Towards scalable ontological reasoning using machine learning

Daniel Ruffinelli

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

Abstract. Ontological reasoning has become a very useful technique for several different applications. However, the use of large knowledge bases has shown that reasoning can be a very resource intensive task which does not scale well. The goal of this work is to explore the development of scalable reasoning approximation methods based on machine learning. Our most important concern is determining in which contexts would such methods be more convenient than currently available approximate reasoning techniques. For this purpose, we will study the use of currently available approximation approaches, and we will develop new machine learning based methods to compete with them. Our preliminary results already provide evidence that this is possible. However, there are several questions that need to be answered, e.g. what reasoning tasks can be efficiently approximated, or what is the appropriate feature representation for such purposes. Finally, it will be important to determine the degree of completeness and correctness of such methods based on machine learning, and compare them with approximate methods based on standard reasoning.

1 Problem statement

Ontologies are commonly used to represent and reason about knowledge in a certain domain. Examples of this are the Gene Ontology [1] which provides a vocabulary to describe genes and their relations, or the Semantic Web which has contemplated the use of ontologies since its conception [2]. However, reasoning over large knowledge bases is very resource demanding. For example, reasoning is in the worst case NExpTime-complete for OWL-DL [13].

One way to address the issue of scalability is to use less expressive languages which allow for more efficient reasoning. Such is the purpose of languages like OWL-EL or OWL-RL [13], all of which have reasoning procedures which are complete within the scope of the language. The problem with this approach is that real world knowledge cannot be limited to such languages, as evidenced by the existence of more expressive and widely used ontologies, e.g. schema.org or the upper level ontology DOLCE. As a consequence, when applied to more expressive datasets, the result is faster but incomplete reasoning. This is usually referred to as approximate reasoning, and it may be convenient for use cases with a high performance requirement or in real time applications, such as information retrieval [18].

While approximate methods are indeed more efficient than using sound and complete reasoning, they come with the intrinsic limitation with which they were designed. This means that their accuracy depends more on the expressiveness of the dataset to which they are applied. Consequently, what we propose in this work is the use of machine learning methods to approximate ontological reasoning in a more flexible way. More specifically, we propose to represent reasoning tasks as supervised learning problems, which require the use of a reasoner to label the training data. Our preliminary results already show that ABox consistency checking can be effectively represented as a binary classification problem [15]. However, we have also found that approximate methods based on standard reasoning are efficient for the same task [10].

All of this leads us to our main research questions: is it possible to approximate ontological reasoning with machine learning methods? And if so, in what context are machine learning methods more convenient than other existing approximate methods? This proposal elicits several further questions, both theoretical and practical, which we will address in more detail in the following sections. We hope that the feedback from the Doctoral Consortium can help us direct our efforts towards answering some of them.

2 Related work

There is already a considerable amount of research done in search of more efficient ways to reason with ontologies. There are early approaches where the deduction process is approximated by simplifying the inference algorithm [17]. However, some of these methods have been found to be ineffective in practical situations [6].

More recently, the use of less expressive languages has been a widely explored approach. An example of this is the *DL-Lite* family of description logics, which allow for the definition of basic ontological languages while providing reasoning tasks in polynomial time [3]. As mentioned before, other examples of this are the OWL profiles [13]. Other approaches achieve more efficient reasoning by relying on assumptions which are context specific, e.g. the SnoRocket reasoner which is designed to reason with the SNOMED CT biomedical ontology [11].

While machine learning has been used at times in reasoning related environments, e.g. for ontology learning [20], there is little work in the direction of our research question. Specifically, Fanizzi et al. [5] define kernel functions to encode similarity between individuals in description logic representations, and use it in combination with Support Vector Machines to generate models for approximate query answering. Similarly, in [4] the same group approximates instance retrieval and query answering by using a dissimilarity measure to extend the k-Nearest Neighbor algorithm. In more recent work, the same group adds terminological extensions to Decision Trees and Random Forests [16].

The works cited in the paragraph above use machine learning methods to approximate reasoning tasks. However, while we may share the approach and could learn from them, we do not in principle share the motivation. These methods, as well as methods employed in relational machine learning [14], are aimed at situations where there is incomplete or uncertain knowledge. We will attempt to develop flexible and scalable reasoning methods for the classical setting where there are both a complete TBox and any number of ABoxes.

3 Approach

We now describe our proposal (Section 3.1), and our preliminary results (Section 3.2).

3.1 Proposed approach

Our basic idea is that given a reasoning task, a TBox, and a set of ABoxes which use the vocabulary defined in the TBox, we could use a reasoner with only a portion of the data and then use this as labelled data to train a machine learning algorithm. This would result in a model that simulates the reasoning task at hand, which could then be applied to the whole dataset. Such an approach would free us from using the reasoner on the whole dataset, which might be considerably more costly.

