Debugging Weighted Ontologies

Heiner Stuckenschmidt

University of Mannheim, Germany

Abstract. We present our work on debugging weighted ontologies. We define this problem as computing a consistent subontology with a maximal sum of axiom weights. We present a reformulation of the problem as finding the most probable consistent ontology according to a log-linear model and show how existing methods from probabilistic reasoning can be adapted to our problem. We close with a discussion of the possible application of weighted ontology debugging to web scale information extraction.

1 Motivation

Probably the most often quoted advantage of logic-based ontologies are the possibility to check the model for different kinds of logical inconsistencies as possible symptoms of modeling errors. Since the work of Schlohbach and Cornet [19] many researchers have investigated the task of debugging description-logic ontologies, which dies not only include the detection of logical inconsistency, but also the identifying minimal sets of axioms causing it and removing axioms from the ontology to make it consistent again (e.g. [18, 16, 6, 8]).

While computing the cause of an inconsistency is relatively well understood and established techniques from diagnostic reasoning like the hitting set algorithm have been successfully applied and adapted to the problem of debugging ontologies, the decision which axioms to discard to retain consistency is still a largely unsolved problem. The classical solution used for instance in the field of believe revision is principle of minimal change that prefers solutions that remove the least number of axioms (compare e.g. [21]. While this approach has theoretical merits, it is not adequate for practical applications. For certain special cases such as debugging ontology mappings we can even observe that the principle of minimal change will remove correct axioms in most cases leaving incorrect ones in. As a consequence, researchers have focused on interactive debugging methods where a human user decides which axioms to remove while being supported by the debugging system [8, 10].

While interactive repair of ontologies is feasible when ontologies are rather small, more recently researchers get interested in debugging ontologies that have been automatically created from text or data sources. The resulting models are typically quite big and contain a high number of inconsistencies. While many classical debugging tools already have trouble in more classical settings as we have shown in our study on the practical applicability of debugging [20], using these tools on sets of automatically generated axioms turns out to be a hopeless endeavor.

In this paper, we summarize work on a new approach to ontology debugging that can be seen as a generalization of classical ontology debugging and that is also better suited for the task of debugging large, highly inconsistent models. Our approach is

based on the idea that axioms in the ontology to be debugged have weights assigned and the task is to remove axioms from this set such that the resulting model is consistent and the sum of weights is maximal. The second part of this definition provides us with an unambiguous criterion for selecting axioms to remove. Further, this definition of debugging weighted ontologies is equivalent to computing the most likely model in log-linear probabilistic models. We can use this correspondence to apply scalable inference mechanisms from the area of statistical relational learning to the task of ontology debugging.

The remainder of the paper is structured as follows: we first introduce a rather generic model of weighted ontologies that applies to different logical formalisms including light weight description logics and explain the relation to log linear models. In the second part of the paper, we discuss different different algorithms for debugging weighted ontologies based on linear integer programming and on Markov Chain Monte Carlo Sampling. We also discuss approach for scaling up these algorithms by distribution and parallel processing. We close with a discussion of open issues and future work.

2 Weighted Ontologies

2.1 Ontologies

We use a rather abstract ontology model that regards an ontology as a set of Axioms $\mathcal{O} = \{A_1, \cdots, A_n\}$. We represent axioms as predicates over constants representing classes, relations and instances. Existing representations of ontologies can be transferred into this representation by first normalizing the logical representation, eventually introducing new concept constants and then translating normalized axioms into literals. A complete translation for the logic $\mathcal{EL}^+$ can be found in [13]. The following example shows our representation of an ontology talking about philosophers and celestial objects:

$$A_1 : type(Pluto, Philosopher) \tag{1}$$
$$A_2 : related(born-in, Pluto, Athens) \tag{2}$$
$$A_3 : domain(born-in, Person) \tag{3}$$
$$A_4 : type(Pluto, DwarfPlanet) \tag{4}$$
$$A_5 : subconcept(Philosopher, Person) \tag{5}$$
$$A_6 : subconcept(Planet, CelestialObject) \tag{6}$$
$$A_7 : subconcept(DwarfPlanet, Planet) \tag{7}$$
$$A_8 : disjoint(CelestialObject, Person) \tag{8}$$

