe the scenario of dataset-specific annotation, since generic annotation is essentially a subset of it. Furthermore, we will use the PURO meta-ontology as more relevant in the dataset-specific mode. (OntoClean would be applied in the same way.) Fig. 2 shows the *B-Annot* interface after the choice of PURO and dataset-specific annotation mode (FOAF has been previously loaded into Protégé as ontology to be annotated). The user can see that of the 14 datasets for which the statistics has been fetched, 10 use some number of FOAF entities, ranging from 1 to 23; these are relevant to the annotation session. After clicking at the 'summary' button for the Geospecies dataset, an ordered listing of frequent 'class-property-class' is displayed, a part of which is in Fig. 3.[8] FOAF entities, here the properties *depiction*, *isPrimaryTopicOf*, *primaryTopic* and *topic*, are displayed in red. Finally, the actual annotation takes place. In Fig. 4 we see that the user, based on the observation that *foaf:topic* is usually valued by biological taxa[9] in this dataset, assigns this property the PURO label 'PrT' ('property whose range is a type') from the pull-down menu (with items picked from the meta-ontology depending on the entity type to be annotated: class, property or individual). Entity annotations are subsequently listed in the bottom part of the window, and can, eventually, be saved (and reloaded) in bulk, as a set of *hasLabel*[10] triples.

[8] We also experiment with 'class-property-class-property-class' paths, but they are not implemented in the current version of the system.

[9] For the sake of this example, we omit the philosophical discussion whether and for what purpose a taxon should indeed be understood as a universal.

[10] Every meta-modelling ontology has its own *hasLabel* property; here it is the one from the PURO ontology.



Dataset URI	Total no. of triples	No. of triples usin...	Entities used	% of all triples	Call summarisation
europeana	117118947	0	0	0	Summary
geonames.org	114390681	69281	23	0,06	Summary
ookaboo.com	51088319	7823311	4	15,31	Summary
worldbank	87351631	14583	2	0,02	Summary
wordnet	3938786	14	1	0	Summary
reactome	7492172	0	0	0	Summary
basekb	147549529	0	0	0	Summary
zitgist.com	7122	79592	18	1 117,55	Summary
dbtune-jamendo	1047950	24401	7	2,33	Summary
dbtune-john-peel-...	271369	18948	2	6,98	Summary
nytimes-people	103496	9958	2	9,62	Summary
geospecies	2201532	249934	9	11,35	Summary
iris	159119	1	1	0	Summary
rkb-cordis	3548975	0	0	0	Summary

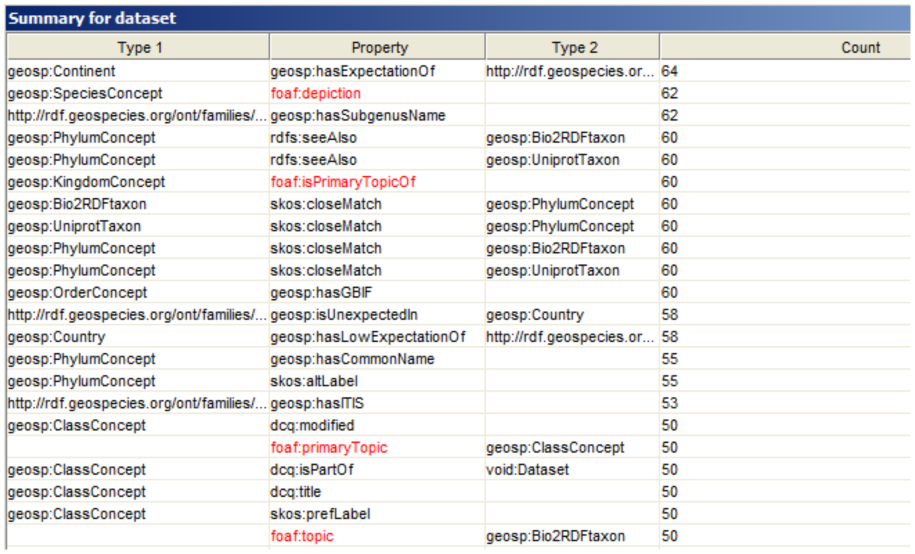
Fig. 2. Dataset choice in dataset-specific annotation

3 Coherence Checking Examples

For each OBML considered, the distinctions underlying a particular OFM can be compared to a predefined set of coherence rules. For OntoClean there are four standard coherence rules [3]: Given two properties, p and q, when q subsumes p then: a) if q is anti-rigid then p must be anti-rigid, b) if q carries an identity criterion then p must carry the same criterion, c) if q carries a unity criterion then p must carry the same criterion, and d) if q has anti-unity then p must also have anti-unity. The PURO OBML, in turn, specifies three constraints: for a) entity coherence, b) type coherence, and c) relation coherence (for details see [9]). We demonstrate the coherence checking on two example annotations.

