Tree-Based OWL Class Expression Learner over Large Graphs

Caglar Demir(✉), Moshood Yekini, Michael Röder, Yasir Mahmood, and Axel-Cyrille Ngonga Ngomo

Data Science Research Group, Department of Computer Science, Paderborn University, Paderborn, Germany
{caglar.demir,moshood.olawale.yekini,michael.roeder, yasir.mahmood,axel.ngonga}@upb.de

Abstract. Learning continuous vector representations for knowledge graphs has significantly improved state-of-the-art performances in many challenging tasks. Yet, deep-learning-based models are only post-hoc and locally explainable. In contrast, learning Web Ontology Language (OWL) class expressions in Description Logics (DLs) is ante-hoc and globally explainable. However, state-of-the-art learners have two well-known limitations: scaling to large knowledge graphs and handling missing information. Here, we present a decision-tree-based learner (tDL) to learn Web Ontology Languages (OWLs) class expressions over large knowledge graphs, while imputing missing triples. Given positive and negative example individuals, tDL firstly constructs unique OWL expressions in $\mathcal{SHOIN}$ from concise bounded descriptions of individuals. Each OWL class expression is used as a feature in a binary classification problem to represent input individuals. Thereafter, tDL fits a CART decision tree to learn Boolean decision rules distinguishing positive examples from negative examples. A final OWL expression in $\mathcal{SHOIN}$ is built by traversing the built CART decision tree from the root node to leaf nodes for each positive example. By this, tDL can learn OWL class expressions without exploration, i.e., the number of queries to a knowledge graph is bounded by the number of input individuals. Our empirical results show that tDL outperforms the current state-of-the-art models across datasets. Importantly, our experiments over a large knowledge graph (DBpedia with 1.1 billion triples) show that tDL can effectively learn accurate OWL class expressions, while the state-of-the-art models fail to return any results. Finally, expressions learned by tDL can be seamlessly translated into natural language explanations using a pre-trained large language model and a DL verbalizer.

Keywords: Decision Tree · OWL Class Expression Learning · Description Logic · Knowledge Graph · Large Language Model · Verbalizer

R. P. Ribeiro et al. (Eds.): ECML PKDD 2025, LNAI 16015, pp. 495–511, 2026.
https://doi.org/10.1007/978-3-032-06066-2_29

1 Introduction

Explainability is quintessential to establish trust in Artificial Intelligence (AI) decisions [32]. Its significance becomes particularly pronounced when AI algorithms rely on large amounts of data, e.g., the Web – an extensive and widely utilized information infrastructure that serves over 5 billion users worldwide. For instance, the recent success of large language models is built upon a crawled Web corpus comprising raw web page data, metadata extracts, and text extracts [5,17,30,34]. A key development over the last decade has been the increasing availability of Web data in the form of large-scale Resource Description Framework (RDF) Knowledge Graphs (KGs) [16]. According to the 2022 crawl of WebDataCommons, roughly 50% of the Web sites now contain (fragments of) RDF KGs.[1] The giant joint Knowledge Graph (KG) that can be extracted from the Web is known to contain at least 82 billion triples [4]. However, when analyzing or processing such a large graph, explainability and scalability remain challenging. For example, while learning continuous vector representations for knowledge graphs has significantly improved state-of-the-art performances in many challenging tasks [10,29], many deep-learning-based models that such solutions rely on, are only post-hoc and locally explainable [25]. In contrast, OWL class expression learning is ante-hoc explainable and showed good performance in areas like ontology engineering [21], bio-medicine [24], and Industry 4.0 [8]. However, most symbolic class expression learners cannot operate well on large KGs having millions of triples.

Our work contributes to a wider domain of designing scalable and explainable Machine Learning (ML) approaches for learning OWL class expressions over large RDF knowledge graphs. An OWL class expression represents a set of individuals by formally specifying conditions on the properties of individuals [26]. Such class expressions[2] in description logic syntax are ante-hoc explainable and intrinsically human-readable, e.g.,$\exists \texttt{award}.\{\texttt{NobelPrizeInPhysics}\}$ represents a set of individuals being awarded a Nobel prize in physics. At the same time, they can be used by machines, e.g., the set of individuals satisfying this expression in the DBpedia KG can be retrieved via the following SPARQL query [12]:

```
SELECT DISTINCT ?x WHERE
{
    ?x dbo:award ?s_1 .
    FILTER (?s_1 IN
        (dbr:Nobel_Prize_in_Physics))
}
```

[1] 1.51 billion of the 3.20 billion URLs crawled by the Web Data Commons contained RDF, see http://webdatacommons.org/structureddata, accessed on April 25th, 2024.

[2] An OWL class expression can also be called a Description Logic (DL) concept. We will stick to the first term throughout this paper.

Thus the given class expression can be seen as a human-readable, interpretable, and binary descriptor, identifying a set of 163 individuals as positive within the DBpedia KG.[3]

Learning class expressions relies on two sets of given examples: a set of positive examples (E^+) that the target class expression should describe while differentiating them from a set of negative examples (E^-).[4] For example, given the positive examples $E^+ = \{\texttt{Roger Penrose}, \texttt{Paul Dirac}\}$ and the negative examples $E^- = \{\texttt{Barack Obama}, \texttt{George R. R. Martin}\}$, a class expression learning algorithm using the DBpedia KG may return the class expression provided above, since the expression's entailment holds for Penrose and Dirac but not for Obama and Martin. However, although state-of-the-art approaches effectively tackle the class expression learning problem on benchmark datasets, their deployment in large-scale applications remains unrealized. Early approaches reformulate the learning problem as a search problem in an infinite, quasi-ordered search space [21,23]. Therein, the search begins by applying a downward refinement operator to the root state (i.e., the most general class expression $\top$). Thereafter, the search is guided by a heuristic function. However, recent results suggest that search-based symbolic learners do not scale well on large KGs [9]. This limitation arises from the fact that search-based symbolic learners must explore the space of OWL class expressions to identify an accurate one. The potentially vast search space introduces significant challenges, serving as the principal barrier to learning class expressions over large KGs [3,9].

