2015

OWL Query Answering using Machine Learning

Todd Huster
Wright State University

OWL Query Answering Using Machine Learning

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science

By

TODD HUSTER
B.S., Cedarville University, 2006

2015
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

September 3, 2015

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Todd Huster ENTITLED OWL Query Answering Using Machine Learning BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Pascal Hitzler, Ph.D.
Thesis Director

Mateen Rizki, Ph.D.
Department Chair

Committee on
Final Examination

Pascal Hitzler, Ph.D

Michelle Cheatham, Ph.D

John Gallagher, Ph.D

Robert Fyffe, Ph.D.
Vice President for Research and
Dean of the Graduate School

ABSTRACT

Huster, Todd. M.S. Department of Computer Science and Engineering, Wright State University, 2015. OWL Query Answering Using Machine Learning.

The formal semantics of the Web Ontology Language (OWL) enables automated reasoning over OWL knowledge bases, which in turn can be used for a variety of purposes including knowledge base development, querying and management. Automated reasoning is usually done by means of deductive (proof-theoretic) algorithms which are either provably sound and complete or employ approximate methods to trade some correctness for improved efficiency. As has been argued elsewhere, however, reasoning methods for the Semantic Web do not necessarily have to be based on deductive methods, and approximate reasoning using statistical or machine-learning approaches may bring improved speed while maintaining high precision and recall, and which furthermore may be more robust towards errors in the knowledge base and logical inconsistencies.

In this thesis, we show that it is possible to learn a linear-time classifier that closely approximates deductive OWL reasoning in some settings. In particular, we specify a method for extracting feature vectors from OWL ontologies that enables the ID3 and AdaBoost classifiers to approximate OWL query answering for single answer variable queries. Amongst other ontologies, we evaluate our approach using the LUBM benchmark and the DCC ontology (a large real-world dataset about traffic in Dublin) and show considerable improvement over previous efforts.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

To Jen, Sam, and Caleb, bringers of comfort and joy.

Thanks to Pascal for his direction and unquenchable optimism and to David Carral for his extensive input.

1

Introduction

The formal semantics of the Web Ontology Language OWL [Hitzler et al. 2012] enables automated reasoning over OWL knowledge bases, which in turn can be used for a variety of purposes including knowledge base development, querying and management. Automated reasoning is usually done by means of deductive (proof-theoretic) algorithms which are either provably sound and complete [Hitzler et al. 2010] or employ approximate methods to trade some correctness for improved efficiency [Groot et al. 2005; Rudolph et al. 2008].

It has been argued, though, that reasoning methods for the Semantic Web do not necessarily have to be based on deductive methods, but that approximate reasoning using statistical or machine-learning approaches, etc., may bring improved speed while maintaining high precision and recall, and which furthermore would be more robust towards errors in the knowledge base and towards logical inconsistencies, under which deductive reasoning is often not feasible or even possible [Fensel and van Harmelen 2007; Hitzler and van Harmelen 2010].

While this general idea of using inductive methods for approximating deductive reasoning has been around for a while, both inside and outside the Semantic Web world, progress has been relatively slow, and only a few recent publications directly address the issue (see Section 4). In fact, most of them, to the best of our knowledge, rely on kernels or distance measures, which can lose the power to discriminate in high dimensional or binary data.

In this thesis, we focus on testing the following proposition:

Proposition 1. *Given a particular query answering task, it is possible to learn an efficiently computable function that closely estimates a deductive reasoner.*

We will demonstrate that this proposition holds in a variety of cases by proposing and testing a particular solution to the problem. We will show how to select a set of features that is compact, retrievable in linear time (w.r.t. the size of the ABox), and expressive enough to allow accurate

classification, with AdaBoost or ID3 serving as the classifier. We will furthermore demonstrate the effectiveness of our approach on a variety of datasets and queries.

The plan of this thesis is as follows. In Section 2 we recall preliminaries regarding the description logic underlying OWL and recast OWL query answering as a machine learning classification problem. In Section 3 we discuss the feature space which is central to our investigations and briefly introduce the well-known classification methods which we use in our evaluation. In Section 4 we discuss related work and in Section 5 we report on our evaluation. Section 6 concludes and suggests future directions for research.

2

Preliminaries

2.1 The DL $\mathcal{SROIQ}$

In this section we introduce the DL language $\mathcal{SROIQ}$ [Horrocks et al. 2006]. We assume basic familiarity with DL and refer the reader to the literature for more details: for a thorough theoretical introduction see [Baader et al. 2007]; a textbook introduction to DL and its relation to Semantic Web technologies and standards is provided in [Hitzler et al. 2010].

A *DL signature* is a tuple $\Sigma = (N_{\mathsf{C}}, N_{\mathsf{R}}, N_{\mathsf{I}})$ where N_{C}, N_{R} and N_{I} are finite and mutually disjoint sets of *concept names*, *role names* and *individuals* such that $\{\bot, \top\} \subseteq N_{\mathsf{C}}$. Given Σ, let $N_{\mathsf{R}}^{-} = N_{\mathsf{R}} \cup \{R^{-} \mid R \in N_{\mathsf{R}}\}$ be the set of *roles*. Let Inv be a function over roles such that $\mathsf{Inv}(R) = R^{-}$ and $\mathsf{Inv}(R^{-}) = R$ for every $R \in N_{\mathsf{R}}$. For the rest of this paper we assume that a signature Σ has been fixed and so omit further references to it when clear from the context.

The set N_{E} of $\mathcal{SROIQ}$ *concept expressions* is as follows

$$N_{\mathsf{E}} ::= \neg N_{\mathsf{E}} \mid (N_{\mathsf{E}} \sqcap N_{\mathsf{E}}) \mid (N_{\mathsf{E}} \sqcup N_{\mathsf{E}}) \mid \exists R.N_{\mathsf{E}} \mid \forall R.N_{\mathsf{E}} \mid$$
$$\geq n\, R.N_{\mathsf{E}} \mid \leq n\, R.N_{\mathsf{E}} \mid A \mid \exists R.\mathsf{Self} \mid \{a\}$$

where $A \in N_{\mathsf{C}}$, $R \in N_{\mathsf{R}}^{-}$ and $a \in N_{\mathsf{I}}$. We refer to $N_{\mathsf{E}} \setminus N_{\mathsf{C}}$ as the set of *complex concepts*.

