Understanding and Improving Ontology Reasoning Efficiency through Learning and Ranking

Yong-Bin Kang[a], Shonali Krishnaswamy[a], Wudhichart Sawangphol[b], Lianli Gao[c], Yuan-Fang Li[d,*]

[a] *Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia*
[b] *Faculty of Information and Communication Technology, Mahidol University, Thailand*
[c] *School of Computer Science and Engineering,*
The University of Electronic Science and Technology of China
[d] *Faculty of Information Technology, Monash University, Australia*

Abstract

Ontologies are the fundamental building blocks of the Semantic Web and Linked Data. Reasoning is critical to ensure the logical consistency of ontologies, and to compute inferred knowledge from an ontology. It has been shown both theoretically and empirically that, despite decades of intensive work on optimising ontology reasoning algorithms, performing core reasoning tasks on large and expressive ontologies is time-consuming and resource-intensive. In this paper, we present the meta-reasoning framework R_2O_2* to tackle the important problems of understanding the source of TBox reasoning hardness and predicting and optimising TBox reasoning efficiency by exploiting machine learning techniques. R_2O_2* combines state-of-the-art OWL 2 DL reasoners as well as an efficient OWL 2 EL reasoner as components, and predicts the most efficient one by using an ensemble of robust learning algorithms including XGBoost and Random Forests. A comprehensive evaluation on a large and carefully curated ontology corpus shows that R_2O_2* outperforms all six component reasoners as well as AutoFolio, a robust and strong algorithm selection system.

Keywords: OWL, Reasoning, Performance prediction, Ontology, Metrics, Learning, Meta-reasoning, Semantic Web

* Author for correspondence
Email addresses: ykang@swin.edu.au (Yong-Bin Kang), skrishnaswamy@swin.edu.au (Shonali Krishnaswamy), wudhichart.saw@mahidol.edu (Wudhichart Sawangphol), lianli.gao@uestc.edu.cn (Lianli Gao), yuanfang.li@monash.edu (Yuan-Fang Li)

1. Introduction

Ontologies are essential building blocks of the Semantic Web. Expressive ontology languages OWL DL and OWL 2 DL are widely used to represent many complex phenomena in a number of application domains, including bioinformatics [1], software engineering [2] and data management [3–6]. In these domains, maintaining the logical correctness of ontologies (i.e. consistency checking) and deducing implicit facts from ontologies (i.e. classification) are both important tasks that may need to be performed repeatedly. However, ontologies as expressed in common ontology languages such as OWL [7] and OWL 2 [8] can be large, complex, or both. The high worst-case complexity of these ontology languages incurs high computational costs on the above core reasoning problems. Checking the logical consistency of an ontology in $\mathcal{SHOIN}(\mathbf{D})$, the description logic (DL) underlying OWL DL, has NExpTime-complete worst-case complexity [7]. The complexity of the same problem for $\mathcal{SROIQ}(\mathbf{D})$, the DL underlying OWL 2 DL, is even higher (2NExpTime-complete) [8].

The past decade has seen the development of highly optimised inference algorithms for description logics, with (hyper) tableau algorithms [9] being a leading exemplar. A number of high-performance DL reasoners have been developed, including FaCT++ [10], HermiT [11], Konclude [12], Pellet [13] and TrOWL [14]. Despite the tremendous progress in both theoretical research and practical implementation, the high theoretical worst-case complexity results for OWL DL and OWL 2 DL still imply that core reasoning services may be computationally very expensive. It has been shown empirically that reasoning on large and complex ontologies in OWL 2 DL and OWL 2 EL (a less expressive profile that enjoys a PTime-complete complexity) can be very time-consuming for state-of-the-art reasoners [15, 16]. Such high difficulty of reasoning and the fundamental role inference plays in ontology-based applications make it highly desirable to be able to accurately predict inference performance for ontologies and reasoners.

It is well-known that worst-case complexity does not necessarily provide useful insights into hardness of individual instances [17, 18]. In this context, it is noteworthy that reasoner benchmarking has been conducted previously [15, 19–22]. Except the two ORE competitions, these works only compared inference performance on a small set of ontologies. Moreover, they did not attempt to correlate characteristics of ontologies with their inference performance. Hence, they do not provide insight into what makes inference difficult on a given ontology.

The *robustness* of ontology reasoners was recently investigated [23], with a particular focus on reasoning efficiency. It was observed that given a corpus of ontologies and a number of state-of-the-art reasoners, it is highly likely that one of the reasoners performs sufficiently well on any given ontology in the corpus. However, this *virtual best reasoner* is only found *a posteriori*, and the paper did not discuss how the best reasoner may be selected automatically. It only stated that this task is not straightforward.

In our previous work we studied the characterisation of ontology's design complexity using metrics [24], the prediction of ontology classification efficiency [25, 26], and proposed a *meta-reasoner* R_2O_2 [27]. A meta-reasoner is one that combines other (component) reasoners. Given an ontology, a meta-reasoner predicts the most efficient component reasoner and selects it to carry out reasoning on that ontology. In this paper, we improve upon our existing work and present a learning- and ranking-based framework for the understanding of sources of ontology reasoning hardness, prediction of ontology reasoning time, and ultimately improving reasoning performance, under a unifying meta-reasoning framework. The main contributions of this paper can be summarised as follows:

- **Accurate prediction models**: A regression based prediction model is learned for each of a number of state-of-the-art OWL 2 DL reasoners. Evaluated with 10-fold cross validation, all the models are highly accurate, with R^2 (*coefficient of determination*) values in $[0.71, 0.95]$.
- **A meta-reasoning framework**: A novel meta-reasoning framework, R_2O_2*, is developed. Building on our previous work, R_2O_2* ranks and selects OWL reasoners with the aim of determining the most efficient reasoner for an unknown ontology. Compared with R_2O_2 [27], R_2O_2* utilises a robust, state-of-the-art prediction model XGBoost [28] based on gradient boosting [29] and an emsemble learning method *stacking* that combines multiple learning algorithms to obtain better predictive per-

formance. R_2O_2* integrates state-of-the-art, *sound* and *complete* reasoners that are both efficient and robust. Moreover, R_2O_2* also incorporates ELK [30], an efficient reasoner for OWL 2 EL ontologies, to further improve reasoning efficiency for a wider variety of ontologies.

- **Comprehensive evaluation**: A comprehensive evaluation on reasoning time has been conducted on a modern, large, and carefully-curated ontology corpus from the ORE 2015 reasoner competition [22]. Our evaluation shows that R_2O_2* outperforms all of the six OWL 2 DL component reasoners in the evaluation. The complete meta-reasoner variant, $R_2O_2*_{\text{(all)}}$, also outperforms our previous meta-reasoners PR and R_2O_2 [27]. More importantly, R_2O_2* outperforms AutoFolio [31], a state-of-the-art portfolio-based algorithm selection model that has demonstrated excellent performance in a number of domains.
- **Ontology metrics**: Furthermore, we give full definitions of a large suite of 91 ontology metrics that have been mentioned but not formally defined in our previous work [24, 26]. The formal definitions provide a valuable insight into ontology engineering and maintenance. We also identify the most important metrics that affect ontology reasoning efficiency, which further informs ontology engineering practices.

The R_2O_2* meta-reasoner, including components that calculate metrics and train prediction models and rankers, has been made freely available for wider dissemination.[1]

The rest of the paper is organised as follows. A brief overview of background knowledge of ontologies and reasoning is given in Section 2, followed by a discussion of closely-related work in Section 3. The suite of all metrics that characterise the design complexity of OWL ontologies are formally defined in Section 4. The four variants of the meta-reasoner R_2O_2* are described in detail in Section 5. Section 6 presents the evaluation framework for building prediction models and the meta-reasoner. Our detailed evaluation results and analysis are presented in Section 7. Lastly, we conclude the paper and discuss future work in Section 8.

[1] `https://github.com/liyuanfang/r2o2-star`

2. Ontologies and Reasoning

Ontologies organise domain knowledge in a structured and logical way. Semantic Web ontologies have been widely used in many different areas as a medium for knowledge representation and data integration. Common ontology languages such as OWL 1 [7] and OWL 2 [8] have formal semantics defined by Description Logics (DL) [32], a family of logics created specifically for the purpose of knowledge representation. In simple terms, knowledge in DL is characterised by abstract *concepts*, which represent sets of entities; *properties* or roles, which are binary relations between entities; and *individuals*, which represent entities themselves. Hence, a concept semantically represents a set of individuals.

The logical nature gives rise to reasoning support for ontologies. These, among others, include concept satisfiability checking, concept subsumption checking, and classification. Concept satisfiability checking ensures that a concept can contain at least one individual. Concept subsumption checks whether two concepts have a sub-class relationship. Classification computes the subsumption relationship between all pairs of (named) concepts in an ontology.

For example, the following axioms in the *DL syntax* describe some knowledge about pizzas. Axioms (1) and (2) state that $AmericanHot$ is a $Pizza$, and that it has some $MozzarellaTopping$. Axiom (3) states that $MozzarellaTopping$ is a $CheeseTopping$. Axiom (4) states that $CheeseyPizza$ is exactly those $Pizza$ that has some $CheeseTopping$. Through classification, the concept $AmericanHot$ is found to bea subclass of $CheeseyPizza$, as it is a Pizza, and that it also has some $MozzarellaTopping$, which is a type of $CheeseyTopping$.

$$AmericanHot \sqsubseteq Pizza \tag{1}$$

$$AmericanHot \sqsubseteq \exists hasTopping.MozzarellaTopping \tag{2}$$

$$MozzarellaTopping \sqsubseteq CheeseTopping \tag{3}$$

$$CheeseyPizza \equiv Pizza \sqcap \exists hasTopping.CheeseTopping \tag{4}$$

A number of different DLs have been proposed over the years. These DLs include different combinations of language constructs, hence they have different expressive power. As a result, they also have different worst-case complexity results for core reasoning tasks. A detailed introduction to the syntax, semantics and complexity of these ontology languages can be found in the literature [7, 8].

Optimisation of ontology reasoning algorithms has been aggressively pursued over the past decades, and a number of highly optimised reasoners have been produced. These include sound and complete reasoners such as FaCT++ [33], HermiT [34], Konclude [12], and Pellet [13]; as well as the sound but incomplete reasoner TrOWL [14]. Despite tremendous progress in optimisation, there has been ample empirical evidence of the actual *hardness* of real-world ontologies [15, 25, 35]. Therefore, efficient reasoning over large and expressive ontologies remains a computationally challenging task. On the other hand, efficient reasoners dedicated to less expressive profiles have also been developed. ELK [30] is such a concurrent reasoner for ontologies in the OWL 2 EL profile.

3. Related Work

Our related work is divided into the three categories according to the focuses in this paper: ontology metrics, prediction models for OWL reasoners, and algorithm selection and meta-reasoners.

Ontology Metrics. There has been research on the development of a series of metrics for analysing ontology complexity [36]. The pioneering work for identifying the proposed metrics in this paper is found in [24] that introduced a suite of 8 metrics with the aim of characterising different aspects of ontology *design complexity*. Further, we identified an additional 19 metrics that can measure different aspects of the size and structural characteristics of an ontology [25] . These 27 metrics were used for predicting discretised reasoning performance of reasoners. These 27 metrics were combined with another set of metrics that capture ontology complexity in [26]. In total, 91 metrics were collected and used to build models for predicting absolute reasoning performance of reasoners. In this work, we use these 91 metrics to build prediction models and the meta-reasoner R_2O_2*. We also give the full definitions of all the 91 metrics, which have not been previously defined formally.

Prediction of Reasoning Performance. Ontology reasoning tasks are hard decision problems that may go beyond NP-hard. For very expressive DLs, ontology reasoning has a very high worst-case complexity of 2NExpTime-complete [8]. For ontolgy reasoning optimisation, the research community have been interested in benchmarking of reasoner performance. In [15, 16], a number of modern reasoners were compared, and it was observed that the reasoners exhibit significantly different performance characteristics, thereby choosing an efficient reasoner for an ontology is a non-trivial task.

In a previous work [25], we developed classifiers to predict ontology classification performance categories for FaCT++, HermiT, Pellet and TrOWL, using ontology metrics as predictors [25]. The raw reasoning time is discretised into 5 increasingly large categories. High prediction accuracy of over 80% is achieved for all the 4 reasoners. Although highly accurate, the limitation of this work is that only the hardness category is predicted, not the actual reasoning time. To overcome this problem, we further investigated regression-based prediction models [26] to predict actual (or absolute) reasoning time of reasoners. In this approach, regression analysis was applied to estimate a numeric *response variable* (i.e. predicted reasoning time) from some *predictor variables* (i.e. 91 ontology metrics). These regression models were built on a small number of ontologies (i.e. 451) for 6 reasoners (FaCT++, HermiT, JFact, MORe, Pellet, TrOWL). These were implemented in R[2]. In this work, we improve upon these models by using a modern, carefully curated dataset of 1,920 ontologies from the ORE 2015 reasoner competition [22], an additional robust learning algrorithm XGBoost [28], and an updated list of reasoners that includes Konclude [12] and ELK [30] (for OWL 2 EL ontologies only) and excludes TrOWL [14] as it is an approximate thus incomplete reasoner.

Sazonau et al. [37] proposed a *local* approach to predicting OWL reasoner efficiency. Small subsets of a given ontology are repeatedly created, on which reasoning

[2]`https://www.r-project.org/`

is performed. Reasoning time data is then used to extrapolate a reasoner's discretised reasoning time on the whole ontology. Principal component analysis (PCA) was also employed to reduce the number of features (metrics). Evaluation conducted on 357 ontologies and 3 reasoners shows that the local prediction method performs as well as the *global* approach [25]. Moreover, they observed that the prediction model based on one feature (number of axioms) has comparable performance as that using a set of 57 features.

In a similar spirit, we investigated the prediction of reasoning time of ABox-intensive OWL 2 EL ontologies [38] and energy consumption of reasoning tasks on the Android platform [39].

