Fast Approximate A-Box Consistency Checking Using Machine Learning

Heiko Paulheim(✉) and Heiner Stuckenschmidt

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

Abstract. Ontology reasoning is typically a computationally intensive operation. While soundness and completeness of results is required in some use cases, for many others, a sensible trade-off between computation efforts and correctness of results makes more sense. In this paper, we show that it is possible to approximate a central task in reasoning, i.e., A-box consistency checking, by training a machine learning model which approximates the behavior of that reasoner for a specific ontology. On four different datasets, we show that such learned models constantly achieve an accuracy above 95 % at less than 2 % of the runtime of a reasoner, using a decision tree with no more than 20 inner nodes. For example, this allows for validating 293M Microdata documents against the schema.org ontology in less than 90 min, compared to 18 days required by a state of the art ontology reasoner.

Keywords: Approximate ontology reasoning · Machine learning

1 Introduction

Ontologies have been a central ingredient of the Semantic Web vision from the beginning on [2,14]. With the invention of OWL, the focus has been on rather expressive ontologies that describe a rather limited set of instances. A number of advanced reasoning systems have been developed to exploit the logical semantics of OWL [11,13,42,47] to support tasks like consistency checking and the derivation of implicit information. While ontological reasoning has been shown to be useful for ensuring the quality of ontological models, it is also known that reasoning is very resource demanding both in theory and in practice. The tableaux algorithm for consistency checking in the Description Logic $\mathcal{SHOIN}$ that provides the formal underpinning of OWL-DL [16] is NExpTime-complete.[1] Further, for large ontologies, the size of the data structures created during the reasoning process easily exceeds the working memory of standard desktop computers.

Although there have been major improvements on the performance of Description Logic reasoners, using them for large real world data sets like DBpedia [20], YAGO [45], or schema.org [28] is not possible in real time. To overcome

[1] http://www.cs.man.ac.uk/~ezolin/dl/.

H. Sack et al. (Eds.): ESWC 2016, LNCS 9678, pp. 135–150, 2016.
DOI: 10.1007/978-3-319-34129-3_9

this problem, tractable fragments, i.e. OWL QL and OWL EL [27] have been proposed and corresponding reasoners have been developed that offer much better performance on models that adhere to these subsets [1,43]. However, real world ontology often do not adhere to the defined sublanguages, but rather use logical operators as needed for the purpose of capturing the intended semantics. Examples for popular ontologies outside the tractable fragments of OWL include FOAF, schema.org, and GoodRelations. This means that we need alternative solutions to be able to perform real time reasoning on large real world datasets.

On the other hand, there are good reasons to assume that in many practical settings, the reasoning process can be drastically simplified. For many ontologies, some constructs defined in the T-box are never or only scarcely used in actual A-boxes. For example, for the *schema.org* ontology, we have shown in [24] that a significant portion of the classes and properties defined in schema.org are never deployed on any web site. Similarly, for DBpedia, we have analyzed the reasoning explanations for A-box inconsistencies in [31]. We have observed that those inconsistencies are not equally distributed; on the contrary, no more than 40 different types of inconsistencies are responsible for 99 % of all inconsistencies in DBpedia. Such findings could help tailoring approximate reasoners for a specific ontology which focus on those parts of the ontology which are actually required to address the majority of all cases.

So far, the major part of research on ontology reasoning focuses on developing reasoning systems that are both sound and complete. However, we argue that reasoning results that are 100 % accurate are not required in many use cases for which ontology reasoning has been proposed and/or applied in the past, e.g., information retrieval [40], recommender systems [26], or activity recognition [4]. On the other hand, many of those use cases have very strict performance requirements, as they are usually applied in real time settings. For these settings, approximations of ontological reasoning are a promising approach. A common approach that has already been investigated in the early days of description logics is the idea of language weakening where a given model is translated into a weaker language that allows for more efficient reasoning [38]. While early ideas on approximating Description Logic reasoning have been shown to not be effective in practice [12], recent work on reducing OWL 2 to tractable fragments and using special data structures for capturing dependencies that cannot be represented in the weaker logic has been reported to provide good results [32].

In this paper, we show how to learn an approximate A-box consistency checking function automatically. To this end, we represent that task as a binary classification problem and apply machine learning method to construct a classifier that efficiently solves the problem. Such a classifier automatically learns to focus on those concepts and axioms in an ontology that are relevant in a particular setting, as discussed above. This idea can essentially be applied to any ontology and does not require any manual ground truth annotations, since we can use a sound and complete reasoner for generating the training data for the classifier. The contributions of this paper are the following:

- We present the general idea of imitating A-box consistency checking using machine learning.
- We apply the approach to the problem of consistency checking of a large number of A-boxes adhering to the same T-box.
- We evaluate the approach on four real world datasets, i.e. DBpedia, YAGO, schema.org Microdata, and GoodRelations RDFa.
- We show that the approach reaches more than 95 % accuracy, and performs the reasoning task at least 50 times faster than a state of the art Description Logic reasoner.