Our model further assumes the existence of an entailment relation $\models$ between sets of Axioms. Often, the entailment relation can be computed using a finite set of derivation rules. This observation corresponds to the investigation of consequence driven reasoning for description logics. In particular, for any description logic supporting consequence driven reasoning, we can compute the entailment relation between sets of using

derivation rules over the predicate representation. A correct and complete set of inference rules for $\mathcal{EL}^+$ can be found in [13]. For our example, we assume the following (incomplete set of) derivation rules for computing the entailment relation.

$$type(X, C) \wedge subclass(C, D) \vdash type(X, D) \quad (9)$$

$$subclass(C, D) \wedge subclass(D, E) \vdash subclass(C, E) \quad (10)$$

$$domain(R, C) \wedge related(X, R, Y) \vdash type(X, C) \quad (11)$$

$$type(X, C) \wedge type(X, D) \wedge disjoint(C, D) \vdash \bot \quad (12)$$

$$subclass(C, D) \wedge disjoint(C, D) \vdash \bot \quad (13)$$

We include the $\bot$ symbol for representing conflicts in the ontology. Abusing notation, we use $\bot$ for any kind of conflicts we want to exclude from the model. Concerning classical debugging the operator can be used to determine the existence of a logical inconsistency as well as incoherent classes in the same framework. We could also include domain-specific types of inconsistencies and detect them using the same algorithms as for the logical inconsistencies.

I our model, the task of ontology debugging can now be defined as finding a minimal subontology $\mathcal{O}' \subseteq \mathcal{O}$ such that $\mathcal{O}' \not\models \bot$ and there is no other subontology $\mathcal{O}' \subseteq \mathcal{O}''$ with this property. In our example such a sub-ontology can be generated by either removing axiom 1 and 2 or any of the axioms 3 to 8.

2.2 Weighted Axioms and log-linear Models

In our work, we consider cases, where not all axioms in an ontology have the same status, but some are preferred over others. We model this preference by a simple weight function $w : \mathcal{O} \rightarrow R \cup \{\infty\}$ where R denotes all real numbers and the weight function maps each axiom of an ontology either on a real number or on $\{\infty\}$ if the axiom should not be removed in any case. In the presence of a weight function, the notion of debugging is slightly changed. It can now be phrased as the task of finding a sub ontology $\mathcal{O}' \subseteq \mathcal{O}$ such that $\mathcal{O}' \not\models \bot$ and the sum of the weights in the axioms is maximal:

$$\sum_{A_i \in \mathcal{O}'} w(A_i) \geq \sum_{A_j \in \mathcal{O}''} w(A_j), \forall \mathcal{O}'' \subseteq \mathcal{O}$$

Let us assume that the first two axioms in our example have been automatically extracted while the other statements have been manually created by an expert. We could model this situation by assigning a lower weight to the first two axioms and higher weights to the other statements to indicate that we have more trust in the manually created parts of the model. So we might define $w(A_i) = 2, i \in \{1, 2\}$ and $w(A_i) = 5, i < 2$. In this case the only debugging of the resulting weighted ontology is $\mathcal{O}' = \{A_3, \cdots, A_8\}$ with a weight-sum of 30, whereas all other possible debuggings have at most a weight sum of 29.

