Praedixi, Redegi, Cogitavi: Adaptive Knowledge for Resource-aware Semantic Reasoning

Carlos Bobed[a,b], Fernando Bobillo[a,b], Ernesto Jiménez-Ruiz[c], Eduardo Mena[a,b], Jeff Z. Pan[d]

[a] *University of Zaragoza, Zaragoza, Spain*
[b] *Aragon Institute of Engineering Research (I3A), Zaragoza, Spain*
[c] *City, University of London, London, UK*
[d] *University of Edinburgh, Edinburgh, UK*

Abstract

Representing knowledge with ontologies and performing reasoning with semantic reasoners is important in many intelligent applications. However, existing reasoners do not take into account the available resources of the device where they run, which can be important in many scenarios such as reasoning with very large ontologies or reasoning on resource-constrained mobile devices.

In this paper, we propose a novel approach to adapt the size of knowledge managed by applications, taking into account several criteria about resources available (such as time, memory, and battery consumption), at the same time. Thus, rather than giving no answer due to the lack of resources needed to deal with a full ontology, we propose a novel architecture to compute a subontology to provide an incomplete answer at least. Our approach makes use of existing approaches to predict the performance of semantic reasoners and to compute ontology modularisation and ontology partition, but taking into account the associated resource consumption. We also propose a novel measure to estimate the semantic loss when replacing the original ontology by a subontology. Finally, we present an implementation and evaluation of the whole pipeline, showing

Email addresses: `cbobed@unizar.es` (Carlos Bobed), `fbobillo@unizar.es` (Fernando Bobillo), `ernesto.Jim\unhbox \voidb@x \bgroup \let \unhbox \voidb@x \setbox \@tempboxa \hbox {e\global \mathchardef \accent@spacefactor \spacefactor }\let \begingroup \let \typeout \protect \begingroup \def \MessageBreak { (Font) }\let \protect \immediate \write \m@ne {LaTeX Font Info: on input line 51.}\endgroup \endgroup \relax \let \ignorespaces \relax \accent 19 e\egroup \spacefactor \accent@spacefactor nez-ruiz@city.ac.uk` (Ernesto Jiménez-Ruiz), `emena@unizar.es` ( Eduardo Mena), `http://knowledge-representation.org/j.z.pan/` (Jeff Z. Pan)

that the semantic loss incurred in the process is acceptable.

Keywords: Ontology reasoning, Semantic reasoning, Resource-constrained devices, Ontology modularisation

1. Introduction

One of the key reasons behind the success of intelligent applications is the exploitation of knowledge. From old knowledge-based systems or expert systems, it has been clear that knowledge representation is a crucial task. Modern intelligent applications usually represent the relevant explicit knowledge using ontologies (see for example (Allemang et al., 2020)), which are formal and shared specifications of the vocabulary of a domain of interest. Ontologies are usually formalised using Description Logics (DLs) (Baader et al., 2003), which have many advantages, including the possibility of using *semantic reasoners* (Khamparia and Pandey, 2017) to answer questions regarding an ontology, compute new knowledge which is not implicitly represented in the ontology, etc. Ontology reasoners solve one or several reasoning tasks. Popular reasoning tasks include *consistency checking* (verifying that an ontology does not have logical contradictions), *instance retrieval* (computing all the instances of a given concept), *classification* (computing a concept hierarchy and a property hierarchy based on the subsumption relationships), and *ontology materialisation* (precomputing some inferences, such as indirect subclass axioms).

Unfortunately, the algorithms currently implemented by the semantic reasoners do not take into account the resources available for the running environment. Let us discuss some examples where resources are important:

- Reasoners do not take into account the memory of computers where they are executed: if the input ontology is very large, the reasoner might abort its execution returning an "out of memory error" and therefore the user would not receive any answer. For example, let us consider GALEN medical ontology, which has been used as a reference terminology for surgical procedures in France, for oral hygienists and dietitians in the Netherlands, and for drugs in the UK (Rogers et al., 2001). When trying to reason with the 20.1 MB GALEN on a desktop computer with an Intel Core i7-6700K@4.00 GHz CPU and 24GB of RAM, HermiT (Glimm et al., 2014) reasoner did not finish in 400 s and, after extending the timeout to 3600 s, it run out of memory. Indeed, in order to support GALEN, people had to manually identify, extract, and use fragments of the ontology. Over the years, different authors identified different fragments of the ontology, with different expressivities and sizes, suitable for their particular hardware resources and information needs. Ideally, such a process should be automatised and generalised so that it can adapt to the user device resources.

MMD ontology, used at Aibel company (Norway), is another example where modularisation is critical to perform efficient reasoning (Skjæveland et al., 2012). In fact, requirements and specifications are represented using generic ontologies, ontologies describing generic concepts in the engineering domain, and domain ontologies. However, existing modules cannot be dynamically adapted to the user device resources.

- In ontology visualisation (which requires using a reasoner to deal with the implicit knowledge) (Dudáš et al., 2018), the ontology could be too large to be displayed completely, so it is a good idea to take hardware resources and user preferences into account. On the one hand, ontology visualisation tools should adapt to the running environment, taking into account, for example, the size of the screen. On the other hand, even if the ontology has a large number of nodes, it is not necessary to display all of them (no user will be able to see the details of thousands of nodes at the same time) but rather limit itself to showing a subset of nodes that are interesting for the user or the most relevant concepts to get a global view.

- Mobile devices typically have limited resources (at least when compared to desktop/server counterparts) in terms of CPU processing, available memory, remaining battery charge, and connectivity (wireless communications are, in general, less reliable than wired ones). This also applies to the Internet of Things (IoT), where the devices deployed to perform so called *edge-computing* would be also limited by their hardware resources. For example, let us consider an emergency-assistance app for mobile devices, such as the one implemented in the SHERLOCK system (Yus and Mena, 2015). In order to assist the health staff to find a proper treatment for a particular patient, it should be able to avoid medication errors by automatically checking that there is no incompatibility between the medicines and the allergies of the patient (according to the knowledge in a certain ontology). Being able to reason locally on such devices is important when connectivity is not guaranteed. This is the case not only for smartphones, but also tablets, laptops, wearable devices, etc. However, due to its limited resources, significantly fewer reasoning tasks are completed on a mobile device than on a desktop computer, as illustrated in the example in Figure 1, adapted from (Bobed et al., 2015), where in 61 out of the 572 evaluated tasks, the reasoning did not end successfully on the mobile device. Another example is the beer recommender system GimmeHop, that required to compute manually a fragment of the ontology to reason on a local mobile device (Huitzil et al., 2020).

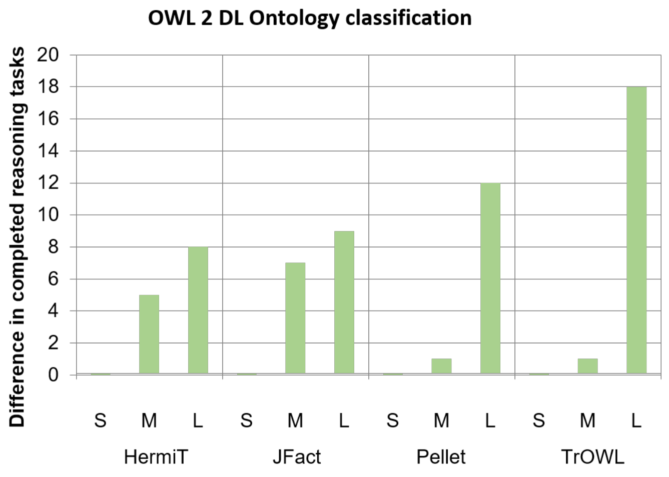


Figure 1: Difference in the number of tasks that finished on a desktop computer but did not finish on an Android device, for different reasoners and ontology sizes: S(*mall*, <500 axiomas), M(*edium*, [500,5000) axiomas) and L(*arge*, ≥5000 axioms).

We claim that an intelligent application cannot really be called "intelligent" if it does not work when available resources are not optimal. For example, coming back to our emergency-assistance app, it should not assume that a stable wireless connection will always exist in a remote mountainous area or inside a tunnel: in such critical cases, the application should implement strategies to guarantee that there will be an acceptable behaviour, such as working with the local knowledge, finding alternative ways to communicate (e.g., an ad-hoc network), etc.

Current ontology tools, such as semantic reasoners (e.g., Pellet (Sirin et al., 2007), HermiT (Glimm et al., 2014), TwOWL (Thomas et al., 2010), or MORe (Armas-Romero et al., 2012)) or ontology editors (e.g., Protégé (Musen, 2015)) start to solve a reasoning task without taking into account whether they will finish such a task or not. If reasoning cannot be finished under the existing resources, current tools just abort their execution and throw an exception (typically, "out of memory" error).

A first solution would be to adapt existing reasoning algorithms to different resource limitations, but this is complex as resource management is heterogeneous and can be contradictory; e.g., to minimise reasoning time, semantic reasoners typically use auxiliary data structures, which increases the memory use.

In this paper, instead, we propose a novel approach to promote *resource-aware semantic reasoning*, deciding at run-time how much knowledge can be processed by the device. As a consequence, it will improve the semantic capabilities of resource-limited devices. In particular, we will mainly focus on adapting the available knowledge to a size that a device is able to handle, to avoid aborting reasoning tasks due to a lack of resources. Ideally, one would like to take into account all the available knowledge, but limited capabilities might force us to restrict to a (as large as possible) subset. Furthermore, our approach will make it possible for applications to detail the percentage of knowledge that is being handled and to explain to the user that the answer of the reasoning is incomplete and its estimated loss of information. To do so, we propose (i) to first estimate the needed resources for a given reasoning task on a given ontology, building on previous approaches to predict the resource consumption of DL reasoners (Guclu et al., 2016b; Kang et al., 2012; Pan et al., 2018), (ii) if needed, considering to the specified limitations for the device, to compute a subset of knowledge to work with, and (iii) to run a semantic reasoner to solve the given reasoning task on the (computed sub)set of knowledge[1].

This context leads to two important research questions:

1. On the one hand, which size of knowledge are we able to manage in a given device? As the resources of the devices are fixed, we turned our attention to the knowledge to be processed, which can be selected somehow.
2. On the other hand, even if we had a general framework to predict the cost in terms of different criteria, how do we measure the consequences of dealing with a subset of available knowledge only? In particular, given an ontology and an extracted subontology, is it possible to measure, for a given reasoning task, the semantic loss we will incur when working only with the latter one? As we will discuss in Section 2.3, existing approaches are not directly applicable.