We see this work as a first step towards a more complete investigation of a principle of compiling complex logical reasoning into an efficient decision model that can be executed in the case of highly limited resources.

The paper is structured as follows. Section 2 discusses previous efforts on fast, approximate reasoning. We introduce our approach in Sect. 3, and an evaluation on four real-world Semantic Web datasets in Sect. 4. We conclude the paper with a summary and an outlook on future research directions.

2 Related Work

Some approaches have been proposed in the past to accelerate ontology reasoning. In essence, those encompass three families of techniques: approximating the deduction process itself (e.g. [3]), weakening the expressivity of the language, or compiling some intermediate results off-line (e.g. [32]).

Some reasoning approaches have been proposed that are specifically tailored to the needs of a particular model. For instance, the FacT reasoner, one of the first reasoners for OWL-DL, was originally designed for reasoning about knowledge in the GALEN Ontology [15]. More recently, the SnoRocket System was created to reason about the SnoMed Terminology [23], and in [46], Suda et al. present a special calculus and reasoning system for logical reasoning on the YAGO Ontology. Nevertheless, optimizing reasoning so far has focused on providing more efficient reasoning methods for a particular logic. In particular the EL fragment for OWL has been proposed as a language for very large models and highly optimized reasoners for OWL EL have been built (e.g. [18,43]) to solve the performance problem.

In contrast to those works, which always require some explicit assumptions on how to simplify a given reasoning problem (e.g., which deduction rules to weaken, which logic language concepts to drop, which intermediate results to materialize, etc.), the approach we follow is to train a machine learning model, where the learning algorithm automatically finds a good approximation for a given ontology, implicitly selecting an appropriate subset of strategies from all of the above.

The use of machine learning techniques to complement reasoning has been proposed for different purposes, e.g., as *ontology learning* for assisting ontology engineering and enrichment [19,48], or to introduce approximate class definitions, which are not part of standard ontology languages [33]. There have been attempts to apply machine learning techniques to reasoning problems in description logics ontologies. The corresponding approaches have addressed the problems query answering, classification, and instance retrieval, all of which can be reduced to deciding whether an instance belongs to a class expression in the ontology.

The connection between the classification and the reasoning problem is established through the definition of Kernel [8] or dissimilarity functions respectively that are defined based on joint membership of instances in ontological classes or their negation, where the membership is either determined using logical reasoning or approximated by a lookup of explicitly membership statements. These functions are then used as a basis for inductive methods, i.e. kNN [5] and Support Vector Machines [7].

In more recent work [36] from the same group, terminological extensions of decision tree [35] and random forest classifiers [34] have been proposed. The nodes of the decision trees may contain conjunctive concept expressions which are also represent the feature space. The induction of the trees is guided by an information gain criterion derived from positive, negative and unknown examples which correspond to instances known to belong to the class, to its negation, or that are not known to belong to any of the former. Similar extensions have been proposed for Bayesian classifiers and regression trees.

These methods were evaluated on a set of rather small ontologies with 50–100 classes (with one exception) and up to 1000 individuals.

The methods proposed in this paper share the motivation with the work above, the efficient approximation of deductive reasoning tasks by inductive methods. There are are some differences, however. While the work above focuses on the development of *new machine learning methods* for terminological data, our approach is based on *existing machine learning techniques*. Further, we address a different reasoning task, namely consistency checking instead of deciding class membership. We chose this task as it is the most basic operation in deductive reasoning. This allows us to later extend our approach to any kind of deductive reasoning by reduction to unsatisfiability checking. In particular, being able to approximately check inconsistency provides us with an approximate entailment operator using the principle of proof by refutation. Finally, to the best of our knowledge, our work is the first attempt to apply the idea to very large real world ontologies that are actually used in many applications. We consider models that are an order of magnitude larger than the ones considered in the works above, and it not clear whether the existing approaches scale to models of the size considered in this paper.

In another line of work, machine learning techniques have been used to predict the efficiency of terminological reasoning [17,37] – that work, however, is only weakly related as it uses different features and its goal is to predict the reasoning performance, not its result.

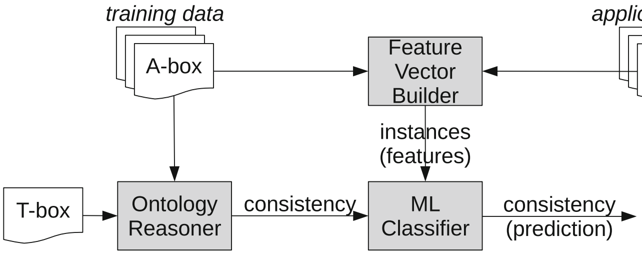


Fig. 1. Schematic view of our approach

3 Approach

With our approach, we try to approximate a reasoner that checks an A-box for consistency, given a T-box. Since the outcome of such a consistency is either true or false, we regard the problem as a binary classification problem when translating it to a machine learning problem.