This idea implies that the reasoning task we want to approximate should be modeled as a supervised learning problem. Moreover, since machine learning algorithms do not take DL assertions as input, but rather feature vectors, a crucial aspect of this approach is the transformation of ontological information into an appropriate feature representation.

As a starting point of this work, and as proof of concept, Paulheim and Stuckenschmidt [15] were able to successfully model the problem of ABox consistency checking as a binary classification problem, i.e. is the ABox consistent or not [15]. For this purpose, they used four different real world datasets: the DBpedia ontology [7] and the YAGO ontology [19] with their respective assertional data, and the Web Data Commons assertional data [12] with the GoodRelations ontology and the schema.org ontology. Both the DBpedia and the YAGO ontologies were used in combination with the upper-level DOLCE-Zero ontology, whose disjointness axioms provided a source for several of the inconsistencies. For each dataset, the authors transformed their assertional data into binary feature vectors, and then used standard machine learning algorithms to generate highly accurate models which behaved as efficient but incomplete reasoners.

An important step in that process was obtaining a number of ABoxes which would be large enough to train an accurate model. For the datasets which came from the Web Data Commons corpus, this was as simple as defining that all assertional axioms belonging to a single website constituted an ABox. As such, the authors were able to build as many ABoxes as the number of websites which were a source of this corpus. This resulted in ABoxes ranging from a small number of assertions to several dozens. For the DBpedia and YAGO datasets, they had to determine a way to systematically break down the ABox into small ABoxes. This meant that any inconsistency that involved the information of more than one ABox would not be detected by their method. For this purpose, the authors defined that an ABox consisted of a relational assertion along with all the type assertions for its subject and object.

For the transformation of ABoxes into feature vectors, the authors used the notion of path kernels as defined by Loesch et al. [9]. This meant taking the graph corresponding to each ABox, and starting from each node, extracting all paths up to a certain length, such that each path would then become a feature in the feature space. Thus, the feature representation of an ABox would consist of all paths which are present in said ABox.

The following example illustrates the importance of having the right the feature transformation procedure in a machine learning based approach. Let $\mathcal{A}_1$ be an ABox made up of the following assertions: $A(a)$, $B(b)$, $C(c)$, $P(a,b)$, $S(a,c)$, $A(d)$, $B(d)$.

Figure 1 shows the graph which corresponds to the ABox $\mathcal{A}_1$. The transformation method used in [15] would provide the following feature representation for this ABox: A, P_B, B, S_C, C. Here the underscore represents an edge in the graph. These are all the distinct paths found in the graph of $\mathcal{A}_1$ (RDF graphs also represent type assertions as edges, even though their arity is 1). Consequently, a binary feature vector corresponding to $\mathcal{A}_1$ would have all of these 5 features set to 1, and the rest to 0.

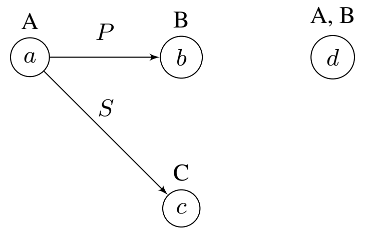

Fig. 1: Graph of the small ABox $\mathcal{A}_1$ with the following assertions: $A(a)$, $B(b)$, $C(c)$, $P(a, b)$, $S(a, c)$, $A(d)$, $B(d)$

However, another ABox $\mathcal{A}_2$ which consists of all the assertions found in $\mathcal{A}_1$ except for $A(d)$ and $B(d)$ would have almost the same corresponding graph as $\mathcal{A}_1$ (the node d would not exist), but this would still result in the exact same feature representation as $\mathcal{A}_1$. This can be a problem in a setting where the TBox has the axiom $A \sqsubseteq \neg B$, because if $\mathcal{A}_1$ is labelled as inconsistent by a reasoner, a machine learning approach would classify $\mathcal{A}_1$ as inconsistent too, even though that would be incorrect. Consequently, these are important considerations for reasoners based on machine learning.

In order to compete with [15], we developed another approximate method for ABox consistency checking which is not based on machine learning [10]. It relies on extending the clash queries for *DL-Lite*$_{\mathcal{A}}$ proposed by Lembo et al. [8]. This method uses a reasoner to build two caches: one which stores assertional patterns of consistent ABoxes, and one which stores assertional patterns of inconsistent ABoxes. The assumption is that these assertional patterns would constitute a partial explanation for the inconsistency of the ABox. For every new ABox, some combinations of its assertions were checked against the caches to determine its consistency. If this was not enough, then a reasoner was used. This way, the more ABoxes were tested, the larger the caches got, and the less a reasoner was required. Moreover, the caches could be built by using an arbitrary number of ABoxes, after which we could stop using a reasoner and instead rely only on the caches for consistency checking. We tested this method with two of the datasets used in [15], and compared the results with the clash queries method by Lembo et al. [8] and the machine learning method in [15].

