Learning Concept Lengths Accelerates Concept Learning in ALC

N'Dah Jean Kouagou(✉), Stefan Heindorf, Caglar Demir, and Axel-Cyrille Ngonga Ngomo

Paderborn University, Paderborn, Germany
nkouagou@mail.uni-paderborn.de, heindorf@uni-paderborn.de, {caglar.demir,axel.ngonga}@upb.de

Abstract. Concept learning approaches based on refinement operators explore partially ordered solution spaces to compute concepts, which are used as binary classification models for individuals. However, the number of concepts explored by these approaches can grow to the millions for complex learning problems. This often leads to impractical runtimes. We propose to alleviate this problem by predicting the length of target concepts before the exploration of the solution space. By these means, we can prune the search space during concept learning. To achieve this goal, we compare four neural architectures and evaluate them on four benchmarks. Our evaluation results suggest that recurrent neural network architectures perform best at concept length prediction with a macro F-measure ranging from 38% to 92%. We then extend the CELOE algorithm, which learns ALC concepts, with our concept length predictor. Our extension yields the algorithm CLIP. In our experiments, CLIP is at least 7.5× faster than other state-of-the-art concept learning algorithms for ALC—including CELOE—and achieves significant improvements in the F-measure of the concepts learned on 3 out of 4 datasets. For reproducibility, we provide our implementation in the public GitHub repository at https://github.com/dice-group/LearnALCLengths.

Keywords: Concept learning · Concept length · Structured machine learning · Description logic · Learning from examples · Prediction of concept lengths

1 Introduction

Knowledge bases have recently become indispensable in a number of applications driven by machine learning [12]. For instance, the Gene Ontology (GO) [1,9], Drug-Bank [36], and the Global Network of Biomedical Relationships (GNBR) [27] are actively being used to find treatments for certain diseases [17,25]. We consider the supervised machine learning task of concept learning[1] [24] on knowledge bases in the description logic (DL) $\mathcal{ALC}$ (attributive language with complements) [32]. We focus on approaches based on refinement operators [3,13,24,28,29].

Recent works on concept learning over DLs [5,14,28] indicate that approaches based on refinement operators often fail to achieve practical runtimes on large real-world knowledge bases, which often contain millions of individuals and concepts with

[1] Also called class expression learning (CEL) [13]. See Sect. 2 for a formal definition.

billions of assertions. As noted by Rizzo et al. [28], this is partially due to the size of the search space that needs to be explored to detect relevant concepts. In this paper, we *accelerate concept learning by predicting the length of the target concept* in advance. By these means, we can prune the search space traversed by a refinement operator and therewith reduce the overall runtime of the concept learning process. To quantify our runtime improvement, we compare our new algorithm—dubbed CLIP—against the state-of-the-art approaches CELOE [23], OCEL [24], and ELTL [7]. The price for our runtime improvement is paid in the prior training of the concept length predictor. Therefore, we also show that the prediction of concept lengths can be carried out using rather simple neural architectures.

To the best of our knowledge, no similar work has been carried out before. Hence, we hope that our findings will serve as a foundation for more investigations in this direction. In a nutshell, our contributions are as follows:

1. We design different neural network architectures for learning concept lengths.
2. We implement a length-based refinement operator to generate training data.
3. We integrate our concept length predictors into the CELOE algorithm, resulting in a new algorithm that we call CLIP. We show that CLIP achieves state-of-the-art performance in terms of F-measure while outperforming the state of the art in terms of runtime.

The remainder of the paper is organized as follows: In Sect. 2, we give a brief overview of the required background in DL, concept learning, knowledge graph embeddings, and refinement operators for DLs. We also present the notation and terminology used in the rest of the paper. Section 3 presents related work on concept learning using refinement operators. In Sects. 4 and 5, we describe our new approach for concept learning in $\mathcal{ALC}$. Our results on different knowledge bases are presented in Sect. 6. Section 7 draws conclusions from our findings and introduces new directions for future work.

2 Background

In this section, we present the background on description logics, concept learning, refinement operators, and knowledge graph embeddings. We also introduce the notation and terminology used throughout the paper.

Description Logics. Description logics [2] are a family of languages for knowledge representation. While there are more powerful variants of description logics [2,20], we focus on the description logic $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplement) because it is the simplest closed description logic with respect to propositional logics. Its basic components are *concept* names (e.g., $Teacher$, $Human$), *role* names (e.g., $hasChild$, $bornIn$) and *individuals* (e.g., $Mike$, $Jack$). Table 1 introduces the syntax and semantics of $\mathcal{ALC}$ (see [24] for more details). In $\mathcal{ALC}$, concept lengths are defined recursively [24]

1. $length(A) = length(\top) = length(\bot) = 1$, for all atomic concepts A
2. $length(\neg C) = 1 + length(C)$, for all concepts C
3. $length(\exists\, r.C) = length(\forall\, r.C) = 2 + length(C)$, for all concepts C
4. $length(C \sqcup D) = length(C \sqcap D) = 1 + length(C) + length(D)$, for all concepts C and D.

Table 1. $\mathcal{ALC}$ syntax and semantics. $\mathcal{I}$ stands for an interpretation, $\Delta^{\mathcal{I}}$ for its domain.

Construct	**Syntax**	**Semantics**
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Existential restriction	$\exists\, r.C$	$\{a^{\mathcal{I}} / \exists\, b^{\mathcal{I}} \in C^{\mathcal{I}}, (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\}$
Universal restriction	$\forall\, r.C$	$\{a^{\mathcal{I}} / \forall\, b^{\mathcal{I}}, (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}} \Rightarrow b^{\mathcal{I}} \in C^{\mathcal{I}}\}$

The pair $\mathcal{K} = (TBox, ABox)$ denotes an $\mathcal{ALC}$ knowledge base. The $TBox$ contains statements of the form $C \sqsubseteq D$ or $C \equiv D$, where C and D are concepts. The $ABox$ consists of statements of the form $C(a)$ and $R(a, b)$, where C is a concept, R is a role, and a, b are individuals. N_C and N_R are the sets of concept names and role names in $\mathcal{K}$, respectively. $\mathcal{K}_I$ stands for the set of all individuals in $\mathcal{K}$. $|.|$ denotes the cardinality function, that is, a function that takes a set as input and returns the number of elements in the set. Given a concept C, we denote the set of all instances of C by $C_{\mathcal{P}}$. $C_{\mathcal{N}}$ stands for the set of all individuals that are not instances of C.

