KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-020-00656-9

TECHNICAL CONTRIBUTION

Learning Description Logic Ontologies: Five Approaches. Where Do They Stand?

Ana Ozaki[1,2]

Received: 6 March 2020 / Accepted: 25 March 2020

Abstract
The quest for acquiring a formal representation of the knowledge of a domain of interest has attracted researchers with various backgrounds into a diverse field called ontology learning. We highlight classical machine learning and data mining approaches that have been proposed for (semi-)automating the creation of description logic (DL) ontologies. These are based on association rule mining, formal concept analysis, inductive logic programming, computational learning theory, and neural networks. We provide an overview of each approach and how it has been adapted for dealing with DL ontologies. Finally, we discuss the benefits and limitations of each of them for learning DL ontologies.

Keywords Ontology learning · Description logic · Logic and learning

1 Introduction

The quest for acquiring a formal representation of the knowledge of a domain of interest has attracted researchers with various backgrounds and both practical and theoretical inquires into a diverse field called *ontology learning* [30, 33]. In this work, we focus on approaches for building description logic (DL) ontologies assuming that the vocabulary and the language of the ontology to be created are known. The main goal is to find how the symbols of the vocabulary should be related, using the logical constructs available in the ontology language. Desirable goals of an ontology learning process include:

1. the creation of ontologies which are *interpretable*; expressions should not be overly complex, redundancies should be avoided;
2. the support for learnability of DL expressions formulated in *rich ontology languages*;
3. *efficient* algorithms for creating ontologies, requiring a *small amount of time and training data*;
4. *limited or no human intervention* requirement;
5. the support for learning in *unsupervised* settings;
6. handling of *inconsistencies and noise*.

Other properties such as explainability and trustability may also be relevant for some approaches. Moreover, once the ontology has been created, it needs to be checked, be maintained, and evolve. This means that other reasoning tasks should also be feasible.

Nearly 20 years after the term "ontology learning" was coined by Maedche and Staab [33], it is not a surprise that no approach could accomplish such ambitious and conflicting goals. However, different approaches have addressed some of these goals. We highlight five approaches coming from machine learning and data mining which have been proposed for (semi-)automating the creation of DL ontologies. These are based on association rule mining (ARM) [1], formal concept analysis (FCA) [19], inductive logic programming (ILP) [35], computational learning theory (CLT) [44], and neural networks (NNs) [34].

The adaptations of the approaches to the problem of learning DL ontologies often come with the same benefits and limitations as the original approach. To show this effect, for each of the five approaches, we start by presenting the original proposal and then explain how it has been adapted for dealing with DL ontologies. Before presenting them, we introduce some basic notions.

✉ Ana Ozaki
ana.ozaki@uib.no

[1] Free University of Bozen-Bolzano, Piazza Università, 1, 39100 Bolzano, BZ, Italy

[2] Department of Informatics, University of Bergen, 5020 Bergen, Norway

Published online: 22 April 2020

2 Definitions

Here we present the syntax and semantics of DLs and basic definitions useful to formalise learning problems.

2.1 Description Logic Ontologies

We introduce $\mathcal{ALC}$ [3], a prototypical DL which features basic ingredients found in many DL languages. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and disjoint sets of *concept* and *role* names. An $\mathcal{ALC}$ ontology (or *TBox*) is a finite set of expressions of the form $C \sqsubseteq D$, called *concept inclusions* (CIs), where C, D are $\mathcal{ALC}$ *concept expressions* built according to the grammar rule

$$C, D ::= A \mid \neg C \mid C \sqcap D \mid \exists r.C$$

with $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. An $\mathcal{EL}$ concept expression is an $\mathcal{ALC}$ concept expression without any occurrence of the negation symbol ($\neg$). An $\mathcal{EL}$ TBox is a finite set of CIs $C \sqsubseteq D$, with C, D being $\mathcal{EL}$ concept expressions.

The semantics of $\mathcal{ALC}$ (and of the $\mathcal{EL}$ fragment) is based on *interpretations*. An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain of* $\mathcal{I}$, and $\cdot^{\mathcal{I}}$ is a function mapping each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each $r \in \mathsf{N_R}$ to a subset $r^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ extends to arbitrary $\mathcal{ALC}$ concept expressions as follows:

$$\begin{aligned}(\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\ (C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}}\\ (\exists r.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \exists e \in C^{\mathcal{I}} \text{ such that } (d,e) \in r^{\mathcal{I}}\}\end{aligned}$$

An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$, in symbols $\mathcal{I} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It satisfies a TBox $\mathcal{T}$, in symbols $\mathcal{I} \models \mathcal{T}$, iff $\mathcal{I}$ satisfies all CIs in $\mathcal{T}$. A TBox $\mathcal{T}$ *entails* a CI α, in symbols $\mathcal{T} \models \alpha$, iff all interpretations satisfying $\mathcal{T}$ also satisfy α.

2.2 Learning Frameworks

By *learning* we mean the process of acquiring some desired kind of knowledge represented in a well-defined and machine-processable form. *Examples* are pieces of information that characterise such knowledge, given as part of the input of a learning process. We formalise these relationships as follows.

A *learning framework* $\mathfrak{F}$ is a triple $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is a set of examples, $\mathcal{L}$ is a set of concept representations,[1] called *hypothesis space*, and μ is a function that maps each element of $\mathcal{L}$ to a set of (possibly classified) examples in $\mathcal{E}$. If the classification is into $\{1, 0\}$, representing positive and negative labels, then μ simply associates elements l of $\mathcal{L}$ to all examples labelled with 1 by l. Each element of $\mathcal{L}$ is called a *hypothesis*. The *target representation* (here simply called *target*) is a fixed but arbitrary element of $\mathcal{L}$, representing the kind of knowledge that is aimed for in the learning process.

Example 1 To formalise the problem of learning DL ontologies from entailments, one can define the learning framework for a given DL L as $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is the set of all CIs $C \sqsubseteq D$ with C, D being L concept expressions; $\mathcal{L}$ is the set of all L TBoxes; and μ is a function that maps every L TBox $\mathcal{T}$ to the set $\{C \sqsubseteq D \in \mathcal{E} \mid \mathcal{T} \models C \sqsubseteq D\}$. In this case, we consider that $C \sqsubseteq D$ is labelled with 1 by $\mathcal{T}$ iff $\mathcal{T} \models C \sqsubseteq D$.

In the next five sections, we highlight machine learning and data mining approaches which have been proposed for (semi-)automating the creation of DL ontologies. As mentioned, for each approach, we first describe the original motivation and application. Then we describe how it has been adapted for dealing with DL ontologies.

