Neuro-Symbolic Class Expression Learning

Caglar Demir, **Axel-Cyrille Ngonga Ngomo**
Data Science Research Group, Paderborn University
caglar.demir@upb.de, axel.ngonga@upb.de

Abstract

Models computed using deep learning have been effectively applied to tackle various problems in many disciplines. Yet, the predictions of these models are often at most post-hoc and locally explainable. In contrast, class expressions in description logics are ante-hoc and globally explainable. Although state-of-the-art symbolic machine learning approaches are being successfully applied to learn class expressions, their application at large scale has been hindered by their impractical runtimes. Arguably, the reliance on myopic heuristic functions contributes to this limitation. We propose a novel neuro-symbolic class expression learning model, DRILL, to mitigate this limitation. By learning non-myopic heuristic functions with deep Q-learning, DRILL efficiently steers the standard search procedure in a quasi-ordered search space towards goal states. Our extensive experiments on 4 benchmark datasets and 390 learning problems suggest that DRILL converges to goal states at least 2.7 times faster than state-of-the-art models on all learning problems. The results of our statistical significance test confirms that DRILL converges to goal states significantly faster (p-value $< 1\%$) than state-of-the-art models on all benchmark datasets. We provide an open-source implementation of DRILL, including pre-trained models, training and evaluation scripts.

1 Introduction

Transparency and explainability are quintessential to establish trust in AI decisions [Holzinger *et al.*, 2019; Samek and Müller, 2019]. Being able to explain Machine Learning (ML) decisions becomes particularly important on the Web, the largest and arguably most used information infrastructure available to humanity with 5.03 billion users [Statista, 2023]. A key development over the last decade has been the increasing availability of Web data in the form of partially large-scale Knowledge Bases (KBs) in RDF [Hogan *et al.*, 2021]. According to the 2022 crawl of WebDataCommons, roughly 50% of the Web sites now contain (fragments of) RDF KBs. The giant joint KB that can be extracted from the Web is known to contain at least 82 billion assertions [Bizer *et al.*, 2022].

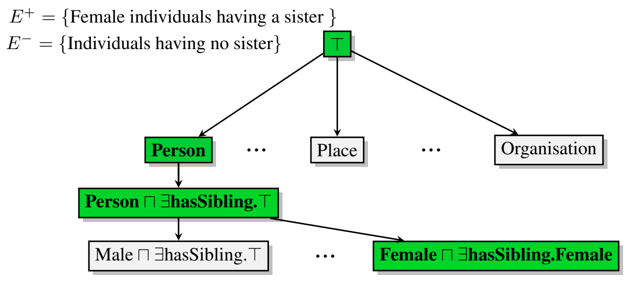
$E^+ = \{$Female individuals having a sister $\}$
$E^- = \{$Individuals having no sister$\}$
$\top$
Person
Place
Organisation
$\text{Person} \sqcap \exists\text{hasSibling}.\top$
$\text{Male} \sqcap \exists\text{hasSibling}.\top$
$\text{Female} \sqcap \exists\text{hasSibling.Female}$

Figure 1: Illustration of traversing a quasi-ordered search space $\mathcal{S}$. Rectangles represent explored $\mathcal{ALC}$ expressions in $\mathcal{S}$. Green filled rectangles represent a sequence of $\mathcal{ALC}$ expressions leading to a goal expression.

This bundle of machine-processable knowledge has led to Web-scale KBs being used in a plethora of applications ranging from the discovery of protein functions to content recommendation systems [Kulmanov *et al.*, 2018; Oramas *et al.*, 2016]. Devising explainable ML approaches for Web-scale RDF KBs is hence an indisputable building block of a trustworthy Web. Here, we focus on scaling up Class Expression Learning (CEL) over Description Logics (DLs) [Lehmann and Hitzler, 2010]. The formal setting of CEL is as follows: Given a knowledge base $\mathcal{K}$ in a DL (e.g. $\mathcal{ALC}$), a set of positive individuals E^+, and a set of negative individuals E^-, the goal is to learn a class expression H in the given DL such that $\forall p \in E^+\ \mathcal{K} \models H(p)$ and $\forall n \in E^-\ \mathcal{K} \not\models H(n)$. State-of-the-art symbolic models attempt to find an H via reformulating the CEL problem as a search problem in an infinite quasi-ordered state space $(\mathcal{S}, \preceq)$ [Bühmann *et al.*, 2018; Lehmann *et al.*, 2011]. This search problem is tackled by iteratively exploring states from $\mathcal{S}$, starting from an initial state (e.g., $\top$) and aiming to terminate in a goal state H (see Example 1 and Figure 1).

Example 1. *Given a set of individuals who have a sister as E^+ and a set of individuals who have no sister as E^-, a CEL models may traverse the sequence* $\top \rightsquigarrow \texttt{Person} \rightsquigarrow \texttt{Person} \sqcap \exists\texttt{hasSibling}.\top \rightsquigarrow \texttt{Person} \sqcap \exists\texttt{hasSibling}.\texttt{Female}$.

The search for a hypothesis H is steered by optimizing a heuristic function that signals how well an expression fits a learning problem and can be used as starting point for the next steps of the search. However, heuristic functions of state-of-the-art models determine the heuristic value of a state *without any consideration for possible future states* (see Section 3). We argue that this is an important drawback of current CEL models, which incur exploring a large number of states to find a goal state [Bin *et al.*, 2016; Hitzler *et al.*, 2020]. Current remedies include applying (a) the redundancy elimination and (b) expression simplification rules [Lehmann *et al.*, 2011]. However, these treatments introduce additional computation and often do not hinder impractical runtimes.