Algorithm selection and meta-reasoner. Algorithm selection [40] is the problem of selecting a well-performing algorithm for a given problem instance. It has been successfully applied to machine learning, combinatorial optimisation and constraint satisfaction problems [41, 42]. SATzilla [43], for instance, a portfolio-based SAT solver, has demonstrated higher efficiency over single solvers. Compared to SAT, ontology languages are more expressive with the inclusion of many more language constructs. As a result, it is more challenging to accurately characterising ontology complexity.

AutoFolio [31] is a state-of-the-art, general-purpose algorithm selection system that performs automatic algorithm selection as well as hyper-parameter tuning. In this paper we use AutoFolio as a strong baseline to evaluate R_2O_2* in Section 7.2.3.

Chainsaw [44] first proposed the notion of a *metareasoner* for OWL ontologies. Given a *query* (i.e. reasoning task) on an ontology, Chainsaw constructs the smallest possible subset of the ontology while guaranteeing completeness of answering the query. This is achieved through the extraction of *locality-based modules* [45] using atomic decomposition [46]. The size of the extracted module is dependent on the reasoning task. For certain tasks such as consistency checking, the entire ontology needs to be extracted, hence not resulting in gains in efficiency. Also, given the potentially substantial overhead of computing modules, Chainsaw may not be competitive for simpler ontologies. As a prototype reasoner, Chainsaw uses FaCT++ version 1.5.3 as the delegate (i.e. component) reasoner. In the ORE 2015 ontology reasoner competition [47], CHAINSAW , CHAINSAW did not perform competitively against state-of-the-art reasoners: it was ranked 10/10 for the task of OWL DL classification and 11/13 for OWL EL classification.

WSReasoner [48] is a hybrid reasoner designed for large and complex ontologies in the description logic $\mathcal{ALCHOI}$. Given an ontology O, WSReasoner builds two approximate ontologies: a weakened version O_{wk} and a strengthened version O_{str}, both of which are in the less expressive (thus less complex) logic $\mathcal{ALCH}$. WSReasoner employs two component reasoners: a consequence-based reasoner that classifies both O_{wk} and O_{str}. As reasoning over O_{str} may not be sound, WSReasoner also employs a tableau-based reasoner to verify these results obtained on O_{str}. In its evaluation on a number of well-known hard ontologies including DOLCE, FMA and variants of Galen, WSReasoner outperforms tableau-based reasoners FaCT++, HermiT, Pellet and the approiimate, consequence-based reasoner TrOWL.

In our preliminary work [27], we proposed a meta-reasoner, R_2O_2, that makes use of regression-based prediction models of six OWL 2 DL reasoners (i.e. FaCT++, HermiT, JFact, Konclude, MORe, TrOWL). R_2O_2 takes two steps in the training phase. First, given training ontologies characterised by a set of metrics [26] and their reasoning time by the reasoners, R_2O_2 constructs a regression-based prediction model for each of the six reasoners. Second, given another set of training ontologies, a *ranking matrix* is generated using the prediction models. In the ranking matrix, each row represents the values of the ontology metrics and a ranking of the reasoners according to their *predicted* reasoning time. Several rankers were trained on this ranking matrix to learn how ontology metrics can be mapped to a relative ordering by the *predicted* performance of the reasoners. In the actual reasoning (testing) phase, given an unknown ontology, R_2O_2 makes performance predictions for the reasoners. It then ranks the reasoners according to their predicted reasoning time. The rankings recommended by the trained rankers are averaged to determine a unique rank of each reasoner. The highest ranked reasoner is chosen to perform the reasoning task for the unknown ontology. The evaluation on R_2O_2 [27] shows that R_2O_2 outperforms all of the six state-of-the-art OWL 2 DL reasoners, including Konclude [12], the most efficient OWL 2 DL reasoner.

As a baseline model to evaluate R_2O_2 [27], we also constructed a portfolio-based

OWL reasoner PR, which always selects the most efficient reasoner for any given ontology according to predicted reasoning time of all component reasoners.

R_2O_2 is different from PR in the following way. Instead of choosing the best reasoner according to predicted reasoning time of reasoners, as in PR, R_2O_2 selects a best possible reasoner from an aggregation of the *rankings* of component reasoners.

Recently a multi-criteria meta-reasoner Multi-RakSOR [49, 50] has been proposed for reasoning about OWL 2 DL and EL ontologies. Multi-RakSOR incorporates two objectives in selecting the best reasoner: reasoning efficiency and robustness. Efficiency is measured by execution time. The robustness of a reasoner is measured by four ordered *termination states*: (1) success ($\mathcal{B}_S$); (2) unexpected ($\mathcal{B}_U$), where the reasoning result is not expected; (3) timeout ($\mathcal{B}_T$), where the reasoner times out on an ontology; and (4) halt ($\mathcal{B}_H$), where the reasoner crashes. The ordering on these states is then defined to be $\mathcal{B}_S \prec \mathcal{B}_U \prec \mathcal{B}_T \prec \mathcal{B}_H$.

Multi-RakSOR encompasses two main components: (1) a multi-label classifier that predicts the termination state of a reasoner, and (2) a multi-target regression model that predicts the ranking of the reasoners (with tie breaking) that respects the above ordering.

A comprehensive evaluation of Multi-RakSOR is performed on the ORE 2015 reasoner competition ontology dataset[3], which we also use for evaluating R_2O_2*, and a set of 10 OWL 2 DL/EL reasoners. The paper also describes an "upgraded" version of Multi-RakSOR, dubbed Meta-RakSOR, that is able to handle both OWL 2 DL (more expressive) and OWL 2 EL (less expressive) ontologies. Meta-RakSOR is evaluated on the task of ontology classification on two datasets: one for OWL 2 DL and for OWL 2 EL. The main evaluation results show that for both datasets, Meta-RakSOR has the highest number of ontologies successfully reasoned over. In terms of efficiency, for OWL 2 DL, Meta-RakSOR demonstrates competitive performance (but not better) than the best single reasoner Konclude. For OWL 2 EL, it is shown that Meta-RakSOR ranks 6^{th} of the eleven reasoners evaluated, in terms of average reasoning time.

The meta-reasoners/hybrid reasoners described so far are all focussed on TBox reasoning (consistency checking or classification). PAGOdA [51] is a hybrid system

[3]`https://zenodo.org/record/50737`

designed for the task of *query answering* over ABox data. Employing an approach similar to WSReasoner [48], PAGOdA uses an efficient reasoner, in this case a Datalog reasoner, to compute a lower bound answer (sound but possibly incomplete) and an upper bound answer (complete but possibly unsound). When the two answers do not completely coincide, PAGOdA extracts relevant subsets from the TBox and the ABox are extracted, which are used to verify the answers by a fully-fledged OWL 2 DL reasoner.

4. Ontology Metrics

Metrics have been proposed to quantitatively measure the quality, complexity, testability, and maintainability of ontologies. Inspired by software metrics [52], we proposed a set of *91 metrics* [24–26] for characterising the *design complexity* of ontologies. However, the definition of many of these metrics were not formally given. In this section, we give a detailed account of this suite of 91 metrics that comprehensively characterise ontologies in terms of their size and syntactic and structural complexity. These metrics serve as distinctive features for learning ontology reasoning prediction models and building the proposed meta-reasoning framework R_2O_2*.

These metrics are organised by what they characterise: (1) *the ontology itself*, (2) *classes*, (3) *anonymous class expressions*, and (4) *properties*. These metrics are proposed with efficient computation as a key consideration. In the calculation of metrics, we adopt a graph-based view of ontologies [24] to capture the complexity of ontologies and generate a set of metrics.

The subsequent subsections present details of the metrics. Note that a metric name without the percent sign (%) is a *count*, and one with it is a *ratio*. A count metric shows how a component of an ontology has impact on the reasoning performance by its occurrences. A *ratio metric* is used to explain the relationship between such a component with respect to the overall structure of the ontology. Intuitively, a count/ratio metric represents the absolute/relative value of a metric of an ontology, respectively. We note that all metrics are computed on the *asserted* ontology hierarchy. In other words, no reasoning is performed prior to computing these ontologies.

4.1. Ontology-level Metrics (ONT)

The 6 ONT metrics were previously defined [24]. Here, we define an additional 18 ONT metrics that have not been described previously. They measure the overall size and complexity of an ontology.

IND counts the number of (named or anonymous) *individuals* in an ontology. The remaining 17 metrics are collected by observing the structure (i.e. language constructs) of a given ontology.

GCI/HGCI: These metrics measure the number of general concept inclusion (GCI) axioms and hidden GCI (HGCI) axioms, respectively. GCI counts the number of subsumption axioms whose subclass is a complex concept (anonymous class expression). HGCI counts the number of (named) concepts that appear as a subclass in some subsumption axioms as well as in some equivalent classes axioms. In general, the presence of GCI axioms may increase reasoning complexity as they may introduce nondeterminism [53]. A GCI axiom is *hidden* when a named class is the LHS of a subclass axiom as well as an equivalent class axiom.

Either an equivalent class axiom or a subclass axiom where the left-hand side of the subclass axiom is a named class.

ESUB%/DSUB%/CSUB%: These metrics measure ratios of subclass axioms that contain (possibly nested) specific types of class expressions, including those that are nested, and all subclass axioms. For these metrics, the subclasses and super classes are *flattened*, and an axiom is considered to contain a specific type of expressions iff one of the flattened expressions is of that type.

Respectively, ESUB%, DSUB% and CSUB% calculate the ratio of subclass axioms that contain existential restrictions ($\exists R.\_\_$), disjunctions ($\_\_ \sqcup \_\_$), and conjunctions ($\_\_ \sqcap \_\_$). Additionally CSUB% requires that at least one of the conjuncts in the subclass is an anonymous class expression. Existential restrictions (ESUB%) and disjunctions (DSUB%) could generate AND-branching and OR-branching, respectively, during the reasoning process. AND- and OR-branching are major sources of complexity for tableau-based algorithms [54], hence their presence may negatively correlate with performance. For the CSUB% metric, anonymous conjuncts in the subclass can generate more axioms during the normalization process, hence it may increase workload for a reasoner.

ELCLS%/ELAX%: These two metrics measure the ratios of (nested) class expressions (ELCLS%) and axioms (ELAX%), respectively, in the OWL 2 EL profile, a sublanguage of OWL that is based on $\mathcal{EL}$ [55], a description logic with efficient PTime-complete algorithms. Our intuition is that as reasoning in the EL profile is in general easier, a reasoner such as MORe [56] that is able to delegate EL reasoning to an efficient EL reasoner could be more efficient with ontologies with a large percentage of EL expressions and axioms.

HLC/HLC%: These metrics are the count (HLC) and ratio (HLC%) of *hard* language constructs, which include disjunctions, transitive properties, inverse properties and property chains, in superclass expressions. These language constructs can potentially negatively influence the performance of ontology reasoners.

SUBCECHN/SUPCECHN: These metrics are the count of top-level class expressions containing *chained* (a sequence of) existential restrictions (i.e., $\exists R.C$ in the DL syntax) as a subclass (SUBCECHN) or a superclass (SUPCECHN). These metrics measure the impact of $\exists R.C$ expressions on the performance of reasoning as they can potentially slow down the reasoning process by increasing the search space.

For example, suppose an ontology contains two subsumption axioms: (1) $\exists R.(A \sqcap \exists R.(B \sqcap \exists P.C)) \sqsubseteq E$, and (2) $\exists R.(D \sqcap \exists R.(F \sqcap \exists P.G)) \sqsubseteq H$, where R, P represent properties and $A, \ldots, H$ represent classes. For this ontology, SUBCECHN = 2 because axiom (1)'s subclass is a chained class expression containing existential restriction $\exists R.(A \sqcap \exists R.(B \sqcap \exists P.C))$ and axiom (2)'s subclass is also a chained existential restriction $\exists R.(D \sqcap \exists R.(F \sqcap \exists P.G))$. On the other hand, there is no chained class expressions containing existential restrictions as the superclass hence SUPCECHN = 0.

DSUBCECHN/DSUPCECHN: These count metrics calculate, in a depth-first manner, the maximum depth of nested class expressions containing existential restrictions as a subclass (DSUBCECHN) or a superclass (DSUPCECHN), with the intuition that deeper subclass chains may increase reasoning time. For example, suppose an ontology contains $\exists R.(A \sqcap \exists R.(B \sqcap \exists P.C)) \sqsubseteq E$, $D \sqcap \exists R.(F \sqcap \exists P.G) \sqsubseteq H$, where R, P represent properties and A, B, C, D, E, F, G, H represent classes. DSUBCECHN is 3 because the depth of nested class expressions containing existential restrictions in the first axiom $\exists R.(A \sqcap \exists R.(B \sqcap \exists P.C)) \sqsubseteq E$ is 3 and that of the second axiom $D \sqcap \exists R.(F \sqcap \exists P.G) \sqsubseteq H$ is 2. In addition, there is no nested class expression containing existential restrictions in the superclass, hence DSUPCECHN is 0.

SUBCCHN/SUPCCHN: These metrics represent the number of class expressions containing *chained* conjunction expression as a subclass/superclass. For tableau-based algorithms, conjunctions of complex concepts in a subclass may not be easily normalised for some reasoners. Hence a subclass expression containing many complex class expressions may slow down the reasoning process.

DSUBCCHN/DSUPDCHN: These metrics represent maximum depth of nesting of class expressions containing disjunction expressions as a subclass/superclass. The idea of these metrics is similar to the metrics DSUBCECHN/DSUPCECHN.

In total there are 24 ONT metrics, which are summarised in Table A.12 in the appendix.