In our work, we exploit the duality of this definition of debugging with log-linear models - probabilistic models where the a priori probabilities are given in terms of real-valued weights that are treated as logarithms of the actual probability. Thus, computing

the joint probability of independent events is done by summing up the weights instead of multiplying the probabilities. This means that the ontology with the highest weight sum is the most probable ontology according to a log-linear model over the weights of the axioms. In the case of only positive weights as in our example, the most probable ontology is always the one that contains all axioms. If we, however, force the probability of any sub ontology $\mathcal{O}' \models \bot$ to be zero, computing the most probable ontology turns out to be equivalent to computing a debugging as defined above. In particular, we define the probability of a subontology as follows:

$$P(\mathcal{O}') = \begin{cases} \frac{1}{Z} \exp\left(\sum_{\{A_i \in \mathcal{O}'\}} w(A_i)\right) & \text{if } \mathcal{O}' \not\models \bot \\ 0 & \text{otherwise} \end{cases}$$

Using this definition, debuggings of an ontology are simply the results of $\mathrm{argmax}_{\mathcal{O}' \subseteq \mathcal{O}}(P(\mathcal{O}'))$.

3 Debugging Algorithms

Actually computing debuggings is quite challenging is requires a combination of logical (for checking whether $\bot$ follows from a subontology) and probabilitic (for computing the probability of a model) inference. It turns out that naive approaches although they work for some special cases such as debugging alignments between small ontologies [9], fail to scale up to real world ontologies. At this point, we directly benefit from the above explained duality of debugging and inference in log-linear models, because we can build upon existing work in the area of probabilistic inference and design reasoning methods that scale up to very large (weighted) knowledge bases.

In the following, we describe two directions of work on algorithms for efficient debugging of weighted ontologies: the first one is based on a translation into an optimization problem that can be computed by solving a linear integer program. This work has already successfully been implemented in the ELOG reasoning system [1] develop at the university of Mannheim and is ready to use with OWL ontologies that have weights assigned as annotation properties [14]. The second direction is based on the idea of Sampling-based approximate inference that has the potential to scale to very large models. This work, that is based on Markov Chain Monte Carlo Sampling of ontologies has so far mostly been investigated on a theoretical level. First experiments have been made that show the potential of the method, but so far no stable reasoner is available.

3.1 Exact Inference using Linear Integer Programming

The first direction of work is based on the simple observation, that computing the most probable model can be phrased as an optimization problem and represented in terms of a linear integer programm. A linear integer program consists of an objective function that consists of the sum of integer variables with weights that has to be maximized. Further, side-conditions on the values of the variables can be stated in terms of linear inequalities over the variables. As we are interested in the presence or absence of axioms in an

[1] http://code.google.com/p/elog-reasoner/

ontology, we only consider Variables that have values from $\{0, 1\}$. A simple example of a linear integer program is $\mathbf{maximize}\, 0.6x_1 + 1.0x_2 + 0.5x_3$, $\mathbf{subjectto}\, x_1 + x_2 + x_3 \leq 1.2$. The solution of the example is: $x_1 = 1, x_2 = 0, x_3 = 1$. Instantiating the variables in the objective function with these values results in an objective value of 1.1.

The main task is now to find an optimal encoding of the problem into an integer linear programme. Riedel has proposed such a translation as a basis for efficient inferene in Markov logic [17]. As our representation of axioms as predicates and well as the corresponding inference rules can be represented as a Markov logic model, we can use the proposed translation as a basis for solving our problem. In particular, we can use the following steps for translating an ontology and the corresponding deduction rules into a linear integer programme:

1. replace non ground formulas with all possible groundings
2. Convert the resulting propositional knowledge base to conjunctive normal form
3. For each ground clause g determine positive $L^+(g)$ and negative $L^-(g)$ literals.
4. Determine the objective function as sum over all ground clause variable z_g and their weights
5. For each ground clause with weight $\neq \infty$ add the following constraints:

$$\sum_{l \in L^+(g)} x_l + \sum_{l \in L^(g)} (1 - x_l) \geq z_g$$

$$x_l \leq z_g, \forall l \in L^+(g)$$

$$(1 - x_l) \leq z_g, \forall l \in L^-(g)$$

6. For each ground clause with weight $= \infty$ add the following constraint

$$\sum_{l \in L^+(g)} x_l + \sum_{l \in L^(g)} (1 - x_l) \geq 1$$

7. Add the constraint $x_\perp = 0$ to enforce that $\perp$ is excluded from the model.

The solution of the corresponding debugging problem can be read from the solution of the linear integer programme. Each axiom in the ontology corresponds to a variable in the objective function, the solution of the debugging problem is the ontology that results from excluding all axioms from the model whose value is 0 in the objective function.