The first is a fragment of the GR ontology annotated with PURO OBML,[11] containing class *ProductOrService* with subclasses *Individual* and *ProductOrSer-*

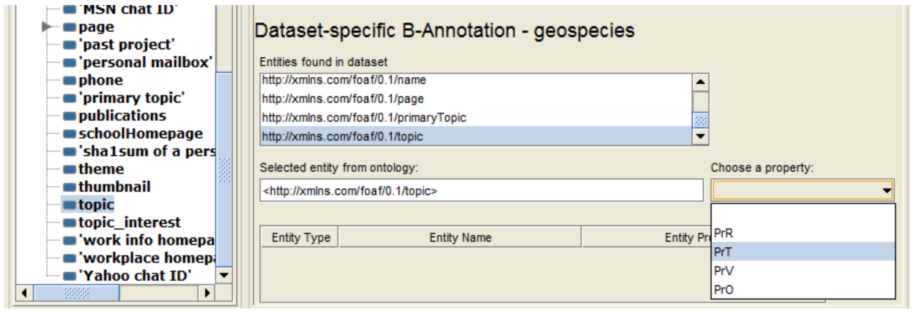
[11] `http://patomat.vse.cz/gr_mm.owl`



Summary for dataset

Type 1	Property	Type 2	Count
geosp:Continent	geosp:hasExpectationOf	http://rdf.geospecies.or...	64
geosp:SpeciesConcept	foaf:depiction		62
http://rdf.geospecies.org/ont/families/...	geosp:hasSubgenusName		62
geosp:PhylumConcept	rdfs:seeAlso	geosp:Bio2RDFtaxon	60
geosp:PhylumConcept	rdfs:seeAlso	geosp:UniprotTaxon	60
geosp:KingdomConcept	foaf:isPrimaryTopicOf		60
geosp:Bio2RDFtaxon	skos:closeMatch	geosp:PhylumConcept	60
geosp:UniprotTaxon	skos:closeMatch	geosp:PhylumConcept	60
geosp:PhylumConcept	skos:closeMatch	geosp:Bio2RDFtaxon	60
geosp:PhylumConcept	skos:closeMatch	geosp:UniprotTaxon	60
geosp:OrderConcept	geosp:hasGBIF		60
http://rdf.geospecies.org/ont/families/...	geosp:isUnexpectedIn	geosp:Country	58
geosp:Country	geosp:hasLowExpectationOf	http://rdf.geospecies.or...	58
geosp:PhylumConcept	geosp:hasCommonName		55
geosp:PhylumConcept	skos:altLabel		55
http://rdf.geospecies.org/ont/families/...	geosp:hasITIS		53
geosp:ClassConcept	dcq:modified		50
	foaf:primaryTopic	geosp:ClassConcept	50
geosp:ClassConcept	dcq:isPartOf	void:Dataset	50
geosp:ClassConcept	dcq:title		50
geosp:ClassConcept	skos:prefLabel		50
	foaf:topic	geosp:Bio2RDFtaxon	50

Fig. 3. Dataset summary for Geospecies, with FOAF entities emphasized



Dataset-specific B-Annotation - geospecies

Entities found in dataset

http://xmlns.com/foaf/0.1/name
http://xmlns.com/foaf/0.1/page
http://xmlns.com/foaf/0.1/primaryTopic
http://xmlns.com/foaf/0.1/topic

Selected entity from ontology: <http://xmlns.com/foaf/0.1/topic>

Choose a property:

Fig. 4. Annotation of *foaf:topic* by a PURO label, for Geospecies

viceModel. Using DL consistency checking over the PURO meta-ontology[12] and this fragment leads to inferred membership of class *ProductOrService* to a special 'diagnostic' class of the PURO ontology: *Incoherent-TPU*. This class which contains meta-models of classes that 'do not have homogeneous instances' in terms of PURO, specifically, whose instances can be both particulars and universals.

The second example is annotation of a fragment of the ontology used to demonstrate OntoClean inconsistencies in [3]. This fragment[13] includes meta-entities representing six classes of the original ontology annotated with OntoClean labels. The OntoClean meta-ontology used for coherence checking[14] allows for validation of all four coherence rules. The ontology contains four classes (Incoherence-Antiunity, Incoherent-Identity, Incoherent-Rigidity, Incoherent-Unity) that are – as result of inference – filled with individuals that represent classes in meta-model that are incoherent with regard to respective OntoClean rules. For example, the class *AmountOfWater* was annotated with OntoClean labels **+O ∼U +R**. Its subclass *LivingBeing* was annotated with OntoClean labels **+O +U +R**. The defect of the model is that a class with anti-unity label (simply said, class of objects whose arbitrary 'section' is again an instance of the same class) cannot subsume a class with unity label (i.e., containing objects that have 'strict boundaries around themselves'). Therefore it is inferred that the individual meta-modelling the class *LivingBeing* belongs to the diagnostic class *Incoherent-Antiunity*.