4.2. Class-level Metrics (CLS)

The CLS metrics capture characteristics of classes, which are first-class citizens in OWL ontologies. Five functions, NOC (number of children), NOP (number of parents), DIT (depth of inheritance tree), CID (class in-degree), and COD (class out-degree), were defined previously [24]. Each of these functions returns, for a (named or possibly nested anonymous) class expression in an ontology, a count value respectively. For a given class C, NOC(C) and NOP(C) return the number of direct subclasses and super classes of C in the ontology, respectively. DIT(C) returns the longest path from C to $\top$, the root class, in a depth-first manner. CID(C) and COD(C) calculate, respectively, the number of incoming and outgoing edges of C.

For each of these five functions, we identify three metrics: the *total*, the *average*, and the *maximum* values across all classes for a given ontology. For example for NOC, the total NOC (tNOC) is calculated by summing the NOC value for all classes in an ontology, and the average NOC (aNOC) is tNOC divided by the total number of class expressions. Similarly, the maximum NOC (mNOC) is the maximum number of children among all classes.

Thus, in total, 15 CLS metrics are identified, which are shown in Table A.13 in the appendix.

4.3. Anonymous Class Expression Metrics (ACE)

The ACE metrics are an important ingredient in building expressive classes. Different types of anonymous class expressions can have different impact on reasoning performance. The 9 ACE count metrics have been previously defined [25], one for each different type of (possibly nested) anonymous class expressions (enumerations, negations, conjunctions, disjunctions, universal restrictions, existential restrictions, and min/max/exact cardinality restrictions). We further define two additional ACE count metrics that represent the number of *value restrictions* (VALUE, for $\exists R.\{a\}$, where a is an individual) and *self references* (SELF, for $\exists R.\mathsf{self}$). We also propose their corresponding *ratio* metrics that measure the percentage of each count ACE metric over all (possibly nested) anonymous class expressions.

Hence in total there are 22 ACE metrics, shown in Table A.14 in the appendix.

4.4. Property Metrics (PRO)

Additional pairs of count and ratio metrics are defined: ASYM/ASYM% (asymmetric properties), REFLE/REFLE% (reflective properties), IRREF/IRREF% (irreflective properties), and CHN/CHN% (property chains).

Similarly, the 6 of the 8 existing count PRO metrics [25] are augmented with their corresponding ratio metrics. For example, the DTP metric counts the number of datatype properties. The metric DTP% records the ratio between the number of datatype properties and the total number of properties. These 6 are: OBP (object properties), DTP (datatype properties), FUN (functional properties), SYM (symmetric properties), TRN (transitive properties), and IFUN (inverse functional properties). Furthermore, four count metrics are defined to record the number of some property axioms, including SUBP (subproperties), DISP (disjoint properties), DOMN (domain), and RANG (range).

Finally, four additional metrics are defined to measure the *usage* of properties in an ontology.

- ELPROP%: This metric measures the ratio, of all property axioms, the number of property axioms allowed in the OWL 2 EL profile, which include subproperty axioms, equivalent property axioms, transitive axioms, reflexive axioms, domain/range axioms, and functional data property axioms. Intuitively, the higher the ELPROP% of an ontology is, the more efficient its reasoning may be.
- IHR, IIR, ITR: These metrics measure the count of class axioms (e.g., subclass axioms and class/property assertions) that make use of some property in some property hierarchy (IHR), inverse properties (IIR), and transitive properties (ITR). The intuition is that the more these types of properties are used in class axioms, the more difficult reasoning may be for this ontology.

There are in total 30 PRO properties as summarised in Table A.15.

5. Meta-Reasoning Models

In this section, as our major contributions of this paper, we propose our meta-reasoning framework R_2O_2* and its four different *meta-reasoning models* (simply *meta-reasoners*). Each meta-reasoner recommends the most efficient reasoner for unknown ontologies using different machine learning techniques. We first introduce basic notations we use in the paper. Then, we present the details of the four meta-reasoners with their learning objectives in the training phase and their utilisation in the recommendation (or testing) phase.

5.1. Notation Definition

The following basic notations are used in the paper.

- Let $R = \{r_1, ..., r_n\}$ be a set of n reasoners, also called *component reasoners* in the paper.
- Let $\hat{R} = \{\hat{r}_1, ..., \hat{r}_n\}$ be a set of *n prediction models* such that $\hat{r}_i$ predicts the reasoning time of r_i.
- Let $OM = \{om_1, ..., om_q\}$ be a set of q ontology metrics.
- Let $O = \{o_1, ..., o_h\}$ be a set of h ontologies that can be reasoned about by at least one reasoner in R without timing out or errors. Each ontology in O is represented using its values of ontology metrics OM in this paper.

- Given a reasoner r and an ontology o, let $\theta(r, o)$ represent the actual reasoning time of r for o for the task of ontology classification. Similarly, let $\theta(\hat{r}, o)$ represent the reasoning time predicted by $\hat{r}$ for o for the same task.
- Two partitioned subsets O_{tr} and O_{te} are drawn from O for training and testing the proposed meta-reasoners, respectively.

5.2. Details of the Meta-Reasoners

In the following, we present the details of each of our four meta-reasoners. For each reasoner, we describe its learning objective and how to build it in detail. All these meta-reasoners are built on the training dataset $O_{tr} \subset O$.

The first meta-reasoner, $\mathrm{R_2O_2*_{(pt)}}$, directly trains prediction models $\hat{R}$ of R on the training data O_{tr}, and uses the predicted reasoning time that has been estimated by $\hat{R}$ to find the most efficient reasoners for unknown ontologies. The underlying idea is to choose a reasoner r, whose predicted reasoning time estimated by $\hat{r}$ is the most efficient among $\hat{R}$, as the most efficient reasoner for an unknown ontology. The second meta-reasoner, $\mathrm{R_2O_2*_{(rk)}}$, trains a ranking algorithm to learn the rankings of the reasoners in R according to the actual reasoning time of the training data O_{tr}, and uses it to predict the best ranked reasoners for unknown ontologies. The third meta-reasoner, $\mathrm{R_2O_2*_{(mc)}}$, trains a classifier that learns the most efficient reasoner on O_{tr}, and uses it to directly predict the most efficient reasoners for unknown ontologies. The forth meta-reasoner, $\mathrm{R_2O_2*_{(all)}}$, is an ensemble classifier that uses the predictions of the above three meta-reasoners.

5.2.1. Meta-reasoner based on the direct use of predicted reasoning time: $\mathrm{R_2O_2*_{(pt)}}$

The meta-reasoner $\mathrm{R_2O_2*_{(pt)}}$ aims to recommend the most efficient reasoners for unknown ontologies based on the *direct use of the predicted reasoning time* of all reasoners in R. Therefore, for all reasoners in R, building their corresponding prediction models $\hat{R}$ is essential prior to making use of $\mathrm{R_2O_2*_{(pt)}}$ for determining such most efficient reasoners. $\mathrm{R_2O_2*_{(pt)}}$ is similar to the non-ranking portfolio reasoner PR described in our preliminary work [27] in that it leverages predicted reasoning time of $\hat{R}$. The difference is that $\mathrm{R_2O_2*_{(pt)}}$ uses an ensemble regression model instead of using a random forest regression algorithm as PR [27]. Also, $R_2O_2*_{(pt)}$ incorporates the average rankings of the reasoners on the training data when there are more than two reasoners that were chosen as the most efficient (i.e. their predicted reasoning time is the same), while PR chooses one in random. The details of resolving a tie-breaking method $R_2O_2*_{(pt)}$ is explained below.

Consequently, the effectiveness of $R_2O_2*_{(pt)}$ relies mainly on the accuracy of the prediction models in $\hat{R}$. In order to build such prediction models, we use *stacking* [57], an ensemble learning technique to combine multiple classification (or regression) models in which (1) *base learners (or regression models)* (or level-0 models) are trained on the training data O_{tr}, and (2) the outputs of the base learners are combined using a *meta-learner (or meta-regression models)* (level-1 model), in our context. More specifically, for each reasoner, each learner is trained to learn a mapping function from the values of ontology metrics on the training data O_{tr} to their actual reasoning time. Then, a meta-learner is trained to learn a mapping function from the predicted outputs of O_{tr}, which have been estimated by the base learners, to actual reasoning time.

Here, our aim is to use the decisions of the individual base learners that employ different learning criteria, and to combine their decisions to outperform each individual base learner using a meta-learner.

More formally, to build each prediction model $\hat{r}_k \in \hat{R}$ for reasoner $r_k \in R$, we represent each ontology $o_i \in O_{tr}$ as follows:

$$o_i = \underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}, \quad \underbrace{\theta(r_k, o_i)}_{\text{actual reasoning time}} \tag{5}$$

where $om_{i,j}$ is the value of the j-th ontology metric om_j of ontology o_i, and $\theta(r_k, o_i)$ denotes the actual reasoning time of r_k on ontology o_i.

Using the above representation scheme, for each reasoner, we train k base learners (level-0 models) on O_{tr}. Then, we generate level-1 data obtained from the predictions of the k base learner over the instances in O_{tr}. The level-1 data have k attributes whose values are the predictions (i.e. predicted time) of the k base learners for every instance in O_{tr}. Thus, each training example for a meta-learner (level-1 model) will be composed of k attributes (e.g. k predictions from the k base learners) and the target which is the actual reasoning time for every instance in O_{tr}. Once the level-1 data have been built from all instances in O_{tr}, any learning regression models can be used to generate the meta-learner. In this paper we choose $k = 2$.

In this paper, we use two robust base learners: (1) the *random forest regression algorithm* and (2) the *XGBoost (eXtreme Gradient Boosting)* algorithm [28]:

- **Random forest regression algorithm:** As a base learner, we build a regression model using *random forest regression algorithm*, an efficient and robust learning model, which has produced good predictive performance in our previous work [26]. In our context, the random forest model combines a number of *decision trees*, each of which is trained using a subset of training ontologies, to build a prediction model for a given reasoner.
- **XGBoost:** As another base learner, we use the state-of-the-art learning algorithm, XGBoost [28], which has recently shown dominant performance on a number of Kaggle competitions for structured or tabular data. It is an implementation of gradient boosted decision trees designed for achieving better computational efficiency and prediction performance.

We again consider random forest and XGBoost as candidates of a meta-learner (level-1 model) due to their strong predictive performance. These level-1 candidates are denoted by meta-RF and meta-XGBoost in this paper. As can be seen in Section 7.1.1, we eventually choose meta-XGBoost as our meta-learner given its best overall performance.

Once we train the meta-regression model on O_{tr}, given an unknown ontology in the testing data O_{te}, the meta-reasoner $R_2O_2*_{(pt)}$ will recommend a reasoner whose predicted reasoning time is the fastest among all of the predictions of the prediction models in $\hat{R}$ as the most efficient reasoner.

If more than two reasoners are chosen as the most efficient, a tie-breaking method is also applied to select one of them. This method takes into consideration the *precision at 1* (P@1) of the reasoners in R that are measured on the training data O_{tr}. In this context, for each reasoner in R, its P@1 is measured by the proportion that the reasoner is the most efficient across all instances in O_{tr}. Our tie-breaking method chooses the reasoner with the highest P@1. This tie-breaking method is applied to all the other meta-reasoners in our R_2O_2*.

5.2.2. *Meta-reasoner based on ranking algorithm:* $R_2O_2*_{(rk)}$

$R_2O_2*_{(rk)}$ is a meta-reasoner that learns the rankings of the reasoners in R. During the training phase, it trains a *ranking algorithm* (simply *ranker*) that learns the rankings of the reasoners on O_{tr} in terms of their reasoning time. Once the ranker is trained, given an unknown ontology in O_{te}, $R_2O_2*_{(rk)}$ ranks and recommends the most efficient reasoner for that ontology.

$R_2O_2*_{(rk)}$ follows a similar spirit of R_2O_2 [27] in that $R_2O_2*_{(rk)}$ uses a *ranking matrix* and uses a ranker. The difference is that $R_2O_2*_{(rk)}$ uses a single ranker, rather than aggregating multiple rankers as R_2O_2, to recommend the most efficient reasoner for an unknown ontology. Given a pool of rankers using different criteria for learning, we have chosen the one with the best ranking performance through our experiments which will be further discussed in Section 7.1.1.

Given the training data O_{tr}, we generate a *ranking matrix*, where each row represents the values of ontology metrics and the rankings of reasoners R according to their actual reasoning time. Then, a ranker is trained on this matrix to learn how the characteristics of the ontologies in O_{tr} can be optimally mapped to the relative ordering of the reasoning performance of the reasoners in R. Initially, we build an $|O_{tr}| \times (q + n)$ data matrix $\mathbf{M_d}$ (recall that $q = |OM|$, $n = |R| = |\hat{R}|$), where row i represents an ontology $o_i \in O_{tr}$ and the actual reasoning time of the reasoners in R for o_i:

$$o_i = \underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}, \underbrace{\theta(r_1, o_i), \ldots, \theta(r_n, o_i)}_{\text{actual reasoning time}}, \tag{6}$$

where $om_{i,j}$ is the value of the j-th ontology metric om_j of o_i, and $\theta(r_s, o_i)$ denotes r_s's actual reasoning time for o_i. From $\mathbf{M_d}$, we build the corresponding $|O_{tr}| \times (q+n)$ ranking matrix $\mathbf{M_r}$, where row i is represented as:

$$o_i = \underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}, \underbrace{\pi(r_1, o_i), \ldots, \pi(r_n, o_i)}_{\text{ranking of reasoners}}, \tag{7}$$

where $\pi(r_s, o_i)$ denotes the *rank* of $r_{s_{[1..n]}} \in R$ for o_i determined by $\theta(r_s, o_i)$. On $\mathbf{M_r}$, the more efficient a reasoner is, the higher ranked it is (the smaller the rank number). To illustrate this, suppose there are 3 reasoners $\{r_1, r_2, r_3\}$, and their actual reasoning time for an ontology o_i is 100s, 90s, and 10s, respectively, i.e., $(\theta(r_1, o_i), \theta(r_2, o_i), \theta(r_3, o_i))$ = (100s, 90s, 10s). Thus, the ranking is $(\pi(r_1, o_i), \pi(r_2, o_i), \pi(r_3, o_i)) = (3, 2, 1)$. If the reasoning time is (10s,10s,100s) instead, the ranking will be $(1, 1, 3)$.