3 Association Rule Mining

3.1 Original Approach

Association rule mining (ARM) is a data mining method frequently used to discover patterns, correlations, or causal structures in transaction databases, relational databases, and other information repositories. We provide basic notions, as it was initially proposed [1].

Definition 1 (*Association rule*) Given a set $I = \{i_1, i_2, \ldots, i_n\}$ of *items*, and a set $\mathbb{D} = \{t_1, t_2, \ldots, t_m\}$ of *transactions* (called *transaction database*) with each $t_i \subseteq I$, an *association rule* is an expression of the form $A \Rightarrow B$ where A, B are sets of items.

The task of mining rules is divided into two parts: (i) mining sets of items which are frequent in the database, and, (ii) generating association rules based on frequent sets of items. To measure the frequency of a set X of items in a transaction database $\mathbb{D}$, one uses a measure called *support*, defined as:

$$\mathsf{supp}_{\mathbb{D}}(X) = \frac{|\{t_i \in \mathbb{D} : X \subseteq t_i\}|}{|\mathbb{D}|}$$

If a set X of items has support larger than a given threshold then it is used in the search of association rules, which have the form $A \Rightarrow B$, with $X = A \cup B$. To decide whether

[1] In the Machine Learning literature, a *concept* is often defined as a set of examples and a concept representation is a way of representing such set. This differs from the notion of a concept in the DL literature and a formal concept in FCA.

Table 1 Transaction database

ID	Product 1	Product 2	Product 3	Product 4
1	✓		✓	✓
2			✓	✓
3	✓	✓	✓	✓
4	✓	✓	✓	
5	✓	✓	✓	✓

an implication $A \Rightarrow B$ should be in the output of a solution to the problem, a confidence measure is used. The confidence of an association rule $A \Rightarrow B$ w.r.t. a transaction database $\mathbb{D}$ is defined as:

$$\mathsf{conf}_{\mathbb{D}}(A \Rightarrow B) = \frac{\mathsf{supp}_{\mathbb{D}}(A \cup B)}{\mathsf{supp}_{\mathbb{D}}(A)}$$

Essentially, support measures statistical significance, while confidence measures the 'strength' of a rule [1].

We parameterize the ARM learning framework $\mathfrak{F}_{\mathsf{ARM}}$ with the confidence threshold $\delta \in [0, 1] \subset \mathbb{R}$. $\mathfrak{F}_{\mathsf{ARM}}(\delta)$ is $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is the set of all database transactions $\mathbb{D}$; $\mathcal{L}$ is the set of all sets $\mathcal{S}$ of association rules; and

$$\mu(\mathcal{S}) = \{\mathbb{D} \in \mathcal{E} \mid \forall \alpha \in \mathcal{S} \text{ we have that } \mathsf{conf}_{\mathbb{D}}(\alpha) \geq \delta\}.$$

ARM Problem Given δ and $\mathbb{D}$, let $\mathfrak{F}_{\mathsf{ARM}}(\delta)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $\mathcal{S} \in \mathcal{L}$ with $\mathbb{D} \in \mu(\mathcal{S})$.

Example 2 Consider the transaction database in Table 1. It contains 5 transactions. For example, the first transaction, t_1, has Product1, Product3 and Product4 (first row of Table 1). Assume that the support and confidence thresholds are resp. 60% and 70%. ARM gives the rules: {Product3, Product4} ⇒ {Product1} (conf. 75%) and {Product2} ⇒ {Product1} (conf. 100%), among others.

3.2 Building DL Ontologies

An immediate way of adapting the ARM approach to deal with DL ontologies is to make the correspondence between a *finite* interpretation and a transaction database. Assume $\mathcal{I}$ is a finite interpretation then the notions of support and confidence can be adapted to:

$$\mathsf{supp}_{\mathcal{I}}(C) = \frac{|C^{\mathcal{I}}|}{|\Delta^{\mathcal{I}}|} \qquad \mathsf{conf}_{\mathcal{I}}(C \sqsubseteq D) = \frac{\mathsf{supp}_{\mathcal{I}}(C \sqcap D)}{\mathsf{supp}_{\mathcal{I}}(C)}$$

The problem of giving logical meaning to association rules is that it may happen that $C \sqsubseteq D$ and $D \sqsubseteq E$ have confidence values above a certain threshold while $C \sqsubseteq E$ *does not* have a confidence value that is above the threshold, even though it is a logical consequence of the first two CIs [7]. This problem also occurs in the original ARM approach if the association rules are interpreted as Horn rules in propositional logic. To see this effect, consider Example 2. We have that both {Product3, Product4} ⇒ {Product1} and {Product1} ⇒ {Product2} have confidence 75% but the confidence of {Product3, Product4} ⇒ {Product2} is only 50%. Another difficulty in this adaptation for dealing with DLs is that the number of CIs with confidence value above a given threshold may be infinite (consider e.g. $\mathcal{EL}$ CIs in an interpretation with a directed cycle) and a finite set which implies such CIs may not exist.

The learning framework here is parameterized with a DL L and a confidence threshold $\delta \in [0, 1] \subset \mathbb{R}$. Then, $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{ARM}}(L, \delta)$ is $(\mathcal{E}, \mathcal{L}, \mu)$ with $\mathcal{E}$ the set of all finite interpretations $\mathcal{I}$; $\mathcal{L}$ the set of all L TBoxes $\mathcal{T}$; and

$$\mu(\mathcal{T}) = \{\mathcal{I} \in \mathcal{E} \mid \forall \alpha \in \mathcal{T} \text{ we have that } \mathsf{conf}_{\mathcal{I}}(\alpha) \geq \delta\}.$$

ARM+DL Problem Given $\mathcal{I}$, L, and δ, let $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{ARM}}(L, \delta)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $\mathcal{T} \in \mathcal{L}$ with $\mathcal{I} \in \mu(\mathcal{T})$.

ARM is an effective approach for extracting CIs with concept expressions of fixed length from RDF datasets. Using this technique, e.g., DeputyDirector ⊑ CivilServicePost and MinisterialDepartment ⊑ Department were extracted from data.gov.uk [13, 45, 46] (see also [41] for expressive DLs with fixed length).

More recently, ARM has been applied to mine relational rules in knowledge graphs [16]. This approach, born in the field of data mining, is relevant for the task of building DL ontologies, as it can effectively find interesting relationships between concept and role names. However, it lacks support for mining CIs with existential quantifiers on the right-hand side [43].

4 Formal Concept Analysis

4.1 Original Approach

Formal Concept Analysis (FCA) is a mathematical method of data analysis which describes the relationship between objects and their attributes [19] (see also [18] for an introduction to this field). In FCA, data is represented by formal contexts describing the relationship between finite sets of objects and attributes. The notion of a transaction database (Definition 1) is similar to the notion of a formal context (Definition 2).