4 Conclusions and Future Work

The *B-Annot* plugin represents the first proof-of-concept implementation of annotation technology for ontologies and vocabularies that is (1) not restricted to a single theoretical framework but supports multiple OBMLs, and (2) interconnects the browsing/editing of ontologies (as supported by common ontological editors) with LD summaries. It is a part of a prospective eco-system of tools (other existing ones include, e.g., pattern-based ontology transformation tools [6]) supporting (informed rather than merely intuitive) reuse and design of ontologies on the semantic web.

Serious usability tests and requirement collection for *B-Annot* is only planned after some of the envisaged enhancements will have taken place.

A straightforward extension of *B-Annot* will be the possibility to also store annotations in OWL annotation properties of a copy of the annotated ontology. This will allow for easy browsing of the annotations in their original context.

Background annotation by distinctions referring to notions like 'rigid' (in OntoClean) or 'particular' (in PURO) risks to discourage even reasonably experienced ontological engineers without philosophical background. The threshold should thus be set as low as possible in the future, via operationalized annotation guidelines. For OntoClean's rigidity alternatives, a promising approach has

[12] `http://patomat.vse.cz/puro_v1.1.owl`
[13] `http://patomat.vse.cz/ontoclean-coherence-check-1.owl`
[14] `http://patomat.vse.cz/ontoclean-v.1.0.owl`

already been shown by Seyed, who designed a wizard relying on common-sense verbalization of the meaning of these alternatives [5]. For the PURO OBML distinctions, textual guidelines with examples have already been designed and tested in an classroom assignment; the experience gained will be used to design verbalisation templates similar to those from [5].

As the amount of mature vocabularies and their stable entities is still low[15] their purely manual annotation via *B-Annot* is feasible. In long term, however, partial automation could be achieved by leveraging on two different sources: (1) via linguistic parsing of associated texts, especially the values of `rdfs:comment`, and, (2) via logically inferring the most likely annotations based on previously assigned annotations of interrelated entities, e.g., from superclasses to subclasses.

This work has been supported from the EU ICT FP7 under no. 257943 (LOD2 project), from the VSE IGA project no. 34/2014, from the Slovak VEGA project no. 1/1333/12, and from project APVV-0513-10.

References

1. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats An Extensible Framework for High-Performance Dataset Analytics. In: EKAW 2012, Galway, Springer LNCS 7603.
2. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: Proc. ISWC 2010.
3. Guarino, N., Welty, C.: An Overview of OntoClean. In: Staab, S., Studer, R., eds.: *The Handbook on Ontologies*, pp. 151–172, Springer-Verlag, 2009.
4. Presutti, V., et al.: Extracting core knowledge from Linked Data. In: Proceedings of the Second Workshop on Consuming Linked Data, COLD2011. (2011)
5. Seyed, P.: A Method for Evaluating Ontologies – Introducing the BFO-Rigidity Decision Tree Wizard. In: FOIS 2012: 191–204.
6. Šváb-Zamazal, O., Dudáš, M., Svátek, V.: *User-Friendly Pattern-Based Transformation of OWL Ontologies.* In: Proc. EKAW'12, Galway.
7. Svátek, V., Homola, M., Kl̆uka, J., Vacura, M.: Ontological Distinctions for Linked Data Vocabularies. Technical Report TR-2013-039. Comenius University, Bratislava, 2013. Available online: `http://kedrigern.dcs.fmph.uniba.sk/reports/display.php?id=54`
8. Svátek, V., Homola, M., Kl̆uka, J., Vacura, M.: Mapping Structural Design Patterns in OWL to Ontological Background Models. In: Proc. K-CAP 2013, ACM.
9. Svátek, V., Homola, M., Kl̆uka, J., Vacura, M.: Metamodeling-Based Coherence Checking of OWL Vocabulary Background Models. In: Proc. OWLED 2013, online `http://ceur-ws.org/Vol-1080/owled2013_6.pdf`.
10. Welty, C.: OntOWLClean: Cleaning OWL ontologies with OWL. In: Proc. FOIS 2006.

[15] The statistics at `http://lov.okfn.org/dataset/lov/stats/` reveals that out of the several thousand entities referenced in LD, there are only about 150 that are at the same time reused by more than one other vocabulary and instantiated by at least 100 LOD instances.