A $\mathcal{SROIQ}$ *axiom* is one of the following formulas

$$C \sqsubseteq D \qquad R_1 \circ \ldots \circ R_n \sqsubseteq R \qquad C(a) \qquad R(a, b)$$

where $C, D \in N_{\mathsf{E}}$, $R_{(i)} \in N_{\mathsf{R}}^{-}$ and $a, b \in N_{\mathsf{I}}$.

As usual, we refer to axioms of the form $C \sqsubseteq D$ as *general concept inclusions (GCIs)*. Axioms such as $R_1 \circ \ldots \circ R_n \sqsubseteq R$ are called *role inclusion axioms (RIAs)*. The remaining types, namely $C(a)$ and $R(a, b)$, are referred as *assertions*. The set of inclusion axioms for an ontology are colloquially referred to as the *TBox*, while the set of assertion axioms are referred to as the *ABox*.

Given a set of Horn-$\mathcal{SROIQ}$ axioms $\mathcal{S}$, we denote with $\sqsubseteq_{\mathcal{S}}$ the minimal relation over roles such that $S \sqsubseteq_{\mathcal{S}} R$ and $\mathsf{Inv}(S) \sqsubseteq_{\mathcal{S}} \mathsf{Inv}(R)$ hold if $S \sqsubseteq R \in \mathcal{S}$. We define $\sqsubseteq^*_{\mathcal{S}}$ as the reflexive-transitive closure of $\sqsubseteq_{\mathcal{S}}$. A role R is *simple with respect to* $\mathcal{S}$ if and only if for every $S \circ V \sqsubseteq R' \in \mathcal{S}$ we have that $R' \not\sqsubseteq^*_{\mathcal{S}} R$.

A $\mathcal{SROIQ}$ *ontology* $\mathcal{O}$ is a set of axioms which additionally satisfies the following conditions:

1. If a role R occurs alongside a concept expression of the form $\leq n\, R.C$ or $\geq n\, R.C$ in $\mathcal{O}$ then R is simple.

2. There exist a strict (irreflexive) partial order $\prec$ over N^-_{R} such that for every R we find $S \prec R$ if and only if $\mathsf{Inv}(S) \prec R$, and every RIA in $\mathcal{O}$ is of one of the following forms

$$R \circ R \sqsubseteq R \qquad R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R \qquad S_1 \circ \ldots \circ S_n \sqsubseteq R$$
$$\mathsf{Inv}(R) \sqsubseteq R \qquad S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$$

   such that $R, S_{(i)} \in N^-_{\mathsf{R}}$, $S_i \prec R$ for all $i = 1, \ldots, n$.

The semantics of an $\mathcal{SROIQ}$ ontology $\mathcal{O}$ is specified by providing a model theory, from which notions such as logical entailment can be derived. The basis for this approach is the notion of an *interpretation*. An interpretation $\mathcal{I}$ for an ontology $\mathcal{O}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that maps each $A \in N_{\mathsf{C}}$ to a subset of $\Delta^{\mathcal{I}}$, each $R \in N_{\mathsf{R}}$ to a binary relation over $\Delta^{\mathcal{I}}$ and each $a \in N_{\mathsf{I}}$ to an element of $\Delta^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is extended to concept expressions, roles and role chains as shown in the upper part of Figure 2.1. An interpretation $\mathcal{I}$ *satisfies* an axiom α, written $\mathcal{I} \models \alpha$, if the corresponding condition shown in the lower part of Figure 2.1 holds.

We proceed now with the introduction of *conjunctive queries*. Let N_{V} be a countably infinite set of variables disjoint from N_{C}, N^-_{R} and N_{I}. A *term* is an element from $N_{\mathsf{V}} \cup N_{\mathsf{I}}$. Let $A \in N_{\mathsf{C}}$, $R \in N_{\mathsf{R}}$ and $t, u \in N_{\mathsf{V}} \cup N_{\mathsf{I}}$. An *atom* is an expression of the form $A(t)$ or $R(t, u)$. A *query* q is set of atoms.

Without loss of generality, we assume that, given an ontology $\mathcal{O}$ defined over a signature Σ, there are no symbols in Σ which do not occur in $\mathcal{O}$.

We write $\mathcal{O} \models \alpha$ (resp. $\mathcal{O} \models q$) to indicate that an ontology $\mathcal{O}$ entails an axiom α (resp. query q). We write $\mathcal{O} \models^q (a_1, \ldots, a_n)$ to indicate that the tuple $(a_1, \ldots, a_n)$ is an answer of ontology $\mathcal{O}$ to a query q. In this thesis, we focus on the problem of conjunctive query answering (CQA) over $\mathcal{SROIQ}$ ontologies. That is, given an ontology $\mathcal{O}$ and a query q, compute all the answers $(a_1, \ldots, a_n)$ of $\mathcal{O}$ to q. We focus our attention on queries with a single answer variable. I.e., queries with answers with answers of the form (a_1) (which may contain arbitrarily complex sets of atoms). Even though in principle our approach could be used to deal with queries containing more answer variables this restriction is in place because of practical reasons (see discussion in Section 2.2).

Syntax	**Semantics**
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists S.C$	$\{t \mid \exists u[(t,u) \in S^{\mathcal{I}}, u \in C^{\mathcal{I}}]\}$
$\forall S.C$	$\{t \mid \forall u[(t,u) \in S^{\mathcal{I}} \rightarrow u \in C^{\mathcal{I}}]\}$
$\geq n\, R.C$	$\{t \mid \#\{(t,u) \in S^{\mathcal{I}} \mid u \in C^{\mathcal{I}}\} \geq n\}$
$\leq n\, R.C$	$\{t \mid \#\{(t,u) \in S^{\mathcal{I}} \mid u \in C^{\mathcal{I}}\} \leq n\}$
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
R	$\{(t,u) \mid (u,t) \in \mathsf{Inv}(R)^{\mathcal{I}}\}$
$R_1 \circ \ldots \circ R_n$	$\{(t_0,t_n) \mid (t_i,t_{i+1}) \in R_i^{\mathcal{I}} \text{ for } 1 \leq i \leq n\}$

Syntax	**Semantics**
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$R_1 \circ \ldots \circ R_n \sqsubseteq R$	$(R_1 \circ \ldots \circ R_n)^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Figure 2.1: $\mathcal{SROIQ}$ Axiom Semantics. In the above: $t_{(i)}, u \in \Delta^{\mathcal{I}}$, $C_{(i)}, D \in N_{\mathsf{E}}$, $R_{(i)} \in N_{\mathsf{R}}^{-}$ and $a, b \in N_{\mathsf{I}}$.