Definition 2 (*Formal context*) A *formal context* is a triple (G, M, I), where G is a set of objects, M is a set of attributes,

Table 2 Formal context

Objects	Attribute 1	Attribute 2	Attribute 3
□	✓		✓
♡	✓	✓	
○			✓
◇	✓	✓	
△			✓

and $I \subseteq G \times M$ is a binary relation between objects and attributes.

A *formal concept* is a pair (A, B) consisting of a set $A \subseteq G$ of objects (the 'extent') and a set $B \subseteq M$ of attributes (the 'intent') such that the extent consists of all objects that share the given attributes, and the intent consists of all attributes shared by the given objects. A formal concept (A_1, B_1) is less or equal to a formal concept (A_2, B_2), written $(A_1, B_1) \leq (A_2, B_2)$ iff $A_1 \subseteq A_2$. It is known that the set of all formal concepts ordered by $\leq$ forms a complete lattice.

Example 3 $(\{\heartsuit, \diamond\}, \{\text{Attribute 1, Attribute 2}\})$ is a formal concept in the formal context shown in Table 2.

In FCA, dependencies between attributes are expressed by *implications*—a notion similar to the notion of an association rule (Definition 1). An implication is an expression of the form $B_1 \rightarrow B_2$ where B_1, B_2 are sets of attributes. An implication $B_1 \rightarrow B_2$ holds in a formal context (G, M, I) if every object having all attributes in B_1 also has all attributes in B_2. A subset $B \subseteq M$ respects an implication $B_1 \rightarrow B_2$ if $B_1 \not\subseteq B$ or $B_2 \subseteq B$. An implication i *follows* from a set $\mathcal{S}$ of implications if any subset of M that respects all implications from $\mathcal{S}$ also respects i. In FCA, one is essentially interested in computing the implications that hold in a formal context. A set $\mathcal{S}$ of implications that hold in a formal context $\mathbb{K}$ is called an *implicational base* for $\mathbb{K}$ if every implication that holds in $\mathbb{K}$ follows from $\mathcal{S}$. Moreover, there should be no redundancies in $\mathcal{S}$ (i.e., if $i \in \mathcal{S}$ then i does not follow from $\mathcal{S} \setminus \{i\}$). Implicational bases are not unique. A well-studied kind of implicational base (with additional properties) is called *stem* (or *Duquenne–Guigues*) base [18, 20].

The learning framework for FCA is $\mathfrak{F}_{\mathsf{FCA}} = (\mathcal{E}, \mathcal{L}, \mu)$ with $\mathcal{E}$ the set of all formal contexts $\mathbb{K}$; $\mathcal{L}$ the set of all implicational bases $\mathcal{S}$; and

$$\mu(\mathcal{S}) = \{\mathbb{K} \in \mathcal{E} \mid \mathcal{S} \text{ is an implicational base for } \mathbb{K}\}.$$

FCA Problem Given $\mathbb{K}$, let $\mathfrak{F}_{\mathsf{FCA}}$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $\mathcal{S} \in \mathcal{L}$ with $\mathbb{K} \in \mu(\mathcal{S})$.

4.2 Building DL Ontologies

Approaches to combine FCA and DL have been addressed by many authors [4, 5, 7, 40]. A common way of bridging the gap between FCA and DL [10] is the one that maps a finite interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and a finite set S of concept expressions into formal context (G, M, I) in such a way that:

- each $d \in \Delta^{\mathcal{I}}$ corresponds to an object o in G;
- each concept expression $C \in S$ corresponds to an attribute a in M; and
- $d \in C^{\mathcal{I}}$ if, and only if, $(o, a) \in I$.

The notion of an implication is mapped to the notion of a CI in a DL. Just to give an idea, if the formal context represented by Table 2 is induced by a DL interpretation then the CI $\mathsf{Attribute2} \sqsubseteq \mathsf{Attribute1}$ would be a candidate to be added to the ontology. The notion of an implicational base is adapted as follows. Let $\mathcal{I}$ be a finite interpretation and let L be a DL language with symbols taken from a finite vocabulary. An *implicational base for* $\mathcal{I}$ *and* L [10] is a non-redundant set $\mathcal{T}$ of CIs formulated in L (for short L-CIs) such that for all L-CIs

- $\mathcal{I} \models C \sqsubseteq D$ if, and only if, $\mathcal{T} \models C \sqsubseteq D$.

We parameterize the learning framework $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{FCA}}$ with a DL L. Then, $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{FCA}}(L)$ is $(\mathcal{E}, \mathcal{L}, \mu)$, where $\mathcal{E}$ is the set of all finite interpretations $\mathcal{I}$, $\mathcal{L}$ is the set of all implicational bases $\mathcal{T}$ for $\mathcal{I} \in \mathcal{E}$ and L, and

$$\mu(\mathcal{T}) = \{\mathcal{I} \in \mathcal{E} \mid \mathcal{T} \text{ is an implicational base for } \mathcal{I} \text{ and } L\}.$$

FCA+DL Problem Given $\mathcal{I}$ and L, let $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{FCA}}(L)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $\mathcal{T} \in \mathcal{L}$ with $\mathcal{I} \in \mu(\mathcal{T})$.

Similar to the difficulty described for the DL adaptation of the ARM approach, there may be no finite implicational base for a given interpretation and DL.

Example 4 Consider the interpretation in Fig. 1. An implicational base for $\mathcal{EL}_{\mathsf{rhs}}$—the $\mathcal{EL}$ fragment that allows only conjunctions of concept names on the left-side of CIs—is $\{A \sqsubseteq \exists r.(A \sqcap B)\}$ [23]. However, if we remove e_3 from the extension of A, B then, for all $n \in \mathbb{N}$, the CI $A \sqsubseteq \exists r^n.\top$ holds and there is no $\mathcal{EL}_{\mathsf{rhs}}$ finite base that can entail all such CIs. More expressive languages can be useful for the computation of finite bases. It is known that, for $\mathcal{EL}$ with greatest fixpoints semantics, a finite implicational base always exists [10].