In this work, we present a solution to the search space exploration problem in state-of-the-art models. Our proposal is a decision tree-based learner (tDL) that can tackle the class expression learning problem over large KGs by performing retrieval operations (i.e., retrieving qualifying individuals of a class expression) only at most $|E^+ \cup E^-|$ times. More specifically, $\forall e \in E^+ \cup E^-$, tDL first retrieves the first hop information about examples (elucidated in 4). From these $|E^+ \cup E^-|$ sets of triples, tDL constructs unique class expressions using $\mathcal{SHOIN}$ Description Logic (DL). These expressions are then treated as features in a binary classification problem to describe examples numerically (as detailed in 4). Subsequently, tDL fits a CART decision tree to distinguish E^+ from E^- within the created feature space. The final class expression is built by traversing the built CART decision tree from the root node to leaf nodes for each positive example. Overall, our experimental results over three benchmark datasets with four state-of-the-art models indicate that tDL learns more accurate class expressions under a time constraint and generalizes better than state-of-the-art models including Drill [9] and EvoLearner [13]. Importantly, we show that tDL can learn OWL class expressions over knowledge graphs involving more than 1.1 billion triples in less than 2 min, while all other baselines lead to out-of-memory errors. Finally, we further improve the explainability by integrating a large language model and a verbalizer to translate potentially complex domain-specific class expressions into natural language sentences.

[3] https://dbpedia.org/sparql.

[4] We give a formal definition for class expression learning in Sect. 2.3.

The main contributions of this paper are as follows:

1. We propose an explainable and scalable ML approach (TDL) to learn OWL class expressions over large RDF KGs.
2. We conducted extensive experiments to benchmark the learning and generalization performance of our proposal against state-of-the-art models.
3. To the best of our knowledge, TDL is the first OWL class expression learner integrating a pre-trained language model and a verbalizer to translate domain-specific expressions into plain natural language sentences.

In the following Section, we introduce background knowledge about RDF, DLs, and class expression learning. After that, we present related work in Sect. 3 before explaining our approach in detail in Sect. 4. The evaluation and its results are described in Sect. 5 and discussed in Sect. 6. We conclude the paper in Sect. 7.

2 Background

2.1 RDF Knowledge Graphs and OWL

RDF is a formal language for describing structured information [15]. The goal of RDF is to enable applications to exchange data on the Web while still preserving their original meaning. OWL is designed to model the semantics of RDF KGs that facilitates machine interpretability of Web content by providing additional expressive power along with formal semantics [15]. OWL has three sublanguages: OWL Full (the most expressive but undecidable), OWL DL (expressive and decidable), and OWL Lite (decidable, less expressive). OWL Full contains OWL DL and OWL Lite, while OWL DL contains OWL Lite. OWL DL coincides with $\mathcal{SHOIN}(D)$ DL [15]. Note that any RDF KG forms an OWL Full ontology [2].

2.2 Description Logics

Description Logics (DLs) are fragments of first-order predicate logic using only unary and binary predicates [1,15,27]. A DL knowledge base corresponding to an OWL ontology is often defined as a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ denotes the set of terminological axioms describing the relationships between defined DL concepts. Every terminological axiom is of the form of $A \sqsubseteq B$ or $A \equiv B$ where A and B are DL concepts and $A, B \in N_C$. N_C denotes a set of atomic concepts corresponding to OWL named classes. $\mathcal{A}$ denotes the set of assertions describing relationships among DL individuals $a, b \in N_I$ via roles $r \in N_R$ as well as instantiation relationships. N_I and N_R denote the set of individuals and the set of DL roles corresponding to OWL properties, respectively. Thus $\mathcal{A}$ contains an assertion of the form $A(a)$ or $r(a, b)$, where $A \in N_C, r \in N_R$, and $a, b \in N_I$. Within this work, we focus on $\mathcal{SHOIN}$ DL. Table 1 depicts the syntax and semantics for $\mathcal{SHOIN}$ concepts.

Table 1. Syntax and semantics for concepts in $\mathcal{SHOIN}$ following Lehmann et al. [21]. $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation where $\Delta^{\mathcal{I}}$ is its domain and $\cdot^{\mathcal{I}}$ is the interpretation function.

Construct	**Syntax**	**Semantics**
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Nominals	$\{o\}$	$o^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}, \lvert o^{\mathcal{I}}\rvert = 1$
Inverse Role	r^{-}	$\{(b^{\mathcal{I}}, a^{\mathcal{I}}) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\}$
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom Concept	$\bot$	$\emptyset$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Exists Restriction	$\exists\, r.C$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \exists\, b^{\mathcal{I}} \in C^{\mathcal{I}}, (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\}$
Universal Restriction	$\forall\, r.C$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \forall\, b^{\mathcal{I}}, (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} \Rightarrow b^{\mathcal{I}} \in C^{\mathcal{I}}\}$
Atmost Restriction	$\geq n\ r.C$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \lvert\{b^{\mathcal{I}} \in C^{\mathcal{I}} : (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\}\rvert \geq n\}$
Atleast Restriction	$\leq n\ r.C$	$\{a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \mid \lvert\{b^{\mathcal{I}} \in C^{\mathcal{I}} : (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\}\rvert \leq n\}$

2.3 OWL Class Expression Learning

Definition 1 (OWL Classical Learning Problem). *Given a DL knowledge base $\mathcal{K}$, a set of positive individuals $E^+ \subset N_I$, and a set of negative OWL individuals $E^- \subset N_I$ s.t. $E^+ \cap E^- = \emptyset$, the learning problem is to find an OWL class expression H s.t.*

$$\forall p \in E^+ :\ \ \mathcal{K} \models H(p)\ \ and\ \ \forall n \in E^- : \mathcal{K} \not\models H(n)\,. \tag{1}$$