Concept Learning. We recall the definition introduced by [24].

Definition 1. *Let $\mathcal{K}$, T, P, and N be a knowledge base, a target concept, and sets of positive and negative examples from $\mathcal{K}_I$, respectively. The learning problem is to find a concept C such that T does not occur in C, and for $\mathcal{K}' = \mathcal{K} \cup \{T \equiv C\}$, we have that $\mathcal{K}' \models P$ and $\mathcal{K}' \not\models N$.*

Since such a concept C does not always exist, we target an approximate definition in this work.

Definition 2. *Given a knowledge base $\mathcal{K}$, a set of positive examples P, and a set of negative examples N, the learning problem is to find a concept C which maximizes the F-measure F, where F is defined by $F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$, with $Precision = \frac{|C_{\mathcal{P}} \cap P|}{|C_{\mathcal{P}} \cap P| + |C_{\mathcal{P}} \cap N|}$ and $Recall = \frac{|C_{\mathcal{P}} \cap P|}{|C_{\mathcal{P}} \cap P| + |C_{\mathcal{N}} \cap P|}$.*

Note that in the above definition, $C_{\mathcal{N}}$ and $C_{\mathcal{P}}$ depend on both the concept C and the learning problem. Following [24], we use the closed-world assumption (CWA) to compute $C_{\mathcal{P}}$ and $C_{\mathcal{N}}$: every individual that cannot be inferred to be an element of $C_{\mathcal{P}}$ is considered to be in $C_{\mathcal{N}}$. In this work, we are interested in finding such a concept C by using refinement operators (see the following subsection). The found concept C might not be unique for a learning problem.

Refinement Operators.

Definition 3. *A quasi-ordering $\preceq$ is a reflexive and transitive binary relation. Let $(\mathcal{S}, \preceq)$ be a quasi-ordered space. A downward (upward) refinement operator on $\mathcal{S}$ is a mapping $\rho : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ such that for all $C \in \mathcal{S}$, $C' \in \rho(C)$ implies $C' \preceq C$ $(C \preceq C')$.*

Example 1. Let $\mathcal{K} = (TBox, ABox)$ be a knowledge base, with

$$TBox = \{Female \sqsubseteq Human, Mother \sqsubseteq Female, Human \sqsubseteq \neg Car\};$$
$$ABox = \{Female(Anna), Mother(Kate), Car(Venza), hasChild(Jack, Paul)\}.$$

Assume the sets of concept names N_C and role names N_R in $\mathcal{K}$ are given by:

$$N_C = \{Car, Female, Human, Mother, Parent\};$$
$$N_R = \{hasChild, manufacturedBy, marriedTo\}.$$

Let $\mathbf{C}$ be the set of all $\mathcal{ALC}$ concept expressions [30] that can be constructed from N_C and N_R (note that $\mathbf{C}$ is infinite and every concept name is a concept expression). Consider the mapping $\rho : \mathbf{C} \rightarrow 2^{\mathbf{C}}$ defined by: $\rho(C) = \{C' \in \mathbf{C} | C' \sqsubseteq C, C' \neq C\}$ for all $C \in \mathbf{C}$. ρ is clearly a downward refinement operator and we have for example,

– $\{Female, Mother, Female \sqcup Mother\} \subseteq \rho(Human)$;
– $\{\exists\ marriedTo.Mother, \forall\ marriedTo.Female\} \subseteq \rho(\exists\ marriedTo.(\neg Car))$.

Refinement operators can have a number of important properties which we do not discuss in this paper (for further details, we refer the reader to [24]). In the context of concept learning, these properties can be exploited to optimize the traversal of the concept space in search of a target concept.

Knowledge Graph Embeddings. The $ABox$ of a knowledge base in $\mathcal{ALC}$ can be regarded as a knowledge graph [16] (see also Sect. 4). A knowledge graph embedding function typically maps a knowledge graph to a continuous vector space to facilitate downstream tasks such as link prediction and knowledge graph completion [10,33]. We exploit knowledge graph embeddings to improve concept learning in DLs. Knowledge graph embedding approaches can be subdivided into two categories: the first category of approaches uses only facts in the knowledge graph [6,26,35], and the second category of approaches takes into account additional information about entities and relations, such as textual descriptions [34,37]. Both approaches typically initialize each entity and relation with a random vector, matrix, or tensor. Then, a scoring function is defined to learn embeddings so that facts observed in the knowledge graph receive high scores, while unobserved facts receive low scores. It can also happen that unobserved facts receive a high score, for instance, if a fact is supposed to hold but it is not observed in the knowledge graph, or if it is a logical implication of the learned patterns. For more details, we refer the reader to the surveys [10,33]. In this work, we use the Convolutional Complex Embedding Model (ConEx) [11], which has been shown to produce state-of-the-art results with fewer trainable parameters.

3 Related Work

Lehmann and Hitzler [24] investigated concept learning using refinement operators by studying combinations of possible refinement operator properties and designed their own refinement operator in $\mathcal{ALC}$. Their approach proved to be competitive in accuracy with (and in some cases, superior to) the state-of-the-art, namely inductive logic programs. Badea and Nienhuys-Cheng [3] worked on a similar topic in the DL $\mathcal{ALER}$. The evaluation of their approach on real ontologies from different domains showed promising results, but it had the disadvantage of depending on the instance data.

DL-Learner [21] is the most mature framework for concept learning. CELOE [23], OCEL [22], and ELTL [7] are algorithms implemented in DL-Learner. CELOE is an extension of OCEL that uses the same refinement operator but with a different heuristic function. It is considered the best algorithm in DL-Learner to date. The algorithm uses a soft syntactic bias in its heuristic function that balances between predictive performance and short, readable concepts. In this work, we opt for a hard syntactic bias that constrains our algorithm to generate expressions shorter than a given threshold. ELTL is designed for the simple description logic $\mathcal{EL}$. Despite its usefulness, DL-Learner suffers from performance issues in certain scenarios [cf. 14,31].