Classical FCA and ARM assume that all the information about the individuals is known and can be represented in a finite way. A '✓' in a table representing a formal

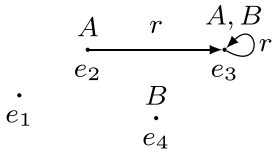

Fig. 1 $\{A \sqsubseteq \exists r.(A \sqcap B)\}$ is a base for $\mathcal{EL}_{\text{rhs}}$ [23]

Table 3 Background knowledge and classified examples

Background knowledge
$\forall x(\text{MedicalDomain}(x) \rightarrow \text{Domain}(x))$
Person(John), MedicalDomain(Allergy)
isExpert(John, Allergy)
Classified Examples
(DomainExpert(John), 1)
(DomainExpert(Allergy), 0)

context means that the attribute holds for the corresponding object and the absence means that the attribute does not hold. In contrast, DL makes the 'open-world' assumption, and so, the absence of information indicates a lack of knowledge, instead of negation. To deal with the lack of knowledge, the authors of [5] introduce the notion of a partial context, in which affirmative and negative information about individuals is given as input and an expert is required to decide whether a given concept inclusion should hold or not.

The need for a finite representation of objects and their attributes hinders the creation of concept inclusions expressing, for instance, that 'every human has a parent that is a human', in symbols

$$\text{Human} \sqsubseteq \exists \text{hasParent.Human}$$

or 'every natural number has a successor that is a natural number', where elements of a model capturing the meaning of the relation are linked by an infinite chain. This limitation is shared by all approaches which mine CIs from data, including ARM, but in FCA this difficulty is more evident as it requires 100% of confidence. This problem can be avoided by allowing the system to interact with an expert who can assert domain knowledge that cannot be conveyed from the finite interpretation given as input [40].

5 Inductive Logic Programming

5.1 Original Approach

ILP is an area between logic programming and machine learning [35]. In the general setting of ILP, we are given a logical formulation of background knowledge and some examples classified into positive and negative [35]. The background knowledge is often formulated with a *logic program*—a non-propositional version of Horn clauses where all variables in a clause are universally quantified within the scope of the entire clause. The goal is to extend the background knowledge $\mathcal{B}$ with a hypothesis $\mathcal{H}$ in such a way that all examples in the set of positive examples can be deduced from the modified background knowledge and none of the elements of the set of negative examples can be deduced from it.

We introduce the syntax of function-free first-order Horn clauses. A term t is either a variable or a constant. An *atom* is an expression of the form $P(\mathbf{t})$ with P a predicate and $\mathbf{t}$ a list of terms $t_1, \ldots, t_a$ where a is the arity of P. An atom is *ground* if all terms occurring in it are constants. A *literal* is an atom α or its negation $\neg\alpha$. A first-order clause is a universally quantified disjunction of literals. It is called *Horn* if it has at most one positive literal. A *Horn expression* is a set of (first-order) Horn clauses. A *classified example* in this setting is a pair $(e, \ell(e))$ where e is a ground atom and $\ell(e)$ (the label of e) is 1 if e is a positive example or 0 if it is negative.

Definition 3 (*Correct hypothesis*) Let $\mathcal{B}$ be a Horn expression and S a set of pairs $(e, \ell(e))$ with e a ground atom and $\ell(e) \in \{1, 0\}$. A Horn expression $\mathcal{H}$ is a *correct hypothesis for $\mathcal{B}$ and S* if

$$\forall (e, 1) \in S,\ \mathcal{B} \cup \mathcal{H} \models e \text{ and } \forall (e, 0) \in S,\ \mathcal{B} \cup \mathcal{H} \not\models e.$$

Example 5 Suppose that we are given as input the background knowledge $\mathcal{B}$[2] and a set S of classified examples presented in Table 3. In this example, one might conjecture a hypothesis $\mathcal{H}$ which states that:

$$\forall xy(\text{isExpert}(x, y) \wedge \text{Domain}(y) \rightarrow \text{DomainExpert}(x)).$$

This form of inference is not sound in the logical sense since $\mathcal{H}$ does not necessarily follow from $\mathcal{B}$ and S. Another hypothesis considered as correct by this approach would be

$$\forall x(\text{Person}(x) \rightarrow \text{DomainExpert}(x)),$$

even though one could easily think of an interpretation with a person not being a domain expert. One could also create a situation in which there are infinitely many hypotheses suitable to explain the positive and negative examples. For this reason, it is often required a non-logical constraint to justify the choice of a particular hypothesis [35]. A common principle is the Occam's razor principle which says that the simplest hypothesis is the most likely to be correct

[2] We use the equivalent representation of Horn clauses as implications.

(simplicity can be understood in various ways, a naive way is to consider the length of the Horn expression as a string).

We parameterize the learning framework for ILP with the background knowledge $\mathcal{B}$, given as part of the input of the problem. We then have that $\mathfrak{F}_{\mathsf{ILP}}(\mathcal{B})$ is the learning framework $(\mathcal{E}, \mathcal{L}, \mu)$ with $\mathcal{E}$ the set of all ground atoms; $\mathcal{L}$ the set of all Horn expressions $\mathcal{H}$; and

$$\mu(\mathcal{H}) = \{e \in \mathcal{E} \mid \mathcal{B} \cup \mathcal{H} \models e\}.$$

Classified examples help to distinguish a target unknown logical theory formulated as a Horn expression from other Horn expressions in the hypothesis space. In the learning framework $\mathfrak{F}_{\mathsf{ILP}}(\mathcal{B})$, positive examples for a Horn expression $\mathcal{H}$ are those entailed by the union of $\mathcal{H}$ and the background theory $\mathcal{B}$.

ILP Problem Given $\mathcal{B}$ and S (as in Definition 3), let $\mathfrak{F}_{\mathsf{ILP}}(\mathcal{B})$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $\mathcal{H} \in \mathcal{L}$ such that $\mathcal{H}$ is a correct (and simple) hypothesis for $\mathcal{B}$ and S. That is, for all $(e, \ell(e)) \in S$, $e \in \mu(\mathcal{H})$ iff $\ell(e) = 1$.

5.2 Building DL Ontologies

In the DL context, ILP has been applied for learning DL concept expressions [12, 15, 22, 26, 27, 29] and for learning logical rules for ontologies [31]. We describe here the problem setting for learning DL *concept expressions*, which can help the designer to formulate the concept expressions in an ontology. As in the classical ILP approach, the learner receives as input some background knowledge, formulated as a *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is a set of *assertions*, that is, expressions of the form $A(a)$, $r(a, b)$ where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and a, b are taken from a set $\mathsf{N_I}$ of individual names. Assertions can be seen as ground atoms and $\mathcal{A}$, in DL terms, is called an *ABox*. A set S of pairs $(e, \ell(e))$ with e an assertion and $\ell(e) \in \{1, 0\}$ is also given as part of the input. In the mentioned works, e is of the form $\mathsf{Target}(a)$, with Target a concept name in $\mathsf{N_C}$ not occurring in $\mathcal{K}$ and $a \in \mathsf{N_I}$.