2.2 A Machine Learning Approach to CQA

One of the central tasks performed in machine learning (ML) is *classification*[1] [Murphy 2012]. Classification is the problem of identifying to which of a set of given categories a new observation belongs. In our context, we use machine learning classification methods to identify whether or not some individual is an answer of an ontology to a query. We proceed with the introduction of some ML related preliminary notions.

A *feature* is some quality that is representable as a numerical value. A *feature vector* is an n-dimensional vector $(v_1, \ldots, v_n)$ of features. A *binary feature vector* is a feature vector $(v_1, \ldots, v_n)$ of binary features (i.e., $v_i \in \{0, 1\}$ for all $1 \leq i \leq n$). A *binary feature space* is the set of all possible binary feature vectors of a certain length.

We define a *classifier* as a function $f : \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ is some feature space and the range represents membership in a given class (with 1 indicating membership). Having selected an appropriate feature space, a function f can be learned by minimizing an error function over a set of training examples. Once this training is complete, the learned function can be employed to determine membership of new elements into the class.

In the context of CQA, a $\mathcal{SROIQ}$ reasoner can be thought of as a function which maps a $\mathcal{SROIQ}$ ontology and a query to a set of individuals. Alternatively, given a particular query q with a single answer variable, we can think of a reasoner as a function $g_q : \mathcal{O}_{\mathcal{SR}} \times N_\mathsf{I} \to \{0, 1\}$ where $\mathcal{O}_{\mathcal{SR}}$ is the set of all $\mathcal{SROIQ}$ ontologies.

Let q be a query and $\mathcal{O} \in \mathcal{O}_{\mathcal{SR}}$. A general framework for learning an estimate function f_q of g_q is as follows :

1. Devise a feature extraction function that projects $\mathcal{O}_{\mathcal{SR}} \times N_\mathsf{I}$ onto a feature space $\mathcal{X}$.
2. Use a deductive reasoner to classify all individuals in $\mathcal{O}_T$ with respect to q where $\mathcal{O}_T$ is some training ontology structurally similar to $\mathcal{O}$.
3. Learn a classifier function, f_q, that minimizes the difference in classification between f_q and g_q on all individuals in $\mathcal{O}_T$ with respect to $\mathcal{X}$.

A diagram of the training framework is shown in Figure 2.2. Once the training phase is complete, the system can classify individuals using only the feature extraction and classifer functions, as shown in Figure 2.2.

This framework can be extended in principle to queries with an arbitrary number n answer variables. In practice, given a query q, instead of classifying $|N_\mathsf{I}|$ individuals, we would need to

[1] Not to be mistaken with the DL reasoning task of classification, which is the comprehensive computation of axioms of the form $A \sqsubseteq B$ entailed by an ontology.

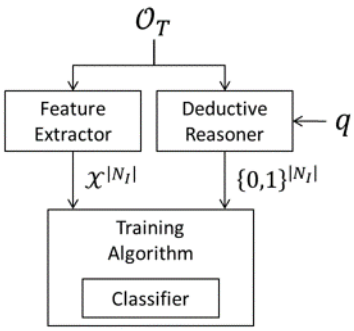

Figure 2.2: Diagram of the training process

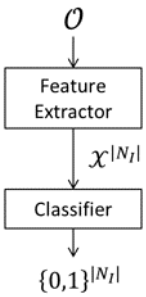

Figure 2.3: Diagram of the classification process

classify $|N_{\mathsf{I}}|^n$ tuples where n is the number of answer variables in q. This would be computationally problematic for both the classifier and the training procedure. Our framework could potentially be used in conjunction with other methods to handle these types of queries, however. A precise formalization of the intuition presented in this general framework can be found in Section 3.

3

Technical Details

3.1 Feature Space

In the given framework, the classification of an individual is a function of the feature vector only. One of the chief challenges in applying machine learning techniques to semantic web reasoning is constructing an appropriate feature space to project the knowledge base onto. In our view, we should consider the following qualities while constructing a feature space:

- Richness: How well does a vector in the space characterize an individual in a knowledge base? Since all other information is lost in the projection, a rich feature space should allow discrimination between a large number of potential target classes. In cases where both members and non-members of the target class have identical feature vectors, perfect classification is impossible.

- Ease of extraction: How much processing is required to perform the projection into the feature space? The chief motivation of an inductive approach to reasoning is that it is more computationally feasible than deductive reasoning, so the computational cost of the projection is a key consideration. A closely related consideration is the number of parameters involved in the extraction procedure. Are there ad hoc parameters being used? Do parameters in the procedure need to be tuned to each dataset?

- Succinctness: How large is the feature space? How does it scale with respect to the number of concept names and roles in the ontology and the number of individuals? In our view, polynomial scaling with respect to the number of named concepts and roles is acceptable, while an ideal feature space would have constant size with respect to the number of individuals.

In the following paragraphs we formalize the feature space chosen while commenting on its particular qualities. We proceed with the introduction of some preliminary notions which will be

used in further definitions.

Definition 1. *Let $\mathcal{O}$ be an ontology and $k, i \in \mathbb{N}_0$. We inductively define the sets $\mathsf{E}^i_\mathcal{O}$, $\mathcal{E}^k_\mathcal{O} \subseteq N_E$ as follows*[1]*:*

$$\mathsf{E}^0_\mathcal{O} = \{A \mid A \in N_C\} \cup \{\exists R.\top \mid R \in N_R^-\}$$
$$\mathsf{E}^{i+1}_\mathcal{O} = \{\exists R.C \mid C \in \mathsf{E}^i_\mathcal{O} \text{ and } R \in N_R^-\}$$
$$\mathcal{E}^k_\mathcal{O} = \bigcup_{i=0}^{k} \mathsf{E}^i_\mathcal{O}$$

Note that the sets of concept names N_C and roles N_R are finite, by definition; and do not contain symbols which do not occur in $\mathcal{O}$, by the assumption made in Section 2. Intuitively, $\mathcal{E}^k_\mathcal{O}$ is a set of concepts that describe the neighborhood graph of a given individual out to depth k.