Traditionally, this learning problem is transformed into a search problem within a quasi-ordered concept space $(\mathcal{C}, \preceq)$ [21], where $\mathcal{C}$ denotes all valid OWL class expressions in a DL. An OWL class expression learner (e.g. OCEL, CELOE [22]) applies a downward refinement operator $\rho : \mathcal{C} \rightarrow 2^{\mathcal{C}}$ to traverse in $\mathcal{C}$, e.g., $\texttt{Mother} \preceq \rho(\texttt{Female})$. To steer the search starting from $\top$ to H satisfying 1, a fixed heuristic function is often applied. Most heuristic functions are based on the quality of the traversed OWL class expressions. One of these metrics is the F1 score, which is defined as

$$F_1(H) = \frac{|E^+ \cap \mathcal{R}(H)|}{|E^+ \cap \mathcal{R}(H)| + \frac{1}{2}(|E^- \cap \mathcal{R}(H)| + |E^+ \setminus \mathcal{R}(H)|)}, \tag{2}$$

where $\mathcal{R}$ denotes a concept retrieval operation that maps a class expression to a subset of N_I. As the size of $\mathcal{K}$ grows, computing the quality of a class expression becomes a bottleneck due to this mapping process [3,9,20]. Recent state-of-the-art models (e.g. Drill [9], EvoLearner [13]) apply reinforcement learning or evolutionary algorithms to find H as elucidated in Sect. 3. In contrast, our proposed approach only needs a refinement operator to generate its training data and does not make use of the mapping process.

3 Related Work

DL-Learner [21] is regarded as the most mature and recent system for class expression learning [6,9]. DL-Learner comprises several state-of-the-art approaches, including OCEL and CELOE [22]. Both consider the OWL class expression learning problem as a search problem in a quasi-ordered space of OWL class expressions in DLs. To traverse the search space, OCEL and CELOE apply a downward refinement operator while relying on different statistical heuristic functions. During the search, CELOE prioritizes syntactically shorter expressions. CELOE and OCEL apply the redundancy elimination and the expression simplification rules to decrease their runtimes. Although applying such fixed rules may reduce the number of explored expressions, long runtimes and extensive memory requirements still prohibit large-scale applications of such refinement-operators-based approaches [9,11,14]. Rizzo et al. [31] follow the general idea of a refinement-operator-based approach but focus on a small number of class expressions, which are used to create several decision trees that are combined similar to a random forest. Recent works often focus on accelerating the learning process. DRILL [9] uses a deep Q-network instead of a fixed heuristic function to steer the search more efficiently towards accurate OWL class expressions. CLIP [18] prunes the search space by introducing an upper bound on the length of the OWL class expressions. NCES [19,20] uses deep neural networks to learn mappings between sets of examples and class expressions without a search process. EvoLearner [13] is based on evolutionary algorithms and initializes its population (i.e., class expressions) by random walks on the input RDF KG.

Compared to the state-of-the-art models, TDL uses multi-hop information about E^+ and E^- to detect relevant OWL class expressions. Such expressions are then used as binary features for a supervised binary classification problem. TDL builds a decision tree algorithm (e.g. CART) to tackle this supervised binary classification problem, where a node corresponds to OWL class expression. Therefore, TDL learns Boolean rules to distinguish positive and negative examples. Instead of using a fixed handcrafted heuristic, TDL can use the Shannon information gain to recursively partition the feature space (i.e., the extracted class expression space) such that the positive examples are grouped together.

4 Methodology

Supervised Binary Classification. Given two ordered sets $E^+, E^- \subset N_I$ with $E^+ \cap E^- = \emptyset$ and a knowledge base $\mathcal{K}$, we firstly extract the first hop information as

$$\mathcal{F} = \bigcup_{x \in E} \{(x, p, o) \mid (x, p, o) \in \mathcal{K}\}, \tag{3}$$

where $E = E^{+} \cup E^{-}$. Given $\mathcal{F}$, a set of relevant atomic concepts can be defined as

$$\mathcal{FC} = \bigcup_{(s,p,o)\in\mathcal{F}} \{o \mid o \in N_C\}. \tag{4}$$

Similarly, relevant restrictions over N_I and N_C can be defined as

$$\mathcal{FRO} = \bigcup_{(s,p,o)\in\mathcal{F}\ \wedge\ o\in N_I} \{\exists p.\{o\}\} \cup \{\forall p.\{o\}\} \tag{5}$$

and

$$\mathcal{FRC} = \bigcup_{(s,p,o)\in\mathcal{F}\ \wedge\ o\in N_I} \{\exists p.\text{type}(o)\} \cup \{\forall p.\text{type}(o)\}, \tag{6}$$

where type : $N_I \rightarrow N_C$ returns a subset of N_C that a given OWL individual belongs to. Since an individual can belong to multiple atomic concepts, a complex conjunctive concept can be constructed and included in $\mathcal{FRC}$. Similarly, since an individual can appear with other nominals with same property, a complex restriction involving multiple nominals can also be included in $\mathcal{FRO}$. Multi-hop expression can be extracted by extending the fillers in $\mathcal{FRO}$ and $\mathcal{FRC}$. By introducing counting over $\mathcal{FRO}$ and $\mathcal{FRC}$, Atmost and Atleast restrictions can be obtained.

Through these extractions over first-hop information about E, we enable a decision tree to leverage multi-hop information for the learning problem. Through $\mathcal{FC}$, $\mathcal{FRO}$ and $\mathcal{FRC}$, we firstly build the training data $\mathbf{X} \in \mathbb{R}^{|E|\times(|\mathcal{F}|+|\mathcal{FRO}|+|\mathcal{FRC}|)}$ and $\mathbf{y} \in \{0,1\}^{|E|}$ where the j-th feature corresponds to a Boolean feature (e.g. $\exists$`hasChild.{Julia}`) of $\mathbf{X}_{i,j} \in \{1.0, 0.0\}$ for the i-th individual in E. After $\mathbf{X}, \mathbf{y}$ are constructed, we fit a decision tree to learn binary decision rules to distinguish the positive individuals E^{+} from the negative individuals E^{-}.