DL-FOIL [13,29] is another concept learning algorithm that uses refinement operators and progressively constructs the solution as a disjunction of partial descriptions. Each partial description covers a part of the positive examples and rules out as many negative/uncertain-membership examples as possible. DL-FOCL1–3 [28] are variants of DL-FOIL that employ meta-heuristics to help reduce the search space. The first release of DL-FOCL, also known as DL-FOCL1, is essentially based on omission rates: to check if further iterations are required, DL-FOCL1 compares the score of the current concept definition with that of the best concept obtained at that stage. DL-FOCL2 employs a look-ahead strategy by assessing the quality of the next possible refinements of the current partial description. Finally, DL-FOCL3 attempts to solve the myopia problem in DL-FOIL by introducing a local memory, used to avoid reconsidering suboptimal choices previously made. EvoLearner [14] is a concept learner based on evolutionary algorithms. In contrast, we propose to learn *concept lengths* from positive and negative examples to boost the performance of *concept learners*.

4 Concept Length Prediction

In this section, we address the following learning problem: Given a knowledge base $\mathcal{K}$, a set of positive examples P and negative examples N, predict the length of the shortest concept C that is a solution to the learning problem defined by $\mathcal{K}$, P, and N according to Definition 2. To achieve this goal, we devise a generator that creates training data for our prediction algorithm based solely on $\mathcal{K}$ and a user-given number of learning problems to use at training time.

4.1 Training Data for Length Prediction

Data Generation. Given a knowledge base, the construction of training data (concepts with their positive and negative examples) is carried out as follows:

1. Generate concepts of various lengths using the length-based refinement operator described in Algorithm 1 and 2. In this process, short concepts are preferred over long concepts, i.e., when two concepts have the same set of instances, the longest concept is left out.
2. Compute the sets $C_{\mathcal{N}}$ and $C_{\mathcal{P}}$ for each generated concept C.
3. Define a hyper-parameter $\mathbf{N} \in [1, |\mathcal{K}_I|]$ that represents the total number of positive and negative examples we want to use per learning problem.
4. Sample positive and negative examples as follows:
   - If $|C_{\mathcal{P}}| \geq \frac{\mathbf{N}}{2}$ and $|C_{\mathcal{N}}| \geq \frac{\mathbf{N}}{2}$, then we randomly sample $\frac{\mathbf{N}}{2}$ individuals from each of the two sets $C_{\mathcal{P}}$ and $C_{\mathcal{N}}$.
   - Otherwise, we take all individuals in the minority set and sample the remaining number of individuals from the other set.

Training Data Features. A knowledge graph is commonly defined as $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ is a set of entities and $\mathcal{R}$ is a set of relations. We convert a given knowledge base $\mathcal{K}$ into a knowledge graph by converting $ABox$ statements of the form $R(a, b)$ into (a, R, b). Statements of the form $C(a)$ are converted into $(a, \texttt{rdf:type}, C)$. In our experimental data, the $TBox$es contained only subsumptions $C \sqsubseteq D$ between atomic concepts C and D, which were converted into triples $(C, \texttt{rdfs:subClassOf}, D)$. Hence, in our experiments, $\mathcal{E} \subseteq N_C \cup \mathcal{K}_I$ and $\mathcal{R} = N_R \cup \{\texttt{rdfs:subClassOf}\}$.

The resulting knowledge graph is then embedded into a continuous vector space to serve for the prediction of concept lengths. On the vector representation of entities, we create an extra dimension at the end of the entries, where we insert $+1$ for positive examples and -1 for negative examples. Formally, we define an injective function f_C for each target concept C

$$\begin{aligned} &f_C : \mathbb{R}^{\mathbf{d}} \longrightarrow \mathbb{R}^{\mathbf{d}+1} \\ &\mathbf{x} = (x_1, \ldots, x_{\mathbf{d}}) \longmapsto \begin{cases} (x_1, \ldots, x_{\mathbf{d}}, 1) & \text{if } ent(\mathbf{x}) \in C_{\mathcal{P}}, \\ (x_1, \ldots, x_{\mathbf{d}}, -1) & \text{otherwise,} \end{cases} \end{aligned} \tag{1}$$

where $\mathbf{d}$ is the dimension of the embedding space, and $ent(\mathbf{x})$ is the entity whose embedding is $\mathbf{x}$. Thus, a data point in the training, validation, and test datasets is a tuple $(M_C, length(C))$, where M_C is a matrix of shape $\mathbf{N} \times (\mathbf{d} + 1)$ constructed by concatenating the embeddings of positive examples followed by those of negative examples. Formally, assume n_1 and n_2 are the numbers of positive and negative examples for C, respectively. Further, assume that the embedding vectors of positive examples are $x^{(i)},\ i = 1 \ldots, n_1$ and those of negative examples are $x^{(i)},\ i = n_1 + 1, \ldots, n_1 + n_2 = \mathbf{N}$. Then, the $i - th$ row of M_C is given by

$$M_C[i, :] = \begin{cases} (x_1^{(i)}, \ldots, x_{\mathbf{d}}^{(i)}, 1) & \text{if } 1 \leq i \leq n_1, \\ (x_1^{(i)}, \ldots, x_{\mathbf{d}}^{(i)}, -1) & \text{if } n_1 + 1 \leq i \leq n_1 + n_2 = \mathbf{N}. \end{cases} \tag{2}$$

We view the prediction of concept lengths as a classification problem with classes $0, 1, \ldots, L$, where L is the length of the longest concept in the training dataset. As shown in Table 2, the concept length distribution can be imbalanced. To prevent concept

Table 2. Number of concepts per length in the training, validation, and test datasets for the four knowledge bases considered