In summary, the goal is to learn rankings of all reasoners in R on the ranking matrix $\mathbf{M_r}$, and to predict a ranking of the reasoners for an unseen ontology. The top-ranked reasoner will be chosen by the meta-reasoner $R_2O_2*_{(rk)}$ to be the most efficient reasoner to reason over the ontology. Comparing to $R_2O_2*_{(pt)}$, the main feature of $R_2O_2*_{(rk)}$ stems from that it is built on the ranking matrix that uses rankings of the reasoners in R, rather than the direct use of the predicted reasoning time estimated from $\hat{R}$.

5.2.3. *Meta-reasoner based on multi-class classification:* $R_2O_2*_{(mc)}$

$R_2O_2*_{(mc)}$ formulates the learning problem into a multi-class classification problem, where its goal is to classify an ontology with one of the reasoners that is able to reason about the ontology the most efficiently. During the training phase, $R_2O_2*_{(mc)}$ learns the most efficient reasoner for each ontology on the training data O_{tr}. The most efficient reasoner is determined by means of the actual reasoning time of the reasoners in R, meaning that the fastest reasoner is chosen as the most efficient reasoner.

More formally, to build $R_2O_2*_{(mc)}$, we represent each ontology $o_i \in O_{tr}$ as follows:

$$o_i = \underbrace{om_{i,1}, \ldots, om_{i,q}}_{\text{ontology metrics}}, \quad \underbrace{\gamma_i}_{\text{the most efficient reasoner}} \tag{8}$$

where $om_{i,j}$ is the value of the j-th ontology metric om_j of o_i, and γ_i denotes the actually most efficient reasoner for o_i. If there is an ontology $o_i \in O_{tr}$ that has k-reasoners (where $k > 1$) that show the equivalently most efficient reasoning time, we generate k instances with k most efficient reasoners for o_i. For example, given on ontology o_i, suppose that there are two most efficient reasoners: Konclude and Pellet. We then generate two instances for o_i as follows: (1): "$om_{i,1}, \ldots, om_{i,q}$, Konclude", and (2) "$om_{i,1}, \ldots, om_{i,q}$, Pellet".

Using the above representation scheme, we train a classifier on the training data O_{tr}. In our experiments, we have considered two classifiers: (1) *random forest algorithm* and (2) *XGBoost*, because of their robust classification performance as in the case of the meta-reasoner $R_2O_2*_{(pt)}$. Based on our cross-validation on O_{tr}, we eventually choose XGBoost which will be further discussed in Section 7.1.1.

In comparison with the meta-reasoner $R_2O_2*_{(pt)}$, $R_2O_2*_{(mc)}$ does not directly use the predicted reasoning time of the reasoners in R, that is, it does not rely on $\hat{R}$. Rather it learns which reasoners have been the most efficient reasoners for ontologies in the training data O_{tr}. The learning goal of $R_2O_2*_{(mc)}$ is similar to the meta-reasoner $R_2O_2*_{(rk)}$ in that its learning is based on the ranks of the reasoners in R on the training data O_{tr}. However, $R_2O_2*_{(mc)}$ differs in that it learns the most efficient reasoner only, not the rankings of all reasoners in R as $R_2O_2*_{(rk)}$.

5.2.4. *Ensemble meta-reasoner:* $R_2O_2*_{(all)}$

$R_2O_2*_{(all)}$ is a stacking classifier that learns from the predictions of the above three meta-reasoners: (1) $R_2O_2*_{(pt)}$ that directly uses the predicted reasoning time of reasoners for the training set O_{tr}, (2) $R_2O_2*_{(rk)}$ that learns the rankings of reasoners by means of their actual reasoning time for O_{tr}, and (3) $R_2O_2*_{(mc)}$ that learns the most efficient reasoners for ontologies in O_{tr}. In other words, in $R_2O_2*_{(all)}$, those three meta-reasoners can be seen as base classifiers (i.e. level-0 models), and $R_2O_2*_{(all)}$ learns a meta-classifier (i.e. level-1 model) from the predictions of the base classifiers.

Thus, $R_2O_2*_{(all)}$ trains a meta-classifier on the training data O_{tr}, where each ontology $o_i \in O_{tr}$ is represented as follows:

$$o_i = \underbrace{R_2O_2*_{(pt)}(o_i), R_2O_2*_{(rk)}(o_i), R_2O_2*_{(mc)}(o_i)}_{\text{predicted reasoners of the meta-reasoners}}, \qquad \underbrace{\gamma_i}_{\text{the most efficient reasoner}} \tag{9}$$

where $R_2O_2*_{(pt)}$ (o_i), $R_2O_2*_{(rk)}$ (o_i) and $R_2O_2*_{(mc)}$ (o_i) are the predicted most efficient reasoners of $R_2O_2*_{(pt)}$, $R_2O_2*_{(rk)}$, $R_2O_2*_{(mc)}$, respectively, for o_i. As in Equation 8, γ_i denotes the actually most efficient reasoner for o_i.

As candidates for a meta-classifier, we also consider the same learning algorithms used to built the meta-reasoner $R_2O_2*_{(pt)}$ because of the same reason: their proven, robust predictive performance: (1) random forest algorithm and (2) XGBoost. The one with the better performance from these two candidates is chosen, where the performance is measured via cross-validation on the training data O_{tr}. Eventually, XGBoost is chosen as our meta-classifier which will be presented in Section 7.1.1.

Thus, $R_2O_2*_{(all)}$ trains the meta-classifier that learns relationships between the predicted reasoners of the previous three meta-reasoners and the actually most efficient reasoners on the training data O_{tr}, and then make predictions about the most efficient reasoners for unknown reasoners for the testing ontologies O_{te}.

5.2.5. Meta-reasoners with ELK

In the previous sections, we have presented the four different meta-reasoners in R_2O_2*. Note that all of these meta-reasoners utilise a number of high-performance DL reasoners. Here, an interesting question is whether incorporating an EL reasoner into the meta-reasoners, such as ELK [30], can further improve reasoning efficiency for the less complex OWL 2 EL ontologies.

To address this question, we augment our meta-reasoners to incorporate an EL reasoner, ELK, that is designed to support the less expressive OWL 2 EL profile. ELK does not support reasoning over non-EL ontologies. Thus, the meta-reasoners only incorporate ELK when predicting the most efficient reasoners for unknown OWL 2 EL ontologies.

First, we train a prediction model for ELK on the training data O_{tr} following the steps presented in Section 5.2.1. Then, given an ontology $o \in O_{te}$ whose reasoning time is unknown, each meta-reasoner (representatively denoted by R_2O_2*) is extended by taking the following further steps to find the most efficient reasoner to reason about o:

1. Check whether o is an EL ontology or not.

2. If o is an EL ontology, compare the predicted reasoning time and the most efficient reasoner determined by R_2O_2*. Then, we choose the fastest one as the most efficient reasoner. Formally,

$$\hat{\gamma}(o) = \underset{\hat{r}_i \in R_2O_2*, \hat{r}_{elk}}{\operatorname{argmin}} \; \theta(\hat{r}_i, o) \tag{10}$$

where $\hat{\gamma}(o)$ is the most efficient reasoner for o determined eventually; $\theta(\hat{r}_i, o)$ is the predicted reasoning time of the reasoner r_i for o; and $\hat{r}_{elk}$ is the prediction model of ELK.

3. If o is not an EL ontology, follow the recommendation of R_2O_2* not considering r_{elk}.

Thus, in our extension for EL ontologies, for each meta-reasoner, the main idea is to simply incorporate predicted reasoning time of ELK and the most efficient reasoner determined by the meta-reasoner, to compare their reasoning time, and finally to recommend the one with the most efficient predicted reasoning time.

Here we summarise the main differences between our meta-reasoning framework R_2O_2* and our preliminary meta-reasoner R_2O_2 and the portofolio-based meta-reasoner PR, which were described in detail in Section 3.

1. As a framework (but not individual meta-reasoners), R_2O_2* adapts and incorporates both R_2O_2 (as $R_2O_2*_{(rk)}$) and PR (as $R_2O_2*_{(pt)}$).

2. R_2O_2* incorporates more advanced prediction models (Random Forests and XGBoost) through *ensembling*.

3. R_2O_2* generates a ranking matrix using the actual reasoning time of the component reasoners. On the other hand, R_2O_2 built a ranking matrix using the predicted reasoning time of the reasoners, where such time was estimated by prediction models.

4. R_2O_2* incorporates a single best ranker instead of using an aggregation of multiple rankers as R_2O_2. The single best ranker is determined empirically through cross-validation on the training data.

5. R_2O_2* invokes the efficient reasoner ELK [30] directly for OWL 2 EL ontologies.

6. R_2O_2* is built using a different mix of component reasoners that includes Pellet (which R_2O_2 does not include) but excludes TrOWL, as TrOWL is an approximate, therefore incomplete reasoner.

7. Finally, R_2O_2* is built and evaluated using a more modern set of ontologies and more recent versions of reasoners.

In the following sections, we discuss our evaluation framework and results that compare the performance of all the proposed meta-reasoners (with and without ELK) using the ORE 2015 competition corpus [22].

6. Evaluation Framework

In this section, we describe the evaluation framework used for evaluating the proposed prediction models and meta-reasoners, including details of the reasoners, ontologies and the evaluation environment. The notations used in this section follow those defined in Section 5.1.

Reasoners: Six state-of-the art OWL 2 DL reasoners that participated in ORE the 2015 reasoner competition [22] are used as component reasoners (simply reasoners) [4]: FaCT++ [10], HermiT [34], JFact,[5] Konclude [12], MORe [56] (with HermiT as the underlying OWL 2 DL reasoner), and Pellet [13]. Besides these six OWL DL reasoners, we also incorporate ELK [30], the efficient reasoner for the less expressive OWL EL profile. The versions of the reasoners are the same as those in ORE 2015. As described in Section 5, we build a prediction model for each reasoner, which is one of the key components in the meta-reasoner framework R_2O_2*.

Target Reasoning Task: For the ontology reasoning task, we choose ontology *classification*. The actual reasoning time (wall-time) of each ontology in the dataset was measured on a high-performance server running CentOS Linux 7.4 (Core) and Java 1.8 on single-core Intel Gold 6140 each at 2.3 GHz, with a maximum of 10 GB memory allocated to the reasoner. A timeout of *30 minutes* of wall-time is imposed on each (reasoner, ontology) pair.

To ensure consistency of the evaluation across reasoners, the ORE 2015 competition framework[6] is used to invoke all reasoners and record their reasoning time. For each ontology, the competition framework converts it into the OWL functional syntax (FSS),

[4]Chainsaw [44] and Racer [58] (two OWL 2 DL reasoners that participated in ORE 2015) are excluded due to reasoning errors in an excessive number of ontologies.

[5]`http://jfact.sourceforge.net`

[6]`https://github.com/andreas-steigmiller/ore-competition-framework`

invokes reasoners for classification using a Bash shell script, and records reasoning time using the GNU `time` command. We follow the framework and include ontology loading time as part of classification time.

Ontologies: We collected all 1,920 ontologies from the ORE 2015 reasoner competition [22] [7]. Prior to building our proposed meta-reasoners, we performed the three preprocessing steps on the 1,920 ontologies, following the steps in [26, 27]. The aim of these steps is to remove duplicate ontologies that may exist in the given ontology collection, normalise their metric values to avoid the high skewness of values of ontology metrics OM, and remove ontology metrics that may influence learning a prediction model with lower prediction accuracy:

1. Cleansing: Since the given ontology collection has been obtained from multiple repositories, it may contain duplicates. All but one ontology is removed from each set of ontologies with duplicate metric values. After removing duplicates, 1,760 ontologies remained.

2. Normalisation: In the given 1,760 ontology collection, values of some of the metrics span a large range and are very skewed as discussed in [26]. We apply a commonly-used log-transformation on the metric values of the ontologies that are greater than 10. The log-transformation is also performed on reasoning time.

3. Metric removal: It is a widely used practice to remove features that have near-zero variance values and features that are highly correlated (with respect to the dataset). In this paper we follow this practice. Following our previous work [26, 27], we consider two metrics with correlation coefficients above 0.9 to be highly correlated. To observe a better generalised distribution of these metrics, we measure their correlation on a larger ontology collection, the one used in the ORE 2014 reasoner competition [21]. It contains 16,555 ontologies, which are split into four groups by percentiles of file size. Ontologies are randomly sampled from within these groups. This is to ensure that files of different sizes are sufficiently represented.

[7] `http://owl.cs.manchester.ac.uk/publications/supporting-material/ore-2015-report/`

Given correlation calculated from this ontology collection, we remove all but 29 metrics: 12 ONT metrics (SOV, ENR, EOG, CYC, RCH, IND, ESUB%, ELCLS%, ELAX%, HLC, HLC%, SUPCECHN); 7 CLS metrics (tNOC, aNOC, aCID, mCID, tCOD, aCOD, aNOP); 4 ACE metrics (CONJ%, UF%, EF, EF%); and 6 PRO metrics (OBP%, DTP%, FUN%, CHN%, ELPROP%, IHR).

Finally, as our ontology collection (O), we used 1,760 ontologies, where each ontology is represented by the 29 metrics and the metric values are log-scaled.

Table 1 shows, for each reasoner, the total number of ontologies it successfully handled, that result in an error, and that time out, and brief statistics of reasoning time (in seconds, and excluding those ontologies that time out). It can be observed that the reasoning time spans a large range for all the reasoners, and that Konclude is the most efficient as well as most robust reasoner (the least number of error and timeout ontologies). In total, 1,269 ontologies (excluding timeout ontologies) were reasoned about successfully (no error, no timeout) by all the six reasoners, and 1,390 ontologies did not result in a runtime error (timeout ontologies are included in this case). Note that timeout ontologies are used to evaluate our meta-reasoners in one set of experiments. Figure 1 in Section 7 depicts the performance characteristics of the component reasoners in more details in violin plots. Of the 1,760 ontologies, 761 are in the OWL 2 EL profile. Hence, we performed classification on these ontologies using ELK.

