What Makes Ontology Reasoning so Arduous? Unveiling the key ontological features

Conference Paper · July 2015

DOI: 10.1145/2797115.2797117

3 authors:

Nourhène Alaya
National Institute for Research in Computer Science and Control
11 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Sadok Ben Ben Yahia
University of Tunis El Manar
400 PUBLICATIONS **2,502** CITATIONS
SEE PROFILE

Myriam Lamolle
Université de Vincennes - Paris 8
86 PUBLICATIONS **370** CITATIONS
SEE PROFILE

What Makes Ontology Reasoning so Arduous?

Unveiling the key ontological features

Nourhène ALAYA [*†] nalaya@iut.univ-paris8.fr
Sadok BEN YAHIA[*] sadok.benyahia@fst.rnu.tn
Myriam LAMOLLE[†] m.lamolle@uit.univ-paris8.fr

[*]LIPAH - Faculty of Sciences
University of Tunis, Tunisia

[†] LIASD - IUT of Montreuil
University of Paris 8, France

ABSTRACT

Reasoning with ontologies is one of the core fields of research in Description Logics. A variety of efficient reasoner with highly optimized algorithms have been developed to allow inference tasks on expressive ontology languages such as OWL(DL). However, reasoner reported computing times have exceeded and sometimes fall behind the expected theoretical values. From an empirical perspective, it is not yet well understood, which particular aspects in the ontology are reasoner performance degrading factors. In this paper, we conducted an investigation about state of art works that attempted to portray potential correlation between reasoner empirical behaviour and particular ontological features. These works were analysed and then broken down into categories. Further, we proposed a set of ontology features covering a broad range of structural and syntactic ontology characteristics. We claim that these features are good indicators of the ontology hardness level against reasoning tasks. In order to assess the worthiness of our proposals, we adopted a supervised machine learning approach. Features served as the bases to learn predictive models of reasoners *robustness*. These models was trained for 6 well known reasoners and using their evaluation results during the *ORE'2014* competition. Our prediction models showed a high accuracy level which witness the effectiveness of our set of features.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; I.2.6 [**Learning**]: *Knowledge acquisition*

Keywords

Ontology, Reasoner, Description Logic, Ontology Features, Supervised Machine Learning

1. INTRODUCTION

WIMS '15, July 13 - 15, 2015, Larnaca, Cyprus

DOI: http://dx.doi.org/10.1145/2797115.2797117

Ontologies are used as conceptual models, for data integration, or to directly represent information in a variety of domain areas. The key component for working with OWL ontologies is the *Reasoner*. This is because knowledge in an ontology might not be explicit, then a reasoner is required to deduce its implicit part. However, the high expressivity of OWL has increased the computational complexity of inference tasks, making the semantic reasoning even more greedier. For instance, it has been shown that the complexity of the consistency checking of $\mathcal{SROIQ}$ ontologies, the description logic (DL) underlying OWL 2, is of worst-case 2NExpTime-complete [14]. Therefore, a considerable efforts has been devoted to make reasoning feasible in practice. A number of highly-optimized reasoners have been developed [24, 9], that support reasoning about ontologies written in expressive description logics.

Despite the remarkable progress in optimizing reasoning algorithms, unpredictable behaviour of reasoner engines is often observed in practice. In several cases, the theoretical worst case complexity does not necessarily unveil real-world performances. In the latest reasoner evaluation competitions [10, 5], reasoner reported computing times have exceeded and sometimes fallen behind expected values. In fact, Gonçalves et al. [11] outlined the performance variability phenomena with OWL ontologies. Roughly speaking, the reasoner performances depend on the success or the failure of optimizations tricks set up by reasoner designers to overcome particular known complexity sources in description logic. However, theses tricks would lead to enormous performance variability across the inputs which is still hardly predictable a priori. Yet, it is not well understood which particular aspect in the ontology is lowering the reasoner performances. As previously highlighted by Wang et al. [27], the actual problem is that both ontology novice and expert users are lacking of theory and tool support helping analysing reasoner's behaviours against case study ontologies. Obviously, a better understanding of ontology complexity factors that may trigger difficulties to reasoners is of a compelling need. Indeed, pointing out what makes reasoning hard in practice, can guide the modelling process of ontologies as to avoid the reasoning performance degrading factors. Moreover, existing ontologies may be revised towards efficient reasoning by detecting, and even repairing its critical components.

In this paper, we focus on studying the ontology expressivity at the level of knowledge representation (KR). Our main purpose is to characterize the ontology authoring language grammar, in order to identify domain independent rel-

evant features, likely to outline the hardness of the ontology against the reasoning tasks. First, we carried out an investigation about existing methods which intended to identify potential correlation between reasoner empirical behaviour and particular ontological features. We tried to give users an overall view of the state of art in this field. The pioneering works were analysed and then broken down into categories. Our investigations have lead us to propose a set of ontology features covering a broad rang of structural and syntactic ontology characteristics. We claim that these features would be key indicators of the ontology hardness level against reasoning tasks. In order to assess the worthiness of our proposals, we carried out a supervised machine learning investigation [18], aiming to learn reasoner ***robustness*** predictive models. The robustness stands for the ability of the reasoner to *correctly* process an ontology within a given cutoff time. We are the first to consider the robustness as a performance criterion for the reasoner prediction. To establish the machine learning study, we choose concise and reliable experimental data conducted during the widely recognised Ontology Reasoner Evaluation Workshop *ORE'2014* [5]. The models was trained for 6 well known reasoners and shown to be highly accurate. We have further improved the accuracy of our models by employing a set of feature selection techniques. We are the first to use *discretization* to identify ontology relevant features. A comparative study showed its strength comparing to techniques used in previous works. Thanks to the predictive investigation, we unveiled sets of *local* and *global* ontology key features likely to have important impact on reasoner empirical behaviour.

The rest of the paper is organized as follows: Section 2 briefly describes basic terms such as ontology, description logic and reasoning. Section 3 scrutinizes the related work approaches. Section 4 details our proposed features, the assessment of the latter ones is the subject of the Section 5. Our concluding remarks as well as a sketching of future works are given in Section 6.

2. BACKGROUND

We focus on OWL ontologies and particularly OWL 2. The latter one was recommended by the W3C as the ontology language for the Semantic Web. It is based on the highly expressive Description Logics $\mathcal{SROIQ}$ [14]. This logical provides a precisely defined meaning for statements made in OWL and thus, makes it possible to automatically reason with the OWL ontologies.

In Description Logics (DLs) [3], an ontology $\mathcal{O}$ is composed of a set of asserted axioms, analogous to FOL formulae, describing relationships between terms in a domain of interest. These terms are basically concept, role, and individual names, organized respectively in three sets N_C, N_R, and N_I. The union of these sets, that is, the set of all terms mentioned in $\mathcal{O}$, is called the signature of $\mathcal{O}$, and denoted $\widetilde{\mathcal{O}}$. In DL, an ontology is basically defined as a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$; where $\mathcal{T}$ denotes *TBox*, which comprises *terminological* axioms describing concepts, $\mathcal{R}$ denotes *RBox* standing for axioms describing roles; and $\mathcal{A}$ stands for *ABox*, which is the assertional part of knowledge base describing individuals. In common notations, A and B can be concept names from N_C (also called atomic concepts), C and D are complex concept descriptions [3], R and S are role names from N_R or complex descriptions and a, b or x are names for individuals. If an individual name is used in a *TBox* axiom, then it's called a nominal. Complex concept descriptions are build based on concept and role constructors, as well as names from $\widetilde{\mathcal{O}}$. Different families of Description Logic provide different sets of constructors, and axiom types. One of the simplest DLs is known as $\mathcal{AL}$ (Attributive Language). This DL supports concept conjunction ($C \sqcap D$, `owl:intersectionOf`), universal quantification ($\forall R.C$, `owl:allValuesFrom`), limited existential quantification ($\exists R.\top$, `owl:someValuesFrom` with a filler restricted to `owl:Thing`) and atomic negation ($\neg A$, `owl:complementOf`). More expressive DLs can be obtained from $\mathcal{AL}$ by adding further constructors. Each constructor is given a specific letter which is used to derive a name for any particular DL. For example, adding a full negation ($\neg C$) to $\mathcal{AL}$ produces the DL $\mathcal{ALC}$, which also contains the disjunction's concept ($C \sqcup D$, `owl:unionOf`) and full existential quantification ($\exists R.C$). Generally speaking, a concept in DL is referred to as a class in OWL. A role in DL is a property in OWL, which could be an `Object Property` (properties for which the value is an individual) or a `Data Property` (properties for which the value is a data literal). Axioms and individuals have the same meaning in DL and OWL. Owe to the closed connections between OWL and DLs, in this paper, we will make no distinction between ontologies (in OWL) and knowledge bases (in DL)[1].