Length	**Carcinogenesis**			**Mutagenesis**			**Semantic Bible**			**Vicodi**		
	Train	**Val.**	**Test**	**Train**	**Val.**	**Test**	**Train**	**Val.**	**Test**	**Train**	**Val.**	**Test**
3	3,647	405	1,013	1,038	115	288	487	54	135	3,952	439	1,098
5	782	87	217	1,156	129	321	546	61	152	2,498	278	694
6	0	0	0	0	0	0	162	18	45	335	37	93
7	1,143	127	318	1,310	146	364	104	12	29	3,597	400	999
8	0	0	0	0	0	0	0	0	0	747	83	207
9	0	0	0	0	0	0	73	8	21	0	0	0
11	0	0	0	0	0	0	41	5	11	0	0	0

length predictors from overfitting on the majority classes, we used the weighted cross-entropy loss

$$\mathcal{L}_w(\bar{y}, y) = -\frac{1}{bs}\sum_{i=1}^{bs}\sum_{k=1}^{L} w_k \mathbb{1}(k, y^i)\log(\bar{y}_k^i), \tag{3}$$

where bs is the batch size, $\bar{y}$ is the batch matrix of predicted probabilities (or scores), y is the batch vector of targets, w is a weight vector, and $\mathbb{1}$ is the indicator function. The weight vector is defined by: $w_k = 1/\sqrt{[k]}$, where $[k]$ is the number of concepts of length k in the training dataset. Table 2 provides details on the training, validation, and test datasets for each of the four knowledge bases. Though the maximal length for the generation of concepts was fixed to 15, many long concepts were equivalent to shorter concepts. As a result, they were removed from the training dataset and the longest remaining are of length 11 (see Table 2).

4.2 Concept Length Predictors

We consider four neural network architectures: Long Short-Term Memory (LSTM) [15], Gated Recurrent Unit (GRU) [8], Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN). Recurrent neural networks (LSTM, GRU) take as input a sequence of embeddings of the positive and negative examples (all positive examples followed by all negative examples). The CNN model takes the same input as recurrent networks and views it as an image with a single channel. In contrast, the MLP model inputs the average embeddings of a set of positive and negative examples. The implementation details and the hyper-parameter setting for each of the networks are given in Sect. 6.1.

5 Concept Learner with Integrated Length Prediction (CLIP)

The intuition behind CLIP is that *if we have a reliable concept length predictor, then our concept learner only needs to test concepts of length up to the predicted length.* Figure 1 illustrates CLIP's exploration strategy.

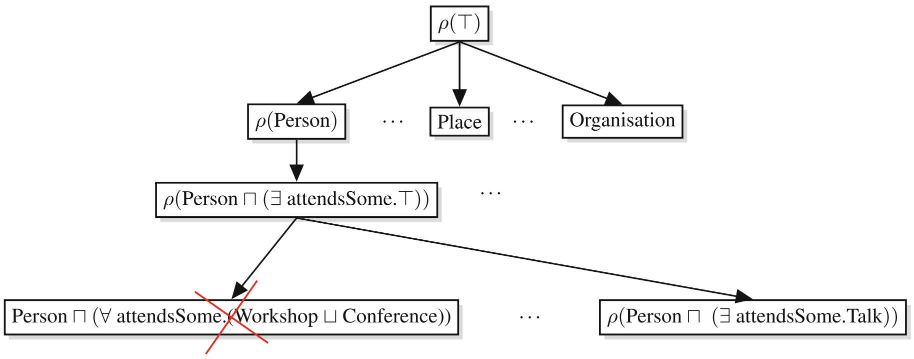

$\rho(\top)$
ρ(Person)
Place
Organisation
ρ(Person $\sqcap$ ($\exists$ attendsSome.$\top$))
Person $\sqcap$ ($\forall$ attendsSome.(Workshop $\sqcup$ Conference))
ρ(Person $\sqcap$ ($\exists$ attendsSome.Talk))

Fig. 1. CLIP search tree when the predicted length is 5. After each refinement, CLIP discards all concepts whose length is larger than the set threshold.

Refinements that exceed the predicted length are ignored during the search. In the figure, the concept `Person` $\sqcap$ ($\forall$ `attendsSome.(Workshop` $\sqcup$ `Conference))` is of length 7 and is therefore neither tested nor added to the search tree.

Remark 1. For concept length prediction during concept learning, we sample n_1 positive examples and n_2 negative examples from the considered learning problem such that $n_1 + n_2 = \mathbf{N}$, as described in Sect. 4.2 (4).

We implemented the intuition behind CLIP by extending CELOE's refinement operator. Our refinement operator differs from CELOE's in how it refines atomic concepts (see Algorithms 1 and 2). For example, it considers all refinements $A' \sqsubset A$ of an atomic concept A whereas CELOE's refinement operator only considers $A' \sqsubset A$ such that there is no A'' with $A' \sqsubset A'' \sqsubset A$. Omitting this expensive check allows more concepts to be tested in the same amount of time. In the following, we describe our method for refining atomic concepts and refer the reader to [23,24] for details on CELOE. In Algorithms 1 and 2, the hyper-parameters max_length, k, and $construct\_frac$ control the refinement operator: max_length specifies how long the refinements can become (Algorithm 2, lines 8, 11, 13); k controls the number of fillers sampled without replacement for universal and existential restrictions (Algorithm 1, line 7); $construct\_frac \in (0, 1]$ specifies the fraction of constructs to be sampled (Algorithm 2, lines 1–3).

Given a knowledge base $\mathcal{K}$, the refinement of an atomic concept A is carried out as follows: (1) obtain the subconcepts $subs$ of A in $\mathcal{K}$; (2) compute the negations neg_subs of all subconcepts of A; (3) construct existential and universal role $restrictions$ where the fillers are in the set made of $\top, \bot, A$, and sample k elements from each of the sets $subs$ and neg_subs; (4) obtain the union $constructs$ of $subs$, neg_subs, and $restrictions$, and finally (5) Algorithm 2 returns the refinements as intersections or unions of the $subs$ and $constructs$ computed before, with the generated refinements having length at most max_length. The refinement operator is designed to yield numerous meaningful downward refinements from a single atomic concept.