Running Example. Fig. 1 visualizes an example tree built on $\mathbf{X}$ and $\mathbf{y}$ for the Aunt benchmark learning problem. The rationale of using a decision tree is that it can be seen as a piece-wise constant approximation, where each decision is ante-hoc explainable [32]. For instance, being a Female and having a sibling being Mother can be important to distinguish E^{+} from E^{-}. The entropy decreases from 1.0 to 0.0 when classifying 25 positive individuals $\texttt{x} \in E^{+}$ as positive class, each of them fulfilling the following: (i) (`x, type, Female`) $\wedge$ (`x, hasSibling, y`) $\wedge$ and (ii) (`y, hasChild, z`) $\wedge$ (`z, type, Person`), where $y, z \in N_I$ and `hasSibling`, `hasChild`, `type` $\in N_R$. Importantly, with a decision tree, we can rank class expressions in descending order w.r.t. their normalized total reduction of entropy.

From a Decision Tree to a DL Concept. After creating a decision tree $T(\cdot)$ using $\mathbf{X}$ and $\mathbf{y}$, we construct a class expression. Let N be the set of decision nodes of $T(\cdot)$ and let the elements of N have two types: leaf nodes L and decision nodes D with $L \cup D = N$. A leaf node $l \in L$ contains a class label $c \in \{-1, +1\}$. We describe a decision node $d_i \in D$ as triple $d_i = (\sigma_i, n_i^+, n_i^-)$, where σ_i is a condition based on a feature in $\mathcal{F}$, while n_i^+ and n_i^- are child nodes of d_i in the tree.

Building Conjunctive DL Concepts. When classifying a given individual, the decision tree algorithm starts at the root node and traverses the tree. The traversal depends on whether the current node is a leaf node or a decision node. If it is a leaf node, the node's class is assigned to the example. If it is a decision node, the example is tested against the node's condition. If the example fulfills the condition, the algorithm traverses to the n_i^+ child node. Otherwise, the traversal continues with the n_i^- child node. We consider each node as a class expression; hence, for a given positive individual, we construct a conjunctive class expression from all nodes seen along the respective traversal. For a given $e \in E^+$, let $\pi = \{n_1, ..., n_{|\pi|} | n_{|\pi|} \in L, \forall i < |\pi| : n_i \in D\}$ denote a sequence of qualifying nodes starting from the root node and ending with a leaf node. From π, a conjunctive class expression can be obtained as

$$C_\pi = \bigsqcap_{i=1} \sigma_i, \tag{7}$$

where $C_\pi(e)$ is the observed explainable features of e at inference by the decision tree classifier. Let Π denote a set of conjunctive class expressions constructed for E^+. For the running example shown in Fig. 2, $\mathit{\Pi}$ contains the following three class expressions:

$$\texttt{Female} \sqcap (\exists\ \texttt{hasSibling.Father}) \tag{8}$$

$$\texttt{Female} \sqcap (\neg(\exists\ \texttt{hasSibling.Father}) \sqcap (\exists\ \texttt{married.Brother}) \tag{9}$$

$$\texttt{Female} \sqcap (\neg(\exists\ \texttt{hasSibling.Father})) \sqcap (\neg(\exists\ \texttt{married.Brother})) \sqcap (\exists\ \texttt{hasSibling.Mother}) \tag{10}$$

Note that since Π is a set, only distinct sequences of decision tree nodes are transformed into class expressions. By this, we aim to reduce redundancy, i.e., the length of generated concepts.

Disjunction of Conjunctive DL Concepts. A final prediction for a given class expression learning problem is then computed as

$$H = \bigsqcup_{C_\pi \in \Pi} C_\pi. \tag{11}$$

For the running example, H corresponds to the disjunction of Equations (8) to (10). Hence, TDL can tackle a class expression learning problem **without a single retrieval operation**. Recall that as the size of the input knowledge base grows, performing retrieval operations to compute the quality of a class expression becomes a computational bottleneck.

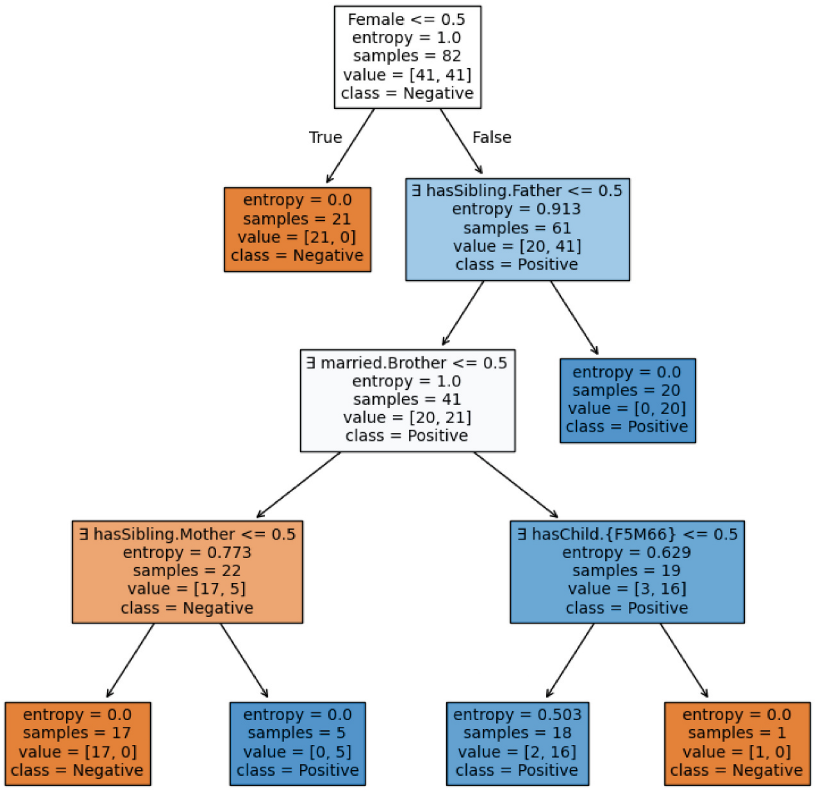

Fig. 1. The built decision tree for the Aunt benchmark learning problem. Outgoing arrows mark the path of examples that either fulfill or neglect the originating node's condition, and `value` array shows counts of observed negative and positive examples