Among the reasoning tasks, *classification* is considered as the key one. It computes the full concept and role hierarchies in the ontology [3]. Explicit and implicit subsumption will be derived to help users navigating through the ontology towards mainly explanation and/or query answering respective tasks. Thus, it's supported by all modern DL reasoners and its duration is often used as a performance indicator to benchmark reasoning engines [1]. From an application point of view, an ontology should be classified regularly during its development and maintenance in order to detect undesired subsumptions as soon as possible. To make this feasible, classification has to be carried out as swiftly as possible.

3. RELATED WORKS

In this section, we tried to draw out the landscape of the state of art works, which discussed the ontology complexity key features, likely to impact reasoner performances. These works were graded into three main categories considering their investigation scopes: first, works assessing the ontology quality; then works evaluating the reasoner quality; and finally works attempting to correlate reasoner empirical behaviours to particular ontology features. Table 1 sum ups the main aspects of the aforementioned categories. In the following, we give a more detailed review of works falling in these categories.

Ontology knowledge richness and conceptualization quality is widely assessed in the literature. Huge stream of ontology evaluation metrics was proposed for this purpose [22, 25, 19]. However, little attention was paid to investigate the effectiveness of these metrics to assess the hardness of ontologies against reasoning tasks [28]. On the other hand, reasoner benchmarks and competitions [1, 10, 5] are annually held to compare the latest innovations in the field of semantic reasoning. The performances of these engines against well selected ontologies are evaluated mainly considering the computational runtime. Reasoner's unpredictable behaviour

[1]In the remain of this paper, by OWL we will mean OWL2.

Table 1: The landscape of tools and methods about ontology features altering reasoner performances

Scope	Purpose	References	Ontology Design	Ontology features
Ontology	Quality evaluation	[22, 25, 28]	Graph, OWL, RDF	Structural, Syntactic, Semantic, etc.
Reasoner	Performance evaluation	[1, 10, 5]	KB	Size + Expressivity
Reasoners Empirical behaviours w.r.t. Ontology Features	Ontology Profiling	[27]	OWL	Patterns
		[20]	OWL	Patterns
		[11]	KB	SAT runtime
	Reasoner Performances Prediction	[16]	Graph	27 features filtered via feature selection algorithms
		[23]	OWL	57 features filtered with PCA technique
		[17]	Graph + OWL	91 features filtered based on correlation.

and performance variability is always reported during these events. Even though, there wasn't any attempt to correlate these phenomena to particular ontology features. Recently, some tools, e.g. *Tweezers* [27] and *Pellint* [20], tried to give insights about reasoner performances bottlenecks w.r.t. the input ontology. To fulfil this task, software profiling techniques were deployed. However, it's hard to agree that these tools' procedures would be possibly generalized, since they were proposed by examining one particular reasoner, i.e. *Pellet* [24]. More recently, Gonçalves et al. [11] suggested that there are ontologies which are *performance homogeneous* and others *performance heterogeneous*. The heterogeneity is clause to particular entanglements between axioms in the ontology, causing the increase of the reasoning cost. They proposed a method to track these entanglements and extract their corresponding ontology modules. The latter ones were called ontology *Hotspots*. Nevertheless, authors did not made clear the nature of these entanglements.

Another stream of works, mainly described in [16, 23, 17], used supervised machine learning techniques aiming at predicting the computational time of a reasoner for a given ontology. Their predictive models take advantage from a large set of pre-computed ontological metrics. The rational behind this choice is to be able to automatically learn future reasoner's behaviours based on what was experienced in their previous executions. Kang et al. [16] were the first to apply machine learning techniques to predict the ontology classification computational time carried by a specific reasoner. 27 metrics were computed for each ontology. These metrics were previously proposed by a work stressing on ontology design complexity [28]. Moreover, they proposed an algorithm to compute the impact factors of ontology metrics according to their effectiveness in predicting classification performances for the different reasoners. Kang et al. have further improved their approach, in a more recent work [17] and demonstrated the strengths of their predictive models by applying them to the problem of identifying ontology performance *Hotspots*. On the other hand, Sazonau et al. [23] claimed that Kang's metrics based on graph translation of OWL ontologies are not effective. Thus, they proposed another set of metrics and used more machine learning techniques to identify the most relevant ontology features likely to have impact on reasoner performances.

Clearly, machine learning methods proposed in the last steam of works are the closest to meet our needs. Indeed, the impact of particular ontology features on reasoner performances are automatically investigated using empirical knowledge about reasoners. Nevertheless, choosing good features is crucial to the training of good predictive models. Unfortunately, our review of state of art confirmed that there is no known, automatic way of constructing good ontology feature sets. Instead, we believe that we must use distinct domain knowledge to identify properties of ontologies that appear likely to provide useful information. Afterwards, applying supervised machine learning techniques would be appropriate to examine the real impact of these features on reasoning performances and to identify the most relevant among them.

4. ONTOLOGY FEATURES ALTERING THE REASONER PERFORMANCES

A wealthy number of ontological features was introduced in literature, particularly to build learning models for reasoner computational time prediction. We reused some of them and defined new ones. Mainly, we discarded those computed based on specific graph translation of the OWL ontology. In fact, Sazonau et al. [17] have previously argued that these kind of features are not reliable as there is no agreement of the way an ontology should be translated into a graph.

We split the ontology features into 4 categories: (*i*) size description; (*ii*) expressivity description; (*iii*) structural features; and (*iv*) syntactic features. Within these categories, features are intended to characterize specific aspect of the ontology design. The third and fourth category are further split into subcategories that provide a finer description of the ontology content. In overall, we proposed 112 ontology features, which we will describe in the next sections:

4.1 Ontology Size Description

To characterize the size of the ontology, we propose 6 features, explained in the following:

Signature size features (SSF) : we design 5 features to assess the amount of terms defined in an ontology. Given an ontology signature $\widetilde{\mathcal{O}} = \langle N_C, N_R, N_I \rangle$, we count the number of names nc_i, that uniquely identify classes in the ontology, $nc_i \in N_C$. We call this feature **SC**, the size of the ontology classes, where SC=$|N_C|$. Analogously, we compute the number of user-defined object and data property names, respectively denoted by **SOP** and **SDP**, where $|N_R| = SOP + SDP$, then the number of named individuals **SI**=$|N_I|$. In addition, we record the number of data types[2] defined in the ontology **SDT**.

Axioms size feature (OAS) : it stands for the number of OWL axioms qualified as logical ones. As commonly known, reasoners only deal with axioms belonging to the

[2]These are RDF literals or simple types defined in accordance with XML Schema datatypes.

subsets *TBox*, *RBox* or *ABox*. Annotations are simply ignored when processing an ontology for a reasoning task.

4.2 Ontology Expressivity Description

We retained two main features to identify the expressivity of the ontology language, namely:

OWL profile name (OPR) : there are four possible OWL profiles[3]: DL, EL, QL and RL. We record the one the ontology language fits in. However, in some particular cases, the language constructs used in an ontology may violate rules of all the OWL profiles. In this case, the profile of the ontology is denoted by a virtual profile name, that we called PNAN. In contrast, we tag by PFULL the ontology that matches all the OWL profiles.

DL family name (DFN) : it is a more strict denomination of the DL constructs group used in the ontology. For instance, an ontology could be $\mathcal{AL}$ or $\mathcal{SH}$ or $\mathcal{SHOIN}$, etc.

4.3 Ontology Structural Description

We paid a special attention to characterize the taxonomic structure of an ontology, i.e. its inheritance hierarchy. The latter sketches the tree like structure of subsumption relationships between names classes $A \sqsubseteq B$ or named properties $R \sqsubseteq S$. We remind that a reasoner classification task infer implicit subsumption from the explicitly defined ones. So, the more the inheritance hierarchy is complex and oversized, the more the reasoning computational time may be important. In this category, we gathered various features that have been defined in literature to describe concept hierarchies. These are, basically, metrics widely used by ontology quality evaluation community [7, 25, 19]. The following subcategories describe the essence of the retained features.

