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 antecedent 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].

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 $K$ 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 $\forall p \in E^+ \ K \models H(p)$ and $\forall n \in E^- \ K \not\models H(n)$. 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 $(S, \preceq)$ [Bühmann et al., 2018; Lehmann et al., 2011]. This search problem is tackled iteratively exploring states from $S$, 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 model may traverse the sequence $\top \leadsto \text{Person} \leadsto \text{Person} \sqcap \exists \text{hasSibling.Female}$.

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-CPU or multi-GPU. Most existing state-of-the-art models (e.g., CELOE, OCEL, ELTL, and SPARCEL) rely 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 be 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., ALC or 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.\footnote{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 \((S, \preceq)\). 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 \(K = (T, A)\). \(T\) denotes the set of terminological axioms describing the relationships between class expressions (also called concepts) \(N_C \subseteq K\). Every terminological axiom is in the form of \(A \sqsubseteq B\) or \(A \equiv B\) where \(A, B\) are class expressions. \(A\) 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 \(A\) 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 \(\text{ALC}\) (Attributive Language with Complements). The model-theoretic semantics of \(\text{ALC}\) are given in the supplemental material.

3.2 Class Expression Learning over \(\text{ALC}\)

We define CEL in a fashion akin to [Lehmann and Hitzler, 2010]. Let \(K = (T, A)\) over \(\text{ALC}\), 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 \(\text{ALC}\) that satisfies

\[
\forall p \in E^+K \models H(p) \land \forall n \in E^-K \not\models H(n). \tag{1}
\]
The problem of finding $H$ is transformed into a search problem within a quasi-ordered state space $(S, \preceq)$, where each state $s \in S$ is a valid $ALC$ class expression from all valid class expressions denoted by $C$. Note that we use $C$ and $S$ interchangeably as $S$ solely represents quasi-ordered $C$. Traversing $S$ is conducted via a downward refinement operator (also called specialization) $\rho: S \rightarrow 2^S$ defined as

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

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 $S$ is a heuristic function steering the search towards an $H$. In current approaches, the heuristic function depends on a retrieval function $R: C \rightarrow 2^{\mathbb{N}}$, which maps a class expression to the set of its individuals. Given $s \in S$ and its a downward refinement state $s' \in \rho(s)$, CELOE computes the heuristic value

$$\phi_{CELOE}(s, s') = Q(s') + \lambda \cdot \left[ Q(s') - Q(s) \right] - \beta \cdot |s'|, \quad (3)$$

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

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

### 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 gameplaying [Mnih et al., 2015] to protein folding [Jumper et al., 2021]. Sequential decision problems in RL are often modeled as Markov Decision Processes (MDPs) applied to model the synchronous interaction between an agent and an environment in RL. Formally, an MDP is a 5-tuple $(S, A, R, T, \gamma)$, where $S$ is a set of states, $A$ is a set of actions, $R$ is a reward function, $T$ is a transition function and $\gamma \in [0, 1)$ is the discount rate. Given a state $s_0 \in S$ at a time $t$, an agent takes an action $a_t$ from the set $A(s_t)$ of actions available on $s_t$. Upon taking an action, the agent receives a reward $r_t$ and reaches the next state $s_{t+1}$. The probability of reaching $s_{t+1}$ and receiving $r_t$ by taking action $a_t$ in a given $s_t$ is assigned by $T$. This synchronous interaction between agent and environment induces a trajectory $\tau$. The discounted return of the $t^{th}$ point in $\tau$ is defined as