We address this drawback by proposing a neuro-symbolic and refinement-based CEL approach, DRILL, that relies on a deep Q-network to efficiently steer the search towards a goal state, while incorporating consideration for future states in immediate actions. We tackle CEL by training DRILL to select actions in a fashion that maximizes *cumulative discounted future rewards*. Learning cumulative discounted future rewards of states decreases the runtimes of CEL and makes the aforementioned remedies unnecessary. Moreover, DRILL does not require length information pertaining to class expressions to converge towards syntactically simpler expressions because it implicitly learns to avoid syntactically complex expressions during training. Importantly, leveraging a deep Q-network enables DRILL to perform estimations in a batch fashion via using multi-CPUs or -GPUs. Most existing state-of-the-art models (e.g., CELOE, OCEL, ELTL, DL-FOIL, and SParCEL) relies on a single CPU to perform estimations in a sequential manner [Bühmann *et al.*, 2018; Fanizzi *et al.*, 2018; Tran *et al.*, 2017]. Not exploiting modern parallel compute architectures can regarded as yet another hindrance to scaling to large real-world KBs.

To train DRILL, we design an unsupervised training procedure based on [Mnih *et al.*, 2015]. Through this training procedure, DRILL can be trained on any KBs in a DL (e.g., $\mathcal{ALC}$ or $\mathcal{SROIQ}$) with no adjustment of its architecture, since the training procedure requires only a refinement operator, a retrieval function, and an embedding look-up operation pertaining to a selected DL (see Algorithm 1). Our experiments suggest that DRILL steers the search towards goal states more efficiently than state-of the art approaches based on refinement operators in 390 CEL problems on 4 benchmark datasets. In particular, DRILL finds goal states **at least 2.7 times faster** than state-of-the-art approaches on all benchmark datasets. The results of one- and two-sided Wilcoxon signed rank tests confirm that the superior performance of DRILL is significant at a confidence level of 99%. The main contributions of this paper are as follows:

1. We model CEL using refinement operators within the framework of reinforcement learning.
2. We present a Q-network and a length-based refinement operator to guide the search for class expressions within an infinite state space.
3. We provide an open-source implementation of our framework to foster research in the direction of combining reinforcement learning with CEL.[1]

[1]https://github.com/dice-group/DRILL

2 Related Work

A plethora of works have investigated class expression learning in Description Logics (DLs) [Hitzler *et al.*, 2009; Lehmann, 2010]. Most symbolic systems differ in the usage of heuristic functions and the design of the refinement operators. For instance, Badea et al. [Badea and Nienhuys-Cheng, 2000] apply a top-down refinement operator to supervised learning. YINYANG combines the previous approaches to learn class expressions [Iannone *et al.*, 2007]. DL-FOIL uses unlabeled individuals to take the open-world assumption into account [Fanizzi *et al.*, 2018]. SParCEL employs upward and downward refinements together [Tran *et al.*, 2017].

DL-Learner [Lehmann, 2010; Lehmann, 2009] is regarded as the most mature and recent system for CEL [Sarker and Hitzler, 2019]. DL-Learner consists of several state-of-the-art models, including OCEL, ELTL, and CELOE. These models apply a proper and complete refinement operator to traverse in a quasi-ordered state space $(\mathcal{S}, \preceq)$. The search of a desired class expression begins with the most general expression ($\top$) and continues to more specific expressions. During the search, CELOE prioritizes syntactically shorter expressions (see Section 3.2), whereas OCEL and ELTL incorporate similar heuristic rules in their search. Aforementioned models apply the redundancy elimination and the expression simplification rules to decrease the 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 state-of-the-art approaches based on refinement operators [d'Amato, 2020; Hitzler *et al.*, 2020; Sarker and Hitzler, 2019]. All aforementioned models rely on myopic heuristics. The goal of this paper is to address exactly this drawback through deep-Q-learning. As DL-Learner is regarded as the most mature and recent system for CEL, we evaluate DRILL against CEL models included in DL-Learner.

3 Preliminaries

3.1 Knowledge Bases and Description Logics

A KB is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. $\mathcal{T}$ denotes the set of terminological axioms describing the relationships between class expressions (also called concepts) N_C in $\mathcal{K}$. Every terminological axiom is in the form of $A \sqsubseteq B$ or $A \equiv B$ where A and B are class expressions. $\mathcal{A}$ denotes the set of assertions describing relationships among individuals $a, b \in N_I$ via roles $r \in N_R$ as well as instantiation relationships between elements of N_I and N_C. Every assertion in $\mathcal{A}$ must be in the form of $A(a)$ or $r(a, b)$, where $A \in N_C$, $r \in N_R$, and $a, b \in N_I$. Here, we consider KBs in the DL $\mathcal{ALC}$ (Attributive Language with Complements). The model-theoretic semantics of $\mathcal{ALC}$ are given in the supplemental material.

3.2 Class Expression Learning over $\mathcal{ALC}$

We define CEL in a fashion akin to [Lehmann and Hitzler, 2010]. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ over $\mathcal{ALC}$, the set E^+ of positive individuals, the set E^- of negative individuals be given. The goal of CEL is to find a class expression H in $\mathcal{ALC}$ that satisfies

$$\forall p \in E^+ \mathcal{K} \models H(p) \wedge \forall n \in E^- \mathcal{K} \not\models H(n). \qquad (1)$$

The problem of finding H is transformed into a search problem within a quasi-ordered state space $(\mathcal{S}, \preceq)$, where each state $s \in \mathcal{S}$ is a valid $\mathcal{ALC}$ class expression from all valid class expressions denoted by $\mathcal{C}$. Note that we use $\mathcal{C}$ and $\mathcal{S}$ interchangeably as $\mathcal{S}$ solely represents quasi-ordered $\mathcal{C}$. Traversing $\mathcal{S}$ is conducted via a downward refinement operator (also called specialization) $\rho : \mathcal{S} \to 2^{\mathcal{S}}$ defined as

$$\forall s \in \mathcal{S} : \rho(s) \subseteq \{s' \in \mathcal{S} \mid s' \preceq s\}. \tag{2}$$