Lemma 1. $|\mathcal{E}^k_\mathcal{O}| = \sum_{i=0}^{k}(|N_C| + |N_R^-|)|N_R^-|^i$.

Proof. It is straightforward to verify that $|\mathsf{E}^0_\mathcal{O}| = |N_\mathsf{C}| + |N_\mathsf{R}^-|$. Also,

$$|\mathsf{E}^{i+1}_\mathcal{O}| = |\mathsf{E}^i_\mathcal{O}||N_\mathsf{R}^-|.$$

Hence,

$$|\mathsf{E}^i_\mathcal{O}| = (|N_\mathsf{C}| + |N_\mathsf{R}^-|)|N_\mathsf{R}^-|^i.$$

Since $\mathsf{E}^i_\mathcal{O} \cap \mathsf{E}^j_\mathcal{O} = \emptyset$ for $i \neq j$,

$$|\mathcal{E}^k_\mathcal{O}| = \sum_{i=0}^{k} |\mathsf{E}^i_\mathcal{O}| = \sum_{i=0}^{k}(|N_\mathsf{C}| + |N_\mathsf{R}^-|)|N_\mathsf{R}^-|^i.$$

□

We introduce a running example which is used to clarify some of the notions presented in this section.

Example 1. *Let $\mathcal{O} = \{A(a), B(b), R^-(b,a), \exists R.B(b)\}$ where $N_C = \{A, B\}$ and $N_R = \{R\}$. Then*

$$\begin{aligned}
E^0_\mathcal{O} =& \{A, B, \exists R.\top, \exists R^-.\top, \} \\
E^1_\mathcal{O} =& \{\exists R.A, \exists R.B, \exists R.\exists R.\top, \exists R.\exists R^-.\top, \\
& \quad \exists R^-.A, \exists R^-.B, \exists R^-.\exists R.\top, \exists R^-.\exists R^-.\top\} \\
\mathcal{E}^0_\mathcal{O} =& E^0_\mathcal{O} \\
\mathcal{E}^1_\mathcal{O} =& E^0_\mathcal{O} \cup E^1_\mathcal{O}
\end{aligned}$$

[1]Note that for $k > 1$, $\mathcal{E}^k_\mathcal{O}$ contains redundant concepts, such as $\exists R.\top$ and $\exists R.R^-.\top$. In practice, we remove concepts of the form $\exists R.\exists \mathsf{Inv}(R).D$ in each iteration without losing any information, but we leave these concepts in $\mathcal{E}^k_\mathcal{O}$ for the duration of this section to simplify the definition.

Definition 2. *Let ρ be some* oracle function *that maps a $\mathcal{SROIQ}$ ontology to a tuple $(F(a_1), \ldots, F(a_m))$ such that $N_I = \{a_1, \ldots, a_m\}$ and $F(a) \subseteq \mathcal{E}^k_{\mathcal{O}}$ for all $a \in N_I$. Given some $a \in N_I$, we define $X^{\rho}_{\mathcal{E}^k_{\mathcal{O}}}(a) = (v_1(a), \ldots, v_n(a))$ as the binary feature vector such that $n = |\mathcal{E}^k_{\mathcal{O}}|$ and, for all $C \in \mathcal{E}^k_{\mathcal{O}}$, $v_{\mathsf{pos}(C)}(a) = 1$ if and only if $C \in F(a)$.*

Note that the length of $X^{\rho}_{\mathcal{E}^k_{\mathcal{O}}}(a)$ is $|\mathcal{E}^k_{\mathcal{O}}|$, which is given in Lemma 1. The size of this vector is independent of both the number of individuals and the number of assertion axioms, so the space required to store the full set of feature vectors is bounded by $|N_{\mathsf{I}}|$ multiplied by a constant. Furthermore, the length of $X^{\rho}_{\mathcal{E}^k_{\mathcal{O}}}(a)$ grows linearly w.r.t. $|N_{\mathsf{C}}|$ and polynomially (with degree k) w.r.t. $|N_{\mathsf{R}}|$.

If we choose a reasonably low value of k (e.g., $k = 2$), this results in a feature vector that is both succinct enough to use with machine learning algorithms and rich enough to accurately approximate conjunctive query answering reasoners, as will be shown in Section 5. Note that this feature space does not capture information about specific nodes in the graph, so it is unlikely that the system would be able to correctly classify individuals for, e.g., "SELECT ?X WHERE { ?X R a }" (for $R \in N_{\mathsf{R}}, a \in N_{\mathsf{I}}$). It may be possible to extend our feature space to capture information about specific nodes mentioned in the query.

In principle, ρ could be *any* function with the appropriate domain and range. Note that *changing ρ changes the feature space.* To illustrate, consider two oracle functions: ρ which checks the entailment of each individual $\times$ concept based on $\mathcal{O}$, and ρ' which checks $\mathcal{O}$ for direct assertions of each individual $\times$ concept. The feature spaces generated by these oracle functions are semantically different, so e.g., a classifier function f_q learned using ρ features is not applicable to ρ' feature vectors. This holds for any two distinct oracle functions.