Algorithm 1. Function REFINEHELPER

Input: Knowledge base $\mathcal{K}$, atomic concept A
Hyper-parameters: Number of subconcepts to be sampled: k, default 5
Output: Subconcepts, negated subconcepts and restrictions of A

```
 1: subs ← SUBCONCEPTS$_{\mathcal{K}}(A)$    # Subconcepts of A in $\mathcal{K}$
 2: neg_subs ← $\{\neg C | C \in$ subs$\}$
 3: restrictions ← $\{\}$
 4: if |subs| $< k$ then
 5:    fillers ← $\{\top, \bot, A\}$
 6: else
 7:    fillers ← $\{\top, \bot, A\} \cup$ RANDSAMPLE(subs, $n = k$) $\cup$ RANDSAMPLE(neg_subs, $n = k$)
 8: end if
 9: for $C$ in fillers do
10:    for $R$ in role names of $\mathcal{K}$ do
11:       restrictions ← restrictions $\cup$ $\{\exists\ R.C\}$
12:       restrictions ← restrictions $\cup$ $\{\forall\ R.C\}$
13:    end for
14: end for
15: constructs ← subs $\cup$ neg_subs $\cup$ restrictions
16: return constructs
```

Algorithm 2. Function REFINEATOMICCONCEPT

Input: Knowledge base $\mathcal{K}$, atomic concept A
Hyper-parameters: Fraction of constructs to be sampled: *construct_frac* $\in (0, 1]$, default 0.8;
max concept length to be generated: *max_length*, default 15
Output: Set of concepts $\{C_1, \ldots, C_n\}$ which are refinements of A

```
 1: constructs = REFINEHELPER($\mathcal{K}$, $A$)
 2: $m$ ← $\lfloor construct\_frac \times$ SIZEOF(constructs)$\rfloor$    # Integer part function
 3: constructs ← RANDSAMPLE(constructs, $n = m$)    # Sample without replacement
 4: subs ← SUBCONCEPTS$_{\mathcal{K}}(A)$    # Subconcepts of A in $\mathcal{K}$
 5: result ← subs    # All subconcepts are of length 1
 6: for $S_1$ in subs do
 7:    for $S_2$ in constructs do
 8:       if $S_1 \neq S_2$ and $length(S_1 \sqcap S_2) \leq max\_length$ then
 9:          result ← result $\cup$ $\{S_1 \sqcap S_2\}$
10:       end if
11:       if $S_1 \neq S_2$ and $S_2 \in$ subs and $length(S_1 \sqcup S_2) \leq max\_length$ then
12:          result ← result $\cup$ $\{S_1 \sqcup S_2\}$
13:       else if $S_1 \neq S_2$ and $length((S_1 \sqcup S_2) \sqcap A) \leq max\_length$ then
14:          result ← result $\cup$ $\{(S_1 \sqcup S_2) \sqcap A\}$  # All refinements are downward refinements
15:       end if
16:    end for
17: end for
18: return result
```

Table 3. Overview of benchmark datasets

Dataset	**|Individuals|**	**|At. Concepts|**	**|Obj. Prop.|**	**|Data Prop.|**	$\lvert TBox \rvert$	$\lvert ABox \rvert$
Carcinogenesis	22,372	142	4	15	138	96,757
Mutagenesis	14,145	86	5	6	82	61,965
Semantic Bible	724	48	29	9	51	3,211
Vicodi	33,238	194	10	2	193	149,634

6 Evaluation

Datasets. We used four datasets for our experiments: Carcinogenesis, Mutagenesis, Semantic Bible, and Vicodi. All datasets are available in our GitHub repository and described in Table 3. We conducted two sets of experiments. First, we wanted to know which neural architecture performs best at predicting concept lengths. Second, we assessed CLIP's performance w.r.t. its runtime and F-measure when compared with the state-of-the-art refinement approaches dubbed CELOE, OCEL, ELTL, and DL-Foil.

Hardware. The training of our concept length learners was carried out on a single 11 GB memory NVIDIA K80 GPU with 4 Intel Xeon E5-2670 CPUs at 2.60 GHz, and 24 GB RAM. During concept learning with CELOE, OCEL, ELTL, and CLIP, we used an 8-core Intel Xeon E5-2695 at 2.30 GHz, and 16 GB RAM to ensure a fair comparison.

6.1 Concept Length Prediction

Hyper-parameter Optimization. In our preliminary experiments on all four knowledge bases, we used a random search [4] to select fitting hyper-parameters (as summarized in Table 4). Our experiments suggest that choosing two layers for the recurrent neural networks (LSTM, GRU) is the best choice in terms of computation cost and classification accuracy. In addition, two linear layers, batch normalization, and dropout layers are used to increase the performance. The CNN model consists of two convolution layers, two linear layers, two dropout layers, and a batch normalization layer. Finally, we chose 4 layers for the MLP model with batch normalization and dropout layers. The Rectified Linear Unit (ReLU) is used in the intermediate layers of all models, whereas the sigmoid function is used in the output layers.

We ran the experiments in a 10-fold cross-validation setting with ten repetitions. Table 4 gives an overview of the hyper-parameter settings on each of the four knowledge bases considered. The number of epochs was set based on the training speed and the performance of the validation dataset. For example, on the Carcinogenesis knowledge base, most length predictors are able to reach 90% accuracy with just 50 epochs, which suggests that more epochs would probably lead to overfitting. Adam optimizer [18] is used to train the length predictors. We varied the number of examples $\mathbf{N}$ between 200 and 1000, and the embedding dimension $\mathbf{d}$ from 10 to 100, but we finally chose $\mathbf{N} = \min(1000, \frac{|\mathcal{K}_{\mathcal{I}}|}{2})$ and $\mathbf{d} = 40$ as best values for both classification accuracy and computation cost on the four datasets considered.

Table 4. Hyper-parameter setting

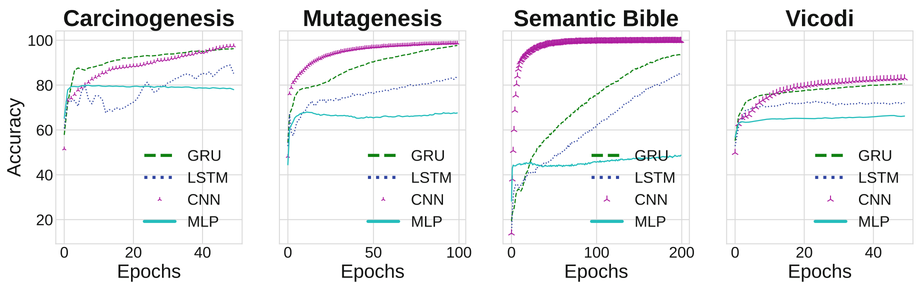
Dataset	|Epochs|	lr	d	Batch Size	N
Carcinogenesis	50	0.003	40	512	1,000
Mutagenesis	100	0.003	40	512	1,000
Semantic Bible	200	0.003	40	256	362
Vicodi	50	0.003	40	512	1,000