Figure 1 depicts a schematic view of our approach. An ontology reasoner is run on a T-box and a small number of A-boxes, denoted as *training data*. At the same time, the A-boxes are translated to feature vectors for the machine learning classifier. The pairs of feature vector representations of the A-box and the consistency detection outcome of the reasoner are used as labeled training examples for the classifier, which then learns a model. This model can be applied to a (potentially larger) set of A-boxes (denoted as *application data*).

Formally, we train a classifier C for a T-box T that, given an A-box a in the set of all A-boxes A, determines whether that A-box is consistent or not:

$$C_T : A \rightarrow \{true, false\} \tag{1}$$

Standard machine learning classifiers do not work on A-boxes, but on *feature vectors*, i.e., vectors of nominals or numbers. Thus, in order to exploit standard classifiers, we require a translation from A-boxes to feature vectors, which is given by a function F (denoted as *feature vector builder* in Fig. 1). In this work, we use binary feature vectors, although other types of feature vectors would be possible:

$$F : A \rightarrow \{0, 1\}^n \tag{2}$$

With such a translation function, a binary machine learning classifier M, which operates on feature vectors, can be used to build C_T:

$$C_T := M(F(a)) \rightarrow \{true, false\}, where\ a \in A \tag{3}$$

In the work presented in this paper, we use *path kernels* (often referred to as *walk kernels* as well) as defined in [22] for the translation function F. We slightly altered the original definition such that for literals, we only use the datatype (`xsd:string` if none exists), but not the literal value as such. Those kernels generate an enumeration of all paths up to a certain lengths that exist

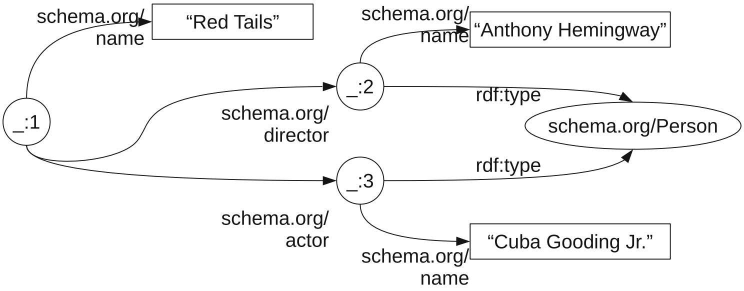


Fig. 2. Example excerpt of an RDF graph

in an RDF graph, and each path is used as a binary feature (i.e., the path is present in a given A-box or not).

Figure 2 depicts an example graph, which is a sub-part of an actual RDF graph in our schema.org Microdata dataset (see below). That graph would be represented by the following six features[2], which represent the number of different paths that exist in the graph:

```
schema.org/name_xsd:string
schema.org/director_schema.org/name_xsd:string
schema.org/director_rdf:type_schema.org/Person
schema.org/actor_schema.org/name_xsd:string
schema.org/actor_rdf:type_schema.org/Person
rdf:type_schema.org/Person
```

4 Evaluation

We evaluate our approach in four different settings:[3]

1. Validating relational assertions in DBpedia with DOLCE
2. Validating type assertions in YAGO with DOLCE
3. Validating entire Microdata documents against the schema.org ontology
4. Validating entire RDFa documents against the GoodRelations ontology

Furthermore, we compare four different learning algorithms for each setting: Decision tree, Naive Bayes, SVM with linear kernel[4], and Random Forest. *HermiT* [11] is used as a reasoner which produces the ground truth, and also as

[2] i.e., for those features, the feature value would be 1, for all others, it would be 0.

[3] All the datasets created in this paper are available online at http://dws.informatik.uni-mannheim.de/en/research/reasoning-approximation.

[4] In essence, since we use root paths, this is equivalent to using an SVM whose kernel function is the count of overlapping root paths of two instances.[6].

Table 1. Datasets used for the evaluation. The table shows the ontology complexity of the schema, the feature set size for the different sample sizes, as well as the percentage of consistent A-boxes in the samples.

Dataset	Complexity	# Features	% consistent
DBpedia-11	$\mathcal{SHIN}(\mathcal{D})$	61	72.73
DBpedia-111		216	84.68
DBpedia-1111		854	87.67
YAGO-11	$\mathcal{SHIN}(\mathcal{D})$	101	63.64
YAGO-111		621	73.87
YAGO-1111		4,344	71.38
schema.org-11	$\mathcal{ALCHI}(\mathcal{D})$	90	27.28
schema.org-111		250	35.14
schema.org-1111		716	35.10
GoodRelations-11	$\mathcal{SHI}(\mathcal{D})$	200	54.55
GoodRelations-111		508	60.36
GoodRelations-1111		862	63.10

a baseline for performance comparisons.[5] All experiments were conducted with *RapidMiner Studio*[6], for Random Forest, we use the implementation in the Weka extension. For the latter, the number of trees was set from 10 to 50 due to the large number of features in some datasets (see below), all other operators were used in their standard settings.