State-of-the-art models often begin their search towards an H after a search tree is initialized with the most general state $(\top)$ as a root node. This search tree is iteratively built by selecting the node with the highest heuristic value and adding its refinements as its children into a search tree. The key to an efficient search in $\mathcal{S}$ is a heuristic function steering the search towards an H. In current approaches, the heuristic function depends on a retrieval function $\mathcal{R} : \mathcal{C} \to 2^{N_I}$, which maps a class expression to the set of its individuals. Given $s \in \mathcal{S}$ and its a downward refinement $s' \in \rho(s)$, CELOE computes the heuristic value

$$\phi_{\text{CELOE}}(s, s') = \mathrm{Q}(s') + \lambda \cdot \big[\mathrm{Q}(s') - \mathrm{Q}(s)\big] - \beta \cdot |s'|, \tag{3}$$

where $\beta > \lambda \geq 0$, $\mathrm{Q}(\cdot)$ denotes a quality function (e.g. F_1 score), and $|s'|$ stands for the length of s'. Through $\mathrm{Q}(\cdot)$ and $|\cdot|$, $\phi_{\text{CELOE}}(\cdot, \cdot)$ steers the search towards more accurate and syntactically shorter expressions. The F_1 score of s is computed as

$$F_1(s) = \frac{|E^+ \cap \mathcal{R}(s)|}{|E^+ \cap \mathcal{R}(s)| + \frac{1}{2}(|E^- \cap \mathcal{R}(s)| + |E^+ \setminus \mathcal{R}(s)|)}. \tag{4}$$

3.3 Reinforcement Learning

Reinforcement Learning (RL) has been successfully applied in learning policies for sequential decision-making problems. Notable examples range from deep Q-learning for Atari game-playing [Mnih *et al.*, 2015] to protein folding [Jumper *et al.*, 2021]. Sequential decision problems in RL are often modeled as Markov Decision Processs (MDPs) applied to model the synchronous interaction between an *agent* and an *environment* in RL. Formally, an MDP is a 5-tuple $\langle \mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{T}, \gamma \rangle$, where $\mathbf{S}$ is a set of states, $\mathbf{A}$ is a set of actions, $\mathbf{R}$ is a reward function, $\mathbf{T}$ is a transition function and $\gamma \in [0, 1)$ is the discount rate. Given a state $\mathbf{s}_t \in \mathbf{S}$ at a time t, an agent takes an action $\mathbf{a}_t$ from the set $\mathbf{A}(\mathbf{s}_t)$ of actions available on $\mathbf{s}_t$. Upon taking an action, the agent receives a reward r_t and reaches the next state $\mathbf{s}_{t+1}$. The probability of reaching $\mathbf{s}_{t+1}$ and receiving r_t by taking action $\mathbf{a}_t$ in a given $\mathbf{s}_t$ is assigned by $\mathbf{T}$. This synchronous interaction between agent and environment induces a *trajectory* τ. The *discounted return* of the t^{th} point in τ is defined as

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{|\tau|-t} r_{|\tau|-t}, \tag{5}$$

where γ determines the present value of future rewards. The goal of an RL agent is to select actions in a fashion that maximizes the cumulate discounted rewards [Sutton and Barto, 2018]. A policy π prescribes which action to take in a given state. An optimal policy π_* hence prescribes actions on any state that maximize G_t. To obtain π_*, value functions are often used. The action-value function $Q_\pi : \mathbf{S} \times \mathbf{A} \to \mathbb{R}$ is defined as

$$Q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[G_t \mid \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right]. \tag{6}$$

Using the Bellman equation, the optimal action-value function Q_* can be approximated:

$$Q_{i+1}(\mathbf{s}, \mathbf{a}) \leftarrow Q_i(\mathbf{s}, \mathbf{a}) + \alpha \big[r + \gamma \max_{\mathbf{a}' \in \mathbf{A}(\mathbf{s}')} Q_i(\mathbf{s}', \mathbf{a}') - Q_i(\mathbf{s}, \mathbf{a}) \big], \tag{7}$$

where $\alpha \in (0, 1]$. Through using the iterative update defined in Equation (7), Q_i converges to Q_* as $i \to \infty$ [Sutton and Barto, 2018]. However, iteratively approximating exact optimal values is often computationally infeasible as $|\mathbf{S}|$ or $|\mathbf{A}|$ increase. In practice, a neural network parameterized with Θ is commonly applied to approximate the optimal action-value function, $Q(\mathbf{s}, \mathbf{a}; \Theta) \approx Q_*(\mathbf{s}, \mathbf{a})$ [Riedmiller, 2005; Mnih *et al.*, 2015]. To this end, trajectories are often accumulated as a RL agent interacts with an environment. A training dataset $\mathcal{D}$ is iteratively built through appending trajectories. Then, Θ is updated via minimizing the following loss function $(r + \gamma \max_{\mathbf{a}' \in \mathbf{A}(\mathbf{s}')} Q(\mathbf{s}', \mathbf{a}'; \Theta) - Q(\mathbf{s}, \mathbf{a}; \Theta))^2$ [Riedmiller, 2005]. Through introducing the experience replay mechanism and using the target network idea, Mnih et al. [Mnih *et al.*, 2015] extend the previous work and design the following Q-loss function

$$\mathcal{L}(\Theta_i) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim U(\mathcal{D})} \left[\Big(r + \gamma \max_{\mathbf{a}' \in \mathbf{A}(\mathbf{s}')} Q(\mathbf{s}', \mathbf{a}'; \Theta_i^-) - Q(\mathbf{s}, \mathbf{a}; \Theta_i) \Big)^2 \right], \tag{8}$$

where $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \sim U(\mathcal{D})$ denotes drawing a data point uniformly at random from a set of most recent data points. Hence, the size of $\mathcal{D}$ is fixed and it contains only most recent interactions between an RL agent and an RL environment. Moreover, Θ_i are the parameters of the neural network at iteration i and Θ_i^- is the parameters of the same neural network that is only updated at every few steps. This framework (known as deep Q-learning and deep Q-network) allows approximating Q_* even in large state-action spaces.