4.3.1 Hierarchical Features

We build both concept hierarchy denoted by **HC** and property hierarchy denoted by **HP**. Then for each hierarchy, we measured the following features:

HC_MDepth, HP_MDepth it is the maximal depth of class and property respective hierarchies. This feature was identified by LePendu et al. [19] as a possible reasoning complexity source.

HC(HP)_MSibling, HC(HP)_ASibing : the maximal and the average number of subclasses (resp. sub-properties). This feature was called by Tartir et al. as *Inheritance Richness* [25]. The authors claimed that higher values of this feature would indicate that the ontology is deep. However, lower values would describe a shallow horizontal ontology having less detailed knowledge.

HC(HP)_Tangledness, HC(HP)_MTangledness : *tangledness* is owe to Gangemi et al. [7]. It measures the number of classes in an ontology with multiple superclasses. We computed this feature for both class and property hierarchy and we also recorded the maximal number of named superclasses (**HC(HP)_MTangledness**).

4.3.2 Cohesion Features

The literature provides a plethora of various metrics to design the *cohesion* in the ontology, otherwise the degree of relatedness of its entities. We opted for the cohesion metrics introduced by Faezeh et al. [6], since they cover different levels of knowledge in the ontology. Seeking for brevity, we did not report the cohesion's formula, yet a short description is given in the following:

HC_Cohesion, HP_Cohesion : these are respectively class and property cohesion. They are computed based on the number of direct and indirect hierarchical links.

OP_Cohesion : it is the object property cohesion. This feature is computed using the number of classes which have been associated through the particular object property (domain and range).

Ont_Cohesion : the ontology cohesion is simply a weighted aggregation of the previously defined cohesion metrics, i.e. **HC_Cohesion**, **HP_Cohesion** and the **OP_Cohesion**.

4.3.3 Schema Richness Features

Finally, we enriched the ontology structural category by two additional features proposed by [25]. These features are well known for ontology evaluation community as they are part of the *OntoQA* tool.

RRichness : Relationship richness reflects the diversity of relations in ontology. Formally, it is the ratio of the number of relations between classes that are not hierarchical w.r.t. total number of different types of the ontology relationships. We slightly modified the formal expression of this metric in order to be able to compute it without any translation of ontology into a graph.

AttrRichness : The attribute richness is defined as the average number of attributes per class. To compute this feature, we considered data properties as class attributes.

4.4 Ontology Syntactic Features

Our main purpose when collecting features for this category, is to quantify some of the general theoretical knowledge about ontology complexity sources, likely to impact reasoner performances and eventually causing unexpected reasoning results. To accomplish this purpose, we conducted an investigation about main reasoning algorithms [4, 21]. Thus, we gathered relevant ontology features, that have inspired the implementation of well known reasoning optimization techniques [13, 26]. Features of the current group are divided in 6 subcategories, covering specific aspects of some of the OWL entities. This organization was inspired by the definition of feature levels provided by Kang et al. [16].

4.4.1 Axioms level Features

In this category, we gathered features that attempt to characterize the different types of axioms as well as to assess their respective relevance in the ontology.

KB sub-parts features (KBF) : in Section 2, we recalled that a knowledge base (KB), which is in our case the ontology, has three main parts *TBox*, *RBox* and *ABox*, each of which has a specific set of axioms. Given this description, we introduced the features, **STBx**, **SRBx** and **SABx**, as the respective sizes in term of number of axioms of *TBox*, *RBox* and *ABox*. Roughly speaking, we recorded the size ratio of these KB subsets w.r.t. the ontology axiom size (**OAS**) and we denoted them **RTBx**, **RRBx**, **RABx**.

Axiom's Types Frequencies (ATF) : this is a set of 28 features, each of which corresponds to a particular OWL axiom type described in the OWL official specification[4]. By frequency we mean, the ratio between the number of

[3]For further details about OWL 2 profiles, the reader is kindly referred to http://www.w3.org/TR/owl2-profiles/.

[4]The OWL specification is available at http://www.w3.org/TR/owl2-syntax/

occurrences of a given axiom type, and the ontology axioms size (OAS).

Axiom Depth Feature (ADF) : we compute the maximal parsing depth of axioms in the ontology **Ax_MDepth**. It is a feature of common use, described by Sazonau et al [17]. An axiom depth is the number of this axiom levels of nested expressions. This feature attempts to capture the extend of structural complexity of an axiom in a given ontology. We added to this information, the average nesting depth **Ax_ADepth** of all axioms, in order to give insight about how common is this complexity type in the ontology.

4.4.2 Classes Constructors level Features

In previous reasoner prediction works [16, 17], authors simply counted axioms that involve potentially hard constructors. However, they missed that one constructor could be invoked more than once in the same axiom. Moreover, we believe that the *density* of use of constructors, may be a valuable indicator of the ontology complexity. In order to characterize these informations, we propose three features that we describe in the following:

Class Constructors Frequencies (CCF) : this is a set of 11 features, where each element is a specific constructor frequency in the ontology. Formally, given a class constructor cc_i belonging to the set of all OWL class constructors ($cc_i \in CC$), **CCR(cc_i)** is defined as the ratio of the cc_i total occurrences in each *TBox* axiom ($A_{tx} \in \mathcal{T}$), divided by the sum of all constructors occurrences. The value of a **CCR** feature ranges within the unit interval $[0, 1]$, i.e

$$CCR(cc_i) = \frac{\sum_{j=1}^{|\mathcal{T}|} Count(cc_i, A_{tx_j})}{\sum_{i=1}^{|CC|}\sum_{j=1}^{|\mathcal{T}|} Count(cc_i, A_{tx_j})}, cc_i \in CC. \quad (1)$$

Ontology Class Constructors Density (OCCD) : we proposed another feature to compute the overall constructors *"density"* in the ontology. Formally, it computes the ratio of the total number of all constructors occurrences, divided by the maximal possible number of constructors in the ontology. The latter is defined as the multiplication result of the total number of *TBox* axioms $|\mathcal{T}|$, by the maximal counted number of constructors in one *TBox* axiom. The value of **OCCD** ranges within the unit interval $[0, 1]$, i.e.

$$OCCD = \frac{\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|CC|} Count(cc_j, A_{tx_i})}{|\mathcal{T}| \times max(\sum_{j=1}^{|CC|} Count(cc_j, A_{tx}), \forall A_{tx} \in \mathcal{T})}. \quad (2)$$

Constructors Coupling Patterns (CCP) : we also defined a more sophisticated feature that examines particular combinations of constructors, that may increase the inference computational cost, whenever used in an axiom. The rational behind this proposition comes from lectures about the well known *Tableau* algorithm [4]. Each class constructor has its own expansion rule. Applied in a specific order, the rules may lead to a sharp increase in the size of the *Tableau* algorithm completion graph, and hence the reasoning cost. We have specified three particular patterns, which describe fragments with *"costly"* class constructors combinations. We detected the occurrences of each of these patterns by *SPARQL* [5] based queries, written for this purpose.

[5]The specification of SPARQL query language is available at http://www.w3.org/TR/sparql11-query/

The *CCP* set of patterns are described in the following.

- **IU** *(Intersection, union Pattern)*: an IU pattern is reported when a conjunction (`owl:intersectionOf`) of class expressions in an axiom is part of a disjunction (`owl:unionOf`) of class expressions in the same axiom, and vice versa. This pattern can manifest in one of the following forms:

$$\sqcap \quad (\ldots, \sqcup(C_1, C_2, \ldots), \ldots) \quad (3)$$

$$\sqcup \quad (\ldots, \sqcap(C_1, C_2, \ldots), \ldots) \quad (4)$$

- **EUvI** *(Existential, universal having intersection Pattern)*: an EUI pattern occurrence is defined by a conjunction of class expressions, that concurrently contains an existential restriction and a universal restriction associated to the same role r. This pattern can be manifested by one of the following forms:

$$\sqcap \quad (\ldots, \exists r.C, \forall r.D, \ldots) \quad (5)$$

$$C_1 \sqsubseteq \exists r.C \quad and \quad C_1 \sqsubseteq \forall r.D \quad (6)$$

- **CUvI** *(Cardinality, universal having intersection Pattern)*: The CUvI pattern is a particular case of the EUvI pattern, where existential restriction is replaced by some restriction forms ($\leq nr.C$, $\geq nr.C$, $= nr.C$).