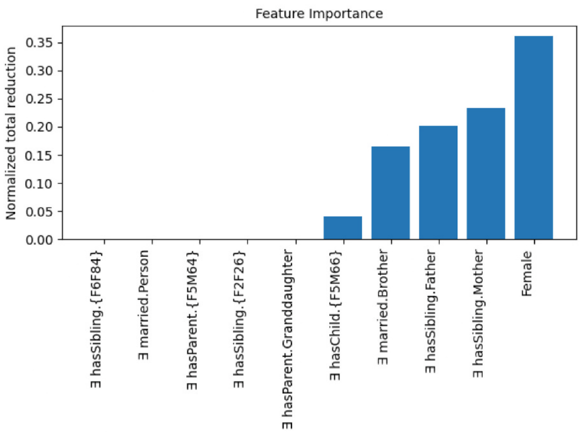

Fig. 2. Normalized total entropy reduction of important features for the Aunt benchmark learning problem with generated goal concept as: `Aunt` $\equiv$ `Female` $\sqcap$ ($\exists$ `hasSibling.Mother` $\sqcup \exists$`hasSibling.Father` $\sqcup$ ($\exists$texttmarried.Brother($\exists$`hasChild`.$\top$))).

From DL Concepts to Natural Language Sentences. We investigate techniques to translate a predicted class expression into natural language sentences. Therewith, we aim to enable non-domain experts to interpret predictions. To this end, we applied a large language model with or without an OWL verbalizer. We explored several system prompts and used the following one:

```
   You are an expert. Be concise in your answers and translate
this description logic concept into English sentences.
   Provide no explanations: $
```

where `$` is replaced with a class expression.

5 Experiments

5.1 Datasets

We use three benchmark datasets—Family, Carcinogenesis, and Mutagenesis—obtained from [9]. In addition, we use the English DBpedia as example of a large KG together with three example learning problems created manually using prominent politicians.[5] Table 2 gives an overview of datasets.

Table 2. Overview of benchmark datasets and total learning problems. $|\mathcal{K}|$, $|N_I|$, $|N_C|$, $|N_R|$ and LP denote the number of triples, individuals, concepts, roles, and learning problems, respectively.

Dataset	$\|\mathcal{K}\|$	$\|N_I\|$	$\|N_C\|$	$\|N_R\|$	LP
Family	2,032	202	18	4	18
Mutagenesis	62,067	14,145	86	11	1
Carcinogenesis	96,939	22,372	142	21	1
DBpedia	1,151,575,981	42,042,875	1,568	1,194	3

5.2 Experimental Setup

We base our experimental setup on [9] and use all learning problems provided by the datasets. We compare approaches based on their F1-scores for predicted class expression and their runtimes. On each dataset, each model is initialized once. For each learning problem, the time needed for the inference of the class expression is measured as runtime. We use two standard stopping criteria for all approaches. (i) We set the maximum runtime to 30 s (60 min for the DBpedia). (ii) Approaches were configured to terminate as soon as they find a goal state (i.e., a state with F1-score = 1.0). Note that (i) is a soft constraint as the runtime

[5] We use the English DBpedia version 2022-12.

criterion is not checked during all the steps of some of the evaluated approaches. If models do not find a goal state, the most accurate state is retrieved.

In a second experiment, we quantify the generalization performance of all approaches using a 10-fold cross-validation on the provided learning problems of the first three datasets. All experiments have been executed on a DELL Precision 3591 with an Intel Core Ultra 7 165H CPU, 64GB RAM, and Ubuntu 22.05. We provide an open-source implementation of TDL, including scripts for training and evaluation to ensure the reproducibility of our results using the ONTOLEARN framework [7].[6]

5.3 Experimental Results

Learning Class Expressions Under Time Restriction. Table 3 reports the concept learning results on the three benchmark datasets with benchmark learning problems as previously done in [9]. Overall, TDL outperforms OCEL, CELOE, Evolearner, and Drill w.r.t. F1-score and runtimes. On all 18 learning problems of the Family dataset, TDL reaches the goal state while requiring less runtime. Compared to OCEL, CELOE, and Drill, EvoLearner underperforms considerably. After these results, we delved into the implementation of EvoLearner and analyzed the learning problems on each datasets. We made the following two observations: (i) If the input knowledge base is not reloaded from the disk into memory for each learning problem, the performance of EvoLearner degenerates. (ii) Goal concepts for some of the benchmark learning problems (e.g. Brother, Daughter, Father, Sister) on the Family benchmark dataset can be found via a linear search over the set of defined class expressions, i.e., a class expression satisfying 1 is already defined in $\mathcal{K}$. In some of these cases, EvoLearner fails in identifying these existing concepts as solutions. To address (1), we reran our experiments on the Family dataset for EvoLearner and we reload the knowledge base for each learning problem. Although this setting increases the runtimes by 1.3 s on average, EvoLearner finds a goal concept having F1-score of 1.0 for all 18 learning problems on the Family benchmark dataset.

The learning problems created for the DBpedia KG and the evaluation results are listed in Table 4. TDL is the only algorithm that is able to solve the learning problems. All other approaches terminate with an out-of-memory error due to the size of the graph and the intermediate results they retrieve from it.