The main contributions of this paper are the following ones:

- We firstly propose a general architecture based on a combination of ontology modularisation strategies (Del Vescovo et al., 2013) and prediction of the cost of a given reasoning task.
- Then, for the sake of concrete illustration, we implement a prototype which focus on a single resource: it adapts the reasoning to the specified maximum running time, using a particular choice of prediction and modularisation techniques.
- We propose and evaluate a measure to estimate the semantic loss incurred, given that our proposal advocate working with subontologies instead of full (but unmanageable) ontologies.
- We empirically demonstrate the feasibility of our approach by studying the accuracy of the prediction when computing the adapted subontologies and the estimated semantic loss.

The rest of the paper is organised as follows. First, in Section 2 we present an overview of the related work. Section 3 describes the proposed architecture, and its

[1]In Latin, "Praedixi, Redegi, Cogitavi" means "I predicted, I reduced, I reasoned."

possible use in a mobile computing scenario. Section 4 presents some possible knowledge extraction strategies, implemented in a prototype. Then, Section 5 presents our proposal to measure the semantic loss of a module. In Section 6, we perform an experimental evaluation of our prototype. Finally, we draw some conclusions and present the future work in Section 7.

2. Related Work

In this section we will overview some related work on computing ontology subsets (Section 2.1), adapting ontology reasoning to the resources (Section 2.2), and measuring semantic loss (Section 2.3). Finally, we identify some open issues (Section 2.4).

2.1. Computing ontology subsets

Several years ago Stuckenschmidt and Klein asserted that the realisation of the Semantic Web depended on the ability to reuse ontologies (Stuckenschmidt and Klein, 2004). They proposed that problems arising from the monolithic nature and size of ontologies could be solved by an automatic *partitioning mechanism*. Ontology partitioning consists of dividing an ontology into several subontologies such that every axiom of the original ontology belongs to exactly one subontology. However, automatic partitioning strategies so far do not take into account either the target environment where the extracted knowledge is going to be deployed or the actual tasks such knowledge is going to be used for. For example, when dealing with SNOMED CT ontology (BioPortal , 2023), automatic partitioning strategies, such as SWOOP (Cuenca Grau et al., 2006), or PATO (Schlicht and Stuckenschmidt, 2006), either produced one very big subontology (due to the inner links between concepts), or one subontology which takes into account parameters that were hardly related to the available resources of our devices, such as *number of partitions*. Other approaches, such as Prompt (Noy and Musen, 2004), rely on *manual partitioning* strategies which require some parameters (again not related to resource consumption) set by the user/developer such as *depth of traversal, relationships to be included, and a starter concept*.

Apart from ontology partitioning, ontology *modularisation* consists of computing a subset of an original ontology such that the inferences related to some terms (a seed signature) are preserved. Some approaches have been proposed for which the ontology conforms an *conservative extension* for a given signature, such as (Armas-Romero et al., 2016; Cuenca Grau et al., 2008; Gatens et al., 2013; Konev et al., 2013). For example, *Syntactic Locality-based Module Extraction* (Cuenca Grau et al., 2008; Jiménez-Ruiz et al., 2008) proposes six different types of modules for an ontology with different assumptions and expectations from a module. A comparison of three logically sound notions of a module (i.e., MEX modules, semantic locality, and syntactic locality) concluded that syntactic locality, which is computationally cheaper to find, can be a good approximation of semantic locality, and that "*in general* there appears to be no or little difference between semantic and syntactic locality" (Del Vescovo et al., 2013). AMEX modules have been generated by implementing a depletion approach on acyclic ontologies using provided signatures (Gatens et al., 2013). However, despite the benefits of their logical grounding, these approaches focus only on the logical properties that the logical modules must satisfy, but they do not consider the resource consumption.

Using partitioning or modularisation techniques to obtain better performance results can be seen in MORe (Armas-Romero et al., 2012), a meta-reasoner which tries to divide the ontology into different modules according to their expressivity and forwards them to the appropriate underlying reasoner. For example, MORe would split a given ontology and use two reasoners, one efficient $\mathcal{EL}$ reasoner (i.e., ELK (Kazakov et al., 2014)) for processing axioms that have the expressivity of the OWL 2 EL profile in the fastest way possible, and a DL reasoner (e.g., HermiT (Glimm et al., 2014)) for processing axioms with higher expressivity. This approach, while similar to the nature of our proposal, does not take into account the available resources.

2.2. *Adapting to resources*

The mobile reasoner mTableaux implements a weighted partial matching approach that This approach would terminate the process when it reaches a given RAM threshold, and would provide the results according to matching conditions provided by the user (Steller et al., 2009). However, why should we deplete all the device resources if we could just work with a specifically task-tailored piece of knowledge? This is specially more important in these scarce-resources scenarios, where we have to optimise everything as much as possible. Besides, this approach expects the user/developer to provide *successful matching conditions* and weights for matching conditions to give priority in their depth-first checking algorithm.

In order to guide the extraction taking into account the resource consumption, we also need to be able to predict the resource consumption of reasoners. In this regard, there have been several different proposals mainly oriented to predict execution times (Guclu et al., 2016a; Kang et al., 2012, 2014; Pan et al., 2018; Sazonau et al., 2014; Zhang et al., 2010) in desktop computers. Other works proposed different set of ontology metrics mainly at TBox reasoning to predict the reasoning time using classification and regression models (Zhang et al., 2010; Kang et al., 2012, 2014). (Pan et al., 2018) explored their capabilities when trying to predict reasoning times with big ABoxes, proposing an extension of the metrics which allowed to improve the time processing prediction. Instead of building global models such as these methods do, (Sazonau et al., 2014) proposed a local prediction method that involves selecting a *suitable* small subset of the ontology and use extrapolation to predict total time consumption of ontology reasoning using the data generated by the processing of such a small subset. We think that using both of them together will be a starting point to get enough information to guide the resource aware knowledge adaptation our proposal requires.

Beyond the semantic reasoning application, (Hutter et al., 2014) showed that it is possible to predict the performance of algorithms for hard problems, in particular propositional satisfiability (SAT), travelling sales person (TSP), and mixed integer programming (MIP) problems. Prediction approaches based on random forests showed the best performance. Identifying metrics for the prediction is problem-dependent: the authors identified 138, 121, and 64 features for SAT, TSP, and MIP, respectively. The prediction of the performance of other discrete problems (such as the travelling thief problem, the quadratic assignment problem, or solving quantified Boolean formulae) but also continuous problems (in particular, both unconstrained and constrained single-objective optimisation problems) has also been investigated (Kerschke et al., 2019).

Regarding reasoning specifically on mobile devices, reasoners can be native (Ruta et al., 2022, 2019; Steller et al., 2009; Van Woensel and Abidi, 2019) (implemented for a specific mobile device or an edge-computing device) or ported (to reuse existing semantic reasoners (Bobed et al., 2015)). (Bobed et al., 2015) adapted some existing reasoners to Android and presented a thorough evaluation of the time performance of DL reasoners, showing that it is feasible to use them up to "medium-size" ontologies. While there have been some approaches that have adapted semantic reasoners to constrained-resource devices (Ruta et al., 2022, 2019; Steller et al., 2009; Van Woensel and Abidi, 2019); none of them is able to adapt the reasoning to the resources of a particular device. In particular, (Van Woensel and Abidi, 2019) proposes to optimise mobile reasoning by de-activating inference rules (for the OWL 2 RL profile) to decrease the running time. However, specific resources are not taken into account, so it could de-active more rules than needed, and even after de-activating some rules, the resource limit could still be reached. Furthermore, (Kleemann, 2006) discussed resource restrictions (computing power, memory, and energy) from the point of view of reasoner development, but the implementation cannot adapt to the resources. Finally, it is worth to mention the only previous work predicting the resource consumption on mobile devices, which presented a battery consumption prediction module for Android devices (Guclu et al., 2016b).

2.3. *Semantic loss*

While there are some works that compute the semantic loss when a query expression is rewritten, such as the OBSERVER system (Mena et al., 2001), computing the semantic loss when replacing an ontology with another one is not always considered. The main reason is that, as already mentioned, ontology modularisation computes a subset of an original ontology such that the inferences related to the seed signature are preserved, so there is no semantic loss (with respect to the seed signature).

(Gobin-Rahimbux, 2022) overviews the metrics to evaluate ontology modules and classifies them into eight categories: naming convention, syntax, module characterisation and distribution, module structure, module richness, logical criteria, module relatedness, and module quality. Module quality includes precision and recall, but they are restricted to taxonomical relations (disregarding other axioms) and do not consider the consequences of the axioms.

In the context of *ontology alignment*, (Euzenat, 2007) proposed the definitions of semantic precision and semantic recall. The idea is that the true and false positives do not only consider the explicit alignments but also their consequences (i.e., implicit alignments). In our scenario, semantic precision would be 1 by definition, as all the axioms in the subontology are part of the original ontology. However, we will consider a similar notion of semantic recall.

While we are interested in computing the semantic loss between an original ontology and a subontology, the notion of *semantic difference* is more general, as it is asymmetric and is thus concerned with both axioms that belong to O_1 and not to O_2, and axioms that belong to O_2 and not to O_1. Measures to compute the semantic difference are also typically based on the percentage of consequences which change with respect to another ontology (Pernisch et al., 2021).

2.4. *Open issues*

Thus, after analysing the available approaches, we have identified two main points that hinder the adoption of semantic reasoning where resources are constrained:

1. *Lack of interaction and collaboration* between reasoning and knowledge extraction for a better optimisation of knowledge processing on resource constrained scenarios. The MORe approach indicates that it is promising to explore such a possible interaction for other purposes such as mobile devices.
2. *Lack of generalisation*, because previous works on the subject are *embedded* in various research prototype implementations and cannot be easily generalised. For example, the weighted matching algorithm (Steller et al., 2009) is embedded in mTableaux DL reasoner and it would essentially require a complete reimplementation if it is expected to be used in an $\mathcal{EL}$ reasoner.

We aim at proposing a *flexible architecture* that can adopt any particular reasoner and knowledge extraction technique. Furthermore, previous approaches to measure the semantic loss must be adapted to our framework.

3. Adaptive Reasoning

In this section, we firstly propose an architecture that evaluates whether a particular device can process an ontology under some resource constraints, and, if it is not the case, adapts the reasoning task to support as much knowledge as possible. Then, we discuss strategies for the prediction of the resources consumption and a possible integration in a more general case, an intelligent framework selecting among local reasoning, global reasoning, or a hybrid approach following (Bobed et al., 2017).