Table 5. Model size and training time

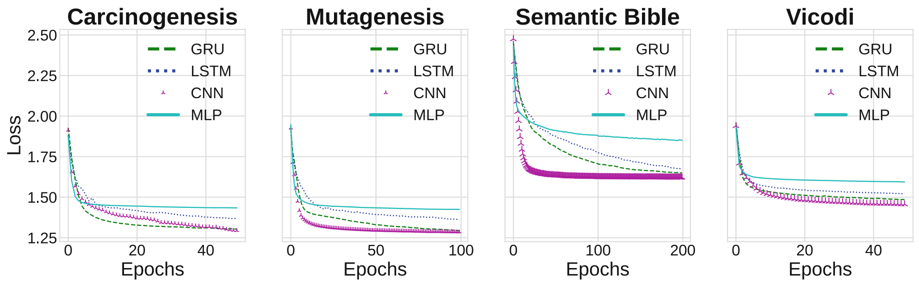
Model	Carcinogenesis		Mutagenesis	
	|Parameters|	Train. Time (s)	|Parameters|	Train. Time (s)
LSTM	160,208	188.42	160,208	228.13
GRU	125,708	191.16	125,708	228.68
CNN	838,968	16.77	838,248	44.74
MLP	61,681	10.04	61,681	14.29
Model	**Semantic Bible**		**Vicodi**	
	|Parameters|	**Train. Time (s)**	**|Parameters|**	**Train. Time (s)**
LSTM	161,012	196.20	160,409	362.28
GRU	125,512	197.86	125,909	367.55
CNN	96,684	18.43	839,377	71.95
MLP	61,933	9.56	61,744	24.61

Results. Table 5 shows the number of parameters and training time of LSTM, GRU, CNN, and MLP architectures on each of the datasets. From the table, we can observe that our concept length predictors can be trained in less than an hour and be used for efficient concept learning on corresponding knowledge bases.

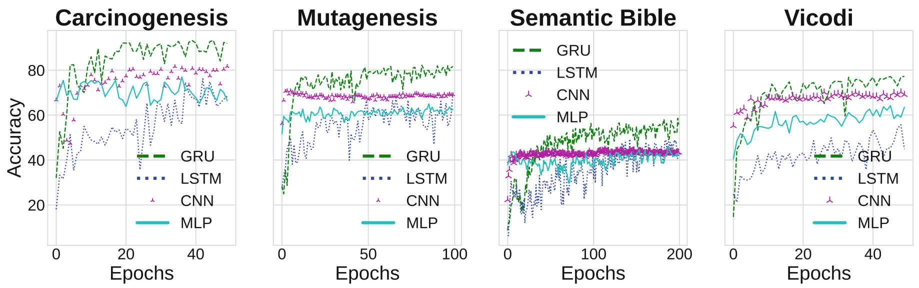
In Fig. 2, we show the training curves for each model on all datasets. We can observe a decreasing loss on all knowledge bases (see Fig. 2b), which suggests that the models were able to learn. Moreover, the Gated Recurrent Unit (GRU) model outperforms the other models on all datasets, see Fig. 2c and Table 6. The input to the MLP model is the average of the embeddings of the positive and negative examples for a concept. This may have caused loss of information in the inputs. As shown in Fig. 2a, MLP curves tend to saturate in the early stages of training. We also assessed the element-wise multiplication of the embeddings and obtained similar results. However, as reflected in Table 6, all our proposed architectures outperform a random model that knows the distribution of the lengths of concepts in the training dataset. A modified version of MLP where the embedding of each example is processed independently before averaging the final output (no interaction) yielded even poorer results. This suggests that the full interaction between examples is the main factor in the increased performance of recurrent neural networks.



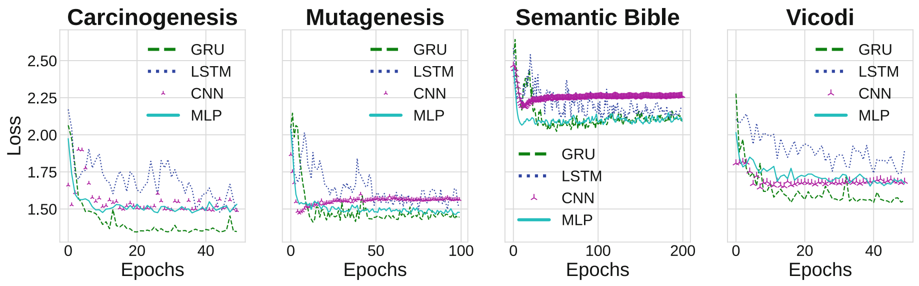
(a) Cross-validation accuracy on the training dataset



(b) Cross-validation loss on the training dataset



(c) Cross-validation accuracy on the validation dataset



(d) Cross-validation loss on the validation dataset

Fig. 2. Training and validation curves

Table 6. Effectiveness of concept length prediction. RM is a random model that makes predictions according to the length distribution in the training dataset, and F1 is the macro F-measure.

Metric	Carcinogenesis					Mutagenesis				
	LSTM	**GRU**	**CNN**	**MLP**	**RM**	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**
Train. Acc	0.89	0.96	0.97	0.80	0.48	0.83	0.97	0.98	0.68	0.33
Val. Acc.	0.76	0.93	0.82	0.77	0.48	0.70	0.82	0.71	0.65	0.35
Test Acc.	0.92	**0.95**	0.84	0.80	0.49	0.78	**0.85**	0.70	0.68	0.33
Test F1	0.88	**0.92**	0.71	0.59	0.33	0.76	**0.85**	0.70	0.67	0.32
Metric	**Semantic Bible**					**Vicodi**				
	LSTM	**GRU**	**CNN**	**MLP**	**RM**	**LSTM**	**GRU**	**CNN**	**MLP**	**RM**
Train. Acc	0.85	0.93	0.99	0.68	0.33	0.73	0.81	0.83	0.66	0.28
Val. Acc.	0.49	0.58	0.44	0.46	0.26	0.55	0.77	0.70	0.64	0.30
Test Acc.	0.52	**0.53**	0.37	0.40	0.25	0.66	**0.80**	0.69	0.66	0.29
Test F1	0.27	**0.38**	0.20	0.22	0.16	0.45	**0.50**	0.45	0.38	0.20

Table 6 compares our chosen neural network architectures and a random model on the Carcinogenesis, Vicodi, Mutagenesis, and Semantic Bible knowledge bases. From the table, it appears that recurrent neural network models (GRU, LSTM) outperform the other two models (CNN and MLP) on three out of four datasets, with the only exception that the LSTM model slightly dropped in performance on Vicodi compared to CNN.