3.2 Selecting an Oracle Function

Now that we have specified the form of a sufficiently succinct feature space, we specify two oracle functions and their implementations. A good oracle function should meet the criteria discussed at the beginning of Section 3.1, specifically *richness* and *ease of extraction.* In short, it should quickly retrieve most of the information afforded by the ontology's assertions to allow discrimination between individuals. In our view, the computational cost of checking concept entailments based on the full ontology undermines the intended benefits of an inductive approach to query answering (i.e., speed and scalability). Instead we propose two algorithms, dubbed ρ_1 and ρ_2, that implement different oracle functions performing fast incomplete reasoning over the concepts in $\mathcal{E}^k_{\mathcal{O}}$. ρ_1 and ρ_2 are shown in Algorithm 1 and Algorithm 2, respectively. For convenience, we write $R(a, b) \in \mathcal{O}$ to indicate

that either $R(a,b) \in \mathcal{O}$ or $\mathsf{Inv}(R)(a,b) \in \mathcal{O}$.

input: $\mathcal{O} \in \mathcal{O}_{SR}$, $k \in \mathbb{N}_0$
Set $F_0(a) = \{\}$, for each $a \in N_\mathsf{I}$;
foreach $C(a) \in \mathcal{O}$ **do** add C to $F_0(a)$;
foreach $R(a,b) \in \mathcal{O}$ **do**
  Add $\exists R.\top$ to $F_0(a)$;
  Add $\exists \mathsf{Inv}(R).\top$ to $F_0(b)$;
end
$i = 0$;
while $i < k$ **do**
  $i++$;
  Set $F_i(a) = \{\}$, for each $a \in N_\mathsf{I}$;
  foreach $R(a,b) \in \mathcal{O}$ **do**
    foreach $C \in F_{i-1}(b)$ **do** add $\exists R.C$ to $F_i(a)$;
    foreach $C \in F_{i-1}(a)$ **do** add $\exists \mathsf{Inv}(R).C$ to $F_i(b)$;
  end
end
Set $F(a) = \bigcup_{i=0}^{k} F_i(a)$, for each $a \in N_\mathsf{I}$;
return $\{F(a) \mid a \in N_I\}$;

Algorithm 1: Oracle Function ρ_1

ρ_1 builds the feature vector by scanning through the ontology $k+1$ times, retrieving assertions about the depth-k neighborhood graph of each individual. This algorithm is well suited for ontologies whose assertions are predominantly of the form $C(a)$ and $R(a,b)$ where $C \in N_\mathsf{C}$, $R \in N_\mathsf{R}$ and $a, b \in N_\mathsf{I}$. If assertions involve complex concepts, however, ρ_1 does not capture any information from them. ρ_2 rectifies this problem by checking whether each individual axiom (i.e., by itself) entails any of the members of $\mathcal{E}^k_\mathcal{O}$. This proves valuable, as will be demonstrated in Section 5, for ontologies that use conjunction and existential role restrictions in their ABox. ρ_2 maintains linear time scaling w.r.t. the *number* of ABox assertions, but could be considerably slower than ρ_1. Both algorithms return a *sparse* representation of $\{X^\rho_{\mathcal{E}^k_\mathcal{O}}(a) \mid a \in N_\mathsf{I}\}$ (for an arbitrary pos), which in practice greatly reduces computation and space requirements.

Let us present two brief examples that clarify the construction of the feature vectors $X^{\rho_1}_{\mathcal{E}^k_\mathcal{O}}$ and $X^{\rho_2}_{\mathcal{E}^k_\mathcal{O}}$ for our example ontology.

input: $\mathcal{O} \in \mathcal{O}_{SR}$, $k \in \mathbb{N}_0$
Set $F(a) = \{\}$, for each $a \in N_\mathsf{I}$;
foreach $Z \in \mathcal{O}$ **do**
  foreach $C \in \mathcal{E}_\mathcal{O}^k$ **do**
    foreach $a \in N_I$ **do**
      if $\{Z\} \models C(a)$ **then** add C to $F(a)$;
    end
  end
end
$i = 0$;
for $i < k$ **do**
  $i++$;
  foreach $R(a, b) \in \mathcal{O}$ **do**
    foreach $C \in F(b)$ **do**
      if $\exists R.C \in \mathcal{E}_\mathcal{O}^k$ **then** add $\exists R.C$ to $F(a)$;
    end
    foreach $C \in F(a)$ **do**
      if $\exists \mathit{Inv}(R).C \in \mathcal{E}_\mathcal{O}^k$ **then** add $\exists \mathsf{Inv}(R).C$ to $F(b)$;
    end
  end
end
return $\{F(a) \mid a \in N_I\}$;

Algorithm 2: Oracle Function ρ_2

Example 2. *Vectors* $X^{\rho_1}_{\mathcal{E}^k_{\mathcal{O}}}(a)$ *and* $X^{\rho_1}_{\mathcal{E}^k_{\mathcal{O}}}(b)$ *for* $0 \leq k \leq 1$ *using a sparse vector representation are as follows:*

$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{A, \exists R.\top\}$$
$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{B, \exists R^-.\top\}$$
$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{A, \exists R.\top, \exists R.B, \exists R.\exists R^-.\top\}$$
$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{B, \exists R^-.\top, \exists R^-.A, \exists R^-.\exists R.\top\}$$

Let the pos *function order concept expressions in the order given in Example 1, starting with* $\mathsf{E}^0_{\mathcal{O}}$ *up to* $\mathsf{E}^k_{\mathcal{O}}$. *I.e.,* $\mathsf{pos}(A) = 1, \mathsf{pos}(B) = 2, ..., \mathsf{pos}(\exists R.A) = 5, ...$ *Dense representations of the above vectors are as follows:*

$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{1, 0, 1, 0\}$$
$$X^{\rho_1}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{0, 1, 0, 1\}$$
$$X^{\rho_1}_{\mathcal{E}^1_{\mathcal{O}}}(a) = \{1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0\}$$
$$X^{\rho_1}_{\mathcal{E}^1_{\mathcal{O}}}(b) = \{0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0\}$$

Example 3. *Vectors* $X^{\rho_2}_{\mathcal{E}^k_{\mathcal{O}}}(a)$ *and* $X^{\rho_2}_{\mathcal{E}^k_{\mathcal{O}}}(b)$ *for* $0 \leq k \leq 1$ *using a sparse vector representation are as follows:*

$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{A, \exists R.\top\}$$
$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{B, \exists R^-.\top, \exists R.\top\}$$
$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{A, \exists R.\top, \exists R.B, \exists R.\exists R.\top, \exists R.\exists R^-.\top\}$$
$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{B, \exists R.\top, \exists R^-.\top, \exists R.B, \exists R^-.A, \exists R^-.\exists R.\top\}$$

Using the pos *function from Example 2, the dense representations of the above vectors are as follows:*

$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(a) = \{1, 0, 1, 0\}$$
$$X^{\rho_2}_{\mathcal{E}^0_{\mathcal{O}}}(b) = \{0, 1, 1, 1\}$$
$$X^{\rho_2}_{\mathcal{E}^1_{\mathcal{O}}}(a) = \{1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0\}$$
$$X^{\rho_2}_{\mathcal{E}^1_{\mathcal{O}}}(b) = \{0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0\}$$

Note that the complex concept assertion $\exists R.B(b)$ leads to information about several concepts absent in the $X^{\rho_1}_{\mathcal{E}^k_{\mathcal{O}}}$ vectors being added to the $X^{\rho_2}_{\mathcal{E}^k_{\mathcal{O}}}$ vectors.