4.1 Datasets

While most type assertions in DBpedia are correct, the main sources of errors are relational and literal assertions [30,49]. Since the latter can hardly be detected by reasoners without literal range definitions (which do not exist in DBpedia), we concentrate on relational assertions. As a dataset, we use all mapping-based properties and types in DBpedia [20]. Since DBpedia as such comes with only few disjointness axioms, which are required for detecting inconsistencies, we add the top-level ontology DOLCE-Zero [9,10] to the reasoner. In this setting, we check whether a relational assertion is consistent with the subject's and object's type assertion. Hence, as proposed in [31], each A-box to be checked consists of a relation assertion and all the type assertions for the subject and object.

In YAGO, types are derived from Wikipedia categories and mapped to WordNet [44]. Thus, in contrast to DBpedia, incompatible class assertions are possible

[5] A reasoner capable of handling the $\mathcal{SHIN}(\mathcal{D})$ complexity class is required for the experiments (cf. Table 1). In a set of preliminary experiments, we tried both Pellet [42] and HermiT, where only the latter was able to handle the DBpedia+DOLCE ontology.

[6] http://www.rapidminer.com.

in YAGO. For example, the instance *A Clockwork Orange* has both the types *1962 Novels* and *Obscenity Controversies*, among others, which are ontologically incompatible on a higher level (a controversy being subsumed under *Activity*, a novel being subsumed under *Information Entity*). Again, since the YAGO ontology as such does not contain class disjointness axioms, we use it in conjunction with the DOLCE top level ontology. Similar to the DBpedia case, the reasoner checks individual A-boxes, each consisting of all the type assertions of a *single entity*. Here, we use types which have already been materialized according to the full hierarchy.

For the schema.org case, we use a sample of documents from the WebDataCommons 2014 Microdata corpus [25], and the corresponding schema.org ontology version.[7] Here, a single document is the set of triples extracted from one Web page. Since schema.org does not come with disjointness axioms, we have defined a set of disjointness axioms between the high level classes as follows: Starting from the largest class, we inspect all the class definitions (both formal and textual), and insert disjointness axioms where appropriate as long as the T-box does not become inconsistent. The set of those inconsistencies is shown in Table 2.

Likewise, we proceed with the GoodRelations RDFa documents. From the WebDataCommons RDFa corpus, we use a subset of documents that use at least one concept from the GoodRelations namespace, and validate it against the GoodRelations 1.0 ontology[8].

In all cases, we use root paths of arbitrary lengths. This is possible since there are no cycles in the graphs, since we are only validating single relational statements for DBpedia, single instances for YAGO, and the documents in the schema.org Microdata are also cycle-free [29]. For the GoodRelations dataset, cycles would be possible in theory, however, we did not encounter any in our sample. If an A-box to test was not proper DL (e.g., an object property was

Table 2. Disjointness Axioms inserted for schema.org. An *X* denotes a disjointness axiom defined, an *e* denotes that the disjointness axiom was expected, but lead to a T-box inconsistency. Since *Local Business* in schema.org is both a subclass of *Organization* and *Place*, the latter two cannot be disjoint. Likewise, *Exercise Plan* is both a subclass of *Creative Work* and *Medical Entity*.

	Product	Place	Person	Organization	Medical Entity	Intangible	Event	Creative Work
Action	X	X	X	X	X		X	X
Creative Work		X	X	X	e		X	
Event	X	X	X	X	X			
Intangible			X					
Medical Entity		X	X	X				
Organization	X	e	X					
Person	X	X						
Place	X							
Product								

[7] We have followed the recommendations on this web page to create the OWL version: http://topbraid.org/schema/.

[8] http://purl.org/goodrelations/v1.

used with a literal object), the instance was directly marked as inconsistent, without passing it to the reasoner.

For each of the three use cases, we extracted three versions, i.e., one with 11, one with 111, and one with 1,111 A-box instances. The rationale for those numbers is that it is possible to evaluate the performance of our approach using 10, 100, and 1,000 training instances, respectively, in a ten-fold cross validation setting. Table 1 summarizes the characteristics of the datasets.