In our work [15] we have further optimized the translation procedure by translating clauses that share literals into a single constraint with counting variables. This approach has been shown to deliver a significant improvement for models with a high number of constants as it exploits symmetries in the resulting ground formulas to avoid repeated computations.

3.2 Approximate Inference using Markov-Chain Monte Carlo Sampling

While the ILP-based approach described above works well for medium sized knowledge-based, it runs into problems for very large models. In particular, if we think about using the methods on web scale, we quickly recognize that an optimal approach like the one described above is bound to fail. In such situations, where optimal algorithms fail, we can still use approximate inference methods for probabilistic models. A class of approximate inference methods that turned out to apply to our problem is Markov Chain Monte Carlo Sampling. In particular, we can adapt methods for sampling in dependent node sets from hypergraphs for our problem. For this purpose, we interpret an ontology as a hypergraph, where every axiom is a node in the hypergraph and nodes are connected by a hyperedge iff they form a diagnosis (i.e. a minimal set of axioms from which $\bot$ follows). A debugging of the ontology then corresponds to finding a maximal independent node set with respect to the weights of the axioms. Such an independent node set can now be determined by a Markov chain [7]. In [12] we proposed the following Markov Chain for computing weight-optimal debuggings in the sense of this paper.

A markov chain is a stochastic process with discrete time steps that is memoryless in the sense that its state at time t only depends on the state in t-1. Markov Chain Monte Carlo Methods are a class of algorithms that sample a probability distribution by constructing a Markov Chain that converges towards the desired distribution. We construct a Markov chain whose states are axiom subsets of the original ontology. It starts with an empty set of axioms and converges towards a state that corresponds to the weight optimal debugging of the ontology. Let $X^{(t)}$ be the state of the Markov Chain at time t, the state of the chain at time t+1 is computed as follows:

- chose and Axiom A uniformly at random
- if A is in $X^{(t)}$ then remove it with probability $\frac{1}{(\exp(w(A))-1)}$
- if A is not in $X^{(t)}$ and it is not in any diagnosis than add it with probability $\frac{\exp(w(A))}{(1+\exp(w(A)))}$
- if A is not in $X^{(t)}$ and it is in a diagnosis, then choose an other axiom from that diagnosis as random and replace it with A with probability $\frac{(m-1)\exp(w(A))}{(2m\exp w(A)-1)}$

First experiments on the PROSPERA Dataset [11] indicate that the method works well also on very large datasets that cannot be handled by optimal algorithms any more.

4 Conclusion: Debugging the Web

In this paper, we discussed the problem of debugging weighted ontologies. The problem can be seen as a generalization of ontology debugging where we have additional information about axiom preference in terms of weights assigned to axioms that can be used to compute a consistent ontology with a maximal sum of weights. We discussed the relation to computing the most probable consistent ontology using log-linear models and showed how we can exploit existing work from the field of probabilistic inference to efficiently compute debuggings. We believe that this method has a lot of potential and a lot of applications, in particular with respect to improving the results of web-scale information extraction.

As already indicated in the previous sections, our aim is to scale up the methods as far as possible. The ultimate goal is to address the web as a source of universal knowledge about the world. Recently a number of large scale knowledge extraction projects have been launched including NELL [2] TEXTRUNNER [3] and Knowitnow [1]. These projects extract more or less accurate facts from webpages building large knowledge bases about the world. Despite the use of high end extarctoin methods, the resulting models still contain mistakes and contradictions that need to be resolved to have a reliable model of world knowledge. In principle, our methods are able to integrate the results of these systems into a single, non conflicting model. For this purpose, however, we have to solve two problems: the first is to make our methods work on the scale of millions of facts as provided by these projects, further we have to model knowledge about conflicts between different facts in terms of a background ontology. While the first one is currently being addressed in terms of implementing the above mentioned sampling approach on a hadoop-based distributed infrastructure, we address the second problem by aligning the results of the extraction projects to the dbpedia ontology by matching objects and relations. If successful, we can use existing work on enriching the DBpedia ontology [5, 4] to determine logical inconsistencies.