4 Methodology

4.1 Motivation

Devising a suitable heuristic function is crucial in CEL. The search of a H is steered by optimizing a heuristic $\phi : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$ that is expected to signal how well refining a quasi-ordered state and transition into one of its refinement state assists to find a H. Equation (3) indicates that $\phi(s, s')$ of state-of-the-art approaches **compute heuristic values without incorporating any information pertaining to** $\rho(s')$. This is analogous to setting $\gamma = 0$ in Equation (5), i.e., to setting the present value of future rewards to 0. This implies that heuristic functions of state-of-the-art CEL models correspond to myopic RL agents, whose only concern is to maximize immediate rewards [Sutton and Barto, 2018]. To address this drawback, we reformulate the CEL problem defined in Equation (1) as a problem of maximizing the cumulative discounted future rewards defined in Equation (5). To maximize the latter, we leverage the deep Q-learning framework.

4.2 Quasi-ordered RL Environment

A RL environment is constructed via continuous vector representations of quasi-ordered states. To obtain these representations, we leverage pre-trained ConEx knowledge graph embedding models provided in the dice-embeddings framework [Demir and Ngomo, 2021] [2]. An embedding of $\mathcal{ALC}$ class expression corresponds to an embedding of a set of individuals obtained via the retrieval function $\mathcal{R}$. This embedding lookup operation is denoted with $\mathcal{E}(\mathcal{R}(\cdot))$, where $\mathcal{E}: 2^{N_I} \mapsto \mathbb{R}^{|2^{N_I}| \times d}$ and d denotes the size of an embedding vector. A RL state $\mathbf{s} \in \mathbb{R}^{|\mathcal{R}(s)| \times d}$ is obtained by using $\mathcal{R}$ with a pre-trained embedding model. For instance, the initial state of this RL environment is represented with embeddings of $\top$, i.e., embedding of all individuals $\mathcal{E}(\mathcal{R}(\top))$. By this, a RL environment can be constructed regardless of selected Open World Assumption (OWA) or Closed World Assumption (CWA), since under any of these assumptions $\mathcal{R}$ returns a set of individuals. In this environment, we consider a RL action $\mathbf{a} \in \mathbf{A}(\mathbf{s})$ as an action of refining a quasi-ordered state and transitioning to $s' \in \rho(s)$. Important to note that a RL environment for more expressive DLs (e.g. $\mathcal{SROIQ}$ [Horrocks *et al.*, 2006]) can be readily constructed provided that respective $\mathcal{R}(\cdot)$, $\mathcal{E}(\cdot)$, and $\rho(\cdot)$ for $\mathcal{K}$ over $\mathcal{SROIQ}$ are given. This is also shown in the requirements of Algorithm 1. Here, we focus on constructing a RL environment based on $\mathcal{ALC}$ as $\mathcal{ALC}$ is often used in the related works.

4.3 DRILL

The standard deep Q-loss function defined in Equation (8) cannot be directly applied in this quasi-ordered RL environment, since the set of possible actions on a given RL state is not fixed as in [Mnih *et al.*, 2015]. To mitigate this issue, we used the state-state Q function [Edwards *et al.*, 2020]. The state-state Q function allows *avoiding redundant states* on RL environments, where the set of possible actions is not fixed. As the number of redundant states increases, using the state-state Q function often leads to more favorable results than using the state-action Q function. The ability of avoiding redundant states without requiring any additional computation is particularly important for our purpose, as many state-of-the-art symbolic models (CELOE, OCEL, and ELTL) apply redundancy elimination techniques to remove redundant states from their search [Lehmann, 2010; Lehmann *et al.*, 2011]. Therefore, minimizing the state-state Q function may permit *incorporating consideration for future states in immediate decisions* and *pruning redundant states without additional computation*. With these considerations in mind, we designed the following loss function

$$\mathcal{L}(\Theta_i) = \mathbb{E}\left[\left(r + \gamma \max_{x \preceq s'} Q(\mathbf{s}', \mathbf{x}, \mathbf{e}_+, \mathbf{e}_-; \Theta_i^-) - Q(\mathbf{s}, \mathbf{s}', \mathbf{e}_+, \mathbf{e}_-; \Theta_i)\right)^2\right], \tag{9}$$

where the expectation w.r.t. $(\mathbf{s}, \mathbf{s}', r, \mathbf{e}_+, \mathbf{e}_-) \sim U(\mathcal{D})$. To minimize Equation (9), we adapt the deep Q-Network proposed in [Mnih *et al.*, 2015] as follows

$$\phi_{\text{DRILL}}([\mathbf{s}, \mathbf{s}', \mathbf{e}_+, \mathbf{e}_-]; \Theta) = f\Big(\operatorname{vec}(f\left[\Psi([\mathbf{s}, \mathbf{s}', \mathbf{e}_+, \mathbf{e}_-]) * \omega\right)] \cdot \mathbf{W}\Big) \cdot \mathbf{H}, \tag{10}$$

where $\Theta = [\omega, \mathbf{W}, \mathbf{H}]$ denotes trainable parameters. Two quasi-ordered RL states $\mathbf{s} \in \mathbb{R}^{|\mathcal{R}(s)| \times d}$, $\mathbf{s}' \in \mathbb{R}^{|\mathcal{R}(s')| \times d}$ are obtained through applying $\mathcal{R}(\cdot)$ and $\mathcal{E}(\cdot)$ consecutively. Similarly, $\mathbf{e}_+ \in \mathbb{R}^{|E^+| \times d}$ and $\mathbf{e}_- \in \mathbb{R}^{|E^-| \times d}$ are obtained via $\mathcal{E}(\cdot)$. Note that $\mathcal{E}(\cdot)$ returns a set of embedding vectors of size d. To obtain permutation-invariant representations, we apply $\Psi(\cdot)$ to convert a input into $\mathbb{R}^{4 \times d}$ by averaging the embeddings of its input. By doing so, the output of ϕ_{DRILL} is invariant to the order of items in $\mathbf{s}, \mathbf{s}', \mathbf{e}_+$, and $\mathbf{e}_-$. Moreover, $f(\cdot)$, $\operatorname{vec}(\cdot)$, $*$ and ω correspond the rectified linear unit function, a flattening operation, the convolution operation, and kernels in the convolution operation. $\mathbf{W}$ and $\mathbf{H}$ denote two linear transformations, respectively.