4.2 Results

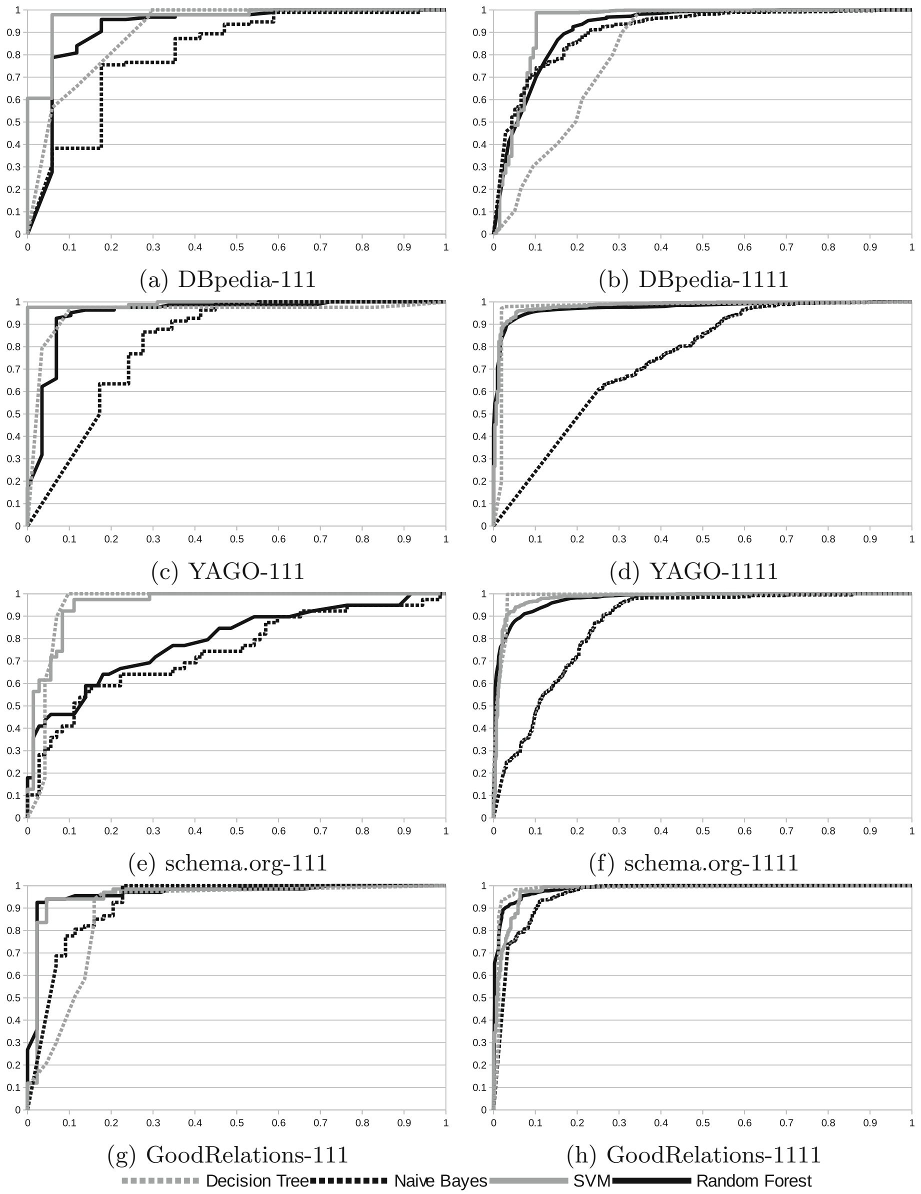
For all four pairs of datasets, we compare the results for the four learning methods. All experiments were conducted in 10-fold cross validation. Table 3 depicts the accuracy of all approaches.

From the results, we can see that in general, 10 training instances are too few to train a classifier significantly outperforming the majority prediction baseline. For the other cases, the decision tree classifier usually provides the best results (on the DBpedia datasets, the difference to SVM is not significant). Unlike the other classifiers, the performance of Naive Bayes is worse when trained with 1,000 instances than when trained with 100. This can be explained with the increase in the number of features when increasing the number of training examples (cf. Table 1). At the same time, there are many interdependent features, which violate the independence assumption of Naive Bayes.

Figure 3 depicts ROC curve plots for the four classifiers and the six problems. Since the table shows that meaningful models are only learned for 100 and more instances, we only depict ROC curves for the 111 and 1,111 datasets. ROC curves

Table 3. Performance of the different classifiers on the three problems. The table depicts the accuracy for all classifiers, including the confidence intervals obtained across a 10-fold cross validation. The best result for each dataset is marked in bold. The baseline is formed by predicting the majority class.