Evaluation method: We evaluate our meta-reasoners with the state-of-the-art algorithm selection framework AutoFolio [31] that is configured to minimise runtime. As discussed earlier in Section 5, meta-reasoners PR and R_2O_2 described in our previous work [27] are similar to $R_2O_2*_{(pt)}$ and $R_2O_2*_{(rk)}$, respectively. Hence this comparison also assesses the performance of $R_2O_2*_{(all)}$ against our previous models. We also attempted to evaluate our meta-reasoners against a recently proposed multi-criteria meta-reasoner Meta-RakSOR [49, 50], which has dual optimisation objectives of reasoning correctness as well as efficiency. However, we are unable to evaluate Meta-RakSOR due to two reasons. Firstly, for the OWL 2 DL classification task, Meta-RakSOR incorporates eight component reasoners but our meta-reasoners only incorporates six. Among the two reasoners that are not considered by our meta-reasoners, TrOWL [14], which is

Table 1: A summary of statistics of the deduplicated ORE 2015 competition dataset containing a total of 1,760 unique ontologies. Note the reasoning time is measured in seconds and without considering timeout ontologies.

Reasoner	No. of Ontologies			Reasoning time (s)			
	Successful	Error	Timout	Min	Max	Median	Mean
FaCT++	1,461	109	190	0.53	1,638.6	1.4	72.4
HermiT	1,658	51	51	0.70	1,622.4	3.6	35.8
JFact	1,292	161	307	1.03	1,788.6	3.4	72.8
Konclude	1,737	8	15	0.03	1,087.2	0.3	3.7
MORe	1,706	18	36	2.00	1,684.5	4.5	87.1
Pellet	1,477	114	169	1.01	1,773.9	3.5	39.5

an approximate hence incomplete reasoner, and RACER [59], which did not execute properly on our evaluation hardware. Secondly, even though Meta-RakSOR has released source for *running* the meta-reasoner[8], it however does not include the source code of Meta-RakSOR itself. Therefore we are unable to modify it and compare with it. However, as can be seen from Table 1 of Meta-RakSOR [50], Meta-RakSOR does not outperform Konclude on average reasoning time, it is thus not unreasonable to hypothesise that our meta-reasoners would outperform Meta-RakSOR, as our meta-reasoners outperform Konclude.

In our evaluation we retain timeout ontologies to realistically assess performance of all reasoners. We assess the impact of those ontologies that result in a runtime error in two different experiments. In the first experiment (hereinafter referred as **ErrorsRemoved**), the error ontologies for each reasoner are removed. In the second experiment (hereinafter referred as **ErrorsReplaced**), the error ontologies for each reasoner are treated as they timeout. Note that in ErrorsRemoved, it may be the case that a reasoner that strictly conforms to OWL semantics may throw many runtime errors

[8] `https://github.com/Alaya2016/Multi-RakSORDemo`

on a corpus of ontologies as it may reject non-conformant constructs (e.g. imaginary numbers). As such, such a reasoner may turn out to be efficient in the experiment ErrorsRemoved than in ErrorsReplaced.

In each experiment, standard 10-fold cross validation is performed to adequately assess the performance of the meta-reasoners. That is, we take the following steps: (1) shuffle the ontologies O randomly; (2) split O into 10 subsets; (3) for each subset, take the subset (i.e 10%) as test set (O_{te}), and take the remaining subsets (i.e. 90%) as a training set (i.e. O_{tr}); (4) fit a model on the training set and evaluate it on the test set; and (5) average the evaluation scores of the model across 10-times.

As evaluation metrics, average runtime (i.e. reasoning time) is used as the main evaluation metric for the meta-reasoner. To evaluate our runtime prediction (regression) models, we used the standard 'coefficient of determination' (R^2) as used in [26]. R^2 denotes the proportion of the variation in the target variable (i.e. reasoning time) that can be explained by each prediction model $\hat{r} \in \hat{R}$. The higher the R^2 value is, the more accurate the model is. Moreover, we evaluate our meta-reasoners using the standard metric precision at 1 (P@1), as the meta-reasoners require the predicted best reasoner. The performance number reported in the rest of the section is the average on the test set over the 10 folds for each experiment.

In the experiment **ErrorsReplaced**, in each fold of the cross-validation, 90% of 1,760 ontologies ($\approx 1,584$) are used for training, and the rest 10% (≈ 176) are used for testing. In the experiment **ErrorsRemoved**, the 1,389 ontologies are randomly divided into the training set ($\approx 1,250$) and the test set (≈ 139) in each fold.

7. Evaluation Results and Analysis

In this section we present the evaluation results and their detailed analysis. The evaluation is conducted in two parts. In Section 7.1, we present the performance evaluation of the prediction models used in R_2O_2* on the training set O_{tr}. In Section 7.2, we present the overall evaluation results, obtained on the test set O_{te}, comparing R_2O_2* with component reasoners and AutoFolio.

7.1. *Performance Evaluation of the Key Learning Components in* R_2O_2*

Here, we present the performance evaluation of the prediction models (regression models, classifiers and rankers) used in our meta-reasoning framework R_2O_2*, obtained through 10-fold cross-validation on the training set O_{tr}.

7.1.1. *Performance of regression models in* $R_2O_2*_{(pt)}$

Here, our goal is to assess the generalizability of the prediction models $\hat{r}_i$ of reasoner $r_i \in R$. As described in Section 5.2.1, these prediction models are central to $R_2O_2*_{(pt)}$. Performance (i.e. generalizability) was measured in terms of the coefficient of determination (R^2), as introduced in Section 6.

As presented in Section 5.2.1, we use a stacking approach to build a prediction model with two base regression models (level-0 models): random forest regression algorithm and XGBoost [28]. As a meta-regression model (level-1 model), we also used these two algorithms.

We employed the widely-used Weka framework [9] as our evaluation environment. For ease of experimentation we chose Weka's version of the random forest algorithm. For XGboost, we used Weka-XGBoost [10] that can easily interface with Weka.

For the random forest algorithm, we use the default configuration in Weka. For XGBoost, we set the following parameters keeping all the others fixed as default throughout this paper: num_round = 50 (the number of rounds for boosting), eta = 0.1 (learning (or shrinkage) parameter that controls how much information from a new tree will be used in the Boosting), max_depth = 10 (controls the maximum depth of the trees: deeper trees have more terminal nodes and fit more data), sub_sample = 0.5 (determines if we are estimating a Boosting or a Stochastic Boosting. A value 1 represents the regular boosting, and a value between 0 and 1 is for the stochastic case. The stochastic Boosting uses only a fraction of the data to grow each tree. For example, if we use 0.5 each tree will sample 50% of the data to grow). Note that these parameter values were chosen empirically. XGBoost uses multiple parameters and determining

[9] `https://www.cs.waikato.ac.nz/ml/weka/`.

[10] `https://github.com/SigDelta/weka-xgboost`.

optimal parameter values is beyond the scope of this paper.

Tables 2 and 3 show the R^2 values of the 7 prediction models obtained from cross-validation on the training data O_{tr} in the two experiments: **ErrorsRemoved** and **ErrorsReplaced**. Note that we have compared the prediction performance of 4 regression models as candidates as a prediction model for each reasoner: (1) random forest regression algorithm (denoted by RF), (2) XGBoost, (3) a stacking meta-regression model using random forest regression algorithm (denoted by meta-RF), and (4) a stacking meta-regression model using XGBoost (denoted by meta-XGBoost). R^2 denotes the proportion of the variation in the target variable (i.e. reasoning time) that can be explained by the model. For example, 0.853 in the model $\hat{r}_{\text{FaCT++}}$, implemented by meta-XGBoost, indicates that 85.3% of the variation in the reasoning time can be accounted for by meta-XGBoost. In Table 2, both RF and meta-XGBoost show the best prediction performance whereas in Table 3, meta-XGBoost has the highest R^2 value. Thus, we choose meta-XGBoost to implement the meta-reasoner $R_2O_2*_{(pt)}$.

Table 2: A summary of prediction model performance as measured by R^2 on the dataset **ErrorsRemoved**.

Model	RF	XGBoost	meta-RF	meta-XGBoost
$\hat{r}_{\text{FaCT++}}$	0.852	0.852	0.852	0.853
$\hat{r}_{\text{HermiT}}$	0.825	0.824	0.831	0.833
$\hat{r}_{\text{JFact}}$	0.904	0.903	0.906	0.907
$\hat{r}_{\text{Konclude}}$	0.910	0.905	0.909	0.909
$\hat{r}_{\text{MORe}}$	0.723	0.705	0.707	0.708
$\hat{r}_{\text{Pellet}}$	0.770	0.763	0.765	0.766
$\hat{r}_{\text{Elk}}$	0.948	0.947	0.949	0.950
Mean	**0.847**	0.843	0.846	**0.847**

The above observations provide insight into how well the 7 prediction models can fit the given data. The similar values of R^2 with the ones in [26] (i.e. averaged R^2 = 0.869) suggest a good generalizability of the models.

Table 3: A summary of prediction model performance as measured by R^2 on the dataset **ErrorsReplaced**.

Model	RF	XGBoost	meta-RF	meta-XGBoost
$\hat{r}_{\text{FaCT++}}$	0.835	0.832	0.834	0.835
$\hat{r}_{\text{HermiT}}$	0.807	0.807	0.812	0.814
$\hat{r}_{\text{JFact}}$	0.843	0.833	0.839	0.840
$\hat{r}_{\text{Konclude}}$	0.909	0.909	0.912	0.913
$\hat{r}_{\text{MORe}}$	0.757	0.744	0.750	0.756
$\hat{r}_{\text{Pellet}}$	0.727	0.724	0.719	0.723
$\hat{r}_{\text{Elk}}$	0.939	0.936	0.939	0.940
Mean	0.831	0.826	0.829	**0.832**

7.1.2. Performance of rankers in $R_2O_2{*}_{(rk)}$

As explained in Section 5.2.2, our meta-reasoner, $R_2O_2{*}_{(rk)}$, incorporates a single ranker which differs from our previous approach in R_2O_2 [27] that uses an aggregation of multiple rankers. To implement $R_2O_2{*}_{(rk)}$, we initially considered the 5 rankers[11] that are used in R_2O_2 [27] on the ranking matrix $\mathbf{M_r}$: (1) *KNNRanker* that aggregates rankings using the nearest neighbors, (2) *PCTRanker* that is based on predictive clustering tree for ranking, (3) *RPCRanker* that is based on a multiple binary pairwise classifier to construct a ranking model, (4) *RegRanker* where the ranking problem is cast to a multi-target regression problem, where the rank position values of each reasoner are the targets in the multi-target regression setting, and (5) *ARFRanker* that is based on using random forests for ensembling multiple binary approximated ranking trees. Then, we choose the one showing the best performance in 10-fold cross-validation on the training data O_{tr}. For each ranker, its performance was measured by precision at 1 (denoted by P@1) that measures the proportion of the reasoners that are correctly recommended by the ranker as most efficient reasoner.

Table 4 shows the performance of the 5 rankers in terms of P@1 on both datasets,

[11]These rankers are available in `http://www.quansun.com`, Readers interested in the details of the rankers are referred to [60].

ErrorsRemoved and **ErrorsReplaced**. For each ranker, P@1 was measured using 10-fold cross-validation from the ranking matrix generated from the training data O_{tr}. The ranking matrix formation was presented in Equation 7 in Section 5.2.2. As seen in the table, all 5 rankers showed high performance, achieving more than 90% of the P@1 values. ARFRanker shows the best performance (denoted in bold): 0.978 and 0.955 on **ErrorsRemoved** and **ErrorsReplaced**, respectively. Consequently, to implement our meta-reasoner $R_2O_2*_{(rk)}$, we use ARFRanker.

Table 4: A summary of performance assessment of the 5 rankers in terms of P@1.

Ranker	**ErrorsRemoved**	**ErrorsReplaced**
KNNRanker	0.973	0.952
PCTRanker	0.974	0.944
RPCRanker	0.970	0.947
RegRanker	0.967	0.947
ARFRanker	**0.978**	**0.955**

7.1.3. Performance of the classifiers in $R_2O_2*_{(mc)}$

As presented in Section 5.2.3, the meta-reasoner: $R_2O_2*_{(mc)}$ learns the most efficient reasoner on the training data, and predicts the most likely efficient reasoner for an unknown ontology. Using the ontology representation scheme in Equation 5, we considered two classifiers: random forest algorithm and XGBoost.

Table 5 shows the prediction performance in terms of classification accuracy on both datasets, **ErrorsRemoved** and **ErrorsReplaced**. As can be seen, the prediction performance is very similar between RF and XGBoost where the best one on each dataset is denoted in bold. In our evaluation, we use XGBoost as it shows the better performance than RF overall.

In the following subsection, we present and analyse the evaluation results of our proposed four meta-reasoners on the testing data O_{te}.

Table 5: A summary of performance of the two prediction models for meta-reasoner $R_2O_2*_{(mc)}$ in terms of accuracy.

Model	**ErrorsRemoved**	**ErrorsReplaced**
RF	**0.984**	0.944
XGBoost	**0.984**	**0.945**

7.2. *Performance Evaluation of R_2O_2* on Reasoning Efficiency*

In this subsection we discuss the evaluation results comparing our meta-reasoners with the various component reasoners as well as AutoFolio [31], a state-of-the-art algorithm selection framework as a strong and robust baseline, following the evaluation framework presented in Section 6.