4.4 Rewards and Unsupervised Training

We base our reward function on the CELOE heuristic:

$$\mathbf{R}(\mathbf{s}, \mathbf{s}') = \begin{cases} maxreward, & \text{if F1-score}(s') = 1; \\ \phi_{\text{CELOE}}(s, s'), & \text{if F1-score}(s') < 1. \end{cases} \tag{11}$$

A reward of transitioning from RL state $\mathbf{s}$ to $\mathbf{s}'$ is based on the heuristic value of refining a quasi-ordered class expression s and transitioning to s' provided that s' is not a class expression maximizing Equation (4). Through minimizing Equation (9) on $\mathcal{D}$, DRILL is expected to steer the search towards shorter class expressions *without computing lengths of class expressions*. This stems from the fact that DRILL is trained on rewards that involves the length information provided within ϕ_{CELOE}. DRILL mitigates the myopic heuristic nature of ϕ_{CELOE} through learning discounted cumulative future rewards, i.e., discounted cumulative future CELOE heuristic values. Importantly, this setting allows to avoid additional computations required to determine a length of a class expression.

To generate learning problems for training, $\rho(\cdot)$ is iteratively applied in a randomized depth-first search manner starting from $\top$. During this randomized process, each state s satisfying the length constraint $1 \leq |s| \leq maxlen$ and $|\mathcal{R}(s)| > 0$ is stored. In our experiments, we set $maxlen = 5$. To ensure the heterogeneity of the set of learning problems, we perform this task m times. For each stored state, we compute all positive $E^+ = \mathcal{R}(s)$ and negative examples $E^- = N_I \setminus \mathcal{R}(s)$, respectively. This operation often results in creating imbalanced E^+ and E^-, with $|E^-| >> |E^+|$ being common. To alleviate imbalanced examples, we randomly undersample the largest set of examples so that $|E^+| = |E^-|$. The training procedure is applied several times to generate several learning problems. For each learning problem, DRILL was trained according to Algorithm 1. Importantly, required inputs of Algorithm 1 confirm that DRILL can be readily used on KB in more expressive DLs provided that $\rho(\cdot)$, $\mathcal{R}(\cdot)$, and $\mathcal{E}(\cdot)$ are given.

We design a length-base refinement operator on $\mathcal{ALC}$. The operator is designed to make no use of subsumption information but rather to consider concept length as ordering. This in contrast to most refinement operators found the in the literature, which order concepts w.r.t. the subsumption relation. By this operator, we measure the impact of the influence of subsumption semantics on DRILL. Let N_C be a finite set of

[2]https://github.com/dice-group/dice-embeddings

Algorithm 1 DRILL with deep Q-learning training procedure

```
1: Require: $E^+, E^-, \rho, \mathbf{R}, \mathcal{R}, \mathcal{E}, \Theta, M, T$
2: for m := 1, M do
3:     $\mathbf{s}, s ::= \mathcal{E}(\mathcal{R}(\top)), \top$
4:     for $t := 1, T$ do
5:         $\mathbf{z} := \{s' \in \rho(s) | \mathcal{E}(\mathcal{R}(s'))\}$
6:         if $\epsilon > .1$ then
7:             Select random $\mathbf{s}' \in \mathbf{z}$
8:         else
9:             Select $\mathbf{s}' := argmax_{\mathbf{s}' \in \mathbf{z}} \phi_{\text{DRILL}}([\mathbf{s}, \mathbf{s}', \mathbf{e}_+, \mathbf{e}_-]); \Theta)$
10:        end if
11:        Compute reward $r := \mathbf{R}(\mathbf{s}, \mathbf{s}')$
12:        Append $[\mathbf{s}, \mathbf{s}', \mathbf{e}_+, \mathbf{e}_-, r]$ to $\mathcal{D}$
13:        Set $s, \mathbf{s} ::= s', \mathbf{s}'$
14:    end for
15:    Reduce $\epsilon$ with a constant
16:    if m % 5 == 0 then
17:        Sample random minibatches from $\mathcal{D}$
18:        Compute loss of minibatches w.r.t. Equation (9)
19:        Update $\Theta$ accordingly
20:    end if
21: end for
```

named concepts and let R be a finite set of roles. We set $N_C^+ = N_C \cup \{\top, \bot\}$. The set S of all $\mathcal{ALC}$ class expressions built upon N_C and R is defined as follows: First, $N_C^+ \subset S$. Now, let $r \in R$. If C and D are elements of S, then so are $(C \sqcap D)$, $(C \sqcup D)$, $\exists r.C$, $\forall r.C$, and $\neg C$. We use the length of concepts to define an ordering over the set S as follows: $\forall C, D \in S : C \preceq D$ iff $|C| \leq |D|$. We define the operator ρ over $(S, \preceq)$ as follows:

$$\rho(C) = \begin{cases} \{\exists r.C, \forall r.C, C \sqcap \top, C \sqcup \top, \neg C, C\} & \text{for any } C \\ \{\exists r.\rho(X)\} & \text{if } C = \exists r.X \\ \{\forall r.\rho(X)\} & \text{if } C = \forall r.X \\ \{\neg \rho(X)\} & \text{if } C = \neg X \\ \{\rho(X) \sqcup \rho(Y)\} & \text{if } C = X \sqcup Y \\ \{\rho(X) \sqcap \rho(Y)\} & \text{if } C = X \sqcap Y \\ N_C^+ & \text{if } C = \top \end{cases} \tag{12}$$

5 Experiments

5.1 Datasets and Experimental Setup

We used four benchmark datasets (Family, Carcinogenesis, Mutagenesis and Biopax) [Bin *et al.*, 2016; Fanizzi *et al.*, 2018]. We compared approaches via the F1-score, accuracy, and the runtime in a manner akin to [Lehmann and Hitzler, 2010]. We used two standard stopping criteria for all approaches. (i) We set the maximum runtime to 3 seconds as models often reach good solutions within 1.5 seconds [Lehmann and Hitzler, 2010]. (ii) Approaches were configured to terminate as soon as they found a goal state (i.e., a state with F1-score = 1.0). Note that (i) was a soft constraint as the runtime criterion is not checked during obtaining refinements in DL-Learner. If models do not find a goal state, a most accurate state is retrieved.