Dataset	Baseline	Decision Tree	Naive Bayes	SVM	Random Forest
DBpedia-11	**.727**	.700 ± .458	600 ± .490	.600 ± .490	.600 ± .490
DBpedia-111	.847	.937 ± .058	.756 ± .101	**.946 ± .073**	.901 ± .049
DBpedia-1111	.877	.951 ± .016	.856 ± .043	**.961 ± .015**	.926 ± .017
YAGO-11	.636	.550 ± .472	**.850 ± .320**	.650 ± .450	.650 ± .450
YAGO-111	.739	**.955 ± .061**	.805 ± .153	.892 ± .066	.901 ± .064
YAGO-1111	.714	**.976 ± .015**	.672 ± .056	.929 ± .052	.913 ± .034
schema.org-11	**.727**	.700 ± .458	.600 ± 0.490	.700 ± .458	.600 ± .490
schema.org-111	.645	**.936 ± .071**	.730 ± .158	.695 ± .066	.749 ± .182
schema.org-1111	.649	**.977 ± .020**	.762 ± .066	.778 ± .060	.906 ± .038
GoodRelations-11	.545	**.700 ± .458**	**.700 ± .458**	**.700 ± .458**	**.700 ± .458**
GoodRelations-111	.604	**.901 ± .076**	.847 ± 0.129	**.901 ± .076**	.892 ± .078
GoodRelations-1111	.631	**.969 ± .016**	.874 ± .027	.945 ± 0.20	.945 ± .027



(a) DBpedia-111

(b) DBpedia-1111

(c) YAGO-111

(d) YAGO-1111

(e) schema.org-111

(f) schema.org-1111

(g) GoodRelations-111

(h) GoodRelations-1111

Fig. 3. ROC curves for the six datasets and four classifiers. A classifier is better than another if its curve runs above the other classifiers curve. A random prediction baseline would lead to to a diagonal.

Table 4. Computation times for 1,000 statements with HermiT vs. Decision Tree, and size of the models learned. For the tree size, the first number is the number of inner nodes, while the number in parantheses depicts the maximum path length to a leaf node.

Dataset	1000 k instances (ms)				All A-boxes (s)		
	HermiT	Features	Training	Classification	HermiT	Trained	DT Size
DBpedia	86,240	259	256	2	1,293,733	4,002	13 (8)
YAGO	20,624	179	814	6	186,022	556	9 (7)
schema.org	7,769	138	285	2	1,574,014	4,110	19 (19)
GoodRelations	15,577	236	229	2	5,545	101	18 (13)

do not take into account only correct and incorrect predictions, but also reflect how good the confidence scores computed by the classifiers are. We can see that the curves often run close to the upper left corner, i.e., the confidence scores are also usually quite accurate.

In addition to the qualitative results, we also look into computation times. Here, we compare the runtime for the reasoner to the runtime of a Decision Tree based classifier, which has been shown to perform best in terms of accuracy. We compare the runtime for classifying 1,000 A-boxes, as well as provide an estimate for classifying all A-boxes in the respective settings. For the latter, we consider all relational statements in DBpedia (ca. 15M in the `dbpedia-owl` namespace), all typed YAGO instances (ca. 2.9M), all URLs from which schema.org Microdata can be extracted (ca. 293M), and all URLs from which RDFa annotations with GoodRelations can be extracted (ca. 356k). For estimating the time for checking all A-boxes, we extrapolated from the time to check 1,000 A-boxes. The extrapolation of the time for the trained model includes the time for running the reasoner and training the Decision Tree on 1,000 instances, as well as the classification with that Decision Tree on the remaining instances. More formally, the extrapolated time for all A-boxes is

$$T_{ext} = |A| \cdot T_F + T_T + (|A| - 1000) \cdot T_C, \tag{4}$$

where T_F is the time to create features on a single A-box, T_T is the time to train the model on 1,000 instances, T_C is the time to check an A-box with the trained model, and $|A|$ is the total number of A-boxes.[9] In total, it can be observed that the computation time based on the trained model is lower than the reasoning based computation time by at least a factor of 50.

For the runtimes, only pure computation times for the consistency determination were measured, but no initialization time for the reasoner (i.e., loading the ontology for the first time) and no I/O times.[10] Furthermore, to make the

[9] Note that the features are generated on each A-box in isolation, so there is only a linear dependency on the number of A-boxes.

[10] All runtimes were measured on a Windows 7 laptop with an Intel i7 quadcore processor and 8 GB of RAM.

comparison fair for YAGO, the reasoner was also provided with fully materialized type statements, and the type hierarchy was not loaded by the reasoner.

Table 4 sums up the computation times. It can be observed that for the Decision Tree based approach, the larger amount of time is spent on the creation of feature vector representations of the instances, rather than the classification itself. For example, we see that checking all of DBpedia's relational assertions[11] with our approach would take less than 90 min, whereas HermiT would take around 15 days. Likewise, checking all the schema.org Microdata documents in the Web Data Commons Corpus would take more than 18 days with HermiT, while our approach would also finish that task in less than 90 min.

Another interesting observation is that the size of the models used for classification is relatively small. While the ontologies are fairly large (loading DBpedia and DOLCE into HermiT for validating a single statement consumes 1.5 GB of RAM), the corresponding Decision Trees are surprisingly small, with no more than 20 inner nodes. Furthermore, the longest path to a leaf node, which corresponds to the number of tests to be performed on the A-box for computing the results, is also strongly constrained, which explains the high computational performance of the approach.