As in the original ILP approach, given $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and S, a concept expression C (in the chosen DL) is *correct* for $\mathcal{K}$ and S if, for all $(\mathsf{Target}(a), \ell(\mathsf{Target}(a))) \in S$, we have that $(\mathcal{T} \cup \{\mathsf{Target} \equiv C\}, \mathcal{A}) \models \mathsf{Target}(a)$ iff $\ell(\mathsf{Target}(a)) = 1$.

Example 6 The background knowledge in Table 3 can be translated into $(\mathcal{T}, \mathcal{A})$, with

$$\mathcal{T} = \{\mathsf{MedicalDomain} \sqsubseteq \mathsf{Domain}\}$$

and $\mathcal{A}$ the set of ground atoms given as background knowledge in Table 3. Assuming that the target concept name is $\mathsf{DomainExpert}$ and the set S of classified examples is the one in Table 3, correct concept expressions would be $\exists \mathsf{isExpert}.\mathsf{Domain}$ and Person.

The learning framework and problem statement presented here is for learning $\mathcal{ALC}$ and $\mathcal{EL}$ concept expressions based on the ILP approach [28, 29]. Here the learning framework is parameterized by a knowledge base $(\mathcal{T}, \mathcal{A})$ and a DL L. Then, $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{ILP}}((\mathcal{T}, \mathcal{A}), L)$ is $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is the set of all ground atoms; $\mathcal{L}$ is the set of all L concept expressions C such that Target does not occur in it; and

$$\mu(C) = \{e \in \mathcal{E} \mid (\mathcal{T} \cup \{\mathsf{Target} \equiv C\}, \mathcal{A}) \models e\}.$$

ILP+DL Problem Given $\mathcal{K}$, L, and S (the classified examples), let $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{ILP}}(\mathcal{K}, L)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $C \in \mathcal{L}$ such that C is correct (and simple) for $\mathcal{K}$ and S. That is, for all $(e, \ell(e)) \in S$, $e \in \mu(C)$ iff $\ell(e) = 1$.

6 Learning Theory

6.1 Original Approach

We describe two classical learning models in CLT which have been applied for learning DL concept expressions and ontologies. We start with the classical PAC learning model and then describe the exact learning model.[3]

In the PAC learning model, a learner receives classified examples drawn according to a probability distribution and attempts to create a hypothesis that approximates the target. The aim is to bound the probability that a hypothesis constructed by the learner misclassifies an example. This approach can be applied to any learning framework. Within this model, one can investigate the complexity of learning an abstract target, such as a DL concept, an ontology, or the weights of a NN.

We now formalise this model. Let $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$ be a learning framework. A *probability distribution* $\mathcal{D}$ over $\mathcal{E}$ is a function mapping events in a σ-algebra E of subsets of $\mathcal{E}$ to $[0, 1] \subset \mathbb{R}$ such that $\mathcal{D}(\bigcup_{i \in I} X_i) = \sum_{i \in I} \mathcal{D}(X_i)$ for mutually exclusive X_i, where I is a countable set of indices, $X_i \in E$, and $\mathcal{D}(\mathcal{E}) = 1$. Given a target $t \in \mathcal{L}$, let $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$ be the oracle that takes no input, and outputs a *classified example* $(e, \ell_t(e))$, where $e \in \mathcal{E}$ is sampled according to the probability distribution $\mathcal{D}$, $\ell_t(e) = 1$, if $e \in \mu(t)$, and $\ell_t(e) = 0$, otherwise. An *example query* is a call to the oracle $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$. A *sample* generated by $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$ is a (multi-)set of indexed classified examples, independently and identically distributed according to $\mathcal{D}$, sampled by calling $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$.

A learning framework $\mathfrak{F}$ is *PAC learnable* if there is a function $f : (0, 1)^2 \to \mathbb{N}$ and a deterministic algorithm such

[3] The expression "Probably Approximately Correct" was coined by Angluin in the paper [2], where she shows the connection between the two learning models.

that, for every $\epsilon, \delta \in (0,1) \subset \mathbb{R}$, every probability distribution $\mathcal{D}$ on $\mathcal{E}$, and every target $t \in \mathcal{L}$, given a sample of size $m \geq f(\epsilon, \delta)$ generated by $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$, the algorithm always halts and outputs $h \in \mathcal{L}$ such that with probability at least $(1-\delta)$ over the choice of m examples in $\mathcal{E}$, we have that $\mathcal{D}(\mu(h) \oplus \mu(t)) \leq \epsilon$. If the number of computation steps used by the algorithm is bounded by a polynomial function $p(|t|, |e|, 1/\epsilon, 1/\delta)$, where e is the largest example in the sample generated by $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$, then $\mathfrak{F}$ is *PAC learnable in polynomial time*.

Example 7 Let $\mathcal{E} = \{\square, \heartsuit, \bigcirc, \diamond, \triangle\}$ and let $\mathcal{D}$ be a probability distribution on $\mathcal{E}$, defined, e.g., by the pairs

$$(\{\square\}, 0.2), (\{\heartsuit\}, 0.1), (\{\bigcirc\}, 0.3), (\{\diamond\}, 0.2), (\{\triangle\}, 0.2).$$

Assume $h, t \in \mathcal{L}$ and assume $\mu(h) = \{\heartsuit, \bigcirc\}$ and $\mu(t) = \{\heartsuit, \triangle\}$ then the probability $\mathcal{D}(\mu(h) \oplus \mu(t))$ that h misclassifies an example according to $\mathcal{D}$ is 0.5.

PAC Problem Given a learning framework decide whether it is PAC learnable in polynomial time.

In the classical PAC approach, the probability distribution $\mathcal{D}$ is unknown to the learner. The algorithm should provide a probabilistic bound for any possible $\mathcal{D}$. We now describe the exact learning model. In this model, a learner tries to identify an abstract target known by a teacher, also called an *oracle*, by interacting with the teacher [2]. The most successful protocol is based on *membership* and *equivalence* queries. As it happens with the PAC learning model, this model can be used to formulate learning problems within the context of any kind of learning framework.

We formalise these notions as follows. Given a learning framework $\mathfrak{F} = (\mathcal{E}, \mathcal{L}, \mu)$, we are interested in the exact identification of a *target* concept representation $t \in \mathcal{L}$ by posing queries to oracles. Let $\mathsf{MQ}_{\mathfrak{F},t}$ be the oracle that takes as input some $e \in \mathcal{E}$ and returns 'yes' if $e \in \mu(t)$ and 'no' otherwise. A membership query is a call to the oracle $\mathsf{MQ}_{\mathfrak{F},t}$. For every $t \in \mathcal{L}$, we denote by $\mathsf{EQ}_{\mathfrak{F},t}$ the oracle that takes as input a *hypothesis* concept representation $h \in \mathcal{L}$ and returns 'yes' if $\mu(h) = \mu(t)$ and a *counterexample* $e \in \mu(h) \oplus \mu(t)$ otherwise, where $\oplus$ denotes the symmetric set difference. There is no assumption regarding which counterexample in $\mu(h) \oplus \mu(t)$ is chosen by the oracle. An equivalence query is a call to the oracle $\mathsf{EQ}_{\mathfrak{F},t}$. In this model, if examples are interpretations or entailments, the notion of 'equivalence' coincides with *logical* equivalence.