3.1. Architecture of the Proposal

Our resource-aware system is flexible and can receive a wide plethora of constraints. For the sake of concrete illustrations, we will provide a non-exhaustive list of examples:

- Hardware constraints such as CPU capacity, memory, battery consumption, connectivity and bandwidth, or screen size. In some cases, such as memory and battery, this includes not only the maximum value but also the available one.
- User preferences, such as a maximum running time.
- Non-functional properties, such as a required user privacy level.
- Feedback from the system, such as a prediction of the cost to reason with the current subontology or the semantic loss with respect to the original one.
- The input ontology (size, expressivity . . .) and the particular reasoning task, as they affect the cost of the reasoning.

Figure 2 shows the main high-level components and steps of our architecture, which we are going to discuss in detail:

1. First, the Constraint Evaluation Module receives the ontology and the constraints, and evaluates the constraints according to the features of the ontology and the resources of the device. If the ontology fulfils them, it is forwarded to the reasoner (step 4); otherwise, the ontology as well as the constraints are passed to the Knowledge Extraction Module.

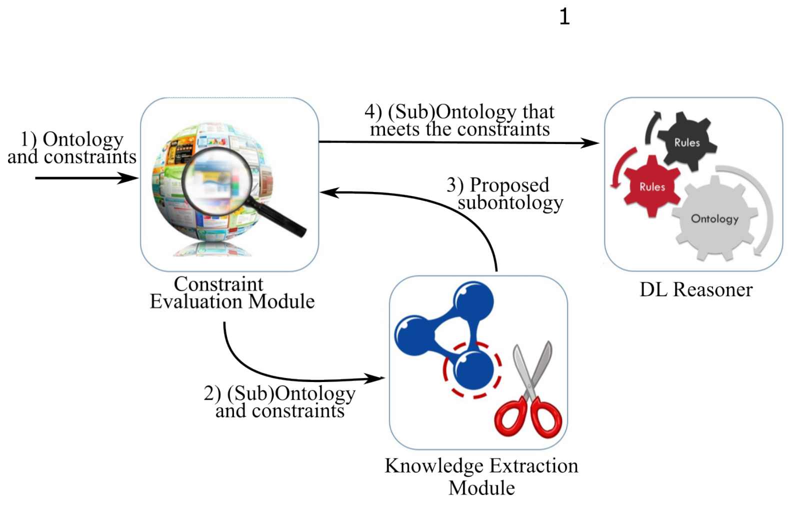

1
1) Ontology and constraints
4) (Sub)Ontology that meets the constraints
3) Proposed subontology
Rules
Rules
Ontology
Constraint Evaluation Module
DL Reasoner
2) (Sub)Ontology and constraints
Knowledge Extraction Module

Figure 2: High-level architecture for the knowledge adaptor module.

2. The Knowledge Extraction Module tailors the knowledge according to the given constraints in an iterative way. In particular, it computes a subontology of the input ontology, taking into account the constraints, trying to keep the most important knowledge to solve the reasoning task, and trying to minimise the semantic loss. This flexible architecture will be instantiated on Section 4.
3. Once extracted, the subontology is returned for its evaluation to the Constraint Evaluation Module. Therefore, we repeat the loop *predict* cost of the reasoning - *evaluate* constraints - *extract* subontology - *predict* again as many times as needed, until obtaining a subontology which has been estimated to be manageable by the semantic reasoner[2].
4. Finally, once the knowledge has been estimated to be manageable by the semantic reasoner, the execution of the reasoning task is performed over the original ontology or a subontology that meets all the constraints.

The proposed architecture assumes that reasoning is performed locally on the user device. However, the resource-aware reasoning module could be integrated in a broader

[2]If the subontology is finally empty (in fact, it would include at least the signature), it means that there was no way to fulfil all the given constraints.

framework, where reasoning does not need to be performed locally. (Bobed et al., 2017) proposed three main strategies to be adopted in a mobile scenario, but they can actually be considered in any scenario with limited resources. Briefly, those strategies are:

- Server-side External Reasoner: External powerful servers perform all the calculations. However, the application might need to send a lot of data through the network, which can be a serious limitation in mobile computing environments, and user privacy can be compromised as the user's data would be sent to an external server.
- Device-side Local Reasoner: Calculations are computed on a local device, which is the best choice to keep privacy or when network communications are not reliable. As limitations, reasoning becomes challenging in devices with limited resources. For instance, there is some evidence that reasoning in mobile devices might be affordable only for small or not very expressive ontologies when using ported reasoners (Bobed et al., 2015).
- Hybrid-reasoning Approach: In this last strategy, some parts can be computed locally and others can be computed on an external server, depending on the resource available in runtime.

The objective would be to automatically determine which one is the best option to use a semantic reasoner. The optimal (or maybe Pareto-optimal) choice depends on the application or, more specifically, on several factors, and is left as future work.

In the following, we detail an important part of our architecture: how resource consumption can be predicted.

3.2. Predicting Resources Consumption

To perform the resource consumption prediction, we could use a model to predict each single resource. One problem of this approach is that we do not have any feedback about which features might be interesting to modify in order to make the ontology fit in the device. This might lead to a blind search, having to apply different extraction heuristics in a blind way. However, the works (Ribeiro et al., 2016, 2018) suggest that we could obtain some explanations in the feature space which could be used to guide the extraction as well. Moreover, some works such as (Sazonau et al., 2014) suggest that some features are strongly correlated and could at least give us some clues about how to proceed (in that example, the number of axioms was one of the most relevant features to predict OWL 2 DL reasoning time)[3].

On the other hand, to be able to act on the source ontology and know which parameters to focus on in order to meet the criteria, we considered having models that, given a (predicted) resource limit, return which feature/s should be modified and how to meet the criteria. We aimed at exploiting the strong correlations that exists between the number of axioms and the predicted values (Sazonau et al., 2014); however, the initial tests about the accuracy were not good, so further analysis using explainability techniques in the feature space must be carried out before considering this option. Thus, as a strategy to reduce the resource consumption, we have adopted an axiom size reduction heuristic, leaving the exploration of these techniques as future work.

Besides, given the heterogeneity of the devices, we might need to train a model for each family of devices: the ontology features do only depend on the input ontology, but the actual resource consumption is dependent on the actual characteristics of the device, thus requiring a model for each group of them (trained once, and shared among all of them). In order to simplify the extraction of training data for new devices, it could be interesting to use Antutu (in mobile devices) or PassMark (in desktop computers) values to scale the values obtained for other devices. Note as well that the prediction depends on the concrete reasoning task, but this will be explained in detail in Section 4.

We are aware that the resource-constrained device might have to reason with incomplete information (depending on the knowledge extraction technique used), but it is important to stress that, since semantic reasoners implement monotonic reasoning, the results would be correct always (although may be incomplete). This approximate

[3]In order to show the feasibility of our approach, in the prototype described in Section 4, we adopted time as our main limiting resource and use the already studied ontology features (Kang et al., 2012; Pan et al., 2018) to predict the reasoning time.

reasoning is not the optimal option, but, when resources are not enough, it is preferable than providing no result at all. Furthermore, one of the novelties of our current proposal is how to measure the semantic loss incurred.

4. Instantiating the Architecture

In this section, we firstly analyse some strategies to instantiate the architecture proposed in the previous section (Section 4.1), and then report our prototype implementation for a particular choice of those strategies (Section 4.2).

4.1. Design of the Solution

First of all, we have to note that it is not possible to apply directly "classical" modularisation or partitioning methods, as they do not take the resources into account.

Moreover, in the previous section we have omitted the requested reasoning task (i.e., what is the knowledge being used for) on purpose. Indeed, the knowledge adaptation module must take into account the nature of this target task. In fact, as we will see in the following, the nature of the target task will give us a starting point to guide the extraction.

Figure 3 shows an instantiation of our proposal, highlighting the different works that could be applied to each stage. It takes as input an ontology, a list of the resource constraints, and the reasoning task that has be performed. Note that the approach might take several iterations: if after computing a subset of the original ontology, the subset is still too big for the device resources, another round of knowledge extraction is performed. The main components are as follows:

1. **Resource Predictor Model/s**: As discussed in Section 2, several metrics to predict the reasoning time (on desktop computers) and the battery consumption (on mobile devices) have been proposed in the literature. These metrics can be considered in our approach as a starting point, building a model for each of the resources that whose consumption we want to predict.
2. **Constraint Evaluation Module**: The predictions obtained by the models for each resource would be taken into account in a multicriteria decision problem

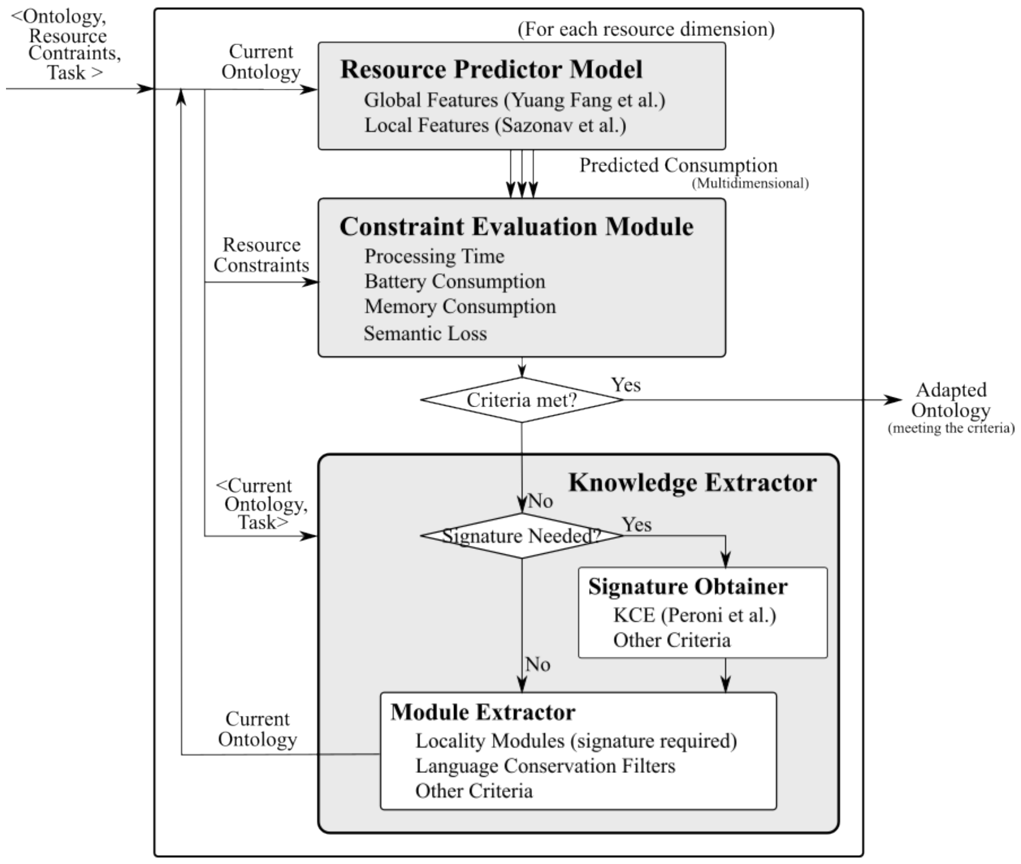


Figure 3: Possible implementation of the knowledge adaptor module.

deciding whether reasoning is feasible or not. In the latter case, it would be necessary to further reduce the amount of knowledge until it is feasible. In principle, the system could interpret all constraints in a conjunctive way and ensure that all of them are satisfied. However, more sophisticated solutions are possible: the user could define the most important constraints and the system could go on if some of the less important constraints are satisfied, there could be an interval of values (for example, for the maximum running time) rather than a strict value, etc.