5 Conclusion and Outlook

In this paper, we have shown that it is possible to approximate a reasoner for A-box consistency checking by training machine learning classifiers. Especially decision trees have turned out to be quite effective for that problem, reaching an accuracy of more than 95 % at computation times at least 50 times as fast as a standard ontology reasoner. The resulting trees are surprisingly small, with no more than 20 decision nodes and tests to perform. This makes the approach appealing to be applied in scenarios with very strongly limited computation resources where not only computation time, but also memory is constrained.

The fact that the learned decision trees use only a very small number of features would allow for even faster implementations of such reasoners, since the tests in the inner nodes could also be performed by pattern matching against the A-box instead of the explicit creation of the entire set of features.

So far, we have focused on consistency checking. However, we believe that the approach is versatile enough to be transferred to other reasoning tasks as well. For example, the task of instance classification could be modeled and solved as a hierarchical multi-label classification problem [41]. Similarly, the task of *explaining* an inconsistency could be modeled as a multi-label classification problem, with the target of predicting the axioms in the explanation.

The evaluation has shown that with decision tree classifiers, it is not only possible to make predictions at high accuracy. The ROC curves furthermore show that the classifier is aware of its quality, i.e., wrong predictions usually go together with low confidence values. This gives way to developing a hybrid

[11] Note that this is *not* equivalent to checking the consistency of the entire A-box of DBpedia as a whole.

system which could deliver high quality results by exploiting active learning [39], i.e., deciding by itself when to invoke the actual reasoner (and incorporate its result in the machine learning model).

For the learning task, a challenging problem is to use an appropriate feature vector representation and configuration of the learning algorithm. For the former, we assume that there is a correlation between the ontology's expressivity and the optimal feature vector representation – e.g., for ontologies incorporating numerical restrictions other than 0 and 1, numeric feature vector representations should be preferred over binary ones. For the latter, we assume that there is a correlation between certain characteristics of the ontology and the machine learning algorithm configuration – for example, the number of required decision trees in a Random Forest is likely to depend on the number of classes and properties defined in the ontology.

The models issued by the learning algorithms have, so far, not been taken into account for any other purpose than the prediction. However, looking at which features are actually used by the decision tree could be exploited as well, e.g., for ontology summarization [21], since they refer to concepts that are likely to influence a reasoner's outcome.

In summary, we have shown that it is possible to train rather accurate approximate reasoners using machine learning. The findings give way to fast implementations of reasoning in scenarios which do not require 100 % accuracy. On a theoretical level, the approach opens up a number of interesting research questions.

Acknowledgements. The authors would like to thank Aldo Gangemi for providing the DOLCE mappings for DBpedia and YAGO, and Robert Meusel for his assistance in providing suitable samples from the WebDataCommons corpora. This work has been supported by RapidMiner in the course of the RapidMiner Academia program.

References

1. Baader, F., Lutz, C., Suntisrivaraporn, B.: CEL — a polynomial-time reasoner for life science ontologies. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 287–291. Springer, Heidelberg (2006)
2. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Sci. Am. **284**(5), 28–37 (2001)
3. Cadoli, M., Schaerf, M.: Approximation in concept description languages. In: KR, pp. 330–341 (1992)
4. Chen, L., Nugent, C.: Ontology-based activity recognition in intelligent pervasive environments. Int. J. Web Inf. Syst. **5**(4), 410–430 (2009)
5. d'Amato, C., Fanizzi, N., Esposito, F.: Query answering and ontology population: an inductive approach. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 288–302. Springer, Heidelberg (2008)
6. de Vries, G.K.D., de Rooij, S.: A fast and simple graph kernel for RDF. In: DMoLD, vol. 1082 (2013)