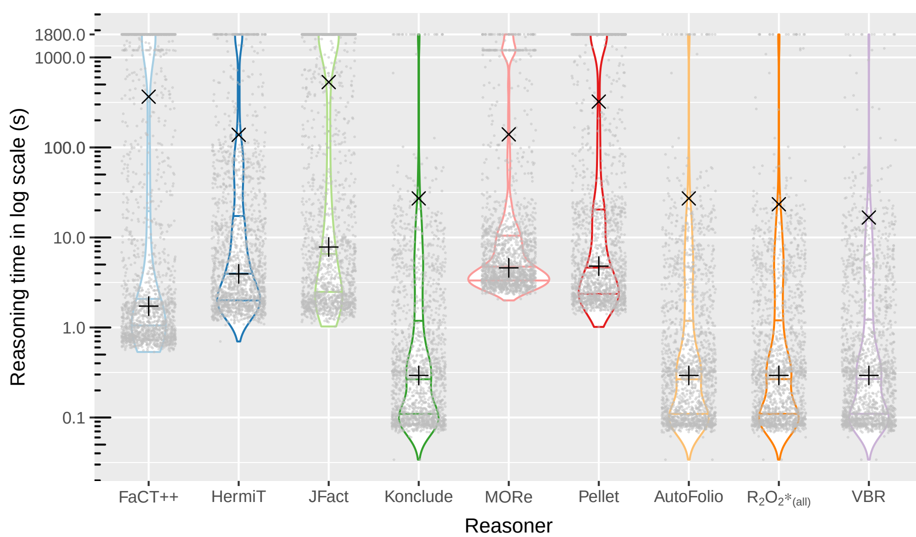

Figure 1: A summary of reasoning time characteristics, in seconds and log-scale, of various reasoners in violin plots.

To summarise the performance characteristics of the various component and meta-reasoners, Figure 1 shows a violin plot of the reasoning time of the reasoners on log scale, for the experiment **ErrorsReplaced**. A violin plot is a combination of a boxplot and a mirrored kernel density plot. As a result, a violin plot visualises the underlying distribution that boxplot does not show. Each shape in Figure 1 contains the following components.

- The (mirrored) violin itself shows the distribution of reasoning time.
- The cross (×) in the middle shows the mean reasoning time of the reasoner.
- The plus symbol (+) in the middle shows the median reasoning time of the reasoner.
- The three horizontal lines within each shape shows the 25%, 50%, and 75% of data, respectively.
- The grey dots represent the actual reasoning time of all ontologies.

As can be seen from the figure, the dominance of Konclude over the other five component reasoners is evident. $R_2O_2*_{(all)}$ shares similar performance characteristics with AutoFolio and VBR, but with a lower mean reasoning time than AutoFolio. The term VBR stands for the virtual best reasoner, which exhibits the optimal efficiency. Even VBR times out on a number of ontologies (grey dots on 1800.0 at the top of the plot), showing the challenging nature of ontology reasoning. The remainder of the section will present more details and discussions on these performance comparisons.

7.2.1. Meta-reasoner time overhead

The four different variants of R_2O_2* all require some additional tasks at both training time and test time. At training time, overall across the 10-fold cross validation, $R_2O_2*_{(pt)}$ needs to learn regression models (Section 5.2.1); $R_2O_2*_{(rk)}$ needs to learn rankers (Section 5.2.2), $R_2O_2*_{(mc)}$ needs to learn a multi-class classifier (Section 5.2.3); and $R_2O_2*_{(all)}$ ensembles all the above (Section 5.2.4), hence needing to learn all those models. At testing time, each of the meta-reasoners will need to apply these models.

We have calculated the time overhead of learning and applying these models. At training time, building a regression model for a reasoner takes 1–2.5 sec (stacking model that combines RF and XGBoost), building a ranker takes 0.3–0.6 sec, building a multi-class classifier takes 0.3–0.5 sec, and building a final stacking model ($R_2O_2*_{(all)}$) takes an additional 0.1 sec. At testing time, making prediction for a given ontology by $R_2O_2*_{(all)}$ takes a negligible < 0.5 millisecond.

We note that the time overhead at training time does not affect R_2O_2*'s performance as an OWL reasoner. It is only the overhead at testing time that does, as for a new ontology, predictions need to be made for the various models. In all the experiments below in the rest of this section, R_2O_2*'s reported reasoning time already includes the testing-time time overhead.

7.2.2. Comparison with our meta-reasoners and component reasoners

We now evaluate our meta-reasoners against the six component reasoners, Autofolio and VBR in detail. The evaluation is conducted for the two experiments described in the previous section, **ErrorsRemoved**, in which error ontologies are removed, and **ErrorsReplaced**, in which error ontologies are treated as they time out. Furthermore, for each experiment, we experiment with the inclusion/exclusion of ELK in our meta-reasoners, as described in Section 5. By incorporating ELK, our meta-reasoners can handle the simpler OWL 2 EL ontologies more efficiently, which will improve the overall performance.

The reasoners are evaluated on two metrics: (1) average precision at 1 (P@1), which measures whether a reasoner is the most efficient on a given test collection O_{te} across 10-fold cross validation, and (2) average runtime (Avg. runtime), which measures the mean reasoning time on a given test collection O_{tr} in seconds across 10-fold cross-validation. The results presented in the rest of this subsection are the average of those obtained on the test set in each of the 10 folds. Note that in each fold, the test collection O_{te} is differently chosen from the total ontology collection O. A more detailed description about our evaluation framework is found in Section 6.

Table 6 and Table 7 show evaluation results for the two experiments, **ErrorsRemoved** and **ErrorsReplaced**, respectively. We note that in the case where ELK is included in our meta-reasoners, the P@1 and Avg. runtime values for ELK are not recorded over the entire test set, but only the subset of OWL 2 EL ontologies. Hence a comparison between ELK and the other reasoners is not meaningful. Note that as we discussed in Section 5.2, $R_2O_2*_{(pt)}$ follows a similar spirit of the non-ranking portfolio reasoner PR [27], and $R_2O_2*_{(rk)}$ follows a similar spirit of R_2O_2 [27]. A number of important observations can be made from these tables.

- In the experiment **ErrorsRemoved**, in all but one case, the best variant of our meta-reasoners outperforms all component reasoners in terms of P@1. Konclude has

Table 6: Performance evaluation in the experiment **ErrorsRemoved**, in which error ontologies are removed. Our meta-reasoners are compared with component reasoners and the virtual best reasoner (VBR). In each column, the best results are **bolded** and the second best results are underlined.

Reasoner	Without ELK		With ELK	
	P@1	Avg. runtime	P@1	Avg. runtime
FaCT++	0.36%	201.62	0.36%	201.62
Hermit	0.14%	54.17	0.14%	54.17
JFact	0.07%	308.47	0.07%	308.47
Konclude	**98.78%**	8.58	97.26%	8.58
MORe	0.79%	89.93	0.58%	89.93
Pellet	0.22%	164.30	0.22%	164.30
ELK	-	-	1.73%	0.69
$R_2O_2{*}_{\text{(pt)}}$	98.56%	8.60	98.20%	8.53
$R_2O_2{*}_{\text{(rk)}}$	**98.78%**	8.58	98.42%	8.51
$R_2O_2{*}_{\text{(mc)}}$	**98.78%**	**7.48**	**98.49%**	**7.40**
$R_2O_2{*}_{\text{(all)}}$	**98.78%**	**7.48**	**98.49%**	**7.40**
VBR	100%	4.50	100%	4.43

the same performance (98.78%) as our meta-reasoners ($R_2O_2{*}_{\text{(rk)}}$, $R_2O_2{*}_{\text{(mc)}}$, and $R_2O_2{*}_{\text{(all)}}$) when ELK is not considered.

- In both experiments **ErrorsRemoved** and **ErrorsReplaced**, in all cases, the best variant of our meta-reasoners outperforms all component reasoners in terms of average runtime, including the highly efficient, parallelising reasoner Konclude.
- Out of all the variants of our meta-reasoners, $R_2O_2{*}_{\text{(all)}}$ exhibits overall best efficiency. Of all the four experimental setups, $R_2O_2{*}_{\text{(all)}}$ exhibits the best performance of three, and second best in the other one. $R_2O_2{*}_{\text{(all)}}$ achieves a speedup of at least 1.10x (over Konclude in experiment **ErrorsReplaced**) and at most 41.69x (over JFact in experiment **ErrorsRemoved**). Its best efficiency is the result of the stacking technique employed in $R_2O_2{*}_{\text{(all)}}$.

Table 7: Performance evaluation of the experiment **ErrorsReplaced**, in which error ontologies are replaced by timeouts (30 minutes). Our meta-reasoners is compared with component reasoners and the virtual best reasoner (VBR). In each column, the best results are **bolded** and the second best results are underlined.

Reasoner	Without ELK		With ELK	
	P@1	Avg. runtime	P@1	Avg. runtime
FaCT++	1.31%	365.92	1.25%	365.92
Hermit	0.68%	138.06	0.68%	138.06
JFact	0.57%	532.11	0.57%	532.11
Konclude	97.22%	27.15	94.55%	27.15
MORe	2.16%	139.68	1.48%	139.68
Pellet	0.91%	322.59	0.85%	322.58
ELK	-	-	3.47%	0.97
$R_2O_2{*}_{(pt)}$	97.10%	**24.50**	96.82%	23.89
$R_2O_2{*}_{(rk)}$	**97.73%**	25.72	97.05%	24.45
$R_2O_2{*}_{(mc)}$	97.39%	26.93	96.82%	25.66
$R_2O_2{*}_{(all)}$	97.56%	24.67	**97.16%**	**23.39**
VBR	100%	16.65	100%	16.22

- The inclusion of ELK in our meta-reasoners indeed improves reasoning efficiency. Even though ELK is only able to handle the less expressive OWL 2 EL profile, our meta-reasoners are able to take advantage of its efficiency in handling such ontologies.

Table 1 in Section 7 shows the vast efficiency dominance of Konclude over the other component reasoners. To better understand the reasoners' performance, we divide the ORE 2015 dataset into four bins of *discretised reasoning time*. The discretisation is performed on the *best reasoning time* (in seconds) of the virtual best reasoner (VBR), into four bins: 'A' (0, 1), 'B' [1, 10), 'C' [10, 100), and 'D' [100, 1,800], which represent ontologies with increasing difficulty.

Tables 8 and 9 below summarise, in each bin and the entire ontology collection O, the percentage of each component reasoner being the most efficient. The component

Table 8: Comparison of percentage of each component reasoner being the most efficient in each bin and overall in experiment **ErrorsRemoved** (Rea: Reasoner, E: ELK, F: FaCT++, H: HermiT, J: JFact, K: Konclude, M: MORe, P: Pellet).

Rank	Overall		A		B		C		D	
	%	Rea	%	Rea	%	Rea	%	Rea	%	Rea
1	96.92	K	100	K	88.62	K	80.56	K	27.27	F,K
2	1.72	E	0	E,F,H,J,M,P	8.94	E	11.11	M	18.18	M
3	0.57	M	–	–	0.81	M,P	5.56	E	9.09	H,J,P
4	0.36	F	–	–	0.41	F,H	2.78	F	0	E
5	0.22	P	–	–	0	J	0	H,J,P	–	–
6	0.14	H	–	–	–	–		–	–	–
7	0.07	J	–	–	–	–		–		–

Table 9: Comparison of percentage of each component reasoner being the most efficient in each bin and overall in experiment **ErrorsReplaced** (Rea: Reasoner, E: ELK, F: FaCT++, H: HermiT, J: JFact, K: Konclude, M: MORe, P: Pellet).

Rank	Overall		A		B		C		D	
	%	Rea	%	Rea	%	Rea	%	Rea	%	Rea
1	91.93	K	100	K	84.41	K	67.31	K	20.83	M
2	3.37	E	0	E,F,H,J,M,P	11.18	E	22.12	E	18.06	F,K
3	1.44	M	–	–	2.35	F	7.69	M	15.28	H
4	1.22	F	–	–	0.88	M,P	1.92	P	13.89	J,P
5	0.83	P	–	–	0.29	H	0.96	F	0	E
6	0.66	H	–	–	0	J	0	H,J	–	–
7	0.55	J	–	–	–	–		–		–

reasoners are ordered by their percentages of being the most efficient, from highest to lowest. Note ELK only performs reasoning on the subset of OWL 2 EL ontologies. Note that the % values in the tables are averaged from the test sets of 10-fold cross validation in the above two experimental setups. Also note that the % values in Tables 8 and 9 are different from the P@1 values in Table 6 and Table 7. The % values show the percentage being the most efficient across all the component reasoners. Thus, the sum of the % values in each of the columns - 'Overall', 'A', 'B', 'C' and 'D' is 1. From these tables, we note a number of interesting observations:

- Konclude's dominance is again evident, as it is the most efficient reasoner for all bins except only in bin D, experiment **ErrorsReplaced**, where MORe is the most efficient.
- Despite its dominance, Konclude does not dominate the most challenging category, bin D. In experiment **ErrorsRemoved** both FaCT++ and Konclude are the most efficient, and in experiment **ErrorsReplaced** MORe is the most efficient.
- For the most difficult category, bin D, many component reasoners are the most efficient for a considerable percentage. In other words, the dominance of any component reasoner is much less pronounced. This shows the challenging nature of algorithm selection in the ontology reasoning context, where for the most challenging instances, a large number of choices are possible. This observation also indicates considerable room for further investigation.

7.2.3. Comparison with AutoFolio

In this section, we evaluate $R_2O_2*_{(\text{all})}$ against AutoFolio for the ontology classification problem[12]. We have chosen $R_2O_2*_{(\text{all})}$ for comparison with AutoFolio as it shows the best performance overall in our evaluation as discussed in the previous section. AutoFolio and $R_2O_2*_{(\text{all})}$ use the same set of metrics, and are trained and tested on the same data splits and the same set of six reasoners. Hence, the comparison is fair. Note that AutoFolio makes use of functionally equivalent components hence it cannot take advantage of ELK. As described in Section 6, 10-fold cross validation is carried out for both experiments. In each fold, an AutoFolio model is trained on the training set, and then evaluated on the test set.

