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.

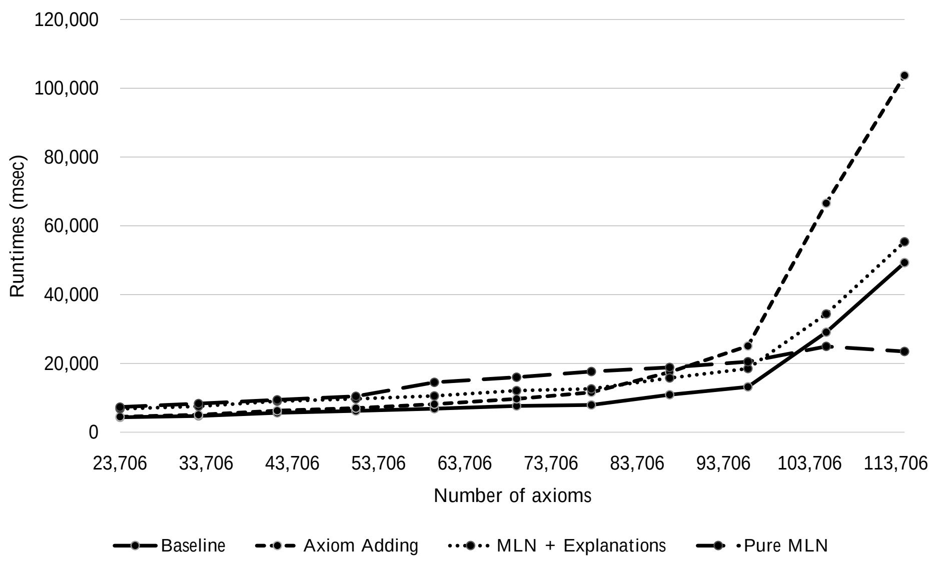

Fig. 2. Runtime behavior

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)