7. Fanizzi, N., d'Amato, C., Esposito, F.: Statistical learning for inductive query answering on OWL ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 195–212. Springer, Heidelberg (2008)
8. Fanizzi, N., d'Amato, C., Esposito, F.: Induction of robust classifiers for web ontologies through kernel machines. J. Web Sem. **11**, 1–13 (2012)
9. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening WORDNET with DOLCE. AI Mag. **24**, 13–24 (2003)
10. Gangemi, A., Mika, P.: Understanding the semantic web through descriptions and situations. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 689–706. Springer, Heidelberg (2003)
11. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an OWL 2 reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
12. Groot, P., Stuckenschmidt, H., Wache, H.: Approximating description logic classification for semantic web reasoning. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 318–332. Springer, Heidelberg (2005)
13. Haarslev, V., Möller, R.: Racer: a core inference engine for the semantic web. In: EON, vol. 87 (2003)
14. Hendler, J.: Agents and the semantic web. IEEE Intell. Syst. **2**, 30–37 (2001)
15. Horrocks, I., Rector, A.L., Goble, C.A.: A description logic based schema for the classification of medical data. In: KRDB, vol. 96, pp. 24–28. Citeseer (1996)
16. Horrocks, I., Sattler, U.: A tableau decision procedure for\ mathcal {SHOIQ}. J. Autom. Reasoning **39**(3), 249–276 (2007)
17. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 198–214. Springer, Heidelberg (2012)
18. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK. J. Autom. Reasoning **53**(1), 1–61 (2014)
19. Lehmann, J., Auer, S., Bühmann, L., Tramp, S. (geb. Dietzold).: Class expression learning for ontology engineering. J. Web Seman. 9(1), 71–81 (2011)
20. 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. Seman. Web J. **5**, 1–29 (2014)
21. Li, N., Motta, E., d'Aquin, M.: Ontology summarization: an analysis and an evaluation. In: International Workshop on Evaluation of Semantic Technologies (2010)
22. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for RDF data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 134–148. Springer, Heidelberg (2012)
23. Metke-Jimenez, A., Lawley, M.: Snorocket 2.0: concrete domains and concurrent classification. In: ORE, pp. 32–38. Citeseer (2013)
24. Meusel, R., Bizer, C., Paulheim, H.: A web-scale study of the adoption and evolution of the schema. org vocabulary over time. In: 5th International Conference on Web Intelligence, Mining and Semantics (WIMS), pp. 15. ACM (2015)
25. Meusel, R., Petrovski, P., Bizer, C.: The WebDataCommons microdata, RDFa and microformat dataset series. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 277–292. Springer, Heidelberg (2014)
26. Middleton, S.E., De Roure, D., Shadbolt, N.R.: Ontology-based recommender systems. Handbook on Ontologies. International Handbooks on Information Systems, pp. 779–796. Springer, Heidelberg (2009)

27. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 web ontology language: profiles. W3C recommendation, vol. 27, p. 61 (2009)
28. Patel-Schneider, P.F.: Analyzing schema.org. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 261–276. Springer, Heidelberg (2014)
29. Paulheim, H.: What the adoption of schema.org tells about linked open data. In: Dataset PROFIling & fEderated Search for Linked Data (2015)
30. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. Int. J. Seman. Web Inf. Syst. (IJSWIS) **10**(2), 63–86 (2014)
31. Paulheim, H., Gangemi, A.: Serving DBpedia with DOLCE – more than justadding a cherry on top. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 180–196. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25007-6_11
32. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for tbox reasoning. In: AAAI, pp. 351–356 (2010)
33. Rizzo, G., dAmato, C., Fanizzi, N.: On the effectiveness of evidence-based terminological decision trees. In: Esposito, F., et al. (eds.) ISMIS 2015. LNCS, vol. 9384, pp. 139–149. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25252-0_15
34. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Tackling the class-imbalance learning problem in semantic web knowledge bases. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 453–468. Springer, Heidelberg (2014)
35. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Towards evidence-based terminological decision trees. In: Laurent, A., Strauss, O., Bouchon-Meunier, B., Yager, R.R. (eds.) IPMU 2014, Part I. CCIS, vol. 442, pp. 36–45. Springer, Heidelberg (2014)
36. Rizzo, G., dAmato, C., Fanizzi, N., Esposito, F.: Inductive classification through evidence-based models and their ensembles. In: Gandon, F., Sabou, M., Sack, H., dAmato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 418–433. Springer, Heidelberg (2015)
37. Sazonau, V., Sattler, U., Brown, G.: Predicting performance of OWL reasoners: locally or globally? In: KR. Citeseer (2014)
38. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. **74**(2), 249–310 (1995)
39. Settles, B.: Active learning literature survey. University of Wisconsin, Madison, vol. 52(55–66), p. 11 (2010)
40. 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)
41. Silla Jr., C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Min. Knowl. Disc. **22**(1–2), 31–72 (2011)
42. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-dl reasoner. Web Seman. Sci. Serv. Agents World Wide Web **5**(2), 51–53 (2007)
43. Steigmiller, A., Liebig, T., Glimm, B.: Konclude: system description. Web Seman. Sci. Serv. Agents World Wide Web **27**, 78–85 (2014)
44. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: 16th International Conference on World Wide Web, pp. 697–706 (2007)
45. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a large ontology from wikipedia and wordnet. Web Seman. Sci. Serv. Agents World Wide Web **6**(3), 203–217 (2008)
46. Suda, M., Weidenbach, C., Wischnewski, P.: On the saturation of YAGO. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 441–456. Springer, Heidelberg (2010)

47. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
48. Völker, J., Niepert, M.: Statistical schema induction. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 124–138. Springer, Heidelberg (2011)
49. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., dAmato, C., Gandon, F., dAquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 504–518. Springer, Heidelberg (2014)