4.4.3 Class level Features

Classes in the ontology could be named or specified via complex expressions. In this subcategory, we will highlight different methods to define classes and track their impact in the ontology *TBox* part.

Class Definition Features (CDF) : in [13], Horrocks established that restricting the Knowledge base (KB), $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ to unique and acyclic definition axioms, makes reasoning much easier, as the *unfolding* technique could be applied to all axioms. Concept definition axioms are primitive ones **PCD** of the form $A \sqsubseteq D$, or non primitive ones **NPCD** of the form $A \equiv D$, where A is an atomic concept name. However, real-world KBs commonly contain general definition axioms **GCI**. These axioms are of the form $C \sqsubseteq D$ or $C \equiv D$, where both C and D are complex class descriptions. They are known to be costly to reason with, due to the high degree of non-determinism that they introduce. Motivated enough, we proposed to record the number of each of these kind of class definitions (PCD, NPCD, GCI).

Cyclic Class Feature (CCyc) : we computed cyclic class definitions in the ontology and retained their ratio w.r.t. total number of named classes, i.e. the **SC** feature. In DL, a cyclic definition axiom is the one that references the same (or equivalent) classes (or properties) on both sides of the subsumption relation (i.e. $\exists P.C \sqsubseteq C$, or $P \circ P \sqsubseteq P$). Such an axiom may be explicit of inferred by a reasoner. We only computed the explicit ones by implementing the method described by Baader et al. [2].

Class Disjointness Feature (CDIJ) : this feature stands for the ratio of named classes declared as disjointed w.r.t. the class size **SC**. Our motivation to compute this feature is based on stated observations by Wang et al. [27]. The latter has conducted empirical studies on reasoners using his profiling tool *Tweezers*. They highlighted that there is a crucial need to characterize the "*right amount*" of disjointness statements to be put in an ontology, as some of them can greatly reduce the computational time when inferring an ontology, but also too many statements would remarkably increase the runtime.

4.4.4 Property level Features

In this subsection, we will characterize special features of the ontology properties, in particular, the ones describing the object properties.

Object Property Characteristics Frequencies (OPCF) : this is a set of 9 features in relation with particular object property characteristics. In OWL, the latter ones are defined using specific axioms that describe object properties transitivity, symmetry, reflexivity and etc. Horrocks and Tsarkov [13, 26] have respectively emphasized on the hardness of managing particular object property description characteristics, since they impact the effectiveness of some reasoning optimization techniques [6]. Hence, more sophisticated and probably costly reasoning procedures would be required to overcome these characteristics hardness level. We denoted the set of all object property characteristics as OPC and we defined $OPCF$ as an object property characteristic frequency. To made clear, when specifying an object property as transitive, only one axiom is required (`owl:TransitiveObjectProperty`). However, this object property name could be repeatedly involved in many other $TBox$ axioms and even more than once in one axiom. Consequently, reasoning techniques dealing with transitivity would be applied as far as this transitive object property is used. Formally, to compute $OPCF$ of a given object property characteristic $C_i^{op} \in OPC$, we start by collecting named object properties sharing C_i^{op}. We designed the latter set as $S(C_i^{op}) = \{OP_j, j \geq 0\}$. Then, we sum up the occurrences of each element in this collection and we denoted it $OPCO(C_i^{op})$. Later, this value is divided by the sum of the total characteristic occurrences, hence we obtain an $OPCF(C_i^{op})$ value ranging in $[0, 1]$. By computing this latter ratio, we would be able to flag out which characteristic of the object property have the highest impact on the reasoning process.

$$OPCO(C_i^{OP}) = \sum_{j=1}^{|S(C_i^{op})|} \sum_{k=1}^{|\mathcal{T}|} Count(OP_j, A_{tx_k}) \quad (7)$$

$$OPCF(C_i^{OP}) = \frac{OPCO(C_i^{OP})}{\sum_{j=1}^{|OPC|} OPCO(C_j^{OP})} \quad (8)$$

Number Restriction Features (NRF) : we studied the impact of using high values with object properties number restrictions. So, we retained for each cardinality type that is min, max and exact cardinality, its highest respective values. For instance, taking a restriction of the form $\geq nR.C$, we have recorded $max(n)$ of all restrictions having the same form. Thus, we build the set of highest values of cardinalities, and we denoted it (**HVC**). In addition, we computed the average value of used numbers for cardinality restrictions, and we denoted it (**AVC**). Worth of cite, the *Pellint* tool [20] reports an ontology pitfall when cardinality values exceed some predefined threshold. However, it's not known how values of this threshold should be set up.

4.4.5 Individual level Features

In this subcategory, we specify some of the interesting characteristics of named individuals that would be declared in the ontology.

[6]For example, *Internalisation* and *Caching* are less effective at the presence of inverse properties in the ontology

Nominal Frequency Features (NFF) : in the remainder Section 2, we mentioned that named individuals could be used in $TBox$ axioms to define new classes. In this case, individuals are designed as nominals. However, this modeling method come at a price, since nominals require specific reasoning procedures that would lower the runtime of the reasoner [26]. To capture the impact of using nominals, we counted their occurrences in $TBox$ axioms and retained its ratio (**NomTB**) w.r.t. the individuals size **SI**. In addition, we recorded **TBNom**, the ratio of axioms having nominals w.r.t. $TBox$ size (**STBx**).

Individual Similarity Features (ISF) : it is simply the ratio of named individuals defined as disjoint ones **IDISJ**, as well as the **ISAM** ratio of individuals declared as equal ones (`owl:sameAS`).

5. CASE STUDY: REASONER ROBUSTNESS PREDICTION

So far, we have described the state of art of works investigating on one hand, reasoner empirical behaviour and on the other hand, particular ontology characteristics. Then, we defined a set of ontology features, that we believe would be good indicator of its hardness against reasoning tasks. Now, we aim at assessing the effectiveness of our proposals. Consequently, we need a procedure to evaluate possible correlation between the empirical behaviours of reasoners and the described features. For this purpose, we choose to operate with supervised machine learning techniques, previously used to predict reasoner computational runtime [16, 17, 23]. Basically, these are mapping functions from a set of features that describe an ontology, to a label or a continuous or a discrete numeric value representing the predicted performance of a given reasoner when processing this ontology. In machine learning, it is well known that the choice of features has a significant impact on the accuracy of the predictive model. Good features, i.e. strongly correlated with the reasoner performances would lead to predictive models highly accurate. Based on this fact, we decided to carry on with the predictive approach to unveil the key features that would impact reasoner performances. Our main goal is to prove that our set of feature are very valuable in learning reasoner performances.

5.1 Evaluation methodology

In this section, we will first specify the performance criterion to be predicted and then steps to automatically learn it from the experimental data.

Robustness as a performance criterion.

In the previous works, the computational runtime or the time-bin of a reasoning task were the reasoner performance criteria considered for the prediction. Thus, these studies predict how fast is a reasoner, without paying any attention to how correct are the results of this engine. Recently in the Ontology Reasoner Evaluation Workshop ORE'2013 [10], the reasoner ***robustness*** is introduced as a more effective criteria for reasoner evaluation. Literally, ***robustness*** is defined as the number of ontologies that were correctly processed within the time limit. The reasoner output was checked for correctness by a majority vote, i.e. the result returned by the most reasoners was considered to be correct. ORE authors affirmed, that this was the most straightforward way to automatically determine correctness without

manual inspecting or artificially generating test cases. In our study, we adopted the ***robustness*** as the performance criterion to build reasoner prediction models. We distinguished four labels that would describe the termination state of a reasoning task carried by a reasoner R on an ontology O. These labels are: 1) *Correct* (**C**) standing for an execution achieved within the time limit and delivered correct results; 2) *Unexpected* **(U)** in the case of execution achieved within the time limit but delivered results that are not expected, otherwise reported incorrect; 3) *Timeout* **(T)** in the case of violating the time limit; and 4) *Halt* **(H)** describing a sudden stop of the reasoning process owing to some error. Thus, the label space $\mathcal{L}$ of our learning process, is expressed by the set $\{C, U, T, H\}$.

Learning steps.

We propose the following methodology for predicting the robustness of a given reasoner on individual ontologies. We believe that our research methodology would lead to unveil the key ontology features, likely to impact the reasoner robustness. Our learning steps are partially inspired by the earlier work of Kang et al. [16].