3. **Knowledge Extractor**: Its purpose is to compute a subontology of the input ontology according to the resource constraints and reasoning task. Firstly, it

might invoke the Signature Obtainer, described below, to get a seed signature. Secondly, it invokes the Module Extractor and returns the computed subontology. At this point, the output ontology could be empty if all the constraints cannot be met given the available resources.

4. **Signature Obtainer**: The reasoning task to solve might require a *signature seed* (i.e., a set of atomic concepts, individuals, object properties and datatype properties to start with) or not. An example with signature is an instance retrieval query, where the user wants to retrieve books that have been written in English. The signature seed would be $\{\mathsf{Book}, \mathsf{writtenIn}, \mathsf{english}\}$. An example without signature is ontology graphical representation, where the user just wants to browse the available knowledge.

   To compute the subontology, there are several possible choices about which knowledge to keep and which to discard. If the task imposes a signature seed, then it should guide the extraction procedure. However, when the target task does not impose a signature or we cannot derive an useful signature from the context of the application, we have two options: trying to find a signature seed, or not using a signature seed. In this latter case, we can use different strategies such as the following ones (the list is by no means complete):

   - To restrict to some OWL 2 profile discarding the other axioms, or a less expressive DL language.
   - To approximate the ontology to some OWL 2 profile, e.g., as in the syntactic-based approximation of TrOWL.
   - To keep just the subclass taxonomy of the ontology.

   More generally, we could define our *preservation language*, defining the axioms and complex concepts to be kept. This is similar to the proposal of (Cuenca Grau et al., 2012). Although in their approach there is no control over the particular values of the ontological features we would like to change, we could act further in the extraction by limiting the applicability of different grammar rules (e.g., limiting the depth of the parsing trees).

If we consider it relevant, we can obtain a seed signature attending to general criteria (Signature Obtainer in Figure 3). For this purpose, we could apply, for example, the detection of Key Concepts proposed by (Peroni et al., 2008). These concepts would comprise the signature seed in the absence of such for the task. In fact, they could be added as metadata (OWL 2 annotation) in the ontology to compute them offline and just once, as it is an expensive task.

5. **Module Extractor**: Apart from restricting the expressivity, if the target task provides somehow a signature to exploit (or we have been able to define a relevant one), we can follow several strategies (again, we do not intend to be exhaustive):

   - Incremental approaches: given an initial signature seed, add more axioms corresponding to the seed signature. After that first step, one could also add axioms corresponding to the new signature introduced by the previous axioms.

   - Pruning approaches: given an initial ontology, remove axioms. Following the spirit of the extraction with no signature, a possible approach could be to limit the "locality" of the axioms up to a certain depth. This would imply to apply the syntactic rules for the acceptance of the different axioms in (Cuenca Grau et al., 2008) parameterised to limit the activations/depth to which a particular rule can be applied.

   - Hybrid approaches or variants of the above approaches.

In the case of multiple criteria, it is necessary to decide an order for the criteria to be relaxed (for example, we could go first for the most restrictive constraint, but other strategies could be studied in the future).

4.2. A Prototype Implementation

As a first step towards our objectives, we have implemented a prototype to show the feasibility of the whole pipeline, with a specific choice of the multiple alternatives discussed in the previous section. Our prototype takes an input ontology and computes a subontology, considering a maximum reasoning time as the only limited resource, on a desktop computer. As we have seen in Section 2.2, there are only prediction models for the reasoning time on desktop computers and for battery consumption on mobile devices. We advocate building our prototype for desktop computers because reasoning on mobile devices is limited to small and medium ontologies (Bobed et al., 2015), so it is not possible to process a meaningful benchmark. The only supported reasoning task is ontology materialisation using the OWLAPI (method `precomputeInferences`).

The prototype uses OWLAPI 3.5.7[4](Horridge and Bechhofer, 2011) as programming API and is able to use any semantic reasoner implementing the `OWLReasoner` interface. In particular, we used HermiT 1.3.8 (Glimm et al., 2014) to gather the data for training the models and to perform all the materialisations required. The reason is that HermiT has already been successfully used to predict the reasoning time (Pan et al., 2018).

Definition 1. *Let O be an ontology. The materialisation of O, denoted mat(O), is the set of consequences that can be derived from O by using the types of InferredAxiomGenerator supported in OWLAPI 3.5.7*[5]*, namely*

- *InferredClassAssertionAxiomGenerator,*
- *InferredDataPropertyCharacteristicAxiomGenerator,*
- *InferredDisjointClassesAxiomGenerator,*
- *InferredEquivalentClassAxiomGenerator,*
- *InferredEquivalentDataPropertiesAxiomGenerator,*
- *InferredEquivalentObjectPropertyAxiomGenerator,*
- *InferredInverseObjectPropertiesAxiomGenerator,*
- *InferredObjectPropertyCharacteristicAxiomGenerator,*

[4]We kept this version as the original prediction features(Kang et al., 2012; Pan et al., 2018) were developed with such one.

[5]https://javadoc.io/doc/net.sourceforge.owlapi/owlapi-distribution/3.5.7/index.html

- *InferredPropertyAssertionGenerator,*
- *InferredSubClassAxiomGenerator,*
- *InferredSubDataPropertyAxiomGenerator, and*
- *InferredSubObjectPropertyAxiomGenerator.*

In the following, we will discuss each step of the prototype in detail.

Resource Predictor Model. In order to train the model, we used the whole set of metrics used in (Pan et al., 2018) to predict the reasoning time, which included the metrics proposed by (Kang et al., 2012). The models have been trained using *scikit-learn 1.2*, which allows to export the models to PMML (Predictive Model Markup Language) (Guazzelli et al., 2009) to import them afterwards in other applications.

The values of those metrics only depend on the input ontology and not on the device. Therefore, in order to train several devices, such values can be computed just once, and possibly on a faster external server. However, recall that each family of devices will require to have their own prediction model. Besides, when actually using the system, it is important that those metrics can be efficiently computed on a limited-resource device or, alternatively, we can assume that the input ontology has been annotated with them (as this is not standard) or the existence of an external service to compute them (as we will need to assume that there is some connectivity).

Constraint Evaluation Module. As above mentioned, our current implementation is focused on time as resource. This module uses the model trained to predict the time and decide, given a timeout, between whether the criteria is met or not, and if not, whether there are more actions (i.e., reduce the size of the ontology) that we can apply to try to meet the requirements.

As reduction criteria, we currently reduce the number of axioms by a 25%[6] at each prediction step iteratively until the criteria is predicted to be met (i.e,. we repeat the

[6]As we will see in Example 2, we also tried a 12.5% reduction with `GALEN-Full-Union_ALCHOI`(D) ontology and obtained the same results of the reasoning, but higher materialisation times.

loop *predict* - *evaluate* constraints - *extract* knowledge - *predict* again as many times as needed).

Signature Obtainer. We have integrated Key Concept Extraction (KCE) technique (Peroni et al., 2008) to find the top most important *k* concepts. To compute them, we adapted the implementation provided by (Peroni et al., 2008): when KCE API retrieved more than the demanded *k* concepts, we just selected the first *k* ones, as they are equally important (according to our own source code inspection). The value *k* could depend on the available resources.

We noticed that KCE implementation is not efficient enough, and it usually does not finish on an average mobile device with medium-size ontologies. Therefore, we assume that KCEs have been computed offline (for example, on a desktop computer or a remote server) and that they are represented as metadata attached to the ontology via an OWL 2 annotation property keyConcepts.

Module Extractor. The extraction technique we have implemented to compute the sub-ontology while having control over its size proceeds as follows:

1. Starting from the given signature, it extracts all the subclass hierarchical information of its entities, obtaining a minimal skeleton[7]. That, it assumes a preservation language built using axioms of type *A SubclassOf B*, with A and B being atomic concepts. Then, it updates the signature with all the terms included in the skeleton.
2. It iteratively adds logical axioms related to the updated signature, expanding the covered part of the original ontology in a breadth-first way, until reaching the maximum number of axioms computed by the Constraint evaluation module. Those logical axioms can be OWL 2 axioms of any type. In each step, the signature is also updated and expanded with the newly added terms.

While pretty straight-forward, this approach allows us to reduce the size of the ontology while keeping the information that we have deemed important. In fact, keeping

[7]This skeleton can also be built in an incremental way to avoid running out of resources

the hierarchical skeleton could be substituted by any conservation language.

Note that, as we reduce knowledge in an iterative way, we cannot recover knowledge discarded in previous iterations. Therefore, knowledge extraction should not be too aggressive in discarding axioms, as a more conservative extraction could be refined in next iterations. An alternative strategy could be, for example, if the computed subontology is not satisfactory, to find a subset of the signature which tries to maximise the amount of knowledge that the module extracts and which is still within the limits. After that, we could add progressively the information available in the module obtained: 1) adding more axioms corresponding to the seed signature, or 2) adding axioms not corresponding to the seed signature. However, we consider the exploration of the different strategies as future work.