3.3 Classifier

We experimented with a variety of algorithms to find a good classifier function, f_q, and selected two algorithms, AdaBoost and ID3, to use in our evaluation. We used the default algorithms implemented in WEKA [Hall et al. 2009].

AdaBoost is an ensemble learning method that produces a linear combination of weak learners as the classifier [Freund and Schapire 1997]. Weak learners are simple binary classifiers taken from the features, each being capable of contributing a small amount of information to the overall classifier. In our case, each feature in $X^{\rho}_{\mathcal{E}^k_{\mathcal{O}}}$ is a weak learner. The algorithm iteratively adds the weak learner that best improves performance on misclassified training samples and stops either after a fixed number of iterations or when performance ceases to improve. AdaBoost gives good performance on a wide variety of problems, and was famously called "the best off-the-shelf classifier in the world" [Hastie et al. 2001].

ID3 is an algorithm for building binary decision trees. Initially, all training data resides in the root node. Branches are added iteratively by greedily selecting the binary feature (again, simply a feature in $X^{\rho}_{\mathcal{E}^k_{\mathcal{O}}}$ in our case) that best reduces the entropy of the training data with respect to the desired class (in our case, membership in the current query).

In our initial investigations, we also tested the random forest [Breiman 2001] and naïve Bayes classifiers, which both underperformed AdaBoost and ID3. These classifiers are generally very good at avoiding over-fitting training data, a common problem in ML that causes poor performance on previously unseen data. We found that ID3, which is the most prone to over-fitting of the classifiers tested, gave the best performance. This is most likely due to the fact that we are using noiseless logical data as features, so fitting the training data very closely, including statistical outliers, actually benefits the classifier's ability to answer queries with very few answers. If we were to apply our approach to the case of inconsistent ontologies with large numbers of assertions, over-fitting may be more of an issue, leading us to favor AdaBoost or random forest.

4

Related work

Other efforts to apply inductive methods to Semantic Web reasoning include Bloehdorn et al. [Bloehdorn and Sure 2007], Lösch et al. [Lsch et al. 2012], d'Amato et al. [dAmato et al. 2008] and Fanizzi et al. [Fanizzi et al. 2008], which all happen to use kernel or distance-based methods. Bloehdorn et al. defines a family of kernel (i.e., similarity) functions that use essentially all relevant information to the individuals at hand to compute the kernel, but prior to using the kernel in his experiments, he manually selects a very small set of features to use in the actual experiments. Lösch et al. defines a set of kernels that are fairly robust and computationally tractable, but only incorporates information about specific node identities, as opposed to the more general knowledge captured in DL concepts.

D'Amato et al. uses a distance measure that essentially counts the number of DL concepts shared (or not shared) by individuals. Individuals that are both described by a large number of the same concepts are close to each other, while individuals that are described by different concepts are far from each other. She also experiments with a scheme for weighting concepts. Fanizzi et al. uses the same idea to create a kernel function. He uses a genetic programming procedure to reduce the size of the feature set in a way that maximizes "discernibility" among individuals. These distance/similarity functions are used in conjunction with suitable classifiers, such as k-nearest neighbors (KNN) and support vector machines (SVMs) to classify individuals.

This work resembles the work of d'Amato et al. and Fanizzi et al. in some respects, but there are a few significant differences. First, d'Amato et al. and Fanizzi et al. are trying to induce classifiers that assess the "true" class of an individual, as opposed to assessing whether an individual is entailed to be in a class or not. As an example, d'Amato et al. is able to correctly infer some knowledge about wines in the popular Wine ontology that is not logically entailed. In this respect, they are trying to go beyond the functionality of pure deduction. We prefer to focus on the problem of "estimating" deductive reasoners, both because we feel this is a valuable service, and because assessing the quality of our answers is then not subjective. A second significant difference is that, while d'Amato et al.

and Fanizzi et al. suggest the possibility of using ABox lookup instead of entailment checking to construct their features, they do not discuss in detail what features (i.e. DL concepts) they use or how much processing is required to generate the feature vectors. We feel the utility of our approach is contingent upon speed and scalablility, and thus focus our efforts on linear-time worst case algorithms.

5

Experimental Evaluation

5.1 Test Setup

We evaluated our system on a variety of ontologies and queries. Ontologies included a Dublin traffic data ontology (DCC) from [Mutharaju et al. 2015], the New Testament Names ontology (NTN),[1] the Semantic Web Service Data ontology (SWSD),[2] the Wine ontology,[3] the Business Process Modeling Notation (BPMN) ontology,[4] and the Lehigh University Benchmark (LUBM) dataset [Guo et al. 2005]. The LUBM dataset is actually an ontology generator with parameters to be specified. We generated two ontologies, denoted LUBM-1 and LUBM-10, which were generated with 1 and 10 universities, respectively. All other parameters were default values. Table 5.1 characterizes the ontologies used in the evaluation. In this table, N_{R} includes OWL object properties only, TBox gives the total number of TBox axioms, and ABox gives the number of ABox axioms.

We randomly generated queries using the same generator[5] employed in previous related papers [Fanizzi et al. 2012]. This generator recursively uses union and existential and universal role restrictions to build DL concepts, then prunes concepts that have no members.