1. **Ontologies selection $\mathcal{C}(\mathcal{O})$:** to ensure the good quality of the prediction models, special attention should be paid to the identification and selection of the test set ontologies. As previously highlighted by Sazonau et al. [23] predictions trained from a biased corpus, towards easy to compute ontologies would cause misleading generalizations. Hence, the ontology corpus $\mathcal{C}(\mathcal{O})$ should be highly diversified in terms of ontology characteristics, in order to reduce the probability for a given reasoner to encounter only problems it is particularly optimised for.
2. **Features identification:** this step was detailed in Section 4, where we introduced a rich and comprehensive set of ontology features depicting our features space $\mathcal{F}^d$, with d denoting the dimension of this space.
3. **Reasoner selection ($R_k \in \mathcal{S}(\mathcal{R})$):** any reasoner part of a reasoners set $\mathcal{S}(\mathcal{R})$, preferably its latest version, would be enough for the study; no knowledge about its algorithm neither details about its internal workings are needed for the learning process. Intuitively, the reasoner performances are close to the reasoning task to be processed. Thus, we should specify which task to be modelled for the learning procedure. In our study, we picked up the ontology classification task, since its duration is often used as a performance indicator in the reasoner benchmarks (Section 2).
4. **Dataset building:** in this step, each ontology belonging to the corpus $O_i \in \mathcal{C}(\mathcal{O})$ is processed by the previously selected reasoner R_k to achieve the previously specified reasoning task. We will retain the termination state of this task, i.e. the label $l_{R_k}(O_i) \in \{C, U, T, H\}$. Then, the identified features will be computed for each ontology O_i in order to obtain its corresponding d-dimensional feature vector $X^{(O_i)} \in \mathcal{F}^d$. Having these steps achieved, the final training dataset of the selected reasoner R_k, designed by $\mathcal{D}_{R_k}$, is built in by gathering the pairs (feature vector, label), i.e. $\mathcal{D}_{R_k} = \{(X^{(O_i)}, l_{R_k}(X^{(O_i)})), i = 1 \dots N\}$, where $N = |\mathcal{C}(\mathcal{O})|$
5. **Preprocessing and Feature selection** machine learning algorithms learn from data, thus we need to make sure that our data are in a useful scale, format, and even that only the meaningful features are included in. In fact, a huge amount of features does not necessarily improve the accuracy of the prediction model. Commonly, feature selection or dimensionality reduction techniques are applied to filter features. For this purpose, we have deployed three different methods of feature selection, that we will describe in Section 5.3. Consequently, three variants of the initial reasoner dataset are established, called featured datasets, each with a different and eventually reduced subset of ontology features. The initial dataset is also maintained and called *"RawData"*. Afterwards, the four datasets will be compared to each other based on the accuracy of their derived predictive models. More explanation are given in step 7.
6. **Learning and assessing the models:** each reasoner dataset $\mathcal{D}_{R_k}^{(j)}$ constructed in the Step 5 is provided to a supervised machine learning algorithm to train its data in order to build a predictive model, M_{R_k}, for the selected reasoner R_k. As previously cited, this would be a function mapping an ontology feature vector $X^{(O)}$ to a prediction of the reasoner robustness label $\hat{l}_{R_k}(X^{(O)})$. Thus, it would be possible to assess the robustness of the reasoner R_k on any unknown[7] ontology provided that its feature vector $X^{(O)}$ is computed.

$$M_R \ : X \in \mathcal{F}^d \mapsto \hat{l}_R(X) \in \mathcal{Y} \tag{9}$$

   Worth to cite, we investigated 5 well known supervised machine learning algorithms. Thus, this step will be further repeated as far as the number of algorithms and the number of datasets of a given reasoner. Then, the established models will be evaluated against a bunch of assessment measures. The main goal of this step is to figure out the *best* predictive model for a given reasoner. Details about the learning algorithms and the assessment measures are given in the Section 5.4.
7. **Unveil the key features**: steps 3-6 are repeated K times to cover all reasoners in the set $\mathcal{S}(\mathcal{R})$, with $K = |\mathcal{S}(\mathcal{R})|$. As a result, K best predictive models are identified each for a reasoner, and each having its own most relevant feature subset. Thus, the key features are those the most involved across the whole set of best models. We denote this subset of features as *Global Key Features*, in contrast to *Local Key Features*, which designs features employed in a given reasoner best predictive model.

5.2 Data collection (Steps 1-4)

To ensure the reliability of reasoner evaluation results, we chose to reuse reported results of the ontology classification challenge, organized during the latest Ontology Reasoner Evaluation Workshop, ORE'2014 [5]. We retained ontologies as well as reasoners which have participated to this challenge. The motivation behind our choice is multifold: first, the event is widely recognized by the Semantic Web community; the ontology corpus is well selected and the competition gathers the latest and the more powerful known reasoners; and finally the description of reasoner results is consistent with the specification of the performance criterion we designed in the previous Section 5.1. In the following, the ontologies, the reasoners as well as the evaluation results will be shortly described.

Ontologies.

[7]This means not part of the initial training dataset.

we have previously mentioned that classification is the main reasoning task for our investigations. Thus, we selected ontologies from the ORE corpora[8] that have been used for this challenge. In overall, we retained 550 ontologies from both OWL DL and EL profiles, which sizes vary from small ([100,1000[axioms), going on to very large ones (>100 000 axioms). More than 75 distinct description logic family are covered in these ontologies. More details about the ontology corpora would be find in [5].

Reasoners.

after each ORE challenge, reasoners were ranked based on their robustness. To carry out our study, we selected the 6 best ranked reasoners[9] in both the DL and EL classification challenges. These reasoners are included in our reasoners set $\mathcal{S}(\mathcal{R})$. They are namely ***Konclude***, ***MORe***, ***HermiT***, ***TrOWL***, ***FaCT++*** and ***JFact***. We excluded ***ELK*** despite its good results and high rank, as it does not support DL ontologies. Indeed, each of the 6 selected reasoners must have an output label for each of the 550 ontologies. This was decided for the sake of generalization. We recall that a three minute time limit was given every reasoner for processing each ontology.

Training datasets.

Table 2 summarizes the ORE classification results [10]. We organised them according to the different prediction labels, that we have already specified for the reasoner robustness criterion. Thus, the table shows the number of ontologies recorded with a given label for a specific reasoner.

Later, we computed the 112 proposed features for each ontology. Thus, we constructed 6 training datasets each for a specific reasoner. It's important to notice that our reasoner's datasets are ***imbalanced***. Based on Bao-Gang et al. description [15], a dataset is considered imbalanced if one of the classes (minority class) contains much smaller number of examples than the remaining classes (majority classes). In our context, the classes are reasoner robustness labels. We take as an example the *Konclude*'s dataset described in Table 2, we can easily notice the huge difference between the number of ontologies labelled as *Correct* (the majority) and those labelled as *Unexpected* or *Timeout* (the minority). However, we would be much more interested in predicting the minor cases, specially the *Unexpected* one, as this is a critical situation for users aiming at using *Konclude* to process their ontology. The learning from imbalanced datasets is considered as one of the most challenging issues in the data mining [15]. We will be considering this aspect when building the predictive models and assessing them.

5.3 Feature Selection (Step 5)

Before training the models for the 6 reasoners, we performed some preprocessing steps. We started by cleaning the datasets. In the case of *Konclude* and *TrOWL*, the ontologies labelled with **(H)** were removed from the respective datasets, since the number of these cases is very small (≤ 2), thus not sufficient for the learning process and would even affect the overall model accuracy. A more crucial procedure is then applied, i.e. *feature selection*. It aims at selecting the most relevant features, by weeding out useless ones. In previous works, Kang et al. [16] have compared 6 feature selection methods. Whereas, Sazonau et al. [23] have deeply investigated the intercorrelation between ontology features, using the *Principal Component Analysis* (**PCA**). In our study, we are more interested in tracking out the subset of features, which correlate the most with the performances of a given reasoner. To fulfil this task, we choose to investigate the utility of a different feature selection technique, i.e. the *discretization* [8]. Basically, this is the transformation of continuous data into discrete bins. More specifically, we opted for the well known Fayyad & Irani's supervised discretization method (**MDL**), available in the machine learning working environment, *Weka* [12]. This method cuts the feature continuous values into categories by referring to the robustness label of the corresponding ontologies. Seeking of more validity, we decided to compare the discretization results to further feature selection techniques. To achieve this work, we carefully chose two well known methods representative of two distinct categories of feature selection algorithms: first, the **Relief** method, which finds relevant features based on their ranks; then, the *CfsSubset* **(CFS)** method, which selects subsets of features that are highly correlated with the target label while having low intercorrelations. These two methods are also available in the *Weka* environment. We remind that the initial dimension of our feature space $\mathcal{F}^d$ is 112. Table 3 summarizes the feature selection results when applied to $\mathcal{F}^d$. The reported values are the size of feature subsets selected by the respective methods. We further investigated the possible presence of common features across these subsets by computing their intersections. It would be observed that feature subsets computed by the *CFS* method