The results from [10] (detailed below) are useful to illustrate that reasoning based approximations could be more effective than a machine learning based approach. It also shows that methods designed for restricted languages could be successful with data which uses more expressive ontologies. Still, as discussed in Section 4, an inductive method might have useful advantages over current reasoning approximation methods.

3.2 Preliminary results

Table 1 details the average results of running each method ten times, as reported in [10]. The column *Training* shows the number of ABoxes used for training and for building the caches. The column under *Runtimes* show the runtimes required for preprocessing steps and for the actual inconsistency checking, and the column *Accuracy* shows the accuracy of each method. In the case of the caching approach, preprocessing implies the caching of assertional patterns found in the ABoxes. In the case of the machine learning method, preprocessing implies the training phase, which includes using the full reasoner HermiT to label the ABoxes. In the case of HermiT, preprocessing means loading the TBox. These results were obtained with two datasets, one made up of 100000 ABoxes created from the DBpedia dataset in combination with the DOLCE-Zero ontology, and the other made up of 5000 ABoxes from the Web Data Commons corpus and using the GoodRelations ontology.

Table 1: Accuracy and runtime results of ABox consistency checking using our caching method, our machine learning method, and the HermiT reasoner

		Training	Runtimes		Accuracy
			Preprocessing (s)	Checking Inc. (ms)	
DBpedia	Caching	1000	77	0.041	98.6%
		10000	172	0.043	99.59%
		50000	253	0.04	99.84%
	ML	1000	98 + 1	0.356	97.62%
		10000	984 + 22	0.383	98.46%
		50000	4919 + 183	0.525	98.52%
	HermiT	-	10	98.38	100%
GoodRelations	Caching	50	0	0.318	99.89%
		500	1	0.315	99.92%
		2500	2	0.321	100%
	ML	50	1 + 0	1.483	95.60%
		500	12 + 0	1.589	99.87%
		2500	61 + 1	1.757	99.9%
	HermiT	-	2	24.48	100%

The results in Table 1 show that both approximation methods require considerably lower time to check the consistency of a single ABox when compared with a full reasoner, but both require considerably more time for preprocessing the data in order to train their approximate reasoner. Moreover, while both approximate methods are highly accurate, the caching method does have an advantage in these datasets.

4 Hypothesis and research questions

Our main research question is to determine whether machine learning can be used to approximate ontological reasoning. Consequently, we are also concerned with the following more specific questions:

1. What feature representation is required in order to accomplish this?
   The feature transformation method used in [15] has some limitations as explained in Section 3. This illustrates the need to find the right way to represent ontological information as feature vectors, such that reasoning can be approximated efficiently. For this purpose, we could define that a feature transformation method is reliable if for any two given ABoxes, one consistent and one inconsistent, it does not generate the same feature representation for both.
2. What other reasoning tasks can be approximated with machine learning?
   Aside from ABox consistency checking, we will try to approximate other reasoning tasks, e.g. instance retrieval, and study how this relates to the feature representation.
3. In what context do machine learning based approaches work best?
   This will require finding real life datasets with different expressiveness in order to test the performance of each method and the flexibility of machine learning based approaches. We could also generate artificial settings in order to test the limits of what the models can learn.
4. What is the relation between the expressiveness of the TBox and the feature representation used in machine learning methods for reasoning approximation?
   Since the correctness and accuracy of a machine learning based reasoner will largely depend on the feature representation which is used, exploring different methods for obtaining feature representations, and studying how these relate to the expressiveness of the TBox, will be a key aspect of our research.
5. Is it possible to use standard machine learning methods? Or are modifications required which allow the learned models to perform better?
   As mentioned in Section 2, there have been proposals where standard machine learning based methods are adapted for the purpose of approximating reasoning. Consequently, an important question is whether such models are more effective for this purpose. Moreover, due to the monotonicity of the reasoning tasks, developing models which consider this property is surely an advantage worth considering.
6. Is it possible to use explanations in order to make these methods more competitive?
   Another possible direction would be to consider working with explanations, either by developing a similar method to the one described in [10] but based on explanations, or by using machine learning algorithms to try to learn explanations and thus provide a more efficient way of obtaining them.
7. Can we develop machine learning based approaches which require training data whose size still allows them to be competitive?
   All machine learning based methods will require the use of a reasoner for training. This has a direct effect in the accuracy of the resulting model, but naturally more training translates to more preprocessing time, which makes these methods less competitive. As a result, studying the relation between the training data and the accuracy of the resulting model might be interesting. Moreover, while in principle an inductive method might learn different models for different datasets, in practice there might be a correlation between the expressiveness in the dataset and the amount of examples required for an inductive method to learn about certain constructs used in the dataset.