A learning framework $\mathfrak{F}$ is *exactly learnable* if there is a deterministic algorithm such that, for every $t \in \mathcal{L}$, it eventually halts and outputs some $h \in \mathcal{L}$ with $\mu(h) = \mu(t)$. Such algorithm is allowed to call the oracles $\mathsf{MQ}_{\mathfrak{F},t}$ and $\mathsf{EQ}_{\mathfrak{F},t}$. If the number of computation steps used by the algorithm is bounded by a polynomial $p(|t|, |e|)$, where $t \in \mathcal{L}$ is the target and $e \in \mathcal{E}$ is the largest counterexample seen so far, then $\mathfrak{F}$ is *exactly learnable in polynomial time*.

Exact Problem Given a learning framework decide whether it is exactly learnable in polynomial time.

In Theorem 1, we recall an interesting connection between the exact learning model and the PAC model extended with membership queries. If there is a polynomial time algorithm for a learning framework $\mathfrak{F}$ that is allowed to make membership queries then $\mathfrak{F}$ is *PAC learnable with membership queries in polynomial time*.

Theorem 1 [2] *If a learning framework is exactly learnable in polynomial time then it is PAC learnable with membership queries in polynomial time. If only equivalence queries are used then it is PAC learnable (without membership queries) in polynomial time.*

The converse of Theorem 1 does not hold [6]. That is, there is a learning framework that is PAC learnable in polynomial time (even without membership queries) but not exactly learnable in polynomial time.

6.2 Building DL Ontologies

The PAC learning model has been already applied to learn DL concept expressions formulated in DL CLASSIC [9, 14] (see also [36]). The main difficulty in adapting the PAC approach for learning DL ontologies is the complexity of this task. In the PAC learning model, one is normally interested in *polynomial time* complexity, however, many DLs, such as $\mathcal{ALC}$, have superpolynomial time complexity for the entailment problem and entailment checks are often important to combine the information present in the classified examples.

It has been shown that the $\mathcal{EL}$ fragments $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ —the $\mathcal{EL}$ fragments that allow only conjunctions of concept names on the right-side and on the left-side of CIs, respectively—are polynomial time exactly learnable from entailments [24, 25, 37],[4] however, this is not the case for $\mathcal{EL}$. The learning framework is the one in Example 1 and the problem statement is the same as in the original approach. By Theorem 1, the results for $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ are transferable to the PAC learning model extended with membership queries. The results show how changes in the ontology language can impact the complexity of searching for a suitable ontology in the hypothesis space. The main difficulty of implementing this model is that it is based on oracles, in particular, on an equivalence query oracle. Fortunately, as already mentioned, such equivalence queries can be simulated by the sampling

[4] The result for $\mathcal{EL}_{\mathsf{rhs}}$ (allowing conjunctions of concept names on the left-side of CIs) appears in [24, Section 4]

oracle of the PAC learning model to achieve PAC learnability (Theorem 1) [2].

7 Neural Networks

7.1 Original Approach

NNs are widespread architectures inspired by the structure of the brain [34]. They may differ from each other not only regarding their weight and activation functions but also structurally, e.g., it is known that feed-forward NNs are acyclic while recurrent NNs have cycles. One of the simplest models is the one given by Definition 4.

Definition 4 (*Neural network*) An NN is a triple (G, σ, w) where $G = (V, E)$ is a graph, with V a set of nodes, called *neurons*, and $E \subseteq V \times V$ a set of (directed) edges; $\sigma : \mathbb{R} \to \mathbb{R}$ is the *activation* function; and $w : E \to \mathbb{R}$ is the *weight* function.

Other parameters that can be part of the definition of an NN are the propagation function and biases. A widely used propagation function is the weighted sum. The propagation function specifies how the outputs of the neurons connected to a neuron n are combined to form the input of n. Given an input to a neuron, the activation function maps it to the output of the neuron. In symbols, the input $\mathsf{in}(n)$ of a neuron n is

$$\sum_{m:(m,n)\in E} \sigma(\mathsf{in}(m)) \cdot w((m,n)).$$

The structure of an NN is organized in *layers*, basically, an *input*, an *output*, and (possibly several) *hidden* layers. The input of an NN is a vector of numbers in $\mathbb{R}$, given as input to the neurons in the input layer. The output of the NN is also a vector of numbers in $\mathbb{R}$, constructed using the outputs of the neurons in the output layer. The dimensionality of the input and output of an NN varies according to the learning task. One can then see an NN as a function mapping an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$. In symbols, $(G, \sigma, w)(\mathbf{x}) = \mathbf{y}$.

The main task is to find a weight function that minimizes the *risk* of the NN $\mathcal{N}$, modelled by a function $L_{\mathcal{D}}(\mathcal{N})$, with $\mathcal{D}$ a probability distribution on a set of pairs $(\mathbf{x}, \mathbf{y})$ of input/output vectors [42]. The risk of an NN represents how well we expect the NN to perform while predicting the classification of unseen examples.

The learning framework can be defined in various ways. Here we parameterize it by a graph structure and an activation function σ. We have that $\mathfrak{F}_{\mathsf{NN}}(G, \sigma)$, with $G = (V, E)$, is $(\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is a set of pairs $(\mathbf{x}, \mathbf{y})$ representing input and output vectors of numbers in $\mathbb{R}$ (respectively, and with appropriate dimensionality); $\mathcal{L}$ is the set of all weight functions $w : E \to \mathbb{R}$; and

$$\mu(w) = \{(\mathbf{x}, \mathbf{y}) \in \mathcal{E} \mid (G, \sigma, w)(\mathbf{x}) = \mathbf{y}\}.$$

One can formulate the NN problem as follows.

NN Problem Given G and σ, let $\mathfrak{F}_{\mathsf{NN}}(G, \sigma)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $w \in \mathcal{L}$ that minimizes the risk $L_{\mathcal{D}}(\mathcal{N})$ of $\mathcal{N} = (G, \sigma, w)$, where $\mathcal{D}$ is a fixed but arbitrary and unknown probability distribution on $\mathcal{E}$.