While the CNN model tends to overfit on all knowledge bases, the MLP model is unable to extract meaningful information from the average embeddings. On the Semantic Bible knowledge base, which appears to be the smallest dataset, all our proposed networks performed less well than expected. This suggests that our learning approach is more suitable for large knowledge bases. Nonetheless, all our proposed models are clearly better than a distribution-aware random model with a minimum performance (macro F1 score) difference on average between 21.25% (MLP) and 41% (GRU).

6.2 Concept Learning

Experimental Settings. The maximal runtime is set to 2 min per learning problem.[2] For all knowledge bases, we generate 100 random learning problems by (1) creating random $\mathcal{ALC}$ concepts C of maximal length 15, (2) computing the sets of instances $C_{\mathcal{P}}$ and $C_{\mathcal{N}}$, (3) providing $C_{\mathcal{P}}$ and $C_{\mathcal{N}}$ to each of the approaches, and (4) measuring the accuracy, the F-measure, the runtime, and the length of the best solution generated within the set timeout. We ran all approaches on the same hardware (see Sect. 6). CLIP was configured to use our best concept length predictor (GRU). Note that a predictor is trained for each dataset (see Table 2). Also note that we add the ELTL algorithm—a

[2] The implementations of OCEL and ELTL in the DL-Learner framework, which we used for our experiments, fail to consider the set threshold accurately. Hence, Table 7 contains values larger than 2 min for these two algorithms.

Table 7. Performance of CLIP compared with CELOE, OCEL, and ELTL on 100 learning problems per knowledge base. The presence of an asterisk indicates that the performance difference is significant between CLIP and the best between CELOE and OCEL. The upward arrow (↑) indicates that the higher is better, whereas the downward arrow (↓) indicates the opposite. All results are average results per knowledge base. The average time is in minutes. ELTL is shown in gray since it learns concepts in $\mathcal{EL}$ instead of $\mathcal{ALC}$ as the others do.

Metric	**Carcinogenesis**			
	CELOE	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.78 ± 0.27	0.89 ± 0.31	0.58 ± 0.46	$\mathbf{0.99} \pm 0.00$
F1 ↑	0.62 ± 0.46	–	0.51 ± 0.47	$\mathbf{0.96}* \pm 0.10$
Runtime (min) ↓	0.93 ± 0.94	3.01 ± 0.72	0.75 ± 0.07	$\mathbf{0.10}* \pm 0.09$
Length ↓	$\mathbf{1.69} \pm 0.89$	7.81 ± 6.88	1.04 ± 0.39	2.00 ± 1.28

Metric	**Mutagenesis**			
Metric	**CELOE**	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.99 ± 0.00	0.71 ± 0.45	0.37 ± 0.43	$\mathbf{0.99} \pm 0.00$
F1 ↑	0.81 ± 0.35	–	0.29 ± 0.40	$\mathbf{0.93}* \pm 0.18$
Runtime (min) ↓	0.70 ± 0.77	2.39 ± 0.18	0.29 ± 0.16	$\mathbf{0.07}* \pm 0.05$
Length ↓	2.79 ± 1.17	12.63 ± 7.03	1.10 ± 0.81	$\mathbf{2.20} \pm 1.16$

Metric	**Semantic Bible**			
	CELOE	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.99 ± 0.02	0.66 ± 0.47	0.59 ± 0.37	$\mathbf{0.99} \pm 0.00$
F1 ↑	0.97 ± 0.10	–	0.57 ± 0.38	$\mathbf{0.98} \pm 0.05$
Runtime (min) ↓	0.47 ± 0.80	22.15 ± 96.55	0.09 ± 0.07	$\mathbf{0.06}* \pm 0.05$
Length ↓	3.85 ± 2.44	9.54 ± 5.73	1.38 ± 1.76	$\mathbf{2.52}* \pm 1.26$

Metric	**Vicodi**			
	CELOE	**OCEL**	**ELTL**	**CLIP**
Acc. ↑	0.29 ± 0.44	0.25 ± 0.43	0.28 ± 0.44	$\mathbf{0.99}* \pm 0.00$
F1 ↑	0.25 ± 0.44	–	0.25 ± 0.44	$\mathbf{0.97}* \pm 0.09$
Runtime (min) ↓	1.30 ± 0.71	4.78 ± 1.12	1.81 ± 0.46	$\mathbf{0.16}* \pm 0.12$
Length ↓	10.79 ± 6.30	11.54 ± 6.00	11.14 ± 6.11	$\mathbf{1.68}* \pm 0.98$

concept learner for the DL $\mathcal{EL}$—to investigate whether our randomly generated concepts are equivalent to concepts in a simpler description logic.

Results. Table 7 presents a comparison of the results achieved by CLIP, CELOE, OCEL, and ELTL; results are formatted *mean* ± *standard deviation*. Note that the table does not contain DL-FOIL because it could not solve the learning problems that we considered. For instance, the first learning problem on the Semantic Bible knowledge base targets $\texttt{SonOfGod} \sqcup (\exists\ \texttt{locationOf.StateOrProvince})$. Here, DL-FOIL was stuck on the refinement of `GeographicLocation` with over 5×10^3 unsuccessful trials. Similar scenarios were observed on other datasets. We also tried running DL-FOCL, but it was not possible using the documentation provided.

Our results suggest that CLIP outperforms the other three algorithms in F1 and in runtime on most datasets. The ELTL algorithm appears to be faster than CELOE and OCEL but slower than CLIP. However, its runtime performance stems from the fact that it detects concepts in the DL $\mathcal{EL}$. Since some of our learning problems can only be solved in $\mathcal{ALC}$ or a more expressive DL, ELTL performs poorly in F1 score on all datasets. This result suggests that we do not generate trivial problems.