K-Fold Cross Validation. Tables 5 and 6 report the 10-fold cross-validation results on the benchmark datasets. Overall, results indicate that TDL outperforms OCEL, CELOE, Drill, and Evolearner in nearly all metrics. Only for the Mutagenesis dataset, TDL achieves a slightly lower F1-score than OCEL and CELOE. The results also show that the generalization performance of OCEL, CELOE, Drill and TDL do not fluctuate, whereas the generalization performance of Evolearner differs extremely (up to 50% F1-score differences between learning problems).

[6] https://github.com/dice-group/Ontolearn/tree/tdl.

Table 3. Class expression learning results on the benchmark datasets with benchmark learning problems. F1 and RT denote the F1-score of learned concept w.r.t. E^+ and E^- and runtime in seconds, respectively. Maximum runtime is set to 30 s. Goal concepts with † denote that the respective concept is defined in $\mathcal{K}$, i.e., a goal concept can be found via a linear search over $\mathcal{K}$. Bold and underlined results indicate the best results and second best results.

Dataset	**Goal Concept**	**OCEL**		**CELOE**		**EvoLearner**		**Drill**		**NCES**		**TDL**	
		F1	RT	F1	RT	F1	RT	F1	RT	F1	RT	F1	RT
Family	Aunt	0.837	15.008	0.911	8.101	1.000	1.523	0.863	30.036	0.804	1.622	1.000	0.061
	Uncle	0.905	19.639	0.905	7.339	0.483	0.254	0.927	30.038	0.884	1.853	1.000	0.043
	Cousin	0.721	12.067	0.793	8.972	0.348	0.335	0.826	30.047	0.686	1.870	1.000	0.060
	Grandgranddaughter	1.000	0.003	1.000	0.001	1.000	0.299	1.000	0.003	1.000	1.902	1.000	0.024
	Grandgrandfather	1.000	0.846	1.000	0.182	0.829	0.295	1.000	0.408	0.944	1.726	1.000	0.032
	Grandgrandmother	1.000	2.506	1.000	0.259	1.000	0.272	1.000	0.400	0.944	1.790	1.000	0.037
	Grandgrandson	1.000	2.082	1.000	0.198	0.486	0.365	1.000	0.224	0.923	1.707	1.000	0.030
	Brother†	1.000	0.033	1.000	0.007	1.000	0.394	1.000	0.051	1.000	1.699	1.000	0.036
	Daughter†	1.000	0.026	1.000	0.009	1.000	0.366	1.000	0.048	1.000	1.752	1.000	0.052
	Father†	1.000	0.004	1.000	0.002	1.000	0.340	1.000	0.008	1.000	1.946	1.000	0.178
	Granddaughter†	1.000	0.003	1.000	0.001	1.000	0.301	1.000	0.005	1.000	1.879	1.000	0.043
	Grandfather†	1.000	0.003	1.000	0.001	1.000	0.277	1.000	0.005	1.000	1.915	1.000	0.037
	Grandmother†	1.000	0.008	1.000	0.002	1.000	0.290	1.000	0.008	0.921	1.648	1.000	0.041
	Grandson†	1.000	0.004	1.000	0.002	1.000	0.336	1.000	0.007	1.000	1.783	1.000	0.043
	Mother†	1.000	0.004	1.000	0.002	0.000	0.277	1.000	0.007	1.000	1.843	1.000	0.055
	PersonWithASibling†	1.000	0.003	1.000	0.001	0.700	0.331	0.737	30.031	1.000	1.985	1.000	0.073
	Sister†	1.000	0.003	1.000	0.001	0.955	0.291	1.000	0.033	1.000	1.939	1.000	0.046
	Son†	1.000	0.005	1.000	0.002	0.905	0.293	1.000	0.008	0.905	1.843	1.000	0.052
	Avg. Results	0.970	2.903	$\underline{0.978}$	1.393	0.817	$\underline{0.380}$	0.964	6.743	0.945	1.817	**1.000**	**0.052**
Mutagenesis	Unknown	0.916	32.30	0.916	30.04	**0.980**	31.31	0.704	30.17	0.704	$\underline{8.324}$	$\underline{0.919}$	**4.050**
Carcinogenesis	Unknown	0.734	30.42	0.734	30.13	$\underline{0.807}$	30.98	0.705	30.19	0.705	$\underline{8.528}$	**0.973**	**3.370**

Translation Into Natural Language. We used Qwen2.5 32B Instruct-AWQ [33,35]and a verbalizer LD2NL [28] to translate the complex, learned class expressions into plain text. The LLM only gets the class expression while LD2NL needs $\mathcal{K}$ as additional input. The outputs of the verbalization given by Qwen2.5 32B Instruct-AWQ and LD2NL for the class expression prediction from TDL for the Aunt learning problem as an example.

Qwen2.5 32B Instruct-AWQ.

– Female who either has a sibling who is a Father and is married to a Brother or has a sibling who is a Mother and is not married to a Brother.'
– Female who has a sibling who is a Father, is married to a Brother or has a child who is F5M66, and Female who does not have a Father sibling, does not marry a Brother, and has a Mother sibling.

Table 4. Learning OWL class expression over 1.1 billion triples. F1 and RT denote the F1-score of learned concept w.r.t. E^+ and E^- and runtime in seconds, respectively. – denotes no results provided due to the out-of-memory error.