Classified examples for training and validation can be obtained by calling the sampling oracle $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$ (recall $\mathsf{EX}^{\mathcal{D}}_{\mathfrak{F},t}$ from Sect. 6), where $t \in \mathcal{L}$ is the (unknown) target weight function. One of the main challenges of this approach is that finding an optimal weight function is computationally hard. Most works apply a heuristic search based on the gradient descent algorithm [42].

7.2 Building DL Ontologies

NNs have been applied to learn CIs from sentences expressing definitions, called *definitorial sentences* [38] (see also [32] for more work on definitorial sentences in a DL context, and, e.g. [8, 48], for work on learning assertions based on NNs). More specifically, the work on [38] is based on *recurrent* NNs, which are useful to process sequential data. The structure of the NN, in this case, takes the form of a grid. The authors learn $\mathcal{ALCQ}$ CIs, where $\mathcal{ALCQ}$ is the extension of $\mathcal{ALC}$ with qualified number restrictions. For example, "A car is a motor vehicle that has 4 tires and transport people." corresponds to

$$\mathsf{Car} \sqsubseteq \mathsf{MotorVehicle} \sqcap = 4\mathsf{has.Tires} \sqcap \exists \mathsf{transport.Person}.$$

The main benefits of this approach is that NNs can deal with natural language variability. The authors provide an end-to-end solution that does not even require natural language processing techniques. However, the approach is based on the *syntax* of the sentences, not on their semantics, and they cannot capture portions of knowledge across different sentences [38]. Another difficulty of adapting this approach for learning DL ontologies is the lack of datasets available for training. Such dataset should consist of a large set of pairs of definitorial sentences and their corresponding CI. The authors created a synthetic dataset to perform their experiments.

The learning framework and problem statement for learning DL CIs based on the NN approach [38] can be formulated as follows. The learning framework for a DL L can be defined as $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{NN}}(G, \sigma, L) = (\mathcal{E}, \mathcal{L}, \mu)$ where $\mathcal{E}$ is a set of pairs $(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}$ a vector representation of a definitorial sentence and $\mathbf{y}$ a vector representation of an L CI; and $\mathcal{L}$ and μ are as in the original NN approach.

NN+DL Problem Given G, L and σ, let $\mathfrak{F}^{\mathsf{DL}}_{\mathsf{NN}}(G, \sigma, L)$ be $(\mathcal{E}, \mathcal{L}, \mu)$. Find $w \in \mathcal{L}$ that minimizes the risk $L_{\mathcal{D}}(\mathcal{N})$ of $\mathcal{N} = (G, \sigma, w)$, where $\mathcal{D}$ is a fixed but arbitrary and unknown probability distribution on $\mathcal{E}$.

8 Where Do They Stand?

We now discuss the main benefits and limitations of ARM, FCA, ILP, CLT, and NNs for building DL ontologies, considering the goals listed in the Introduction.

Interpretability refers to the easiness of understanding the learned DL ontology/concept expressions and obtaining insights about the domain. In ARM, the requirement for computing CIs with high support often results in highly interpretable CIs (at the cost of fixing the length of concept expressions). The FCA approach classically deals with redundancies, which is often not considered in ARM approaches. However, the CIs generated with this approach can be difficult to interpret [7]. The ILP approach follows the Occam's razor principle, which contributes to the generation of interpretable DL expressions, although there is no guarantee for the quality of the approximation. Such guarantees can be found in CLT, where the goal is to approximate or exactly identify the target. However, the focus of these approaches is on accuracy rather than interpretability. Regarding NNs, the complex models can deal with high variability in the data but may lose on interpretability.

Expressivity refers to the expressivity of the DL language supported by the learning process. As we have seen, many previous approaches for learning DL ontologies focus on Horn fragments such as $\mathcal{EL}$ [4, 7, 11, 24, 25, 28] (or Horn-like fragments such as $\mathcal{FLE}$ [40]). Non-horn fragments have been investigated for learning DL ontologies [41, 46] and concept expressions [12, 22, 29] (fixing the length of concept expressions). As mentioned, $\mathcal{ALCQ}$ CIs can be learned with NNs [38] (see also [49]).

Efficiency refers to the amount of time and memory consumed by algorithms in order to build a DL ontology (or concept expressions) in the context of a particular approach or a learning model. In CLT one can formally establish complexity results for learning problems. In ARM the search space is heavily constrained by the support function, which means that usually large portions of the search space can be eliminated in this approach. The Next-closure algorithm used in FCA is polynomial in the output and has polynomial delay, meaning that from the theoretical point of view it has interesting properties regarding efficiency. However, in practice, there may be difficulties in processing large portions of data provenient of knowledge graphs, such as DBpedia [7].

Human interactions may be required to complete the information given as input or to validate the knowledge that cannot be represented in a finite dataset or in a finite interpretation (recall the case of an infinite chain of objects in Sect. 4.2). Since the input is simply a database or an interpretation, the ARM and FCA approaches require limited or no human intervention. It is worth to point out that some DL adaptations of the FCA approach depend on an expert which resembles a membership oracle. The difference is that in the exact learning model the membership oracle answers with 'yes' or 'no', whereas in FCA the oracle also provides a counterexample if the answer is 'no' [40]. In ILP, examples need to be classified into positive and negative, which may require human intervention to classify the examples before learning takes place. The same happens with the CLT models presented. The exact learning model is purely based on interactions with an oracle, which can be an expert (or even a neural network [47]).

Unsupervised learning is supported by the ARM and FCA approaches, as well as some NNs (but not by the DL adaptation we have seen in the literature [38]). As already mentioned, the approaches based on ILP and CLT fall in the supervised setting. That is, examples receive some sort of (usually binary) classification.

Inconsistencies and noise are often present in the data. The ARM approach deals with them by only requiring that the confidence of the CI is above a certain threshold (instead of requiring that the CI is fully satisfied, as in FCA). ILP and CLT classically do not support inconsistencies and noise, though, the PAC model has an agnostic version in which it may not be possible to construct a hypothesis consistent with the positive and negative examples (due e.g. to noise in the classification). NNs can deal very well with data variability, including cases with inconsistencies and noise.

9 Conclusion

We discussed benefits and limitations of, namely, ARM, FCA, ILP, CLT, and NNs for DL settings. Not many authors have applied NNs for learning DL ontologies (when the focus is on building the logical expressions), even though NNs are widespread in many areas. We believe that more works exploring this approach are yet to come. One of the challenges is how to capture the *semantics* of the domain. Promising frameworks for capturing the semantics of logical expressions [17, 39] and modelling logical rules [21] have been recently proposed. Each approach addresses some of the desired properties of an ontology learning process. An interesting question is whether they can be combined so as to obtain the best of each approach [36].