6 Results

Benchmark Learning Problems Table 1 reports results on 20 benchmark learning problems provided within DL-Learner [Lehmann, 2009]. These results suggest that approaches yield similar performances in terms of F1-score and accuracy on benchmark datasets. Yet, DRILL outperforms all other approaches on all datasets w.r.t. its runtime. On all benchmark datasets, DRILL requires at most **3.3 seconds** to reach state-of-the-art performance, while CELOE, OCEL, and ELTL require at most **21 seconds**, **23.5 seconds**, and **22.1**, respectively. Overall, DRILL is at least **2.7 times** more time-efficient than CELOE, OCEL and ELTL on all standard learning problems. Moreover, during our experiments, we observed that the F1-score of the best found concept in OCEL are not reported. Results also indicate that CELOE, OCEL, and ELTL do not terminate within the set maximum runtime. We delved into their implementations in the DL-Learner framework and observed that the maximum runtime criterion is not checked until refinements of a given class expression are obtained (also mentioned in Section 5). Note that we also evaluated DL-FOIL and SParCEL in our initial experiments. However, both approaches failed to terminate. Consequently, we could not include DL-FOIL and SParCEL in our final set of experiments.

Convergence over Shorter Class Expressions Table 2 shows that DRILL and CELOE often converge towards shorter concepts compared to OCEL and ELTL on 18 learning problems. This indicates that DRILL **learned to converge over shorter concepts without any additional computation**, i.e., the overhead of computing lengths of concepts is mitigated.

In Section 6, we reported a sequence of quasi-ordered class expressions terminating a final prediction for the Uncle learning used in Table 2. Section 6 highlights that for a given learning problem, (a) each intermediate decision leading to a prediction can be obtained, and (b) the trade-off between runtimes and effectiveness. Note that (a) can be enriched through collecting different sequences of quasi-ordered class expressions terminating in Male $\sqcap$ $\forall$hasSibling.$\top$.

Random Learning Problems Table 4 reports results on 370 learning problems generated on benchmark datasets. These results confirm that DRILL consistently finds a goal concept faster than all other approaches. Importantly, DRILL was always able to find goal concepts in all learning problems. To exclude any possible impact of the process of learning problem generation, we conduct an experiment based on fully random learning problems. Although, in practice, CEL problems are not randomly generated, we were interested in quantify CEL performance on random learning problems with different sizes. Table 5 reports results on random learning problems with different sizes. Overall results indicate that OCEL does not find any adequate solution as the size of the random inputs increase, while ELTL performs poorly compared to CELOE and DRILL. Moreover, DRILL and CELOE often return expressions having similar F1-scores. Yet, DRILL often finds expressions having higher accuracy in a better runtime. Hence, we can conclude that it is improbable that the better performance of DRILL is due to the experimental setting used in Section 5.

Dataset	#LP	DRILL			CELOE			OCEL			ELTL		
		F1	Acc.	RT	F1	Acc.	RT	F1	Acc.	RT	F1	Acc.	RT
Family	18	0.96	0.95	**1.2**	**0.97**	**0.97**	3.6	*	0.94	6.1	0.96	0.95	3.4
Carcinogenesis	1	**0.71**	0.56	**3.3**	**0.71**	0.56	21.1	†	†	23.5	**0.71**	**0.57**	22.1
Mutagenesis	1	**0.70**	**0.54**	**3.0**	**0.70**	**0.54**	13.9	†	†	13.2	**0.70**	**0.54**	13.2

Table 1: Results on benchmark datasets. #LP, F1, Acc, and RT denote the number of benchmark learning problems, the F1-score, the accuracy, and runtime in seconds, respectively. † stands for no solution found by the respective approach. $*$ indicates that respective value is not reported in DL-Learner. Bold entries denote best results.

Expression	DRILL				CELOE				OCEL				ELTL			
	L	F1	Acc	RT	L	F1	Acc	RT	L	F1	Acc	RT	L	F1	Acc	RT
Aunt	6	**0.83**	**0.79**	**3.3**	6	**0.83**	**0.79**	5.7	16	*	**1.00**	5.8	1	0.80	0.76	2.8
Brother	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.9	1	*	**1.00**	5.8	5	**1.00**	**1.00**	3.8
Cousin	4	0.73	0.65	**2.9**	5	**0.79**	0.74	5.9	21	*	**1.00**	6.2	1	0.66	0.50	3.0
Daughter	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.9	1	*	**1.00**	5.9	3	**1.00**	**1.00**	2.9
Father	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	3.0	1	*	**1.00**	6.0	3	**1.00**	**1.00**	3.0
Granddaughter	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	3.1	1	*	**1.00**	5.3	1	**1.00**	**1.00**	2.9
Grandfather	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.9	1	*	**1.00**	5.7	1	**1.00**	**1.00**	3.0
Grandgranddaughter	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.9	1	*	**1.00**	5.9	7	**1.00**	**1.00**	3.0
Grandgrandfather	1	0.94	0.94	**1.2**	5	**1.00**	**1.00**	3.0	5	*	**1.00**	5.8	7	**1.00**	**1.00**	3.7
Grandgrandmother	9	0.94	0.94	**2.3**	5	**1.00**	**1.00**	3.1	5	*	**1.00**	5.9	7	**1.00**	**1.00**	3.7
Grandgrandson	1	0.92	0.92	**3.5**	5	**1.00**	**1.00**	5.7	5	*	**1.00**	6.6	7	**1.00**	**1.00**	3.1
Grandmother	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.8	1	*	**1.00**	5.9	1	**1.00**	**1.00**	3.1
Grandson	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.8	1	*	**1.00**	6.0	1	**1.00**	**1.00**	3.0
Mother	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.9	1	*	**1.00**	5.9	5	**1.00**	**1.00**	3.1
PersonWithASibling	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.8	1	*	**1.00**	7.0	1	**1.00**	**1.00**	3.1
Sister	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	2.8	1	*	**1.00**	5.8	5	**1.00**	**1.00**	3.0
Son	1	**1.00**	**1.00**	**0.2**	1	**1.00**	**1.00**	3.0	1	*	**1.00**	5.7	3	**1.00**	**1.00**	2.9
Uncle	5	**0.90**	**0.89**	**2.9**	5	**0.90**	**0.89**	5.9	†	†	†	5.8	1	0.88	0.87	3.9

Table 2: Results of single learning problems on the Family benchmark dataset. F1, Acc, and RT denote the F1-score, the accuracy, and runtime in seconds, respectively. † stands for no solution found by the respective approach. $*$ indicates that respective value is not reported in DL-Learner. Bold entries denote best results.