Finally, our hypothesis is that ontological reasoning can be approximated by using machine learning, that machine learning based methods can be at least as efficient as current methods, and that machine learning based methods can be more flexible by adapting to each dataset and learning what is required for them. All of this means that we will explore the use of existing approximation methods, we will develop new methods based on machine learning, and we will test their strengths and weaknesses.

5 Research plan

As this research is in its very early stages, we propose a reseach plan with two initial stages. After that, different directions might be considered depending on the results obtained in the first two stages.

For the first stage, we will continue working with consistency checking as the reasoning task to be approximated. In this stage we will focus on studying the role of feature representation in this context. This implies finding a feature representation which is adequate for fully representing ontological information, and also compiling a list of specific examples of assertional data which could be used as benchmarks to see whether a given feature representation accurately represents them. Additionally, in this stage we plan on extending our caching method presented in [10] by caching the assertional elements in the explanations of inconsistencies. This should result in a complete but less scalable model which we plan to use as baseline. Finally, in this stage we should find new datasets of different expressiveness and test our current methods in combination with some new feature transformation methods. We expect this stage to take about a year, after which a publication could result detailing the findings regarding the importance of feature representation in this context and their effect in approximating reasoning in different datasets.

In a second stage, we plan on approximating new reasoning tasks and seeing what effect this has on the feature representation. Moreover, on this stage we plan to explore the possibility of generating monotone models, and to consider extensions of standard machine learning approaches, such as the ones presented in [16]. We expect this stage to take about a year as well.

Regarding evaluation, if successful our method should be able to be trained on different datasets and learn a model which efficiently approximates specific ontological reasoning tasks. Moreover, since rather than relying on design restrictions to make it more efficient, our method would learn what is required for each dataset, it should be competitive with different approximate methods which may be designed for the specific datasets which we will use for testing.

Acknowledgments

I would like to thank Prof. Dr. Heiner Stuckenschmidt, Prof. Dr. Heiko Paulheim and Dr. Christian Meilicke for their guidance and support in the realization of this work.

References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene Ontology: tool for the unification of biology. Nature genetics 25(1), 25–29 (2000)
2. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific American 284(5), 28–37 (2001)
3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated reasoning 39(3), 385–429 (2007)
4. d'Amato, C., Fanizzi, N., Esposito, F.: Query answering and ontology population: An inductive approach. In: European Semantic Web Conference. pp. 288–302. Springer (2008)
5. Fanizzi, N., d'Amato, C., Esposito, F.: Statistical learning for inductive query answering on OWL ontologies. In: International Semantic Web Conference. pp. 195–212. Springer (2008)
6. Groot, P., Stuckenschmidt, H., Wache, H.: Approximating description logic classification for semantic web reasoning. In: European Semantic Web Conference. pp. 318–332. Springer (2005)
7. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: DBpedia– a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2), 167–195 (2015)
8. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent DL-Lite ontologies. In: International Conference on Web Reasoning and Rule Systems. pp. 155–169. Springer (2011)
9. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for RDF data. In: Extended Semantic Web Conference. pp. 134–148. Springer (2012)
10. Meilicke, C., Ruffinelli, D., Nolle, A., Paulheim, H., Stuckenschmidt, H.: Fast ABox consistency checking using incomplete reasoning and caching. In: International Joint Conference on Rules and Reasoning, RuleML+RR (2017), to appear
11. Metke-Jimenez, A., Lawley, M.: Snorocket 2.0: Concrete domains and concurrent classification. In: ORE. pp. 32–38. Citeseer (2013)
12. Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, RDFa and microformat dataset series. In: International Semantic Web Conference. pp. 277–292. Springer (2014)
13. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., et al.: OWL 2 web ontology language profiles. W3C recommendation 27, 61 (2009)
14. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE 104(1), 11–33 (2016)
15. Paulheim, H., Stuckenschmidt, H.: Fast approximate A-Box consistency checking using machine learning. In: Extended Semantic Web Conference. pp. 135–150. Springer (2016)
16. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Inductive classification through evidence-based models and their ensembles. In: European Semantic Web Conference. pp. 418–433. Springer (2015)
17. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artificial Intelligence 74(2), 249–310 (1995)
18. Shah, U., Finin, T., Joshi, A., Cost, R.S., Matfield, J.: Information retrieval on the semantic web. In: Proceedings of the eleventh international conference on Information and knowledge management. pp. 461–468. ACM (2002)
19. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a large ontology from wikipedia and wordnet. Web Semantics: Science, Services and Agents on the World Wide Web 6(3), 203–217 (2008)
20. Völker, J., Niepert, M.: Statistical schema induction. In: Extended Semantic Web Conference. pp. 124–138. Springer (2011)