Acknowledgements Open Access funding provided by University of Bergen. This work is supported by the Free University of Bozen-Bolzano.

References

1. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. SIGMOD 22(2):207–216
2. Angluin D (1988) Queries and concept learning. Mach Learn 2(4):319–342
3. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P (eds) (2007) The description logic handbook: theory, implementation, and applications, 2nd edn. Cambridge University Press, Cambridge
4. Baader F, Distel F (2009) Exploring finite models in the description logic. In: ICFCA, pp 146–161
5. Baader F, Ganter B, Sertkaya B, Sattler U (2007) Completing description logic knowledge bases using formal concept analysis. In: IJCAI, pp 230–235
6. Blum AL (1994) Separating distribution-free and mistake-bound learning models over the boolean domain. SIAM J Comput 23(5):990–1000
7. Borchmann D, Distel F (2011) Mining of ${\cal{E}}{\cal{L}}$-GCIs. In: ICDM workshops
8. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. NeurIPS, pp 2787–2795
9. Cohen WW, Hirsh H (1994) Learning the CLASSIC description logic: theoretical and experimental results. In: KR, pp 121–133
10. Distel F (2011) Learning description logic knowledge bases from data using methods from formal concept analysis. Ph.D. thesis, Dresden University of Technology
11. Duarte MRC, Konev B, Ozaki A (2018) Exactlearner: a tool for exact learning of EL ontologies. In: KR, pp 409–414
12. Fanizzi N, d'Amato C, Esposito F (2008) DL-FOIL concept learning in description logics. In: ILP, pp 107–121
13. Fleischhacker D, Völker J, Stuckenschmidt H (2012) Mining RDF data for property axioms. In: OTM, pp 718–735
14. Frazier M, Pitt L (1996) Classic learning. Mach Learn 25(2–3):151–193
15. Funk M, Jung JC, Lutz C, Pulcini H, Wolter F (2019) Learning description logic concepts: when can positive and negative examples be separated? In: IJCAI, pp 1682–1688
16. Galárraga L, Teflioudi C, Hose K, Suchanek FM (2015) Fast rule mining in ontological knowledge bases with AMIE+. VLDB J 24(6):707–730
17. Galliani P, Kutz O, Porello D, Righetti G, Troquard N (2019) On knowledge dependence in weighted description logic. In: GCAI, pp 68–80
18. Ganter B, Rudolph S, Stumme G (2019) Explaining data with formal concept analysis. In: RW, pp 153–195
19. Ganter B, Wille R (1997) Formal concept analysis: mathematical foundations. Springer, Berlin
20. Guigues JL, Duquenne V (1986) Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math Sci Hum 95:5–18
21. Gutiérrez-Basulto V, Schockaert S (2018) From knowledge graph embedding to ontology embedding? An analysis of the compatibility between vector space representations and rules. In: KR, pp 379–388
22. Iannone L, Palmisano I, Fanizzi N (2007) An algorithm based on counterfactuals for concept learning in the semantic web. Appl Intell 26:139–159
23. Klarman S, Britz K (2015) Ontology learning from interpretations in lightweight description logics. In: ILP
24. Konev B, Lutz C, Ozaki A, Wolter F (2018) Exact learning of lightweight description logic ontologies. JMLR 18(201):1–63
25. Konev B, Ozaki A, Wolter F (2016) A model for learning description logic ontologies based on exact learning. In: AAAI, pp 1008–1015
26. Lehmann J (2009) DL-learner: learning concepts in description logics. JMLR 10:2639–2642
27. Lehmann J (2010) Learning OWL class expressions, vol 6. IOS Press, Amsterdam
28. Lehmann J, Haase C (2009) Ideal downward refinement in the EL description logic. In: ILP, pp 73–87
29. Lehmann J, Hitzler P (2010) Concept learning in description logics using refinement operators. Mach Learn 78(1–2):203–250
30. Lehmann J, Völker J (2014) Perspectives on ontology learning, vol 18. IOS Press, Amsterdam
31. Lisi FA (2011) Al-quin: an onto-relational learning system for semantic web mining. Int J Semant Web Inf Syst 7:1–22
32. Ma Y, Distel F (2013) Learning formal definitions for Snomed CT from text. In: AIME, pp 73–77
33. Maedche A, Staab S (2001) Ontology learning for the semantic web. IEEE Intell Syst 16:72–79
34. McCulloch WS, Pitts W (1988) A logical calculus of the ideas immanent in nervous activity. In: Neurocomputing: foundations of research. MIT Press, pp 15–27
35. Muggleton S (1991) Inductive logic programming. New Gen Comput 8(4):295–318
36. Obiedkov S, Sertkaya B, Zolotukhin D (2019) Probably approximately correct completion of description logic knowledge bases. In: DL
37. Ozaki A, Persia C, Mazzullo A (2020) Learning query inseparable ELH ontologies In: AAAI
38. Petrucci G, Ghidini C, Rospocher M (2016) Ontology learning in the deep. In: EKAW, pp 480–495
39. Porello D, Kutz O, Righetti G, Troquard N, Galliani P, Masolo C (2019) A toothful of concepts: towards a theory of weighted concept combination. In: DL
40. Rudolph S (2004) Exploring relational structures via FLE. In: ICCS
41. Sazonau V, Sattler U (2017) Mining hypotheses from data in OWL: advanced evaluation and complete construction. In: ISWC, pp 577–593
42. Shalev-Shwartz S, Ben-David S (2014) Understanding machine learning: from theory to algorithms. Cambridge University Press, Cambridge
43. Stepanova D, Gad-Elrab MH, Ho VT (2018) Rule induction and reasoning over knowledge graphs. In: RW, pp 142–172
44. Valiant LG (1984) A theory of the learnable. Commun ACM 27(11):1134–1142
45. Völker J, Fleischhacker D, Stuckenschmidt H (2015) Automatic acquisition of class disjointness. J Web Semant 35:124–139

46. Völker J, Niepert M (2011) Statistical schema induction. In: The semantic web: research and applications. Springer, Berlin, pp 124–138
47. Weiss G, Goldberg Y, Yahav E (2018) Extracting automata from recurrent neural networks using queries and counterexamples. In: ICML, pp 5244–5253
48. Yang B, Yih W, He X, Gao J, Deng L (2015) Embedding entities and relations for learning and inference in knowledge bases. In: ICLR
49. Zhu M, Gao Z, Pan JZ, Zhao Y, Xu Y, Quan Z (2015) Tbox learning from incomplete data by inference in belnet${}^{\text{+ }}$. Knowl Based Syst 75:30–40