		DRILL		
Step	**Prediction**	**F1**	**Acc.**	**RT**
1	$\top$	0.67	0.67	0.1
2	Male	0.88	0.86	1.2
3	Male $\sqcap \exists$ hasSibling.$\top$	0.90	0.89	2.9

Table 3: A sequence of quasi-ordered concepts terminating in a most accurate concept found for the `Uncle` learning problem.

Length-based vs Subsumption-based Refinement Operator We were interested in quantifying the impact of using our length-based refinement operator instead of a subsumption-based refinement operator on DRILL's performance. To that end, we replaced our length-based refinement operator with CELOE's refinement operator in the implementation of DRILL. We chose this refinement operator because it performed best in all of our previous experiments. All other parameters remained constant. Our results are shown in Table 6 and suggest that DRILL with the length-based refinement operator (see Equation (12)) performs better in terms of F1, Acc, and RT than DRILL with CELOE's refinement operator. These results suggest that our length-based refinement operator allows DRILL to steer the search towards more accurate expressions efficiently, although F1 and Acc scores are on a par. This is an important result as it suggests that the semantics of the data are well captured in the embeddings used as input to DRILL. Hence, DRILL follows more of a linear construction of the class expression instead of an inference-based selection as in previous works.

Statistical Hypothesis Testing We performed a Wilcoxon signed-rank test. Our null hypothesis was that the performances of DRILL and CELOE come from the same distribution provided that a goal state is found. The alternative hypothesis was that these results come from different distributions. To perform the Wilcoxon signed-rank test (one-and two-sided), we used the runtimes of DRILL and CELOE on benchmark datasets provided both approaches found a goal state. We were able to reject the null hypothesis with a p-value $< 1\%$. Ergo, the superior runtime performance of DRILL is statistically significant.

7 Discussion

Our results on all benchmark datasets suggest that DRILL achieves state-of-the-art performance w.r.t. the quality of the expressions found during the search, while outperforming the state of the art significantly w.r.t. its runtime.

Dataset	#LP	DRILL			CELOE			OCEL			ELTL		
		F1	Acc.	T	F1	Acc	RT	F1	Acc	RT	F1	Acc	RT
Family	74	**1.00**	**1.00**	**1.1**	**1.00**	**1.00**	3.6	*	**1.00**	6.2	**1.00**	**1.00**	3.5
Carcinogenesis	100	**1.00**	**1.00**	**2.2**	**1.00**	**1.00**	17.3	*	**1.00**	20.6	**1.00**	**1.00**	19.1
Mutagenesis	100	**1.00**	**1.00**	**1.4**	**1.00**	**1.00**	10.0	*	0.98	12.9	0.97	0.97	10.2
Biopax	96	**1.00**	**1.00**	**1.1**	0.99	0.99	3.7	*	**1.00**	6.74	0.99	0.98	3.7

Table 4: Results on automatically generated learning problems. #LP, F1, Acc, and RT denote the number learning problems, the F1-score of prediction, accuracy of prediction, and runtime in seconds, respectively. $*$ indicates that respective value is not reported in DL-Learner. Bold entries denote best results.

Dataset		DRILL			CELOE			OCEL			ELTL		
	N	F1	Acc	RT	F1	Acc	RT	F1	Acc	RT	F1	Acc	RT
Family	1	**0.97**	0.95	**0.4**	**0.97**	0.95	3.6	*	**1.00**	6.1	**0.97**	0.95	3.6
	10	0.77	**0.73**	**1.3**	**0.78**	0.72	6.3	†	†	†	0.67	0.51	3.3
Carcinogenesis	1	**0.93**	0.90	**1.2**	**0.93**	0.90	19.3	*	0.80	20.1	**0.93**	0.90	19.3
	10	**0.71**	**0.60**	**2.2**	**0.71**	0.58	22.7	†	†	†	0.67	0.50	19.6
Mutagenesis	1	**0.97**	0.85	**2.8**	**0.97**	0.95	11.4	*	0.90	14.0	**0.93**	0.90	11.6
	10	**0.71**	**0.61**	**3.2**	0.70	0.58	13.9	†	†	†	0.67	0.50	10.9
Biopax	1	**0.93**	0.90	**1.0**	**0.93**	0.90	4.9	*	0.80	7.3	**0.93**	0.90	4.7
	10	**0.73**	**0.65**	**3.7**	0.72	0.60	7.1	†	†	†	0.67	0.50	4.4

Table 5: Results on fully randomly generated learning problems. E^+ and E^- are constructed via drawing N random individuals N_I without replacement, where $|E^+| = |E^-|$. † stands for no solution found by the approach and for the number learning problems, respectively. $*$ indicates that respective value is not reported in DL-Learner. Bold entries denote best results.

Dataset	#LP	Length-Based Ref.			CELOE's Ref.		
		F1	Acc	RT	F1	Acc	RT
Family	74	**1.00**	**1.00**	**1.1**	0.97	0.97	1.25
Carcinogenesis	100	**1.00**	**1.00**	**2.2**	0.93	0.93	12.8
Biopax	96	**1.00**	**1.00**	**1.1**	0.99	0.99	1.68

Table 6: Results on comparing refinement operators in DRILL.