Learning Problem	**OCEL**		**CELOE**		**Evo**		**Drill**		**TDL**	
	F1	RT	F1	RT	F1	RT	F1	RT	F1	RT
$E^+ = \{$`B. Obama`$\}$, $E^- = \{$`A. Merkel`$\}$	–	–	–	–	–	–	–	–	1.000	62.18
$E^+ = \{$`B. Obama,A. Merkel`$\}$, $E^- = \{$`E. Macron`$\}$	–	–	–	–	–	–	–	–	1.000	67.33
$E^+ = \{$`B. Obama,A. Merkel,E. Macron`$\}$, $E^- = \{$`P. Sánchez`$\}$	–	–	–	–	–	–	–	–	1.000	71.41

Table 5. 10-fold cross-validated class expression learning results for the learning problems of the Family dataset. F1 and RT denote the average F1-scores of learned concepts on the 10-test folds and runtime in seconds, respectively. Maximum runtime is set to 30 s. Goal concepts with † denote that the respective concept is defined in $\mathcal{K}$, i.e., a goal concept can be found via a linear search over $\mathcal{K}$. Bold and underlined results indicate the best results and second best results.

Dataset	**Goal Concept**	**OCEL**		**CELOE**		**EvoLearner**		**Drill**		**NCES**		**TDL**	
		F1	RT	F1	RT	F1	RT	F1	RT	F1	RT	F1	RT
Family	Aunt	0.631	16.190	0.855	16.190	1.000	2.723	0.791	30.106	0.812	3.696	1.000	0.107
	Cousin	0.708	12.520	0.789	12.520	0.344	0.279	0.784	30.105	0.557	4.090	0.956	0.153
	Uncle	0.891	20.731	0.891	20.731	0.642	0.276	0.899	30.161	0.877	4.551	0.986	0.103
	Grandgranddaughter	1.000	0.010	1.000	0.010	0.800	0.236	1.000	0.003	1.000	4.202	1.000	0.077
	Grandgrandfather	1.000	0.657	1.000	0.657	0.860	0.195	1.000	0.795	0.947	4.118	0.913	0.079
	Grandgrandmother	1.000	5.060	1.000	5.060	1.000	0.233	1.000	0.816	0.947	4.314	0.880	0.076
	Grandgrandson	1.000	1.826	1.000	1.826	0.463	0.199	1.000	0.722	0.931	4.108	0.862	0.083
	Brother†	1.000	0.006	1.000	0.006	1.000	0.314	1.000	0.008	1.000	4.022	1.000	0.091
	Daughter†	1.000	0.006	1.000	0.006	0.900	0.323	1.000	0.015	0.967	3.966	1.000	0.113
	Father	1.000	0.004	1.000	0.004	1.000	0.280	1.000	0.007	1.000	4.225	1.000	0.109
	Granddaughter†	1.000	0.003	1.000	0.003	0.700	0.244	1.000	0.005	1.000	4.140	1.000	0.103
	Grandfather†	1.000	0.003	1.000	0.003	0.780	0.256	1.000	0.005	0.980	4.088	1.000	0.088
	Grandmother†	1.000	0.003	1.000	0.003	0.556	0.240	1.000	0.005	0.964	4.563	1.000	0.088
	Grandson†	1.000	0.004	1.000	0.004	0.666	0.240	1.000	0.007	1.000	4.377	1.000	0.107
	Mother†	1.000	0.004	1.000	0.004	0.625	0.219	1.000	0.007	0.929	4.441	1.000	0.113
	PersonWithASibling†	1.000	0.004	1.000	0.004	0.564	0.244	0.725	30.072	0.976	4.680	1.000	0.145
	Sister†	1.000	0.003	1.000	0.003	0.731	0.247	1.000	0.008	1.000	4.361	1.000	0.103
	Son†	1.000	0.004	1.000	0.004	0.710	0.212	1.000	0.007	0.943	4.363	1.000	0.106
Avg. Results		0.957	3.170	0.974	3.170	0.741	0.387	0.955	6.820	0.935	4.239	**0.978**	**0.102**

Table 6. 10-fold cross-validated class expression learning results on the Mutagenesis and Carcinogenesis datasets. F1 and RT denote the average F1-scores of learned concepts on the 10-test folds and runtime in seconds, respectively. Maximum runtime is set to 30 s.

Dataset	**Goal Concept**	**OCEL**		**CELOE**		**Evo**		Drill		NCES		TDL	
		F1	RT	F1	RT	F1	RT	F1	RT	F1	RT	F1	RT
Mutagenesis	Unknown	**0.918**	31.551	**0.918**	31.551	0.742	32.317	0.704	30.046	0.704	**10.062**	0.855	11.467
Carcinogenesis	Unknown	0.706	30.899	0.701	30.899	0.707	33.059	0.704	30.280	0.714	**9.844**	**0.747**	10.670

– A female who either has a sibling who is a father, is married to a brother but has no children, or has a sibling who is a mother but is not married to a brother and has no children.

LD2NL. Every predicted individual is that something that a female that has not as sibling has as child a person and that marries something that has as sibling a mother or that a female that has as sibling something that has as child a person or that something that a female that has not as sibling has as child a person and that does not marry has as sibling a mother and that marries something that has as sibling a son. Although the translation via Qwen2.5 32B Instruct-AWQ arguably is more fluent than the translation of LD2NL, LD2NL verbalizes the input at least 10 times faster and requires less memory.

6 Discussion

Although TDL finds more accurate concepts in less time in most of cases, the length of the extracted class expressions created from the decision tree can grow quite large in comparison to the results of the other approaches. A large, complex class expression may require more domain expert knowledge for the interpretation. Moreover, TDL currently does not support datatype properties, data values, or data types. Consequently, we expect that TDL may perform poorly if a goal concept is based on the usage of one of these features, e.g., the class of persons that are taller than 1.80 m.

We conjecture that the performance of TDL can be further improved in an iterative fashion. More specifically, after $\mathcal{F}$ is constructed and a decision tree is built, the most important features can be refined through a refinement operator and the refinements can be added into $\mathcal{F}$. Consider`married.Brother.hasChild` as shown in Fig. 2, many possible feature candidates can be inferred depending on schema design of $\mathcal{K}$, e.g. ($\exists$`married.Brother`($\exists$`hasChild.Male`))) or ($\exists$`married.Brother`($\exists$`hasChild.Person`))) provided `Male` $\sqsubseteq$ `Person`. Although this iterative process may allow the creation of a decision tree to find more compact rules to distinguish E^+ from E^-, it may require more careful hyperparameter optimization to alleviate possible overfitting.