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

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

$$Q_\pi(s, a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]. \quad (6)$$

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

$$Q_{t+1}(s, a) \leftarrow Q_t(s, a) + \alpha \left[ r + \gamma \max_{a' \in A(s')} Q_t(s', a') - Q_t(s, a) \right], \quad (7)$$

where $\alpha \in (0, 1]$. Through using the iterative update defined in Equation (7), $Q_t$ converges to $Q_*$ as $i \rightarrow \infty$ [Sutton and Barto, 2018]. However, iteratively approximating exact optimal values is often computationally infeasible as $|S|$ or $|A|$ increase. In practice, a neural network parameterized with $\Theta$ is commonly applied to approximate the optimal action-value function, $Q(s, a; \Theta) \approx Q_*(s, a)$ [Riedmiller, 2005; Mnih et al., 2015]. To this end, trajectories are often accumulated as a RL agent interacts with an environment. A training dataset $D$ is iteratively built through appending trajectories. Then, $\Theta$ is updated via minimizing the following loss function

$$\mathcal{L}(\Theta_t) = \mathbb{E}_{(s, a, r, s') \sim U(D)} \left[ (r + \gamma \max_{a' \in A(s')} Q_a(s', a'; \Theta_{t-1}) - Q(s, a; \Theta_t))^2 \right], \quad (8)$$

where $(s, a, r, s') \sim U(D)$ denotes drawing a data point uniformly at random from a set of most recent data points. Hence, the size of $D$ is fixed and it contains only most recent interactions between an RL agent and an RL environment. Moreover, $\Theta_t$ are the parameters of the neural network at iteration $i$ and $\Theta_{t-1}$ 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 $\phi : S \times S \rightarrow \mathbb{R}$ 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 an RL environment via continuous vector representations of quasi-ordered states. This embedding lookup operation is denoted with $\mathcal{E}(\mathcal{R}(\cdot))$, where $\mathcal{E}: 2^N \rightarrow \mathbb{R}^{2^{|\mathcal{L}|} \times d}$ and $d$ denotes the size of an embedding vector. A RL state $s \in \mathbb{R}^{\mathcal{R}(s) \times d}$ 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 $\mathbb{T}$, i.e., embedding of all individuals $\mathcal{E}(\mathcal{R}(\mathbb{T}))$. Through minimizing Equation (4), we design the following loss function to convert a input into $\mathbb{R}^{d \times 4}$ by averaging the embeddings of its input. By doing so, the output of $\phi_{\text{DRL}}$ is invariant to the order of items in $s, s', e_\omega$, and $e_\omega$. Moreover, $f(\cdot), \psi(\cdot)$, $\ast$ and $\omega$ correspond the rectified linear unit function, a flattening operation, the convolution operation, and kernels in the convolution operation. $W$ and $H$ denote two linear transformations, respectively.

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 ($\mathcal{CELOE}$, $\mathcal{OCLE}$, and $\mathcal{ELTL}$) 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

$$L(\Theta) = \mathbb{E}\left[ r + \gamma \max_{\omega} Q(s', x, e_\omega, e_\omega; \Theta) - Q(s, s', e_\omega, e_\omega; \Theta) \right]^2,$$

where the expectation w.r.t. $(s, s', r, e_\omega) \sim U(D)$. To minimize Equation (9), we adopt the deep Q-Network proposed in [Mnih et al., 2015] as follows

$$\phi_{\text{DRL}}([s, s', e_\omega, e_\omega]; \Theta) = f \left( \text{vec} \left( \psi([s, s', e_\omega, e_\omega]) \ast \omega \right) \right) \cdot W \cdot H,$$

where $\Theta = [\omega, W, H]$ denotes trainable parameters. Two quasi-ordered RL states $s \in \mathbb{R}^{\mathcal{R}(s) \times d}$, $s' \in \mathbb{R}^{\mathcal{R}(s') \times d}$ are obtained through applying $\mathcal{R}(\cdot)$ and $\mathcal{E}(\cdot)$ consecutively. Similarly, $e_\omega \in \mathbb{R}^{|\mathcal{L}| \times d}$ and $e_\omega \in \mathbb{R}^{|\mathcal{L}| \times d}$ are obtained via $\mathcal{E}(\cdot)$. Note that $\mathcal{E}(\cdot)$ returns a set of embedding vectors of size $d$.