5. Measuring the Semantic Loss

Apart from leading to a lower resource consumption (due to the simplification and reduction of the size of the processed ontologies), these potential approximations will have an associated *semantic loss* which has to be measured (and possibly be predicted) to be part of the decision process and to provide users with a confidence degree on the system answer.

We advocate to adapt the notion of *semantic recall* proposed by Euzenat (Euzenat, 2007) in the context of *ontology alignment*.

Definition 2. *Given an ontology O, let $Cons(O)$ be the set of consequences that follow from O.*

The *syntactic recall* is defined as the proportion of axioms that had been extracted from the original ontology.

Definition 3. *Let two ontologies O and O', where O' is the result of any extraction process over O (i.e., $O' \subseteq O$ syntactically). The syntactic recall of O' w.r.t. O is given by:*

$$SynRecall(O', O) = \frac{|O'|}{|O|}$$

While syntactic recall compares original axioms, semantic recall compares consequences.

Definition 4. *Let two ontologies O and O', where O' is the result of any extraction process over O (i.e., $O' \subseteq O$ syntactically). The theoretic semantic recall of O' w.r.t. O is given by:*

$$SemRecall_T(O', O) = \frac{|Cons(O')|}{|Cons(O)|}$$

Potentially, $Cons(O)$ and $Cons(O')$ can be infinite, so we have to restrict it to the set of axioms that we can materialise from the extracted module:

Definition 5. *Let O and O' be two ontologies, where O' is the result of any extraction process over O (i.e., $O' \subseteq O$ syntactically). The practical semantic recall of O' w.r.t. O is given by:*

$$SemRecall(O', O) = \frac{|mat(O')|}{|mat(O)|}$$

where $mat(O) \subseteq Cons(O)$ is the ontology containing all the axioms that can be materialised by a reasoner from O.

The definition of the ontology that can be materialised by a reasoner from an input ontology $mat(O)$ could be defined in many different ways, depending on the axiom types that the reasoner takes into account. In this paper, we use the strategy implemented in our prototype, presented in Definition 1.

Note that, following Definition 5, we can work with different versions of the ontologies. In particular, we could choose to work under different materialisation regimes, where we could choose to have different kinds of inferences materialised in order to improve the semantic loss analysis[8].

Depending on the particular task, it can be convenient to extend further the definition to take the signature of the extracted subontology into account:

Definition 6. *Let O and O' be two ontologies, where O' is the result of any extraction process over O (i.e., $O' \subseteq O$ syntactically). The signature-aware practical semantic*

[8]Of course, this requires the cost of materialising the different ontologies, so it has to be defined carefully.

recall of O' w.r.t. O is given by:

$$SemRecall_{SIG}(O', O) = \frac{|mat(O')|}{|restrictSig(mat(O), O')|}$$

where restrictSig (O, O') is the set of axioms in O which refer to any element in the signature of O'.

The signature of the extracted ontology must not be confused with the *seed signature* of the subontology, i.e., the signature used as starting point for the subontology extraction. This leads, for the sake of completeness, to define the generalisation of the definition to work with any particular signature externally defined:

Definition 7. *Let O and O' be two ontologies, where O' is the result of any extraction process over O (i.e., $O' \subseteq O$ syntactically). The externally defined signature-aware practical semantic recall of O' w.r.t. O is given by:*

$$SemRecall^{E}_{SIG}(O', O, S) = \frac{|mat(restrictSig(O', S))|}{|mat(O), S)|}$$

where restrictSig (O, S) is the set of axioms in O which refer to any element in S. Indeed, it directly follows that $SemRecall_{SIG}(O', O) = SemRecall^{E}_{SIG}(O', O, Sig(O')$.

Finally, note that we could use the above definitions to compare two different subontologies O' and O'' extracted from the same original ontology O by relaxing the condition of syntactical inclusion (as they might extract different parts of the original ontology). In this case, the precision would be always 1, as all the axioms comes from the same source, but we would be calculating the *Syntactic* and *Semantic Overlap*:

Definition 8. *Let two ontologies O' and O'', where both O' and O'' are the result of any extraction process over O (i.e., $O' \subseteq O$ and $O'' \subseteq O$ syntactically). The syntactic overlap of O' and O'' is given by:*

$$SynOverlap(O', O'') = \frac{|O' \cap O''|}{|O' \cup O''|}$$

Definition 9. *Let two ontologies O' and O'', where both O' and O'' are the result of any extraction process over O (i.e., $O' \subseteq O$ and $O'' \subseteq O$ syntactically). The semantic overlap of O' and O'' is given by:*

$$SemOverlap(O', O'') = \frac{|mat(O') \cap mat(O'')|}{|mat(O') \cup mat(O'')|}$$

Note that we have considered the semantic loss between a pair of ontologies, but one could also define semantic loss for specific reasoning tasks.

In the following section, we will apply these measures in order to evaluate how our initial implementation of the pipeline behaves in terms of semantic loss. As a baseline, we will consider the syntactic recall and the relaxed definitions (*Overlaps*) in order to evaluate our proposed technique to a safe-module extraction one (locality-based modularisation (Cuenca Grau et al., 2008)).

6. Evaluation

In order to show the feasibility of our approach, we have focused our experiments on several particular aspects: 1) we first test how accurate the prediction of the execution time was with our dataset (Section 6.1), 2) then, we evaluate how such accuracy affects our approach regarding materialising ontologies within a restricted timeout (Section 6.2), 3) we evaluate the extraction technique regarding the semantic loss incurred when compared to the original ontologies and safe-modules (Section 6.3), and 4) we discuss some detailed examples involving particular ontologies (Section 6.4).

Experimental Setup. For the experiments, we used the ORE 2015 dataset (Parsia et al., 2016), composed of 16,555 independent ontologies (many of them, real ontologies). The dataset was developed by independent researchers and was not designed for this task, so there is not any bias. We adopted Random Forests as a machine learning model, using 500 trees as hyperparameter. We explored other models such as Multilayer Perceptron and Linear Regression, but Random Forests were the best ones out of the shelf providing results that were good enough for our purposes, and we chose them for the sake of simplicity. (Hutter et al., 2014) also found that random forest models showed the best results to predict the performance of some algorithms to solve combinatorial problems (SAT, TSP, and MIP). We used Hermit 1.3.8 as reasoner and established a maximum timeout of 300 seconds for materialising the ontologies. Figure 4 shows the distribution of the execution times for the ontologies that were materialised under such a timeout: around 95% of ontologies are materialised before 100 seconds, around 98% before 200 seconds. The reasoning task to solve was to compute all the inferences available using the OWLAPI (we materialised all the inferences OWLAPI allows for, adding all the axioms to both the TBox and ABox). Finally, all the experiments were run on a desktop computer with an Intel Core i7-6700K processor (4 cores, 8 threads, although no parallelism was applied) at 4.00 GHz and 32 GB of RAM memory.

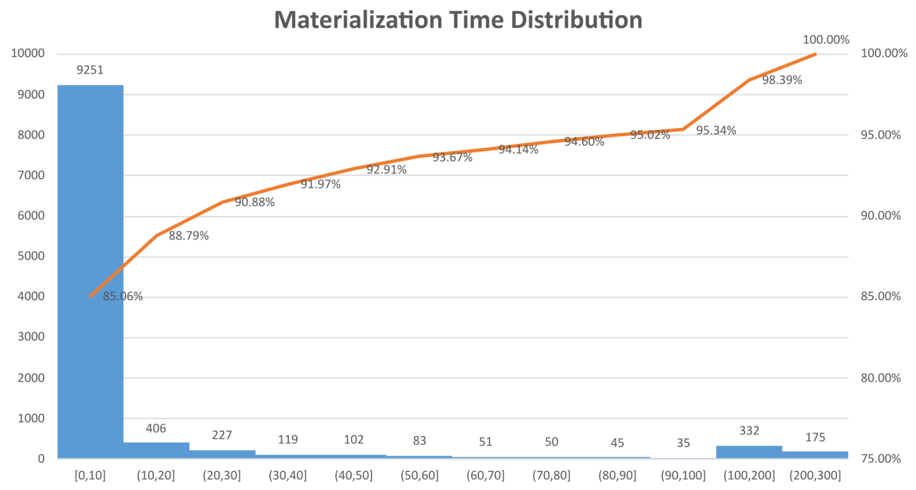


Figure 4: Distribution of ontology materialisation time. Axis-x shows materialisation time intervals, in seconds. Axis-y shows number of ontologies analysed (left) and percentage of materialised ontologies (right).

Our extraction method provides us with the capability of establishing a size for the module in number of axioms, but we needed to assess its semantic conservation capabilities. Thus, as a baseline to compare it in the semantic loss experiments, we have considered locality-based modularisation, using the Locality Module Extractor (Cuenca Grau et al., 2008)[9], which also has a Protégé plug-in called ProSÉ (Jiménez-Ruiz et al., 2008): Starting from a signature, we extract the Upper Modules (UMs) from $\perp - \textit{locality}$ (as suggested for the authors to reuse terminologies). Because UMs does not take into account the device resources, we needed to fix the number of axioms considered for the sake of comparability: the number of axioms to be extracted by our method is given by the size of the correspondent UM module, and we set $k \in \{5, 10, 15, 20\}$.

[9]http://www.cs.ox.ac.uk/isg/tools/ModuleExtractor

6.1. Precision of the Resource Prediction

First of all, we double checked that the 143 ontology features proposed in (Kang et al., 2012) and (Pan et al., 2018) were appropriate for our purposes. For this, we first materialised the whole dataset. Establishing the timeout of 300 s, we managed to obtain the materialisation time for 10,932 ontologies.

With such data, we adopted a 10-fold cross-validation approach and two setups: 1) using just the data (ontological features + materialisation time) for the ontologies materialised to train the models, and 2) augmenting the data with subontologies obtained from the original ontologies and materialised under the same conditions (i.e., the training test contains the original ontologies plus a set of subontologies). The rationale was to check whether giving further insight about intermediate points for the ontologies provided the models with relevant information, as suggested in (Sazonau et al., 2014). In particular, the data augmentation was carried out using the locality-based modularisation (UM) and our extraction method with 5, 10, 15 and 20 concepts as signature (obtained with KCE).

Table 1 shows the R^2 score, MAE (Mean Average Error), and RMSE (Root Mean Square Error) obtained for both setups. Split size states the training and the test sizes for each case. Recall that $R^2 \in [0, 1]$, whereas MAE and RMSE are measured in seconds.