We used a Wilcoxon Rank Sum test to check whether the difference in performance between CLIP, CELOE, and OCEL was significant. Significant differences are marked with an asterisk. The null hypothesis for our test was as follows: "the two distributions that we compare are the same". The significance level was $\alpha = 0.05$. The performance differences in F1 between CLIP and the other algorithms are significant on 3 out of 4 datasets.[3] With respect to runtimes, we significantly outperform all other algorithms on all datasets. Large time differences correspond to scenarios where CLIP detects short solution concepts while other algorithms explore longer concepts. Low time differences correspond to either simple learning problems, where all algorithms find a solution in a short period of time, or complex learning problems where CLIP explores long concepts as other algorithms.

The average runtimes of CELOE, OCEL, and CLIP across all datasets are 0.85, 8.08, and 0.1 min, respectively. Given that the average training time of the length predictor GRU is 4.10 min, we can conjecture the following: (1) the expected number of learning problems from which CLIP should be preferred over CELOE is 5, and (2) CLIP should be preferred over OCEL for any number of learning problems.

7 Conclusion and Future Work

We investigated the prediction of concept lengths in the description logic $\mathcal{ALC}$, to speed up the concept learning process using refinement operators. To this end, four neural network architectures were evaluated on four benchmark knowledge bases. The evaluation results suggest that all of our proposed models are superior to a random model, with recurrent neural networks performing best at this task. We showed that integrating our concept length predictors into a concept learner can reduce the search space and improve the runtime and the quality (F-measure) of solution concepts.

Even though our proposed learning approach was very efficient when dealing with concepts of length up to 11 (in $\mathcal{ALC}$), its behavior is not guaranteed when longer concepts are considered. Moreover, the use of generic embedding techniques might lead to suboptimal results. In future work, we plan to jointly learn the embeddings of a given knowledge graph and the lengths of its complex (long) concepts. We will also explore further network architectures such as multi-set convolutional networks [38] and neural class expression synthesis [19].

Acknowledgements. This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860801. This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the project DAIKIRI under the grant no 01IS19085B and by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the project RAKI under the grant no 01MD19012B. The authors gratefully acknowledge the funding of this project by computing time provided by the Paderborn Center for Parallel Computing (PC^2).

[3] Note that we ran OCEL with its default settings and F1 scores are not available.

References

1. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. Nat. Genet. **25**(1), 25–29 (2000)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
3. Badea, L., Nienhuys-Cheng, S.-H.: A refinement operator for description logics. In: Cussens, J., Frisch, A. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44960-4_3
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2) (2012)
5. 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)
6. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data. Mach. Learn. **94**(2), 233–259 (2013). https://doi.org/10.1007/s10994-013-5363-6
7. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner—a framework for inductive learning on the Semantic Web. J. Web Semant. **39**, 15–24 (2016)
8. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
9. Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource. Nucleic Acids Res. **32**(suppl1), D258–D261 (2004)
10. Dai, Y., Wang, S., Xiong, N.N., Guo, W.: A survey on knowledge graph embedding: approaches, applications and benchmarks. Electronics **9**(5), 750 (2020)
11. Demir, C., Ngomo, A.C.N.: Convolutional complex knowledge graph embeddings. arXiv preprint arXiv:2008.03130 (2020)
12. Deshpande, O., et al.: Building, maintaining, and using knowledge bases: a report from the trenches. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 1209–1220 (2013)
13. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85928-4_12
14. Heindorf, S., et al.: EvoLearner: Learning description logics with evolutionary algorithms. In: Proceedings of the ACM Web Conference (2022)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Hogan, A., et al.: Knowledge graphs. Synth. Lect. Data Semant. Knowl. **12**(2), 1–257 (2021)
17. Ioannidis, V.N., et al.: DRKG-drug repurposing knowledge graph for COVID-19 (2020)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Kouagou, N.J., Heindorf, S., Demir, C., Ngomo, A.N.: Neural class expression synthesis. CoRR abs/2111.08486 (2021)
20. Krötzsch, M., Simancik, F., Horrocks, I.: A description logic primer. CoRR abs/1201.4089 (2012)
21. Lehmann, J.: DL-Learner: learning concepts in description logics. J. Mach. Learn. Res. **10**, 2639–2642 (2009)
22. Lehmann, J.: Learning OWL Class Expressions, vol. 22. IOS Press (2010)
23. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. J. Web Semant. **9**(1), 71–81 (2011)

24. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Mach. Learn. **78**(1–2), 203 (2010)
25. MacLean, F.: Knowledge graphs and their applications in drug discovery. Expert Opin. Drug Discov. **16**(9), 1057–1069 (2021)
26. Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing YAGO: scalable machine learning for linked data. In: Proceedings of the 21st international conference on World Wide Web, pp. 271–280 (2012)
27. Percha, B., Altman, R.B.: A global network of biomedical relationships derived from text. Bioinformatics **34**(15), 2614–2624 (2018)
28. Rizzo, G., Fanizzi, N., d'Amato, C.: Class expression induction as concept space exploration: from DL-Foil to DL-Focl. Future Gener. Comput. Syst. **108**, 256–272 (2020)
29. Rizzo, G., Fanizzi, N., d'Amato, C., Esposito, F.: A framework for tackling myopia in concept learning on the web of data. In: Faron Zucker, C., Ghidini, C., Napoli, A., Toussaint, Y. (eds.) EKAW 2018. LNCS (LNAI), vol. 11313, pp. 338–354. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03667-6_22
30. Rudolph, S.: Foundations of description logics. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23032-5_2
31. Sarker, M.K., Hitzler, P.: Efficient concept induction for description logics. In: AAAI, pp. 3036–3043 (2019)
32. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. **48**(1), 1–26 (1991)
33. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
34. Wang, Z., Li, J., Liu, Z., Tang, J.: Text-enhanced representation learning for knowledge graph. In: Proceedings of International Joint Conference on Artificial Intelligent (IJCAI), pp. 4–17 (2016)
35. Weston, J., Bordes, A., Yakhnenko, O., Usunier, N.: Connecting language and knowledge bases with embedding models for relation extraction. arXiv preprint arXiv:1307.7973 (2013)
36. Wishart, D.S., et al.: DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res. **46**(D1), D1074–D1082 (2018)
37. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
38. Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J.: Deep sets. In: NIPS, pp. 3391–3401 (2017)