Table 2: Ontology classification results. The time is given in seconds. #C, #U, #T and #H stand respectively for the number of ontologies labelled by *Correct*, *Unexpected*, *Timeout* or *Halt*

Reasoner	#C	#U	#T	#H	Runtime		
					Min	Max	Mean
Konclude	521	12	16	1	0.04	182.75	3.37
MORe	470	12	68	0	1.09	150.68	8.03
HermiT	452	1	97	0	0.59	150.12	16.83
TrOWL	455	51	42	2	0.47	173.26	10.54
FaCT++	388	5	143	14	0.47	157.50	7.50
JFact	294	5	218	33	0.63	141.44	14.3

Table 3: Summary of feature selection results. ($\cap$) stands for the intersection of feature sets.

Dataset	MDL	Relief	CFS	($\cap$)
$\mathcal{D}$(Konclude)	27	97	8	8
$\mathcal{D}$(MORe)	72	104	18	17
$\mathcal{D}$(HermiT)	67	94	21	16
$\mathcal{D}$(TrOWL)	85	97	17	17
$\mathcal{D}$(FaCT++)	76	95	17	15
$\mathcal{D}$(JFact)	94	96	17	15
($\cap$)	17	82	0	0

[8]The whole corpus of ontologies is available for download at http://zenodo.org/record/10791

[9]All ORE'2014 reasoners are available for download at https://zenodo.org/record/11145/

[10]A detailed description of ORE 2014 results is available at https://zenodo.org/record/11142/

are the most compact ones. In contrast, *Relief*'s subsets are almost equal to the initial set of features. On the other hand, the size of feature subsets decided by *MDL* scales differently from one reasoner dataset to another. Moreover, given a reasoner dataset, common features were found across the different computed features subset. However, given a selection method, less common features are identified over the datasets of the different reasoners, and more specially no common features is found using the *CFS* method. Intuitively, this would be up to the diversity in reasoning techniques deployed by the respective reasoning engines. Even they all share the same core reasoning algorithm, i.e. the *Tableau algorithm*, each reasoner designer have evolved it using distinct optimization techniques. Certainly, at this stage, it is hard to decide which feature subset is having the most relevant features for a given reasoner. This would be concluded only after training predictive models from the respective distinct feature subsets. The one leading to the most accurate predictive model, will be recognized as the key ontology features, with respect to reasoner robustness criterion. To conduct the comparison, for each reasoner three additional datasets are built, called *featured datasets*. As mentioned in the Section 5.1, the initial dataset with the full feature set is maintained and called ***RAWD***. The others will got the same name as the employed feature selection technique, i.e. **MDL**, **RELIEF** and **CFS**.

5.4 The Learning Methods (Step 6)

We start by describing the set of supervised machine learning algorithms selected for our study, as well as the assessment measures. Then, we will explain the training steps and the procedure of reasoner best predictive model selection.

Supervised machine learning algorithms.

Seeking for diversity in the learning process, we selected 5 candidate algorithms, each one is representative of a distinct and widely recognized category of machine learning algorithms [18]. These algorithms are available in the *Weka* environment, and was applied with the default parameters of this tool. They are namely: *1) Random Forest* **(RF)** a combination of C4.5 decision tree classifiers; *2) Simple Logistic* **(SL)** a linear logistic regression algorithm; *3) Multilayer Percetron* **(MP)** a back propagation algorithm to build an Artificial Neural Network (ANN) model; *4)* ***SMO*** the Support Vector Machine (SVM) learner with a sequential minimal optimization; and finally *5)* ***IBk*** a K-Nearest-Neighbour algorithm with normalized euclidean distance.

Prediction accuracy measures.

When looking for accuracy measures to assess the effectiveness of the built in models, we paid special attention to select the ones known for their appropriateness in the case of imbalanced data. In previous works [16, 17], the accuracy standing for the fraction of the correct predicted labels out of the total number of the dataset samples, was adopted as main evaluation metric for assessing the predictive models. However, accuracy would be misleading in the case of imbalanced datasets as it places more weight on the majority class(es) than the minority one(s). Consequently, high accuracy rates would be reported, even if the predictive model is not necessarily a good one. Based on the comparative study conducted by Bao-Gang et al. [15], we chose the most powerful assessment metrics in the case of imbalanced data. Worth to cite, all of these measures are computed based on the *confusion matrix*, that we will describe at first:

- ***Confusion Matrix*** it is a square matrix, $L \times L$, where L is the number of labels to be predicted. It shows how the predictions are made by the model. The rows correspond to the known labels of the data, i.e. in our case $\{C, U, T, H\}$. The columns correspond to the predictions made by the model. Each cell (i, j) of the matrix contains the number of ontologies from the dataset that actually have the label i and were predicted as with label j. A perfect prediction model would have its confusion matrix with all elements on the diagonal.
- ***Precision*** (**PR**), ***Recall*** (**RC**), and ***F-Measure*** (**FM**): these are common measures of model effectiveness. *Precision* is a measure of how many errors we make in predicting ontologies as being of some label l. On the other hand, *Recall* measures how good we are in not leaving out ontologies that should have been predicted with the label l. To be noted, these two measures are misleading when used in isolation. Usually, we rather employ the *F-Measure* (**FM**) to assess the model. This measure combines both *Precision* and *Recall* into a single value: $FM = \frac{2 \times RC \times PR}{RC + PR}$.
- ***Kappa coefficient*** (κ): it is used to measure the agreement between the predicted labels and the real labels. The value of *Kappa* lies between -1 and 1, where 0 represents agreement due to chance only. 1 represents complete agreement between the two values. In rare situations, *Kappa* can be negative.
- ***Matthews correlation coefficient*** (**MCC**): it is performance measure barely influenced by imbalanced test sets since it considers mutually accuracies and error rates on all labels, so it involve all values of the confusion matrix. *MCC* ranges from 1 for a perfect prediction to -1 for the worst possible prediction. *MCC* close to 0 indicate a model that performs randomly.

Training and selecting the best predictive models.

A reasoner dataset, D_R, is trained by each of the abovementioned supervised machine learning algorithm. Thus, five distinct predictive models are learned. We made use of the standard *cross-validation* technique for evaluating these models. We applied a stratified 10-fold cross-validation to ensure the generalizability of the results. For the sake of simplicity, we only retained the average values of the computed assessment measures over all the cross-validation steps.

We recall, our study cover 6 reasoners, for each 4 datasets were built in and then provided to 5 supervised learning algorithms, each of which have train it separately and produced its corresponding predictive model. In overall, 20 models were learned for every reasoner, and assessed using 3 distinct evaluation measures. All these steps were put forward to reveal the reasoner best predictive model, according to the aforementioned assessment measures. Being aware of the amount of data to analyse, we propose to compute a score index per model. By referring to it, we will be able to establish a total order of the reasoner's predictive models. We called it, the *Matthews Kappa Score* (***MKS***). As the name suggests, *MKS* is a weighted aggregation of the *MCC* and the *Kappa* values computed to assess a reasoner model M_R.

$$MKS(M_R) = \frac{\alpha * MCC(M_R) + \beta * Kappa(M_R)}{\alpha + \beta} \quad (10)$$

Table 4: Assessment summary of the reasoners' models learned from *RAWD* datasets