	Original data	Augmented data
Split size	9839/1093	86113/9575
R^2	0.904	0.848
MAE	3.118	0.801
RMSE	12.690	6.709

Table 1: R^2, MAE and RMSE values for both setups using Random Forests with 500 trees. The splits for CV including the modules were done splitting the ontologies and then selecting the modules to avoid data leaking.

We can see that the high R^2 values show that the features used are informative enough to predict the time for materialisation[10]. Observe that although augmented data have a smaller R^2 score, MAE and RMSE values suggest that the model trained on this dataset is more accurate. However, given that the data augmentation technique might be computationally expensive, we will use original data to avoid such a cost. The errors for the original data are low as well, and we will check that their precision is enough for our purposes in the next section.

6.2. Adjusting the Available Knowledge to the Resources

We now turn our attention to whether the previous precision is good enough for our proposal. In this case, we use the model trained with just the ontologies (modules are not used), and fix the signature size to 10 concepts (the most important concepts according to KCE signature extraction) and a timeout of 10 seconds (which seems a reasonable time for a final user to wait). We split the results into two different sets: the 10876 ontologies that were materialised within the 300 s timeout (those used in the previous section), and the 5,679 other ones which were not handled within such a timeout.

Table 2 shows two confusion matrices for the ontologies within the timeout. The left matrix show the contrast between the predicted time and the real one for the original ontologies. In this matrix, the worst situation would be the false positives (41 ontologies predicted to be under the 10 seconds limit, but which actually required more time, which is a 0.38% of the whole dataset). The right matrix shows the results after iterating to reduce the size of the ontologies, as explained in Section 4.2 (i.e., reducing iteratively the size of the ontology a 25% at each step, until the prediction consumption met the criteria).

We can see how, for the final results (after having predicted and extracted the modules), summing up the "$\leq$ 10" column of right matrix, we would have 10,654 ontologies which we estimated that will be processed under 10 seconds (97.9% of the set of mate-

[10]As already mentioned, we explored as well the possibility of training a model to predict the number of axioms, but it was not accurate enough to do the way back for the prediction and the iterative approach was deemed to be more suitable for showing the feasibility of the approach.

		Real Time	
		≤ 10	> 10
Predicted Time	≤ 10	8943 (82.23%)	41 (0.38%)
	> 10	308 (2.83%)	1584 (14.46%)

(a) Predicted Time: Initial prediction

	Final Time	
	≤ 10	> 10
≤ 10	8969 (82.47%)	15 (0.14%)
> 10	1685 (15.49%)	207 (1.90%)

(b) Predicted Time: Final results

Table 2: Confusion matrices for the 10,876 ontologies that were materialised within the timeout of 300 s.

rialised ontologies). Note that this success ratio includes both the ontologies predicted correctly from the beginning (8969), and those processed and which ended taking less than 10 s (1685).

For the ontologies that were not processed within 300 s (5,679 ontologies, 34.30% of the dataset), we checked how many were predicted to be out of the 300 s timeout and of our 10 s limit. None was predicted to be above the 300 s timeout, showing a limitation of our selected machine learning model (we leave the use of better generalising models as future work). On the other hand, 899 ontologies (5.43%) were predicted to be within the 10 s. limit which was obviously wrong, but curiously they all were ontologies that HermiT could not process[11]. Nevertheless, assuming that all of them were above the 10 s timeout, we managed to reduce and process 3,465 (20.93%) out of 4,490 ontologies (27.12%) with our approach within the established time out (the rest of the ontologies, 1,189 (7.18%), were not handled by that version of HermiT).

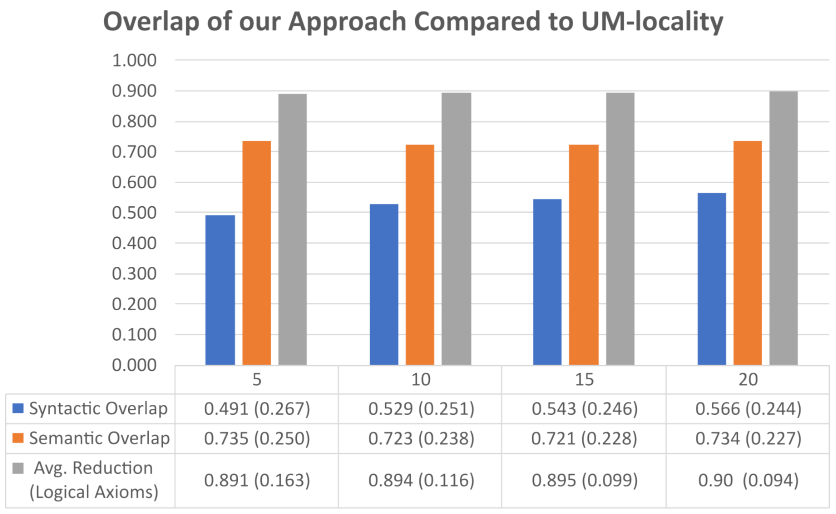
6.3. *Evaluation of the Semantic Loss*

Once we tested the feasibility of the knowledge extraction pipeline, we focused on measuring the semantic loss of: 1) our extraction technique when compared to logic-based module techniques, and 2) the actual application of our pipeline when compared to using the full ontologies.

[11]Mainly due to inconsistencies, malformed literals, and unsupported characteristics.

Comparison to Locality-based Modules

As commented in the experimental setup, in order to check if our extraction method was good enough for our purposes, we compared the overlap of our extracted partitions to locality-based modularisation (Cuenca Grau et al., 2008) extracting the Upper Modules (UM). In particular, out of the 16,555 ontologies, we were able to extract and materialise both modules (ours and UM) for 14,255 of them. Figure 5 shows the comparison in terms of *syntactic* and *semantic overlap* as defined in Section 5. We add the *reduction* percentage as we noticed that, even though we established the size of UM modules as an upper bound for our extraction procedure, in general UM modules were still bigger than ours as ours exhausted the extraction earlier.



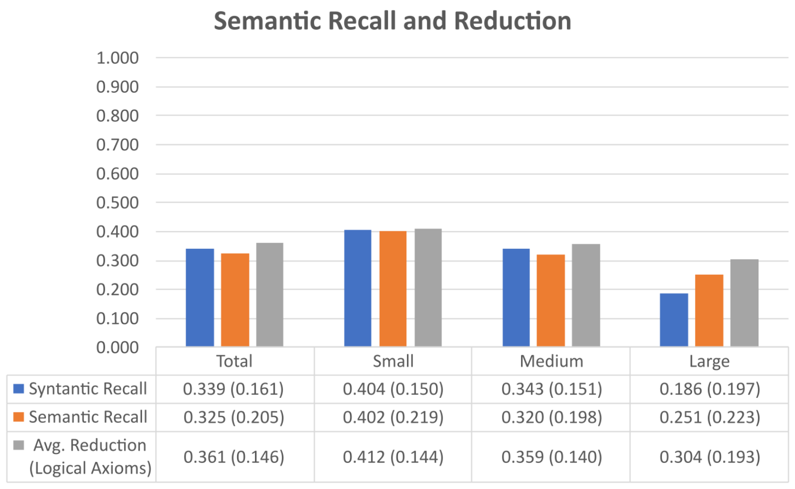
	5	10	15	20
🟦 Syntactic Overlap	0.491 (0.267)	0.529 (0.251)	0.543 (0.246)	0.566 (0.244)
🟧 Semantic Overlap	0.735 (0.250)	0.723 (0.238)	0.721 (0.228)	0.734 (0.227)
⬜ Avg. Reduction (Logical Axioms)	0.891 (0.163)	0.894 (0.116)	0.895 (0.099)	0.90 (0.094)

Figure 5: Syntactic and semantic overlap between UM modules and ours, along with the relative size of our modules (Reduction) compared to UM ones (our modules are about 10% smaller than UM ones). The X axis depicts the size of the signature, and the results in the table are read as *average (standard deviation)*.

We can see how the *syntactic overlap* increases as the signature does: bigger signatures lead to bigger UM modules, which in turn allows for bigger signatures and more choices for our extraction approach. However, note how the *semantic overlap* remains stable, implying that we keep important knowledge (in this case, the taxonomy skeleton which we decided to include in our conservation language, is important for both extraction techniques). Moreover, we have to take into account as well that our modules are about a 10% smaller than UM ones, so it can be argued that we still have some room for adding more axioms we deem important, to increase the overlap without exceeding the upper bound number of axioms to be extracted.

Comparison to the Original Ontologies

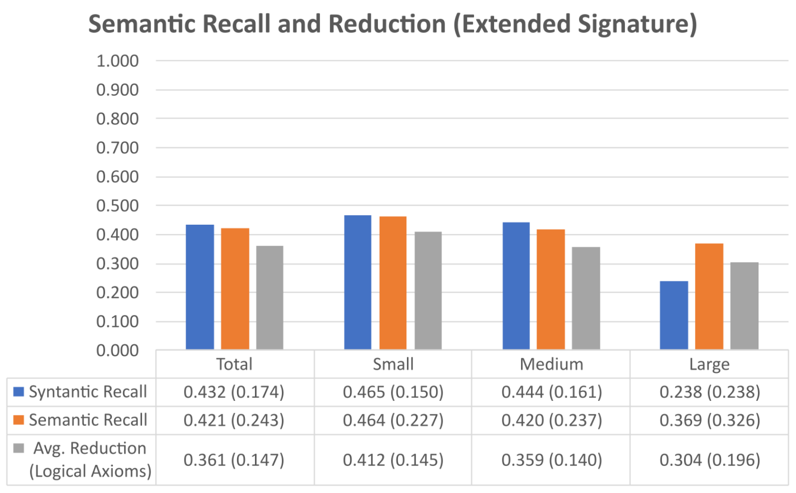
In this case, we focus on the 1,892 ontologies that were predicted to be above the 10 s timeout as they were the ones subject to reduction. We established a combined timeout of 1,200 seconds for materialising both ontologies (the original and the extracted one), and calculating both syntactic and semantic recall values (which, following the work by Euzenat et al. (Euzenat, 2007), required checking whether each axiom was inferred from the original ontology), which lead to a final set of 1,360 analysed ontologies.