Table 10 summarises the mean reasoning time for the two experiments. As can be observed in the table, $R_2O_2*_{(\text{all})}$ outperforms AutoFolio in both experiments. The

[12] AutoFolio is available at `https://github.com/mlindauer/AutoFolio`.

best performance is achieved when ELK is incorporated in $R_2O_2*_{(all)}$, where $R_2O_2*_{(all)}$ outperforms AutoFolio by 15.95% and 16.08% respectively. However, even without ELK, $R_2O_2*_{(all)}$ also outperforms AutoFolio, by 14.71% and 10.05% respectively. These results demonstrate the effectiveness of $R_2O_2*_{(all)}$ as AutoFolio represents the state-of-the-art method in automated algorithm selection.

Table 10: Summary of mean reasoning performance (in seconds) comparison between AutoFolio and $R_2O_2*_{(all)}$ ($R_2O_2*_{(all)}$, both with and without ELK). In each row, best performance in the test set is highlighted in **bold**, and second best performance in the test set is underlined.

Experiment	AutoFolio	$R_2O_2*_{(all)}$ (without ELK)	$R_2O_2*_{(all)}$ (with ELK)
ErrorsRemoved	8.58	<u>7.48</u>	**7.40**
ErrorsReplaced	27.15	<u>24.67</u>	**23.39**

We further analysed the component reasoners selected by AutoFolio and $R_2O_2*_{(all)}$ in both experiments. In **ErrorsRemoved**, AutoFolio exclusively selects Konclude for all 1,389 instances. In **ErrorsReplaced**, AutoFolio selects MORe in 11 (0.625%) of the 1,760 instances, and Konclude 1,749 (99.375%) instances.

On the other hand, $R_2O_2*_{(all)}$ is able to select a more diverse set of efficient reasoners. Table 11 shows, with ELK included, the number and percentage of times each component reasoner is selected by $R_2O_2*_{(all)}$ for each bin as well as the entire dataset. These results provide additional evidence that always selecting the most-efficient-on-average reasoner(s) is not the most optimal approach. For example, Table 1 shows that the mean reasoning time of MORe is more than 20x slower than Konclude. However, in experiments **ErrorsReplaced**, MORe is selected almost 30% among the most difficult ontologies (bin D). Thus, this analysis further validates the effectiveness of the sophisticated meta-reasoning framework $R_2O_2*_{(all)}$.

7.3. Key Metrics Identification

Lastly, we investigate the identification of each of the 29 metrics' influence on the performance of our meta-reasoners. This will provide insight into the contribution of individual metrics we have chosen (i.e. 29 metrics) on their performance. We measure

Table 11: The number and percentage of each component reasoner being selected by $R_2O_2*_{(all)}$ in both experiments, with ELK included. Percentages are calculated column-wise.

	Overall (%)	A (%)	B (%)	C (%)	D (%)
ErrorsRemoved					
Konclude	1,354 (97.48)	1,101 (100)	219 (88.66)	28 (82.35)	6 (85.71)
MORe	1 (0.07)	0	0	0	1 (14.29)
ELK	34 (2.45)	0	28 (11.34)	6 (17.65)	0
ErrorsReplaced					
FaCT++	6 (0.34)	0	6 (1.80)	0	0
Konclude	1,682 (95.57)	1,294 (100)	291 (87.12)	79 (75.24)	18 (66.67)
MORe	13 (0.74)	0	0	5 (4.76)	8 (29.63)
Pellet	2 (0.11)	0	0	1 (0.95)	1 (3.70)
ELK	57 (3.24)	0	37 (11.08)	20 (19.05)	0

the relative metric importance by using the feature importance values provided by the XGBoost classifier used in $R_2O_2*_{(mc)}$. Figure 2 shows the most important metrics for the prediction task of $R_2O_2*_{(mc)}$, which is to predict the most efficient reasoner, in both experiments. The weights are estimated by calculating the average of importance of the metrics through 10-fold cross validation and normalised into $[0, 1]$. The weight of each metric is measured based on the number of times it is used to split the data across all trees in XGBoost.

As can be seen from the figure, different metrics are important for each experiment. However, there are some common important metrics. Two metrics, SOV and mCID, are common to both experiments' top 5 metrics. Six metrics, IND, SOV, mCID, aCID, RHLC, and EOG, are common to both experiments' top 10 metrics. For both experiments, OBP% is the least important among the 29 metrics. Curiously, DTP% is the most important for experiment **ErrorsRemoved**, while it is the second least important for **ErrorsReplaced**.

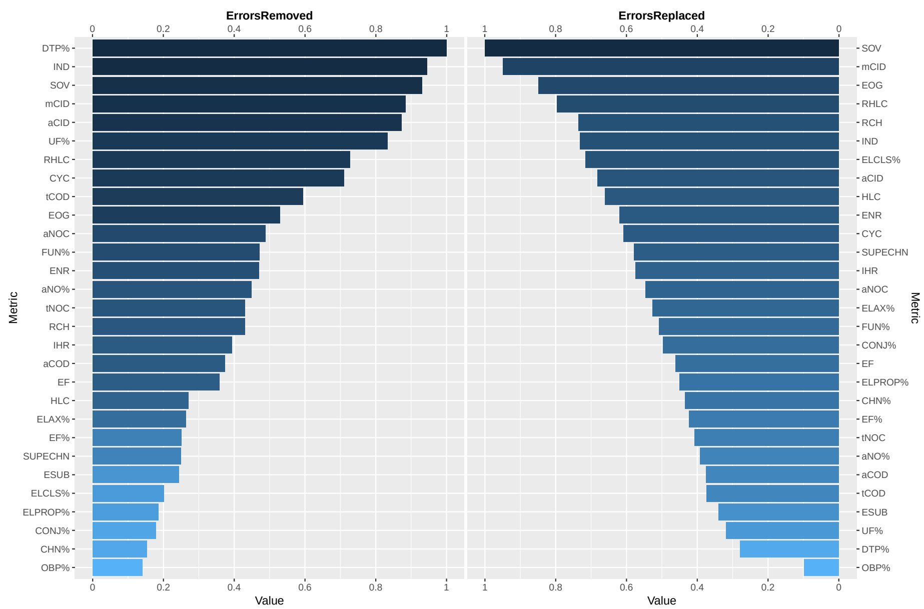

Figure 2: Importance of ontology metrics for $R_2O_2*_{(mc)}$ in both experiments. For each experiment the metrics are ordered in descending order by their importance values.

8. Conclusion

Reasoning support for OWL ontologies is essential for ensuring the correctness of ontologies, and for inferring implicit knowledge from them. For an expressive ontology language such as $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SROIQ}(\mathbf{D})$, worst-case complexity is very high. Moreover, ample empirical evaluation has also confirmed the hardness of actual, real-world ontologies, even on state-of-the-art ontology reasoners such as Konclude and HermiT.

This paper presented R_2O_2*, a novel, robust meta-reasoning framework that automatically ranks component reasoners by efficiency and selects the one that is most likely the most efficient for any given ontology. The R_2O_2* framework comprises a number of novel contributions: (1) we learn regression models that accurately predict reasoning time for a number of state-of-the-art ontology reasoners; and (2) we propose a learning- and ranking-based meta-reasoner that ensembles base prediction models and thus combines component reasoners based on their predicted reasoning efficiency, and (3) we formally define a large suite of syntactic and structural metrics that describe ontologies.

We performed a comprehensive evaluation on six state-of-the-art OWL 2 DL reasoners and a large corpus of carefully curated ontologies. Our evaluation shows that R_2O_2* significantly outperforms all six component reasoners as well as AutoFolio, a strong, general-purpose, and state-of-the-art algorithm selection system. Compared to component reasoners, R_2O_2* achieves a speedup of at least 1.10x (over Konclude) and up to 41.69x (over JFact).

Extending our R_2O_2* meta-reasoning framework to a multi-criteria setting as done in Multi-RakSOR [49, 50] is a natural next step. We plan to extend our methodology to support ABox reasoning, and investigate support of other non-standard reasoning problems. Continuing on our work on ABox-intensive EL ontologies [38] and reasoning on the Android platform [39], we will further investigate sources of ABox reasoning hardness by studying structural and syntactic properties of ABoxes. Performance prediction and optimisation utilising machine learning techniques is particularly interesting and relevant in the context of ontology-based data access (OBDA) [5], where a large database is enhanced by an ontology, and (conjunctive) query answering on the database requires ontology reasoning [61]. We will also investigate the generation of synthetic, yet realistic benchmark ontologies (TBoxes) and instances (ABoxes) to assist in the evaluation and optimisation of reasoners. Finally, investigating the correlation of efficiency between different reasoning tasks by a same reasoner is also an interesting problem worthy of investigation.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, et al., Gene Ontology: Tool for the Unification of Biology, Nat Genet 25 (1) (2000) 25–29. `doi:10.1038/75556`. URL `http://dx.doi.org/10.1038/75556`

[2] J. Z. Pan, S. Staab, U. Aßmann, J. Ebert, Y. Zhao (Eds.), Ontology-Driven Software Development, Springer, 2013.

[3] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '02, ACM, New York, NY, USA, 2002, pp. 233–246. doi:http://doi.acm.org/10.1145/543613.543644.
URL http://doi.acm.org/10.1145/543613.543644

[4] A. Calı, D. Calvanese, G. De Giacomo, M. Lenzerini, Data integration under integrity constraints, Information Systems 29 (2) (2004) 147–163.

[5] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, M. Zakharyaschev, The combined approach to ontology-based data access, in: IJCAI, Vol. 11, 2011, pp. 2656–2661.

[6] Y.-F. Li, G. Kennedy, F. Ngoran, P. Wu, J. Hunter, An ontology-centric architecture for extensible scientific data management systems, Future Gener. Comput. Syst. 29 (2) (2013) 641–653. doi:10.1016/j.future.2011.06.007.
URL http://dx.doi.org/10.1016/j.future.2011.06.007

[7] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen, From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language, Journal of Web Semantics 1 (1) (2003) 7–26.

[8] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 6 (2008) 309–322. doi:10.1016/j.websem.2008.05.001.
URL http://portal.acm.org/citation.cfm?id=1464505.1464604

[9] F. Baader, U. Sattler, An overview of tableau algorithms for description logics, Studia Logica 69 (1) (2001) 5–40.

[10] D. Tsarkov, I. Horrocks, FaCT++ description logic reasoner: System description, in: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), Springer, 2006, pp. 292–297.

[11] R. Shearer, B. Motik, I. Horrocks, HermiT: A Highly-Efficient OWL Reasoner, in: Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008), 2008.

[12] A. Steigmiller, T. Liebig, B. Glimm, Konclude: System description, J. Web Sem. 27 (2014) 78–85. `doi:10.1016/j.websem.2014.06.003`.
URL `http://dx.doi.org/10.1016/j.websem.2014.06.003`

[13] E. Sirin, B. Parsia, B. Cuenca-Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semantics: Science, Services and Agents on the World Wide Web 5 (2) (2007) 51–53. `doi:10.1016/j.websem.2007.03.004`.
URL `http://dx.doi.org/10.1016/j.websem.2007.03.004`

[14] J. Z. Pan, Y. Ren, Y. Zhao, Tractable approximate deduction for OWL, Artificial Intelligence 235 (2016) 95–155.

[15] K. Dentler, R. Cornet, A. ten Teije, N. de Keizer, Comparison of reasoners for large ontologies in the OWL 2 EL profile, Semantic Web Journal 2 (2) (2011) 71–87.

[16] Y.-B. Kang, Y.-F. Li, S. Krishnaswamy, A rigorous characterization of reasoning performance – a tale of four reasoners, in: Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), 2012.

[17] M. Y. Vardi, Boolean satisfiability: Theory and engineering, Commun. ACM 57 (3) (2014) 5–5. `doi:10.1145/2578043`.
URL `http://doi.acm.org/10.1145/2578043`

[18] K. Leyton-Brown, H. H. Hoos, F. Hutter, L. Xu, Understanding the empirical hardness of *NP*-complete problems, Commun. ACM 57 (5) (2014) 98–107. `doi:10.1145/2594413.2594424`.
URL `http://doi.acm.org/10.1145/2594413.2594424`

[19] J. Z. Pan, Benchmarking DL reasoners using realistic ontologies, in: B. Cuenca-Grau, I. Horrocks, B. Parsia, P. F. Patel-Schneider (Eds.), OWLED, Vol. 188 of CEUR Workshop Proceedings, CEUR-WS.org, 2005.

[20] T. Gardiner, I. Horrocks, D. Tsarkov, Automated benchmarking of description logic reasoners, in: Proceedings of the 2006 International Workshop on Description Logics (DL2006), 2006.

[21] S. Bail, B. Glimm, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, A. Steigmiller, Summary ORE 2014 competition, in: Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE 2014), Vol. 1207, CEUR Workshop Proceedings, 2014, pp. iv–vii.

[22] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, A. Steigmiller, The OWL reasoner evaluation (ORE) 2015 competition report, Journal of Automated Reasoning 59 (4) (2017) 455–482.

[23] R. S. Gonçalves, N. Matentzoglu, B. Parsia, U. Sattler, The empirical robustness of description logic classification, in: T. Eiter, B. Glimm, Y. Kazakov, M. Krötzsch (Eds.), Description Logics, Vol. 1014 of CEUR Workshop Proceedings, CEUR-WS.org, 2013, pp. 197–208.

[24] H. Zhang, Y.-F. Li, H. B. K. Tan, Measuring design complexity of Semantic Web ontologies, Journal of Systems and Software 83 (5) (2010) 803–814. `doi:DOI:10.1016/j.jss.2009.11.735`.
URL `http://www.sciencedirect.com/science/article/B6V0N-4XV06RX-2/2/1024981a176a351b9dbb0cc4c487c17e`

[25] Y.-B. Kang, Y.-F. Li, S. Krishnaswamy, Predicting reasoning performance using ontology metrics, in: Cudré-Mauroux et al. [62], pp. 198–214.