	Konclude			MORe			HermiT			TrOWL			FaCT++			JFact		
M_R	FM	MCC	κ	FM	MCC	κ	FM	MCC	κ	FM	MCC	κ	FM	MCC	κ	FM	MCC	κ
RF	0.95	0.37	0.30	**0.95**	**0.78**	**0.78**	**0.95**	**0.85**	**0.84**	0.90	0.66	0.64	**0.92**	**0.83**	**0.83**	**0.92**	**0.87**	**0.87**
SL	**0.96**	**0.49**	**0.39**	0.94	0.78	0.78	0.95	0.83	0.83	0.80	0.56	0.54	0.88	0.74	0.73	0.90	0.83	0.82
ML	0.95	0.32	0.28	0.94	0.78	0.77	0.95	0.82	0.82	0.90	0.65	0.66	0.86	0.69	0.67	0.77	0.58	0.58
IBk	0.95	0.32	0.27	0.93	0.71	0.71	0.92	0.74	0.74	**0.90**	**0.67**	**0.65**	0.87	0.72	0.71	0.86	0.74	0.74
SMO	0.96	0.40	0.40	0.94	0.78	0.77	0.94	0.78	0.78	0.89	0.63	0.64	0.88	0.83	0.83	0.84	0.71	0.70
M^d_{RBest}	**SL (MKS: 0.44)**			**RF (MKS: 0.78)**			**RF(MKS: 0.84)**			**IBk (MKS: 0.66)**			**RF (MKS: 0.83)**			**RF (MKS: 0.87)**		

where $(\alpha + \beta) = 1$. Thus, the best predictive model for a given reasoner is the one having the maximal value of the *MKS* score $M_{RBest} = arg \max_{M^i_R \in \mathcal{MR}} (MKS(M^i_R))$, where $\mathcal{MR}$ denotes the set of predictive models learned for a particular reasoner R. In the case where multiple maximal models are identified, the model with the highest *F-Measure* (FM) is selected. We believe that using *MKS* is the straightforward way to automatically determine the best robustness predictive model for a given reasoner. The rational behind this proposal is twofold: first, *MCC* and *Kappa* are widely recognized as powerful assessment measures, more effective that *FM*, specially in the case of imbalanced datasets; second both measures are ranging in $[-1, 1]$, which makes the aggregation coherent.

5.5 The Learning Results and Discussion

The selection of a reasoner best predictive model is achieved within two stages: the best model given a specific reasoner dataset variant; then the best model across all the datasets. We start by discussing the results of the first stage in the case of RAWD datasets. Next, we report the final selection the best predictive model of each reasoner. Afterwards, the identified key features will be examined and analysed.

Best models from RAWD datasets.

For the sake of brevity, we only report the assessment results of the predictive models learned from reasoner RAWD datasets. We remind, in this type of datasets, ontologies' feature vectors are full ones, counting 112 distinct feature. Table 4 shows the distributions of *F-Measures* (FM), *MCC* and *Kappa* (κ) across the 5 learned models, trained for every reasoner. The ending line of the table displays the name of the reasoner best predictive model under this dataset description, denoted M^d_{RBest}. The selection was made according to the MKS values[11].

A number of important observations can be made from this table. First, it would be observed that the MAX value of the F-Measure (FM) for the 6 reasoner ranges from 0.90 by IBK in the case of TrOWL, to 0.96 by SL in the case of Konclude. These close to optimum FM values indicate that our proposed set of ontology features entails highly accurate predictive models, even when no feature selection or dimensionality reduction techniques are deployed a priori. Thereby, the high correlation between our ontology features and the reasoners' robustness is confirmed. However, the examination of the MCC and Kappa values spots some weakness in the Konclude's models, as these values are mostly less than 0.5. This reveals the hardness of predicting Konclude's minor classes, i.e. *Timeout* and *Unexpected*. We recall the Konclude's dataset, described in table 2, is highly biased towards *Correct* cases, as the reasoner has fewer times failed to respect the time limit ($\leq 2.9\%$), or to correctly infer ontologies ($\leq 2.2\%$). To overcame the model weakness, we need to increase the minor classes' ratio of samples. This would be accomplished by conducted much more experiments where Konclude would be faced to more resisting ontologies, or by using an over sampling technique. In overall, reasoner best models have achieved good MCC and Kappa values, as their MKS scores are ranging between 0.66 (by IBK for TrOWL) to 0.87 (by RF for JFact). It would also be noticed that there is no dominant learning algorithm for all the reasoners. Obviously, the RF algorithm is the most performing one, as it derived the best predictive models for 4 over 6 reasoners. However, SL and IBK algorithms have outperformed RF when handling the most biased datasets, i.e. the Konclude's dataset and the TrOWL's one.

Best models across the dataset types.

In Table 5, the assessment results of best models derived from the original datasets, RAWD, are compared with their respective counterparts learned from featured datasets, i.e. MDL, RELIEF and CSF. For each reasoner, the best model given a dataset type is reported as well as its measured MKS and FM values. The last line of Table 5 sum ups the comparison by revealing, the across datasets, reasoner best predictive model, denoted M_{RBest}. The dataset type and the size of the ontology feature vector of M_{RBest} are also indicated. Its assessment results are denoted in boldface.

For all reasoners, it could be noticed that predictive models derived by the discretization technique, MDL, are better than the ones learned from RAWD. This is according to both MKS and FM measures. However, the accuracy of models trained from CFS and RELIEF featured datasets, are less stable. In some cases, they could exceed or eventually fall behind the FM or the MKS values measured for the RAWD models. Yet, not once the RAWD dataset, with its entire set of 112 ontology features, is listed in the final selection of the reasoners' best models. The size of feature vectors of the M_{RBest} models varies from 8 in the case of Konclude going to 94 in the case of JFact. This fact pinpoints that for each reasoner there is a specific number of key ontology features that have more impact on his robustness. Therefore, it is quite certain, that restricting the full initial set of ontology features to some particular subsets would improve the reasoners' predictive models. Nevertheless, we are not

[11] In our experimentations, we set up $\alpha = \beta = 0.5$ in order to compute a model's MKS score.

Table 5: Best reasoners' models across the different types of datasets

Dataset variant	Konclude			MORe			HermiT			TrOWL			FaCT++			JFact		
	M^d_{best}	MKS	FM	M^d_{best}	MKS	FM	M^d_{best}	MKS	FM	M^d_{best}	MKS	FM	M^d_{best}	MKS	FM	M^d_{best}	MKS	FM
RAWD	SL	0.44	0.96	RF	0.78	0.95	RF	0.84	0.95	IBk	0.66	0.90	RF	0.83	0.92	RF	0.87	0.92
MDL	SL	0.48	0.96	**SL**	**0.83**	**0.96**	**SMO**	**0.88**	**0.96**	**SL**	**0.78**	**0.93**	**RF**	**0.84**	**0.93**	**SMO**	**0.91**	**0.95**
RELIEF	SL	0.44	0.96	RF	0.78	0.94	RF	0.84	0.95	RF	0.64	0.90	RF	0.83	0.92	RF	0.87	0.93
CFS	**SL**	**0.53**	**0.96**	RF	0.76	0.94	RF	0.82	0.95	RF	0.73	0.92	RF	0.83	0.92	RF	0.87	0.92
M_{RBest}	SL+CSF(8f)			SL+MDL(72f)			SMO+MDL(67f)			SL+MDL(85f)			RF+MDL(76f)			SMO+MDL(94f)		

sure which feature selection technique would be the most suited to discover these features. In fact, the MDL method have outperformed in major cases, but beaten by the CFS's technique in one reasoner instance, i.e. Konclude. For the latter one, CFS have delivered more significant and compact feature set which leaded to the Konclude's best predictive model. On the other hand, SL, SMO and RF are sharing the podium of the most performing supervised learning algorithms. Considering these findings, we would confirm that there is no ultimate best combination, i.e. feature selection technique and learning algorithm, that suits all the reasoners. The latter ones deploy different reasoning techniques and therefore, their robustness against the same ontologies may also be significantly different. Accordingly, we admit that even if the learning process may be maintained the same, the learning techniques must be diversified to better capture the reasoner-specific empirical behaviour.

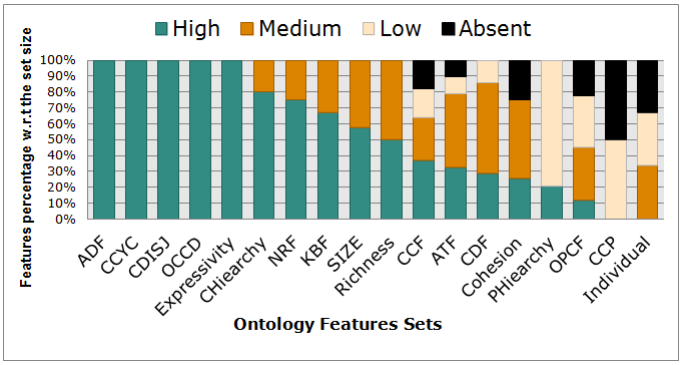

Figure 1: Frequency levels of ontology feature sets

Unveil the key features.