	Total	Small	Medium	Large
🟦 Syntantic Recall	0.339 (0.161)	0.404 (0.150)	0.343 (0.151)	0.186 (0.197)
🟧 Semantic Recall	0.325 (0.205)	0.402 (0.219)	0.320 (0.198)	0.251 (0.223)
⬜ Avg. Reduction (Logical Axioms)	0.361 (0.146)	0.412 (0.144)	0.359 (0.140)	0.304 (0.193)

Figure 6: Syntactic and semantic recall between the original ontologies and our subontologies, along with the relative size (Reduction): global recall without taking into account any signature.The results in the table are read as *average (standard deviation)*.

Figures 6– 8 show the *syntactic* and *semantic recall* for these ontologies calculated in three settings: a) globally (Definition 5) in Figure 6, b) restricted to the signature of the extracted subontologies (Definition 6) in Figure 7, and c) restricted to the original seed signature (Definition 7) in Figure 8. We show the total averaged values along with



	Total	Small	Medium	Large
🟦 Syntantic Recall	0.432 (0.174)	0.465 (0.150)	0.444 (0.161)	0.238 (0.238)
🟧 Semantic Recall	0.421 (0.243)	0.464 (0.227)	0.420 (0.237)	0.369 (0.326)
▨ Avg. Reduction (Logical Axioms)	0.361 (0.147)	0.412 (0.145)	0.359 (0.140)	0.304 (0.196)

Figure 7: Syntactic and semantic recall between the original ontologies and our subontologies, along with the relative size (Reduction): recall conditioned to the extended signature. The results in the table are read as *average (standard deviation)*.

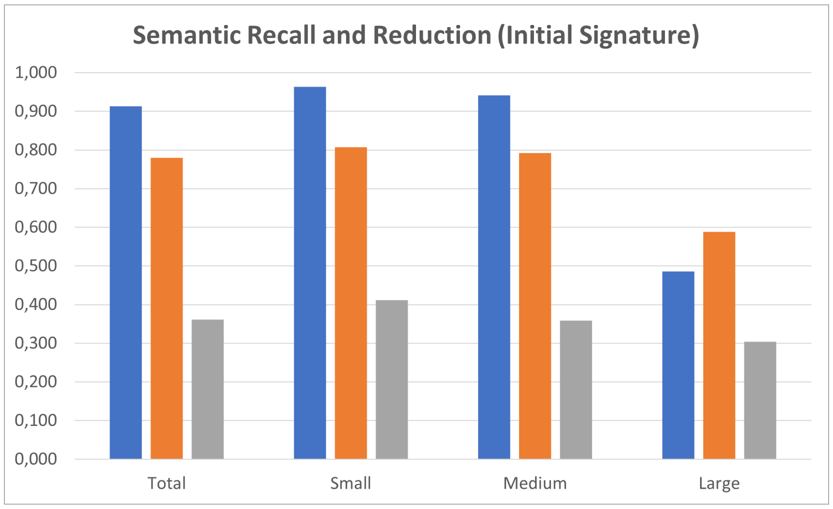


Figure 8: Syntactic and semantic recall between the original ontologies and our subontologies, along with the relative size (Reduction): recall conditioned to the initial signature. The results in the table are read as *average (standard deviation)*.

the standard deviations (between brackets, as well as the values aggregated regarding the size of the original ontologies, namely, *small* ontologies (#*axioms* $\leq$ 500), *medium* ones ($500 < \#axioms \leq 5,000$), and *large* ones ($\#axioms > 5,000$)[12].

We can see how the results are stable across all the ontology sizes, and strongly correlated to the reduction achieved conditioned to the signature of the subontology[13], but for the initial signature, which show that the main information about the initial signature is kept in our extracted subontology. When analysing the data clustered by ontology size, we can see that, for small and medium ontologies, both syntactic and semantic recall values are quite similar, but syntactic recall is notably lower than semantic one for large ontologies, which remarks that the information we keep in subontologies is semantically relevant. These results, along with the overlap achieved by our approach with UM-modules, show that our proposal is capable of adapting the knowledge without losing the most important parts, regarding the given signatures. Therefore, our definitions of semantic loss, based on the proportion between the number of inferences with respect to the ideal number of inferences, seem realistic.

6.4. Detailed examples

In this section we show two concrete cases of our prototype and architecture, using real ontologies for ontology visualisation and retrieval, respectively.

Example 1 (Ontology visualisation). *A final user is navigating through and ontology and want to visualise its relevant classes and their individuals. More precisely, `00104` (3,573 logical axioms, 641KB file), one of the ontologies in the ORE 2015 dataset which includes information about geographic locations (`#GEOREF`), types of food commodities (`#FOODS-COMMODITY`), and different languages (`#LANGUAGE`), among many others things. The user is located at class `#FOODS-COMMODITY` and wants to navigate through more specific classes. To do so, the ontology navigation would call a semantic reasoner to retrieve the subclasses of the original class. (Note that retrieving the subclasses of class could be interesting in other scenarios, such as to refine the results of an instance retrieval query when there are too many results.) Unfortunately, semantic*

[12]We classified them as in the ORE 2013 Workshop, by counting the logical axioms in the original ontology (that is, before doing any reasoning) (Gonçalves et al., 2013)

[13]Note that this signature might be quite larger than the seed one.

reasoning does not finish in a 300 seconds timeout: the ontology is too complex for his/her device (this actually happens using the environmental setup described in this section). Therefore, using a traditional reasoner, the user would receive an empty answer (reasoners do not retrieve subclasses as they compute them, but at the end of the process, where all of them have been computed).

Instead, using our prototype (with `#GEOREF`, `#LANGUAGE`, and `#FOODS-COMMODITY` as signature), reasoning finishes within 10 seconds, so the final user could happily navigate to any of the subclasses of `#FOODS-COMMODITY`.

In fact, if we query the subontology retrieved by our prototype to obtain the descendants and instances of the following concepts, we get the following results:

- *`#GEOREF`: 100% (265/265) of the instances (the original concept did not have subclasses).*
- *`#LANGUAGE`: 100%% (77/77) of the descendants and a 100% (263/263) of the instances.*
- *`#FOODS-COMMODITY`: 100% (45/45) of the descendants (the original concept did not have instances).*

While the available knowledge is not complete, at least, we managed to provide the user with information about these concepts and an application could use it, maybe notifying the user that is not complete. For example, if we take SWEET Ontology Phenomena Atmosphere[14], and we would like to retrieve all the phenomena that have a planetary scale (`Phenomena and hasSpatialScale value PlanetaryScale`), our approach it is able to retrieve a 3.81% (13/341) of the subclasses, being able to answer some concepts such as `Climate` or `AtmosphericPhenomena`. This is due the fact that the prediction is not so accurate in this case and reduces the amount of axioms too conservatively (we have checked that our approach, adding more axioms, increases the retrieval results). Note that while the percentages might seem low sometimes, the alternative was to raise directly an error and not providing any answer at all.

[14] `00412d8c-f97f-4e3d-b184-9a8133e74e61_phenWave.owl_functional.owl` in the dataset.

Example 2 (Knowledge Retrieval). *In this example, a final user wants to know as much as possible about any pathology that can occur in the Liver. In this case, the application would be using* `GALEN-Full-Union_ALCHOI`(D) *(37411 logical axioms, 11MB file), which cannot be processed by HermiT*[15]*. For this ontology, our prediction module was timing out and we found out that calculating the metrics for such an ontology was quite expensive sometimes (they are quadratic in the size of the underlying graph), but we opted to leave their optimisation as future work. This said, we directly reduced the size of the ontology to a 25% percent of the original logical axioms*[16]*. With this setup, our approach, using* `Liver` *and* `hasLocation` *as signature, is able to extract a subontology where asking for* `hasLocation some Liver` *retrieves 9 classified concepts, among which we can find* `Hepatitis`, `HepaticNecrosis` *and* `FattyLiver`*. Note that those are not directly related to* Liver *by* `hasLocation` *but by* `hasSpecificLocation`*, its subproperty. Extracting a subontology in this case is not only about performance, but a matter of enabling the application to work (approximating the reasoning).*

In both real-world examples, the original ontologies could not be processed by a regular desktop computer. However, our prototype was able to compute a subontology so that the original problem can be partially solved, even when some inferences are missing.

7. Conclusions and Future Work

In this paper, we have discussed the develop strategies to adapt ontology reasoning to the limitations of the device where such a reasoning will take place, particularly in the case of devices with heavily constrained resources, such as mobile devices or IoT infrastructure.

[15]In fact, we have used different versions of HermiT and Pellet to handle it, but the original one could not be processed.

[16]We tested it with 25% and 12.5% with exactly the same results of the reasoning, but higher materialisation times for 12.5%.

We have proposed an architecture to perform adaptive reasoning, taking into account several criteria (such as the maximum running time or memory/battery limits). The key idea is to perform a knowledge extraction step that is suitable for the device, possibly after several iterations of the process. Reasoning will thus be executed against a subontology that can be processed on the device, although, in general, the reasoning results would be incomplete. We have discussed several issues to be taken into account during the process, such as the use of a signature (possibly including automatically discovered key concepts) as a starting point, or a preservation language as a strategy to decide which axioms to be kept. We have also discussed how to measure the impact of our approximations. In particular, we proposed a novel definition of semantic loss, adapted from the ontology alignment field.

To illustrate some important steps of our architecture, we have implemented a prototype that is able to compute subontologies on desktop computers according to the predicted values of a single criterion (the reasoning time), which obviously depends on the device resources. The prototype illustrates the joint use of feature selection, strategies to compute a signature (KCE), and modularisation to give a possibly incomplete answer to the ontology materialisation problem with limited resources.

We think that this paper clearly shows the potential of this line of research but also that there is a long road ahead, and there exist many directions for our future research. More experiments on mobile devices (or other devices with constrained resources) will also be necessary to properly evaluate the feasibility of the approach. In particular, although considering as many possible criteria as possible is desirable, first it has to be confirmed empirically that their costs can be successfully predicted.

To do so, our prototype could be extended in several ways. Firstly, using more complex (but efficient) prediction strategies, possibly including features selected for mobile devices, and different machine learning techniques. Secondly, to compute the signature locally, we need a more efficient implementation of KCE or another algorithm. Thirdly, our prototype could be generalised to decide whether to perform local reasoning or not. Last but not least, we would like to test the pipeline on resource-constrained devices such as mobile phones. However, to do so, it is firstly necessary to develop models to predict the ontology reasoning time, which so far has only considered desktop computers.

Acknowledgment

The work of C. Bobed, F. Bobillo, and E. Mena was supported by the I+D+i project PID2020-113903RB-I00 (funded by MCIN/AEI/10.13039/501100011033) and the project T42_23R (funded by Gobierno de Aragón). We also thank the anonymous reviewers for their helpful comments to improve the paper.