This improvement in performance is due to the following: (i) deep Q-learning with the state-state Q loss function, (ii) the length-based refinement operator and (iii) the efficient computation of heuristic values. Deep Q-learning endows DRILL with the ability of considering future rewards, while selecting the next state for the refinement. State-of-the-art approaches lack this ability. Through minimizing the state-state Q-loss function, DRILL can detect redundant states without any additional computation. To detect redundant states, CELOE, OCEL, and ELTL require additional computations. Moreover, DRILL learns to converge to simple expressions without computing lengths of expressions, whereas other approaches yet again perform addition computations (see Equation (3)). These additional computations inherently increase their runtimes. DRILL achieved state-of-the-art performance on all datasets without over-parameterization and extensive parameter optimization. Throughout our experiments, DRILL was trained with a fixed configuration: 32 input channels, (3x3) kernel. Consequently, we conjecture that performing a more extensive hyperparameter optimization (e.g. via Hyperband [Li *et al.*, 2017]) is likely to improve the results even further. Moreover, the Q-network used in DRILL can assign scores for the refinements in a batch manner, while the sequential nature of state-of-the-art models precludes batch computation.

To exclude that our results were due to the approach used to generate learning problems being akin to that used to train DRILL, we carried out experiments with learning problems whose positive and negative examples were selected randomly. Although CEL problems are rarely constructed at random, we aimed to check the hypothesis that our runtime improvement was due to our approach to problem generation. Our results clearly indicate that this is not the case.

8 Conclusion

We introduced DRILL, a novel neuro-symbolic CEL approach to accelerate the problem of learning $\mathcal{ALC}$ concepts. DRILL effectively detects adequate $\mathcal{ALC}$ concepts without excessive exploration. Our experiments show that DRILL outperforms state-of the art models in 390 CEL problems on 4 benchmark datasets w.r.t. the quality of found concepts and the total computational time. The results of our statistical tests (one- and two-sided Wilcoxon signed rank tests) confirm the superior performance of DRILL. We strongly believe that incorporating neural models in CEL is worth pursuing further. Here, we focused mainly on learning permutation-embeddings tailored towards predicting quality of predefined $\mathcal{ALC}$ expressions. Yet, learning embeddings tailored towards more expressive DLs expressions may also lead to interesting insights.

Acknowledgments

This work has been supported by the HORIZON Europe research and innovation programme (GA No 101070305), by the Ministry of Culture and Science of North Rhine-Westphalia (GA No NW21-059D), by the German Research Foundation (GA No TRR 318/1 2021 – 438445824) and by the H2020 Marie Skłodowska-Curie programme (GA No 860801).

References

[Badea and Nienhuys-Cheng, 2000] Liviu Badea and Shan Hwei Nienhuys-Cheng. A refinement operator for description logics. In James Cussens and Alan Frisch, editors, *Inductive Logic Programming*, pages 40–59, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[Bin *et al.*, 2016] Simon Bin, Lorenz Bühmann, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Towards sparql-based induction for large-scale rdf data sets. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1551–1552. IOS Press, 2016.

[Bizer *et al.*, 2022] Christian Bizer, Robert Meusel, Anna Primpeli, and Alexander Brinkmann. 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: 2023-05-15.

[Bühmann *et al.*, 2018] Lorenz Bühmann, Jens Lehmann, Patrick Westphal, and Simon Bin. Dl-learner structured machine learning on semantic web data. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 467–471, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[Demir and Ngomo, 2021] Caglar Demir and Axel-Cyrille Ngonga Ngomo. Convolutional complex knowledge graph embeddings. In *European Semantic Web Conference*, pages 409–424. Springer, 2021.

[d'Amato, 2020] Claudia d'Amato. Machine learning for the semantic web: Lessons learnt and next research directions. *Semantic Web*, 11(1):195–203, 2020.

[Edwards *et al.*, 2020] Ashley D. Edwards, Himanshu Sahni, Rosanne Liu, Jane Hung, Ankit Jain, Rui Wang, Adrien Ecoffet, Thomas Miconi, Charles Isbell, and Jason Yosinski. Estimating q(s,s') with deep deterministic dynamics gradients. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2825–2835. PMLR, 2020.

[Fanizzi *et al.*, 2018] Nicola Fanizzi, Giuseppe Rizzo, Claudia d'Amato, and Floriana Esposito. Dlfoil: Class expression learning revisited. In *European Knowledge Acquisition Workshop*, pages 98–113. Springer, 2018.

[Hitzler *et al.*, 2009] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. 2009.

[Hitzler *et al.*, 2020] Pascal Hitzler, Federico Bianchi, Monireh Ebrahimi, and Md Kamruzzaman Sarker. Neural-symbolic integration and the semantic web. *Semantic Web*, 11(1):3–11, 2020.

[Hogan *et al.*, 2021] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021.

[Holzinger *et al.*, 2019] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[Horrocks *et al.*, 2006] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible sroiq. *Kr*, 6:57–67, 2006.

[Iannone *et al.*, 2007] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[Jumper *et al.*, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[Kulmanov *et al.*, 2018] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

[Lehmann and Hitzler, 2010] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203, 2010.

[Lehmann *et al.*, 2011] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9(1):71–81, 2011.

[Lehmann, 2009] Jens Lehmann. Dl-learner: learning concepts in description logics. *The Journal of Machine Learning Research*, 10:2639–2642, 2009.

[Lehmann, 2010] Jens Lehmann. *Learning OWL class expressions*, volume 22. IOS Press, 2010.

[Li *et al.*, 2017] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[Oramas *et al.*, 2016] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–21, 2016.

[Riedmiller, 2005] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[Samek and Müller, 2019] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 5–22. Springer, 2019.

[Sarker and Hitzler, 2019] Md Kamruzzaman Sarker and Pascal Hitzler. Efficient concept induction for description logics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3036–3043, 2019.

[Statista, 2023] Statista. Number of internet and social media users worldwide as of january 2023 (in billions). https://www.statista.com/statistics/617136/digital-population-worldwide/, 2023. Accessed: 2023-05-15.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Tran *et al.*, 2017] An C. Tran, Jens Dietrich, Hans W. Guesgen, and Stephen Marsland. Parallel symmetric class expression learning. *J. Mach. Learn. Res.*, 18:64:1–64:34, 2017.