[26] Y.-B. Kang, J. Z. Pan, S. Krishnaswamy, W. Sawangphol, Y.-F. Li, How long will it take? Accurate prediction of ontology reasoning performance, in: C. E. Brodley, P. Stone (Eds.), AAAI, AAAI Press, 2014, pp. 80–86.

[27] Y.-B. Kang, S. Krishnaswamy, Y.-F. Li, R2O2: an efficient ranking-based reasoner for OWL ontologies, in: M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, S. Staab (Eds.), The Semantic Web - ISWC 2015 - 14th International Semantic

Web Conference, Part I, Vol. 9366 of Lecture Notes in Computer Science, Springer, 2015, pp. 322–338. doi:10.1007/978-3-319-25007-6_19.
URL http://dx.doi.org/10.1007/978-3-319-25007-6_19

[28] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, ACM, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
URL http://doi.acm.org/10.1145/2939672.2939785

[29] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 1189–1232.

[30] Y. Kazakov, M. Krötzsch, F. Simancik, The incredible ELK - from polynomial procedures to efficient reasoning with $\mathscr{EL}$ ontologies, J. Autom. Reasoning 53 (1) (2014) 1–61. doi:10.1007/s10817-013-9296-3.
URL http://dx.doi.org/10.1007/s10817-013-9296-3

[31] M. Lindauer, H. H. Hoos, F. Hutter, T. Schaub, AutoFolio: An automatically configured algorithm selector, Journal of Artificial Intelligence Research 53 (2015) 745–778.

[32] F. Baader, W. Nutt, Basic description logics, in: F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The description logic handbook: theory, implementation, and applications, Cambridge University Press, 2003, pp. 43–95.

[33] I. Horrocks, The FaCT system, Tableaux’98, LNCS 1397 (1998) 307–312.

[34] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, HermiT: An OWL 2 reasoner, J. Autom. Reason. 53 (3) (2014) 245–269. doi:10.1007/s10817-014-9305-1.
URL http://dx.doi.org/10.1007/s10817-014-9305-1

[35] R. S. Gonçalves, B. Parsia, U. Sattler, Performance heterogeneity and approximate reasoning in description logic ontologies, in: Cudré-Mauroux et al. [62], pp. 82–98.

[36] D. Zhang, C. Ye, Z. Yang, An evaluation method for ontology complexity analysis in ontology evolution, in: Managing Knowledge in a World of Networks, Vol. 4248, 2006, pp. 214–221.

[37] V. Sazonau, U. Sattler, G. Brown, Predicting OWL reasoners: Locally or globally?, in: M. Bienvenu, M. Ortiz, R. Rosati, M. Simkus (Eds.), Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014., Vol. 1193 of CEUR Workshop Proceedings, CEUR-WS.org, 2014, pp. 713–724.
URL `http://ceur-ws.org/Vol-1193/paper_12.pdf`

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

[39] I. Guclu, Y.-F. Li, J. Z. Pan, M. J. Kollingbaum, Predicting energy consumption of ontology reasoning over mobile devices, in: P. T. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, Y. Gil (Eds.), The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I, Vol. 9981 of Lecture Notes in Computer Science, 2016, pp. 289–304. `doi:10.1007/978-3-319-46523-4_18`.
URL `http://dx.doi.org/10.1007/978-3-319-46523-4_18`

[40] J. R. Rice, The algorithm selection problem, in: Advances in computers, Vol. 15, Elsevier, 1976, pp. 65–118.

[41] K. A. Smith-Miles, Cross-disciplinary perspectives on meta-learning for algorithm selection, ACM Computing Surveys (CSUR) 41 (1) (2009) 6.

[42] F. Hutter, L. Xu, H. H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: Methods & evaluation, Artif. Intell. 206 (2014) 79–111. `doi:10.1016/j.artint.2013.10.003`.
URL `http://dx.doi.org/10.1016/j.artint.2013.10.003`

[43] L. Xu, F. Hutter, H. H. Hoos, K. Leyton-Brown, SATzilla: Portfolio-based algorithm selection for SAT, J. Artif. Int. Res. 32 (1) (2008) 565–606.
URL http://dl.acm.org/citation.cfm?id=1622673.1622687

[44] D. Tsarkov, I. Palmisano, Chainsaw: a metareasoner for large ontologies, in: I. Horrocks, M. Yatskevich, E. Jiménez-Ruiz (Eds.), Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), Manchester, UK, July 1st, 2012, Vol. 858 of CEUR Workshop Proceedings, CEUR-WS.org, 2012.
URL http://ceur-ws.org/Vol-858/ore2012_paper2.pdf

[45] B. Cuenca-Grau, I. Horrocks, Y. Kazakov, U. Sattler, Modular reuse of ontologies: Theory and practice, J. Artif. Intell. Res. (JAIR) 31 (2008) 273–318.

[46] C. Del Vescovo, B. Parsia, U. Sattler, T. Schneider, The modular structure of an ontology: Atomic decomposition, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11, AAAI Press, 2011, pp. 2232–2237. doi:10.5591/978-1-57735-516-8/IJCAI11-372.
URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-372

[47] B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, A. Steigmiller, The OWL reasoner evaluation (ORE) 2015 competition report, in: T. Liebig, A. Fokoue (Eds.), Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015., Vol. 1457 of CEUR Workshop Proceedings, CEUR-WS.org, 2015, pp. 2–15.
URL http://ceur-ws.org/Vol-1457/SSWS2015_paper1.pdf

[48] W. Song, B. Spencer, W. Du, WSReasoner: A prototype hybrid reasoner for ALCHOI ontology classification using a weakening and strengthening approach, in: OWL Reasoner Evaluation Workshop (ORE 2012), 2012, p. 1.

[49] N. Alaya, S. B. Yahia, M. Lamolle, Raksor: Ranking of ontology reasoners based on predicted performances, in: 28th IEEE International Conference on Tools with

Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016, IEEE Computer Society, 2016, pp. 1076–1083. doi:10.1109/ICTAI.2016.0165.
URL https://doi.org/10.1109/ICTAI.2016.0165

[50] N. Alaya, M. Lamolle, S. B. Yahia, Multi-label based learning for better multi-criteria ranking of ontology reasoners, in: C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I, Vol. 10587 of Lecture Notes in Computer Science, Springer, 2017, pp. 3–19. doi:10.1007/978-3-319-68288-4_1.
URL https://doi.org/10.1007/978-3-319-68288-4_1

[51] Y. Zhou, B. C. Grau, Y. Nenov, M. Kaminski, I. Horrocks, Pagoda: Pay-as-you-go ontology query answering using a datalog reasoner, Journal of Artificial Intelligence Research 54 (2015) 309–367.

[52] N. Fenton, J. Bieman, Software Metrics: A Rigorous and Practical Approach, Third Edition, 3rd Edition, CRC Press, Inc., Boca Raton, FL, USA, 2014.

[53] B. Motik, R. Shearer, I. Horrocks, Optimized reasoning in description logics using hypertableaux, in: International Conference on Automated Deduction, Springer, 2007, pp. 67–83.

[54] F. M. Donini, Complexity of reasoning, in: The Description Logic Handbook: Theory, Implementation, and Applications, 2003, pp. 96–136.

[55] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), Proceedings of IJCAI 2005, Professional Book Center, 2005, pp. 364–369.
URL http://dblp.uni-trier.de/db/conf/ijcai/ijcai2005.html#BaaderBL05

[56] A. A. Romero, B. Cuenca-Grau, I. Horrocks, MORe: Modular combination of OWL reasoners for ontology classification, in: Cudré-Mauroux et al. [62], pp. 1–16.

[57] M. P. Sesmero, A. I. Ledezma, A. Sanchis, Generating ensembles of heterogeneous classifiers using stacked generalization, Wiley Int. Rev. Data Min. and Knowl. Disc. 5 (1) (2015) 21–34. `doi:10.1002/widm.1143`.
URL `http://dx.doi.org/10.1002/widm.1143`

[58] V. Haarslev, K. Hidde, R. Möller, M. Wessel, The RacerPro knowledge representation and reasoning system, Semantic Web 3 (3) (2012) 267–277.

[59] V. Haarslev, R. Möller, RACER system description, in: Proceedings of Automated Reasoning : First International Joint Conference, no. 2083 in Lecture Notes in Computer Science, Siena, Italy, 2001, pp. 701–706.

[60] Q. Sun, B. Pfahringer, Pairwise meta-rules for better meta-learning-based algorithm ranking, Machine Learning 93 (1) (2013) 141–161. `doi:10.1007/s10994-013-5387-y`.

[61] A. Cali, G. Gottlob, A. Pieris, Towards more expressive ontology languages: The query answering problem, Artificial Intelligence 193 (2012) 87–128.

[62] P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist (Eds.), The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I, Vol. 7649 of Lecture Notes in Computer Science, Springer, 2012.

Appendix A. Metric Definitions

Table A.12: Definitions of the 24 ontology-level (ONT) metrics. Note that "__" represents "don't cares".

Metric	Definition
SOV	No. of named entities (classes, properties & individuals)
ENR	Ratio between number of (named and anonymous) entities and number of edges
TIP	Difference between number of subclass axioms and number of (named or anonymous) classes
EOG	The *entropy* of the ontology graph, measuring the diversity of the edge distribution
CYC	The *cyclomatic complexity* of the ontology, measuring the number of linearly independent paths
RCH	The ratio of (possibly nested) anonymous class expressions and all (named or anonymous) class expressions
IND	No. of (named or anonymous) individuals
GCI	No. of GCI axioms
HGCI	No. of hidden GCI axioms
ESUB%	Ratio of subclass axioms that contain (nested) existential restrictions ($\exists R.\_\_$)
DSUB%	Ratio of subclass axioms that contain (nested) class unions (__ $\sqcup$ __)
CSUB%	Ratio of subclass axioms that contain (nested) class intersections (__ $\sqcap$ __) and the subclass is anonymous
ELCLS%	Ratio of (nested) class expressions that are in OWL 2 EL profile
ELAX%	Ratio of subclass or equivalent class axioms that only contain expressions in the OWL 2 EL profile

Table A.12: Definitions of the 24 ontology-level (ONT) metrics. Note that "__" represents "don't cares".

Metric	**Definition**
HLC	Count hard language constructs containing in superclass expressions
HLC%	Ratio of HLC and subclass axioms
SUBCECHN	No. of top-level subclass expressions that contain chained existential restrictions
SUPCECHN	No. of top-level superclass expressions that contain chained existential restrictions
DSUBCECHN	Max depth of chained existential restrictions in a subclass expression
DSUPCECHN	Max depth of chained existential restrictions in a superclass expression
SUBCCHN	No. of top-level subclass expressions that contain chains of conjunctions
SUPCCHN	No. of top-level superclass expressions that contain chains of conjunctions
DSUBCCHN	Max depth of nested conjunctions in a subclass expression
DSUPCCHN	Max depth of nested conjunctions in a superclass expression

Table A.13: Definitions of the 15 class-level (CLS) metrics. Note that C represents a (named or possibly nested) class expression in an antology, and N_C denotes the total number of (named or possibly nested) class expressions in a given ontology.

Metric	**Definition**
tNOC	$\sum_C NOC(C)$
aNOC	$\frac{tNOC}{N_C}$
mNOC	$\mathtt{max}_C NOC(C)$
tNOP	$\sum_C NOP(C)$
aNOP	$\frac{tNOP}{N_C}$
mNOP	$\mathtt{max}_C NOP(C)$
tCID/tCOD	$\sum_C$ incoming/outgoing edges of C
aCID/aCOD	$aCID = \dfrac{tCID}{N_C}, aCOD = \dfrac{tCOD}{N_C}$
mCID/mCOD	$mCID = \mathtt{max}_C CID(C), mCOD = \mathtt{max}_C COD(C)$
tDIT	$\sum_C$ distance of C from `owl:Thing` in a depth-first manner
aDIT	$\dfrac{tDIT}{N_C}$
mDIT	$\mathtt{max}_C DIT(C)$

Table A.14: Definition of the 22 (possibly nested) anonymous class expression (ACE) metrics. Note that each row represents a count metric and a ratio metric (represented by [%]) for the same type of class expressions.

Metric	**Definition**
ENUM[%]	For enumerations/nominals ($\{a, b, c\}$, where a, b, c are individuals)
NEG[%]	For class negations ($\neg C$)
CONJ[%]	For class intersections (conjunctions, $C_1 \sqcap C_2$)
DISJ[%]	For class unions (disjunctions, $C_1 \sqcup C_2$)
UF[%]	For universal restrictions ($\forall R.C$)
EF[%]	For existential restrictions ($\exists R.C$)
VALUE[%]	For value restrictions ($\exists R.\{a\}$), where a is an individual
SELF[%]	For self references
MNCAR[%]	For min cardinality restrictions ($\geq n\ R.C$)
MXCAR[%]	For max cardinality restrictions ($\leq n\ R.C$)
CAR[%]	For (exact) cardinality restrictions ($= n\ R.C$)

Table A.15: Definitions of the 30 property-level (PRO) metrics.

Metric	**Definition**
OBP[%]	Count and ratio of object-properties
DTP[%]	Count and ratio of datatype-properties
FUN[%]	Count and ratio of functional properties
SYM[%]	Count and ratio of symmetric properties
TRN[%]	Count and ratio of transitive properties
IFUN[%]	Count and ratio of inverse functional properties
ASYM[%]	Count and ratio of asymmetric properties
REFLE[%]	Count and ratio of reflexive properties
IRREF[%]	Count and ratio of irreflexive properties
CHN[%]	Count and ratio of property chain axioms
SUBP	Count of subproperty axioms
EQVP	Count of equivalent property axioms
DISP	Count of disjoint property axioms
INV	Count of inverse property axioms
DOMN	Count of domain axioms
RANG	Count of range axioms
ELPROP%	Ratio of property axioms that are in the OWL 2 EL profile
IHR	Count of class axioms that involve property hierarchies
IIR	Count of class axioms that involve inverse properties
ITR	Count of class axioms that involve transitive properties