References

References

Allemang, D., Hendler, J., Gandon, F., 2020. Semantic Web for the Working Ontologist, 3rd edition. Morgan & Claypool.

Armas-Romero, A., Cuenca Grau, B., Horrocks, I., 2012. MORe: Modular combination of OWL reasoners for ontology classification. In: Proceedings of the 11th International Semantic Web Conference (ISWC 2012). pp. 1–16.

Armas-Romero, A., Kaminski, M., Cuenca Grau, B., Horrocks, I., 2016. Module extraction in expressive ontology languages via Datalog reasoning. Journal of Artificial Intelligence Research 55, 499–564.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. F., 2003. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press.

BioPortal, 2023. SNOMED CT. URL: http://bioportal.bioontology.org/ontologies/SNOMEDCT.

Bobed, C., Bobillo, F., Mena, E., Pan, J. Z., 2017. On serializable incremental semantic reasoners. In: Proceedings of the 9th International Conference on Knowledge Capture (K-CAP 2017). ACM, pp. 1–4.

Bobed, C., Yus, R., Bobillo, F., Mena, E., 2015. Semantic reasoning on mobile devices: Do androids dream of efficient reasoners? Journal of Web Semantics 35, 167–183.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U., 2008. Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research 31, 273–318.

Cuenca Grau, B., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., 2012. Ontology evolution under semantic constraints. In: Proceedings of the 13th International conference on Principles of Knowledge Representation and Reasoning (KR 2012). pp. 137–147.

Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A., 2006. Modularity and web ontologies. In: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006). pp. 198–209.

Del Vescovo, C., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Tsarkov, D., 2013. Empirical study of logic-based modules: Cheap is cheerful. In: Proceedings of the 12th International Semantic Web Conference (ISWC 2013). Springer, pp. 84–100.

Dudáš, M., Lohmann, S., Svátek, V., & Pavlov, D. (2018). Ontology visualization methods and tools: A survey of the state of the art. The Knowledge Engineering Review, 33, E10.

Euzenat, J., 2007. Semantic precision and recall for ontology alignment evaluation. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI 2007). pp. 348–353.

Gatens, W., Konev, B., Wolter, F., 2013. Module extraction for acyclic ontologies. In: Proceedings of the 7th International Workshop on Modular Ontologies (WoMO 2013). Vol. 1081 of CEUR Workshop Proceedings. pp. 49–60.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z., 2014. HermiT: An OWL 2 reasoner. Journal of Automated Reasoning 53 (3), 245–269.

Gobin-Rahimbux, B., 2022. Evaluation metrics for ontology modules: Short report. In: Proceedings of the 2022 IEEE International Conference on Data Science and Information System (ICDSIS 2022). IEEE, pp. 1–6.

Gonçalves, R. S., Bail, S., Jiménez-Ruiz, E., Matentzoglu, N., Parsia, B., Glimm, B., Kazakov, Y., 2013. OWL Reasoner Evaluation (ORE) workshop 2013 results: Short report. In: Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013). Vol. 1015. CEUR Workshop Proceedings, pp. 1–18.

Guazzelli, A., Zeller, M., Lin, W.-C., Williams, G., et al., 2009. PMML: An open standard for sharing models. R Journal 1 (1), 60.

Guclu, I., Bobed, C., Pan, J. Z., Kollingbaum, M. J., Li, Y., 2016a. How can reasoner performance of ABox intensive ontologies be predicted? In: Proceedings of the 6th Joint International Conference on Semantic Technology (JIST 2016). pp. 3–14.

Guclu, I., Li, Y., Pan, J. Z., Kollingbaum, M. J., 2016b. Predicting energy consumption of ontology reasoning over mobile devices. In: Proceedings of the 15th International Semantic Web Conference (ISWC 2016). pp. 198–214.

Horridge, M., Bechhofer, S., 2011. The OWL API: A Java API for OWL ontologies. Semantic web 2 (1), 11–21.

Huitzil, I., Alegre, F., Bobillo, F., 2020. GimmeHop: A recommender system for mobile devices using ontology reasoners and fuzzy logic. Fuzzy Sets and Systems, 401, 55–77.

Huitzil, I., Straccia, U., Bobed, C., Mena, E., Bobillo, F., 2020. The serializable and incremental semantic reasoner fuzzyDL. In: Proceedings of the 29th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2020). IEEE Press, pp. 1–8.

Hutter, F., Xu, L., Hoos, H. H., Leyton-Brown, K., 2014. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence 206, 79–111.

Jiménez-Ruiz, E., Grau, B., Sattler, U., Schneider, T., Berlanga, R., 2008. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: Proceedings of the 5th European Semantic Web Conference (ESWC 2008). Springer, pp. 185–199.

Kang, Y.-B., Li, Y.-F., Krishnaswamy, S., 2012. Predicting reasoning performance using ontology metrics. In: Proceedings of the 11th International Semantic Web Conference (ISWC 2012). pp. 198–214.

Kang, Y.-B., Pan, J. Z., Krishnaswamy, S., Sawangphol, W., Li, Y.-F., 2014. How long will it take? Accurate prediction of ontology reasoning performance. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014). pp. 80–86.

Kazakov, Y., Krötzsch, M., Simančík, F., 2014. The incredible ELK. Journal of Automated Reasoning 53, 1–61.

Kerschke, P., Hoos, H. H., Neumann, F., Trautmann, H., 2019. Automated algorithm selection: Survey and perspectives. Evolutionary Computation 27(1), 3–45.

Khamparia, A., Pandey, B., 11 2017. Comprehensive analysis of semantic web reasoners and tools: a survey. Education and Information Technologies 22, 3121–3145.

Kleemann, T., 2006. Towards mobile reasoning. In: Proceedings of the 2006 International Workshop on Description Logics (DL 2006). pp. 231–238.

Konev, B., Lutz, C., Walther, D., Wolter, F., 2013. Model-theoretic inseparability and modularity of description logic ontologies. Artificial Intelligence 203, 66–103.

Mena, E., Illarramendi, A., 2001. Ontology-Based Query Processing for Global Information Systems. Kluwer Academic Publishers.

Musen, M. A., 2015. The Protégé project: a look back and a look forward. AI Matters 1 (4), 4–12.

Noy, N. F., Musen, M. A., 2004. Specifying ontology views by traversal. In: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004). pp. 713–725.

Pan, J. Z., Bobed, C., Guclu, I., Bobillo, F., Kollingbaum, M. J., Mena, E., Li, Y.-F., 2018. Predicting reasoner performance on ABox intensive OWL 2 EL ontologies. International Journal on Semantic Web and Information Systems 14 (1), 1–30.

Parsia, B., Matentzoglu, N., Gonçalves, R. S., Glimm, B., Steigmiller, A., 2016. The OWL reasoner evaluation (ORE) 2015 resources. In: Proceedings of the 15th International Semantic Web Conference ISWC 2016), Part II. Vol. 9982 of Lecture Notes in Computer Science. Springer, pp. 159–167.

Pernisch, R., , D., Bernstein, A., 2021. Beware of the hierarchy — An analysis of ontology evolution and the materialisation impact for biomedical ontologie. International Journal on Semantic Web and Journal of Web Semantics 70, 100658.

Peroni, S., Motta, E., D'Aquin, M., 2008. Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In: Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008). pp. 242–256.

Ribeiro, M. T., Singh, S., Guestrin, C., 2016. Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2016). pp. 1135–1144.

Ribeiro, M. T., Singh, S., Guestrin, C., 2018. Anchors: High-precision model-agnostic explanations. In: Proceedings of the 32nd Conference on Artificial Intelligence (AAAI 2018). pp. 1527–1535.

Rogers, J., Roberts, A., Solomon, D., van der Haring, E. J., Wroe, C., Zanstra, P. E., Rector, A. L., 2001. GALEN Ten Years On: Tasks and Supporting Tools. In: Proceedings of the 10th World Congress on Medical Informatics (MEDINFO 2001). Vol. 84 of Studies in Health Technology and Informatics. IOS Press, pp. 256–260.

Ruta, M., Scioscia, F., Bilenchi, I., Gramegna, F., Loseto, G., Ieva, S., Pinto, A., 2022. A multiplatform reasoning engine for the semantic web of everything. Journal of Web Semantics 73, 100709.

Ruta, M., Scioscia, F., Gramegna, F., Bilenchi, I., Sciascio, E. D., 2019. Mini-ME Swift: The first mobile OWL reasoner for ios. In: Proceedings of the 16th Extended

Semantic Web Conference (ESWC 2019). Vol. 11503 of Lecture Notes in Computer Science. Springer, pp. 298–313.

Sazonau, V., Sattler, U., Brown, G., 2014. Predicting performance of OWL reasoners: Locally or globally? In: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR 2014). pp. 661–664.

Schlicht, A., Stuckenschmidt, H., 2006. Towards structural criteria for ontology modularization. In: Proceedings of the 1st International Workshop on Modular Ontologies (WoMO 2006). Vol. 232 of CEUR Workshop Proceedings. pp. 85–97.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y., 2014. Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5 (2), 51–53.

Martin G. Skjæveland, M., G., Gjerver, A., Hansen, C. M., Klüwer J. W., Strand, M. R., Waaler, A., *O*verli, P. *O*., 2018, Semantic Material Master Data Management at Aibel. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018). CEUR Workshop Proceedings 2180, CEUR-WS.org.

Steller, L. A., Krishnaswamy, S., Gaber, M. M., 2009. A weighted approach to partial matching for mobile reasoning. In: Proceedings of the 8th International Semantic Web Conference (ISWC 2009). pp. 618–633.

Stuckenschmidt, H., Klein, M., 2004. Structure-based partitioning of large concept hierarchies. In: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004). pp. 289–303.

Thomas, E., Pan, J. Z., Ren, Y., 2010. TrOWL: Tractable OWL 2 reasoning infrastructure. In: Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010). Vol. 2. pp. 431–435.

Van Woensel, W., Abidi, S. S. R., 2019. Optimizing and benchmarking OWL2 RL for semantic reasoning on mobile platforms. Semantic Web Journal 10, 637–663.

Yus, R., Mena, E., 2015. Emergency Management Using SHERLOCK. In: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 2015). ACM, pp. 495–495.

Zhang, H., Li, Y.-F., Tan, H. B. K., 2010. Measuring design complexity of semantic web ontologies. Journal of Systems and Software 83, 803–814.