The high accuracy of the reasoner robustness predictive models trained from featured datasets confirmed the validity of our assumption that particular ontology features would help tracking reasoner empirical behaviour. We must recognize another valuable advantage of these selected features, since in major cases they have induced linear best predictive models, i.e. those based on SL or SMO. Besides being highly effective, these linear models are known to be intuitive and simple to explain. Henceforth, the robustness of reasoners would be explained in terms of the reduced set of ontology features, involved in their respective best predictive models. We have called these subsets, a reasoner *local key features*. We have also investigated possible presence of common features across reasoners' best predictive models. We called these features, *global key features*. We tracked their contribution to the reasoner robustness prediction, by computing their occurrences in the best models. A group of 6 features were recognized to be the most frequent ontology features as they were involved in each of the 6 best predictive models. They are namely: the average depth of the ontology axioms **Ax_ADepth**, the highest value of the min and the exact number restrictions, i.e. **HVC(min)** and **HVC(exact)**, and the three members of the set of class constructors frequency(**CCF**): **intersection**, **exact cardinality** and **hasSelf**. To have further insights about the contribution of the other features in building the best models, we distinguished four levels of feature frequencies. Given O_f the number of occurrences of the feature f in the set of the 6 best models, f has *high* frequency if $O_f \geq 5$, *medium* one if $3 \leq O_f \leq 4$, *low* in case $1 \leq O_f \leq 2$ and eventually a feature would be *absent* having $O_f = 0$. In overall, 46 features were highly frequent, 37 were medium ones, 17 had low frequencies and 12 never been used in the set of reasoner best predictive models. As we can not detail the frequencies of each of the 112 feature, we choose to report the distribution of feature subcategories, described in Section 4, over the four frequencies' levels. Figure 1 illustrates these distributions.

We would observe that features describing the axiom's depth (**ADF**), the amount of cyclic (**CCYC**) and disjoint (**CDISJ**) classes as well as the ontology expressivity and the class constructors density (**OCCD**) are absolutely frequent ones. In other words, none of these features were found in less than 5 out of the 6 best predictive models. These findings agree with subsequent observation established in previous works about the significance of these particular features (Section 4). On the other hand, feature subcategories such as the number restriction features (**NRF**) and the class constructor frequency (**CCF**) are including high but also some medium and even low frequent members. Thus, they must be further analysed to pinpoint their most relevant elements. Whereas, features describing individuals and class constructor patterns were the less involved in the best models. We must stress that we are not concluding that the low frequent features are unimportant for the reasoners. Quite the contrary, these features could be highly specific to a particular reasoner and thus a strong indicator of its behaviour.

According to the established inspections, we would assume that the highly frequent features among best reasoner predictive models form a core group of good indicators of reasoner robustness considering the ontology classification task. These features would be recommended as starting points to conduct further improvement in ontology modelling as well as reasoning optimization. However, we must retain that each reasoner has its particular ontology key features, likely to have the strongest impact on his performances.

6. CONCLUSION

In this paper, we conducted a rigorous investigation about ontology key features likely to impact reasoner performances. Our main purpose was to be able to explain the empirical behaviour of reasoners when inferring particular ontologies. To fulfil this purpose, we learned predictive models of the robustness of 6 well known reasoners. We put into comparison various supervised learning algorithms and feature selection techniques in order to train best reasoner predictive models. Thanks to this predictive study, we unveiled sets of local and global ontology key features likely to give insights of ontology hardness level against reasoners. We want to stress that these features are key ones under the selected reasoner performance criterion, i.e. the *robustness*. However, we cannot confirm that it would be the same ones when moving to different criteria. Investigating this point would be interesting, as to gain more insights about the most relevant features across different reasoning performances criteria. In our case, this would be easily established, since the whole learning steps described in this paper are implemented in one single prototype. Our implementation is generic enough, that it would be applied to any reasoner, with the only requirement of providing enough amount of its running results against diverse ontologies. Further research perspectives would be opened by our present work. We assume that the most important one would be extending our learning steps by a ranking stage where reasoners could be compared based on their predicted robustness. Such ranking would made it possible to recommend the most robust reasoner for any given ontology.

7. REFERENCES

[1] S. Abburu. A Survey on Ontology Reasoners and Comparison. *International Journal of Computer Applications*, 57(17):33–39, 2012.

[2] F. Baader, S. Borgwardt, and B. Morawska. Unification in the description logic $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. Ltcs-report, Institute for Theoretical Computer Science, Technische Universität Dresden, Germany, 2011.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, USA, 2003.

[4] F. Baader and U. Sattler. Tableau algorithms for description logics. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, 2000.

[5] S. Bail, B. Glimm, E. JimÃl'nez-Ruiz, N. Matentzoglu, B. Parsia, and A. Steigmiller. Summary ORE 2014 Competition. In *the 3rd Workshop on OWL Reasoner Evaluation, Vienna, Austria*, 2014.

[6] E. Faezeh and D. Weichang. Canadian semantic web. chapter A Modular Approach to Scalable Ontology Development, pages 79–103. Springer US, 2010.

[7] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. In *Proceedings of the 3rd European Semantic Web Conference*, 2006.

[8] S. Garcia, J. Luengo, J. SaÌAez, V. LoÌApez, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25:734–750, 2013.

[9] B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos. A novel approach to ontology classification. *Web Semant.*, 14:84–101, 2012.

[10] R. S. Gonçalves, S. Bail, E. Jiménez-Ruiz, N. Matentzoglu, B. Parsia, B. Glimm, and Y. Kazakov. Owl reasoner evaluation (ore) workshop 2013 results: Short report. In *ORE*, pages 1–18, 2013.

[11] R. S. Gonçalves, B. Parsia, and U. Sattler. Performance heterogeneity and approximate reasoning in description logic ontologies. In *Proceedings of the 11th International Conference on The Semantic Web*, pages 82–98, 2012.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, and P. Reutemann. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11:10–18, 2009.

[13] I. Horrocks. Implementation and optimisation techniques. In *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 9, pages 306–346. Cambridge University Press, 2003.

[14] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence*, pages 57–67, 2006.

[15] B. Hu and W. Dong. A study on cost behaviors of binary classification measures in class-imbalanced problems. *CoRR*, abs/1403.7100, 2014.

[16] Y.-B. Kang, Y.-F. Li, and S. Krishnaswamy. Predicting reasoning performance using ontology metrics. In *Proceedings of the 11th International Conference on The Semantic Web*, pages 198–214, 2012.

[17] Y.-B. Kang, Y.-F. Li, and S. Krishnaswamy. How long will it take? accurate prediction of ontology reasoning performance. In *Proceedings of Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 80–86, 2014.

[18] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the Emerging Artificial Intelligence Applications in Computer Engineering Conference.*, pages 3–24, The Netherlands, 2007. IOS Press.

[19] P. LePendu, N. Noy, C. Jonquet, P. Alexander, N. Shah, and M. Musen. Optimize first, buy later: Analyzing metrics to ramp-up very large knowledge bases. In *Proceedings of The International Semantic Web Conference*, pages 486–501. Springer, 2010.

[20] H. Lin and E. Sirin. Pellint - A Performance Lint Tool for Pellet. In *Proceedings of the OWL Experiences and Directions Workshop*, volume 432, Germany, 2008.

[21] B. Motik, R. Shearer, and I. Horrocks. Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[22] L. Obrst, B. Ashpole, W. Ceusters, I. Mani, and B. Smith. Semantic web. chapter The evaluation of ontologies - Toward Improved Semantic Interoperability, pages 139–158. Springer US, 2007.

[23] V. Sazonau, U. Sattler, and G. Brown. Predicting Performance of OWL Reasoners: Locally or Globally? In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 2014.

[24] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semant.*, 5:51–53, 2007.

[25] S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. OntoQA: Metric-based ontology quality analysis. In *Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

[26] D. Tsarkov, I. Horrocks, and P. F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *J. of Automated Reasoning*, 39(3):277–316, 2007.

[27] T. D. Wang and B. Parsia. Ontology Performance Profiling and Model Examination: First Steps. In *Proceedings of The 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, pages 595–608, 2007.

[28] H. Zhang, Y.-F. Li, and H. B. K. Tan. Measuring design complexity of semantic web ontologies. *J. Syst. Softw.*, 83(5):803–814, 2010.