Additionally, for the LUBM dataset, we manually wrote a set of SPARQL queries that seemed reasonable to pose using the classes and roles in the ontology. The LUBM ontologies describe individuals related to a set of universities. Individuals include students, professors, organizations, courses, etc. Our queries requested, for example, individuals in a specific $C \in N_{\mathsf{C}}$, taking a specific type of course, taking a course from a specific type of professor, and having membership in a specific type of organization. Not all queries were expressible as DL concepts, and we included some cyclic

[1] http://www.semanticbible.com/ntn/ntn-view.html
[2] http://www.uni-koblenz-landau.de/de/koblenz/fb4/ AGStaab/Projects/x-media/dl-tree/index_html
[3] http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#
[4] https://dkm-static.fbk.eu/resources/ontologies/ BPMN/BPMN_1.1_ontology.owl
[5] http://lacam.di.uniba.it/ nico/research/snippet1.html

Table 5.1: Characterization of Tested Ontologies

Ontology	$\lvert N_C \rvert$	$\lvert N_R \rvert$	$\lvert N_I \rvert$	TBox	ABox
WINE	138	16	206	552	494
BPMN	95	101	36	730	81
NTN	86	29	724	3699	735
SWSD	256	25	512	300	514
LUBM-1	43	25	17k	101k	101k
LUBM-10	43	25	207k	88	1273k
DCC	42	49	1752	180	7790

queries. All queries were built based solely on background knowledge of the LUBM ontology. Of the queries we wrote, some had empty result sets and were excluded from the evaluation, while other queries, denoted *ABox* queries, could be correctly answered by doing simple ABox lookup without reasoning. The remaining queries are denoted *Reas*. In all cases, queries involved only the named concepts and roles of the ontologies without naming specific individuals or data values.

Test parameters for our system included the value of k (k=2 tested) and the classifier (AdaBoost and ID3 tested). We used ρ_1 as the default oracle function in Sections 5.2, 5.3, and 5.4. We used a partially implemented version of ρ_2 defined earlier, but for the ontologies tested, our implementation faithfully matched the output of ρ_2. In all cases, the reasoner we were trying to match was Pellet [Sirin et al. 2007], version 2.3.1.

For assessment, we use 10-fold cross validation. In this approach, the data is partitioned into 10 equally-sized random subsets of individuals. The classifier is trained on 9 of them at a time and assessed on the hold-out data. This process is repeated for each partition, and the results are combined. We feel this approach is appropriate to assess utility for our designed use case of query answering on an ontology with a large number of individuals relative to the number of named classes and roles; a scalable system with accurate retrieval on generic reasoning tasks should be able to fill this role. We also demonstrate our system functioning within the target use case by training the classifier on the LUBM-1 ontology and testing it on the LUBM-10 ontology.

We were not able to obtain the classifiers or queries used in prior papers, so we compare our results to the best results reported in [Fanizzi et al. 2012], which used a support vector machine in conjunction with a genetic programming feature selection algorithm (here denoted SVM-GP). We use *error percentage*, i.e., the percentage of misclassified individuals, for the comparison. This corresponds to 100 minus the *match rate* reported in [Fanizzi et al. 2012]. We ran our system on 5 of the 8 ontologies used in [Fanizzi et al. 2012]. The three excluded ontologies were the Surface Water

Table 5.2: Comparison of Error Rates

Ontology	SVM-GP	AdaBoost	ID3
WINE	4.5	5.6311	2.4515
BPMN	4.7	8.9	8.9
NTN	0.9	0.0760	0.0552
SWSD	2	1.7578	0.0977
LUBM-1	1.1	0.0006	0.0006

Model (SWM) ontology, which we were unable to obtain, and the Finance and BioPax ontologies, which produced inconsistent ontology exceptions in Pellet.

In addition to *error percentage*, we report precision, recall, and F-measure on all tested ontologies and query sets to robustly characterize system performance.

5.2 Comparison to Previous Work

Table 5.2 compares our approach using both AdaBoost and ID3 with the reported results for SVM-GP. It is apparent that our algorithm is not well-suited for the BPMN ontology. Section 5.3 will show that our algorithm actually gives a recall of zero. The cause of this is that the BPMN uses complex concepts in the majority of assertions, which are not exploited by ρ_1. We show in Section 5.5 that complex concepts can be used to populate our feature vector and achieve good classification performance.

With the exception of BPMN, though, the algorithm compares well. The error rates on NTN, SWSD, and LUBM-1 are an order of magnitude lower for ID3 than for SVM-GP, while the error rates for WINE are all comparable. Based on our experience with the data, two reasons for this seem probable. First, our feature vector, though simpler to compute, may capture more information relevant to answering the query than theirs. Second, distance measures, which form the basis of their classification, are not always meaningful for binary data and thus may prevent the classifier from making optimal decisions.

5.3 In Depth Results

Table 5.3 gives average precision and recall for the ontologies tested in section 5.2. These results paint a somewhat different picture: the queries generated tend to have a small number of results, so algorithms with very poor recall can be masked by error rate.

Table 5.3: Average Precision and Recall for Randomly Generated Queries

	AdaBoost		ID3	
	Prec	Rec	Prec	Rec
WINE	0.88	0.56	0.91	0.69
BPMN	–	0	–	0
NTN	1.00	0.76	1.00	0.77
SWSD	1.00	0.5	1.00	0.99
LUBM-1	1.00	1.00	1.00	1.00

Looking at the *type* of errors the algorithm made, several of the queries with low recall had only one example, so due to the cross-validation scheme, when the test set has a positive example for the query, there are no positive examples in the training set. The NTN ontology had four such queries, and removing them from consideration raised ID3's recall to 0.96.

A second type of error occurred when the algorithm had positive training examples of the query that were different from the positive test example. For instance, one of the queries randomly generated for the SWSD ontology was $BankCode \sqcup WeightUnit$. While there were 14 individuals in the query concept, there were only a few members of $WeightUnit$. Since these were of a different nature than the bulk of the the results, they were missed.

Finally we would expect the system to perform best when $|N_{\mathsf{I}}|$ is large relative to $|N_{\mathsf{C}}|$ and $|N_{\mathsf{R}}|$. While the system performs reasonably outside this situation, e.g., the Wine ontology, performance degrades relative to ontologies with large ABoxes.

These types of errors are rooted in the fundamental limitations of induction, so in order to mitigate them, one must pick suitable problems for this approach.