Acknowledgement

The work summarized in this abstract has been joint work with Christian Meilicke, Mathias Niepert and Jan Noessner

References

1. Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of the Conference on Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, 2005.
2. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr, and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, page 13061313, 2010.
3. O. Etzioni, M. Banko, S. Soderland, and D.S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
4. Daniel Fleischhacker and Johanna Vlker. Inductive learning of disjointness axioms. In *On the Move to Meaningful Internet Systems: OTM 2011 : Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011*, Lecture Notes in Computer Science, pages 680–697. Springer, 2011.
5. Daniel Fleischhacker, Johanna Vlker, and Heiner Stuckenschmidt. Mining rdf data for property axioms. In *On the Move to Meaningful Internet Systems: OTM 2012 : Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012*, Lecture notes in computer science, pages 718–735. Springer, 2012.
6. Gerhard Friedrich and Kostyantyn Shchekotykhin. A general diagnosis method for ontologies. In *Proceedings of 4th International Conference on Semantic Web (ISWC?05)*, pages 232–246, Galway, Ireland, 2005.

7. M. Jerrum and A. Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. In *Approximation algorithms for NP-hard problems*, pages 482–520. PWS Publishing, 1996.
8. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. Repairing unsatisfiable concepts in owl ontologies. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184, Budva, Montenegro, June 2006.
9. Christian Meilicke and Heiner Stuckenschmidt. Applying logical constraints to ontology matching. In *KI 2007: Advances in Artificial Intelligence : 30th Annual German Conference on AI*, 2007.
10. Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. Supporting manual mapping revision using logical reasoning. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, Illinois, USA, July 2008. AAAI Press.
11. N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the 4th International Conference on Web Search and Data Mining (WSDM)*, pages 227–236, 2011.
12. Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. Towards distributed mcmc inference in probabilistic knowledge bases. In *NAACL-HLT 2012 Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, Montreal, 2012.
13. Mathias Niepert, Jan Noessner, and Heiner Stuckenschmidt. Log-linear description logics. In Toby Walsh, editor, *IJCAI*, pages 2153–2158. IJCAI/AAAI, 2011.
14. Jan Noessner and Mathias Niepert. Elog: A probabilistic reasoner for owl el. In *Lecture Notes in Computer Science Web Reasoning and Rule Systems : 5th International Conference, RR 2011*, pages 281–286, Galway, Ireland, 2011. Springer.
15. Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. Rockit: Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proceedings of the 27th Conference on Artificial Intelligence (AAAI)*, 2013.
16. Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging owl ontologies. In *Proceedings of the 14th international World Wide Web Conference*, page 633?640, Chiba, Japan, 2005.
17. Sebastian Riedel. Improving the accuracy and efficiency of map inference for markov logic. In David A. McAllester and Petri Myllymki, editors, *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 468–475, Helsinki, Finland, July 9-12 2008. AUAI Press.
18. Stefan Schlobach. Diagnosing terminologies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, page 670?675, 2005.
19. Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 355–362, Acapulco, Mexico, August 2003. Morgan Kaufmann.
20. Heiner Stuckenschmidt. Debugging owl ontologies - a reality check. In Raul Garcia-Castro, Asuncin Gmez-Prez, Charles J. Petrie, Emanuele Della Valle, Ulrich Kster, Michal Zaremba, and M. Omair Shafiq, editors, *Proceedings of the 6th International Workshop on Evaluation of Ontology-based Tools and the Semantic Web Service Challenge (EON-SWSC-2008)*, volume 359 of *CEUR Workshop Proceedings*, Tenerife, Spain, June 2008. CEUR-WS.org.
21. Renata Wassermann. An algorithm for belief revision. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*. Morgan Kaufmann, 2000.