7 Conclusion

In this work, we proposed TDL—a decision-tree-based learner for OWL class expressions. We explained how we use the decision tree to learn Boolean rules from a feature space comprising class expressions. Furthermore, we illustrated that the Boolean rules learned by said tree can be adeptly converted into class expressions. Additionally, we showed that domain-specific class expressions can be seamlessly translated into natural language sentences by employing a sophisticated language model enhanced with a verbalizer. Our evaluation showed that our approach TDL outperforms previous state-of-the-art approaches on all benchmarking datasets except one case. We also were able to show that tDL is the only approach that is able to provide results for learning problems on a large KG comprising 1.1 billion triples. In future work, we want to improve tDL further to cover its shortcomings discussed in Sect. 6. For example, we plan to extend the feature generation to be able to learn class expression in the $\mathcal{SHOIN}(D)$ DL.

Acknowledgements. This work has been supported by the European Union's Horizon Europe research and innovation programme within the project ENEXA under the grant No. 101070305, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 - 438445824, the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within projects SAIL under the grant No. NW21-059D and WHALE (LFN 1-04) funded under the Lamarr Fellow Network programme.

References

1. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
2. Bechhofer, S., et al.: OWL Web Ontology Language. W3c recommendation, W3C (February 2004). https://www.w3.org/TR/2004/REC-owl-ref-20040210/
3. Bin, S., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.C.: Towards sparql-based induction for large-scale rdf data sets. In: ECAI 2016, pp. 1551–1552. IOS Press (2016)
4. Bizer, C., Meusel, R., Primpeli, A., Brinkmann, A.: Web data commons – rdfa, microdata, embedded json-ld, and microformats data sets – october 2022. https://webdatacommons.org/structureddata/2022-12/stats/stats.html (2022). Accessed 15 May 2023
5. Brown, T., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)
6. Bühmann, L., Lehmann, J., Westphal, P., Bin, S.: Dl-learner structured machine learning on semantic web data. In: Companion Proceedings of the The Web Conference 2018. pp. 467–471. WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2018)
7. Demir, C., et al.: Ontolearn–a framework for large-scale owl class expression learning in python. J. Mach. Learn. Res. **26**(63), 1–6 (2025)
8. Demir, C., Himmelhuber, A., Liu, Y., Bigerl, A., Moussallem, D., Ngomo, A.C.N.: Rapid explainability for skill description learning. In: ISWC (Posters/Demos/Industry) (2022)
9. Demir, C., Ngomo, A.C.N.: Neuro-symbolic class expression learning. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pp. 3624–3632 (2023)
10. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
11. d'Amato, C.: Machine learning for the semantic web: lessons learnt and next research directions. Semantic Web **11**(1), 195–203 (2020)

12. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 Query Language. Recommendation, W3C (March 2013). http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
13. Heindorf, S., et al.: Evolearner: Learning description logics with evolutionary algorithms. In: Proceedings of the ACM Web Conference 2022, pp. 818–828 (2022)
14. Hitzler, P., Bianchi, F., Ebrahimi, M., Sarker, M.K.: Neural-symbolic integration and the semantic web. Semantic Web **11**(1), 3–11 (2020)
15. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of semantic web technologies (2009)
16. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (Csur) **54**(4), 1–37 (2021)
17. Jiang, A.Q., et al.: Mistral 7b. arXiv preprint arXiv:2310.06825 (2023)
18. Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Learning concept lengths accelerates concept learning in ALC. In: ESWC. Lecture Notes in Computer Science, vol. 13261, pp. 236–252. Springer (2022)
19. Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Neural class expression synthesis in $\mathcal{ALCHIQ(D)}$. In: ECML/PKDD (4). Lecture Notes in Computer Science, vol. 14172, pp. 196–212. Springer (2023)
20. Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.C.: Neural class expression synthesis. In: European Semantic Web Conference, pp. 209–226. Springer (2023)
21. Lehmann, J.: Learning OWL class expressions, vol. 22. IOS Press (2010)
22. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. J. Web Seman. **9**(1), 71–81 (2011). https://doi.org/10.1016/j.websem.2011.01.001, https://www.sciencedirect.com/science/article/pii/S1570826811000023
23. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Mach. Learn. **78**(1–2), 203 (2010)
24. Lehmann, J., Völker, J.: Perspectives on ontology learning, vol. 18. IOS Press (2014)
25. Marques-Silva, J., Ignatiev, A.: Delivering trustworthy AI through formal XAI. In: Thirty-Sixth AAAI Conference on Artificial Intelligenc, pp. 12342–12350. AAAI Press (2022)
26. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). Recommendation, W3C (December 2012). http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
27. Nardi, D., Brachman, R.J., et al.: An introduction to description logics. Descrip. Logic Handbook **1**, 40 (2003)
28. Ngomo, A.C.N., Moussallem, D., Bühmann, L.: A Holistic Natural Language Generation Framework for the Semantic Web. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, pp. 8. ACL (Association for Computational Linguistics) (2019)
29. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2015)
30. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
31. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Tree-based models for inductive classification on the web of data. J. Web Semantics **45**, 1–22 (2017). https://doi.org/10.1016/j.websem.2017.05.001, https://www.sciencedirect.com/science/article/pii/S1570826817300173

32. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Mach. Intell. **1**(5), 206–215 (2019)
33. Team, Q.: Qwen2.5: A party of foundation models (September 2024). https://qwenlm.github.io/blog/qwen2.5/
34. Touvron, H., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
35. Yang, A., et al.: Qwen2 technical report. arXiv preprint arXiv:2407.10671 (2024)