4.4 Rewards and Unsupervised Training

We base our reward function on the CELOE heuristic:

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

A reward of transitioning from RL state $s$ to $s'$ 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 $D$, $\text{DRILL}$ is expected to steer the search towards shorter class expressions without computing lengths of class expressions. This stems from the fact that $\text{DRILL}$ is trained on rewards that involves the length information provided within $\phi_{\text{CELOE}}$. $\text{DRILL}$ mitigates the myopic heuristic nature of $\phi_{\text{CELOE}}$ 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 $\mathbb{T}$. During this randomized process, each state $s$ satisfying the length constraint $1 \leq |s| \leq \text{maxlen}$ and $|\mathcal{R}(s)| > 0$ is stored. In our experiments, we set $\text{maxlen} = 5$. 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 $E^- = N_j \setminus \mathcal{R}(s)$, respectively. This operation often results in creating imbalanced $E^+$ and $E^-$, with $|E^-| >> |E^+|$. 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, $\text{DRILL}$ was trained according to Algorithm 1. Importantly, required inputs of Algorithm 1 confirm that $\text{DRILL}$ is readily used on KB in more expressive DLs provided that $\phi_{\text{CELOE}}, \mathcal{R}(\cdot)$, 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 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 $\text{DRILL}$. Let $N_C$ be a finite set of

²https://github.com/dice-group/dice-embeddings
We used four benchmark datasets (Family, Carcinogenesis, D2018). We compared approaches via the F1-score, according to 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.

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.
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.
with a fixed configuration: 32 input channels, (3x3) kernel.

with the ability of considering future rewards, while selecting

(i) deep Q-learning with the state-state Q loss function, (ii) the

hyperparameter optimization (e.g. via Hyperband [Li et al.

Throughout our experiments, D_RILL

additional computations inherently increase their runtimes.

again perform addition computations (see Equation (3)). These

putting lengths of expressions, whereas other approaches yet

additional computation. To detect redundant states, CELOE,

lack this ability. Through minimizing the state-state Q-loss

the next state for the refinement. State-of-the-art approaches

putation of heuristic values. Deep Q-learning endows D

length-based refinement operator and (iii) the efficient com-

E

| entries denote best results.

| prediction, accuracy of prediction, and runtime in seconds, respectively. * indicates that respective value is not reported in DL-Learner. Bold entries denote best results.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#LP</th>
<th>DRILL</th>
<th>CELOE</th>
<th>OCEL</th>
<th>ELTL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>Acc</td>
<td>RT</td>
<td>F1</td>
</tr>
<tr>
<td>Family</td>
<td>74</td>
<td>1.00</td>
<td>1.00</td>
<td>1.1</td>
<td>1.00</td>
</tr>
<tr>
<td>Carcinogenesis</td>
<td>100</td>
<td>1.00</td>
<td>1.00</td>
<td>2.2</td>
<td>1.00</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>100</td>
<td>1.00</td>
<td>1.00</td>
<td>1.4</td>
<td>1.00</td>
</tr>
<tr>
<td>Biopax</td>
<td>96</td>
<td>1.00</td>
<td>1.00</td>
<td>1.1</td>
<td>0.99</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DRILL</th>
<th>CELOE</th>
<th>OCEL</th>
<th>ELTL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>F1</td>
<td>Acc</td>
<td>RT</td>
</tr>
<tr>
<td>Family</td>
<td>1</td>
<td>0.97</td>
<td>0.95</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.77</td>
<td>0.73</td>
<td>1.3</td>
</tr>
<tr>
<td>Carcinogenesis</td>
<td>1</td>
<td>0.93</td>
<td>0.90</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.71</td>
<td>0.60</td>
<td>2.2</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>1</td>
<td>0.97</td>
<td>0.85</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.71</td>
<td>0.61</td>
<td>3.2</td>
</tr>
<tr>
<td>Biopax</td>
<td>1</td>
<td>0.93</td>
<td>0.90</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.73</td>
<td>0.65</td>
<td>3.7</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#LP</th>
<th>Length-Based Ref.</th>
<th>CELOE's Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F1</td>
<td>Acc</td>
</tr>
<tr>
<td>Family</td>
<td>74</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Carcinogenesis</td>
<td>100</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Biopax</td>
<td>96</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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 ALC concepts. DRILL effectively detects adequate 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 ALC expressions. Yet, learning embeddings tailored towards more expressive DLs expressions may also lead to interesting insights.

3630
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