5.4 SPARQL Query Answering

Table 5.4 gives average precision and recall for the manually written SPARQL queries on the LUBM ontologies. For LUBM-1, we tested using cross-validation as in previous sections. For LUBM-10, we trained the classifier on LUBM-1 and used it to classify individuals in LUBM-10. All errors were on a single query from the ABox set. This query, "SELECT ?X WHERE { ?X Advisor ?Y. ?X TeachingAssistantOf ?Z. ?Y TeacherOf ?Z}" had a very small number of positive training examples (8 of 17166 individuals), and the low error rate may have caused the training algorithm to stop before reaching perfect classification.

Table 5.4: Results on SPARQL Queries for ID3, k=2

Query Set	Ontology	Prec	Recall
ABox	LUBM-1	0.94	0.94
ABox	LUBM-10	0.95	0.97
REAS	LUBM-1	1.0	1.0
REAS	LUBM-10	1.0	1.0

5.5 Results with Complex Concept Assertions

We implemented the ρ_2 function using Pellet as the reasoner. While ρ_2 maintains linear time complexity w.r.t. the number of axioms, in practice, retrieving entailed individuals for each member of $\mathcal{E}_{\mathcal{O}}^k$ for each axiom in a brute force fashion was slow. This process could be sped up by tailoring the reasoning process to the concepts in $\mathcal{E}_{\mathcal{O}}^k$, but here we merely aim to demonstrate that the process can be extended to handle complex concept assertions. We were able to achieve perfect precision and recall for the BPMN and DCC ontologies on the 20 generated queries using this oracle function and $k = 1$.

6

Conclusions

We have shown that the proposition in Section 1 holds in some settings. Our approach extracts features directly from the ABox in linear-time and exploits a sparse feature space to further increase speed. Additionally, precision and recall values are rather good, especially in settings with many individuals and at least a handful of positive examples of the query. Our approach outperforms previous inductive query answering results, both in terms of accuracy and scalability.

A challenge innate to statistical approaches such as ours is demonstrating that the approach performs well in a variety of useful settings. While our evaluations have shown promising performance on some variety of datasets, further investigations using more extensive real-world data and queries would be worthwhile. Additionally, possibly beneficial extensions of the approach include supporting multiple answer variables, tailoring the feature space to a given query, and developing additional oracle function algorithms. We intend to pursue these lines of investigation as we go forward.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., Eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*, Second ed. Cambridge University Press.

Bloehdorn, S. and Sure, Y. 2007. Kernel methods for mining instance data in ontologies. In *The Semantic Web*, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudr-Mauroux, Eds. Lecture Notes in Computer Science, vol. 4825. Springer Berlin Heidelberg, 58–71.

Breiman, L. 2001. Random forests. *Machine Learning 45,* 1 (Oct.), 5–32.

dAmato, C., Fanizzi, N., and Esposito, F. 2008. Query answering and ontology population: An inductive approach. In *The Semantic Web: Research and Applications*, S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds. Lecture Notes in Computer Science, vol. 5021. Springer Berlin Heidelberg, 288–302.

Fanizzi, N., dAmato, C., and Esposito, F. 2008. Statistical learning for inductive query answering on owl ontologies. In *The Semantic Web - ISWC 2008*, A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, Eds. Lecture Notes in Computer Science, vol. 5318. Springer Berlin Heidelberg, 195–212.

Fanizzi, N., d'Amato, C., and Esposito, F. 2012. Induction of robust classifiers for web ontologies through kernel machines. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web 11,* 0.

Fensel, D. and van Harmelen, F. 2007. Unifying reasoning and search to web scale. *IEEE Internet Computing 11,* 2, 94–96.

Freund, Y. and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55,* 1 (Aug.), 119–139.

Groot, P., Hitzler, P., Horrocks, I., Motik, B., Pan, J. Z., Stuckenschmidt, H., Turi, D., and Wache, H. 2005. Methods for approximate reasoning. Tech. Rep. Deliverable D2.1.2 (WP2.1), EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB. Available from http://daselab.cs.wright.edu/pub/kweb-212.pdf.

Guo, Y., Pan, Z., and Heflin, J. 2005. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics 3,* 2-3 (Oct.), 158–182.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. 2009. The WEKA data mining software: An update. *SIGKDD Explorations Newsletter 11,* 1 (Nov.), 10–18.

Hastie, T., Tibshirani, R., and Friedman, J. 2001. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S., Eds. 11 December 2012. *OWL 2 Web Ontology Language Primer (Second Edition).* W3C Recommendation. Available at http://www.w3.org/TR/owl2-primer/.

Hitzler, P., Krötzsch, M., and Rudolph, S. 2010. *Foundations of Semantic Web Technologies.* Chapman & Hall/CRC.

Hitzler, P. and van Harmelen, F. 2010. A reasonable Semantic Web. *Semantic Web 1,* 1–2, 39–44.

Horrocks, I., Kutz, O., and Sattler, U. 2006. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006).* AAAI Press, 57–67.

Lsch, U., Bloehdorn, S., and Rettinger, A. 2012. Graph kernels for rdf data. In *The Semantic Web: Research and Applications*, E. Simperl, P. Cimiano, A. Polleres, O. Corcho, and V. Presutti, Eds. Lecture Notes in Computer Science, vol. 7295. Springer Berlin Heidelberg, 134–148.

Murphy, K. P. 2012. *Machine Learning: A Probabilistic Perspective.* The MIT Press.

Mutharaju, R., Hitzler, P., Mateti, P., and Lécué, F. 2015. Distributed and scalable OWL EL reasoning. In *The Semantic Web. Latest Advances and New Domains – 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31 – June 4, 2015. Proceedings*, F. Gandon, M. Sabou, H. Sack, C. d'Amato, P. Cudré-Mauroux, and A. Zimmermann, Eds. Lecture Notes in Computer Science, vol. 9088. Springer, Heidelberg, 88–103.

Rudolph, S., Tserendorj, T., and Hitzler, P. 2008. What is approximate reasoning? In *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, Germany, October/November 2008*, D. Calvanese and G. Lausen, Eds. Lecture Notes in Computer Science, vol. 5341. Springer, Heidelberg, 150–164.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. 2007. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web 5,* 2, 51 – 53. Software Engineering and the Semantic Web.