

# Leveraging Class Abstraction for Commonsense Reinforcement Learning via Residual Policy Gradient Methods

Niklas Höpner<sup>1\*</sup>, Ilaria Tiddi<sup>2</sup> and Herke van Hoof<sup>1</sup>

<sup>1</sup>University of Amsterdam

<sup>2</sup>Vrije Universiteit Amsterdam

{n.r.hopner, h.c.vanhoof}@uva.nl, i.tiddi@vu.nl,

## Abstract

Enabling reinforcement learning (RL) agents to leverage a knowledge base while learning from experience promises to advance RL in knowledge intensive domains. However, it has proven difficult to leverage knowledge that is not manually tailored to the environment. We propose to use the subclass relationships present in open-source knowledge graphs to abstract away from specific objects. We develop a residual policy gradient method that is able to integrate knowledge across different abstraction levels in the class hierarchy. Our method results in improved sample efficiency and generalisation to unseen objects in commonsense games, but we also investigate failure modes, such as excessive noise in the extracted class knowledge or environments with little class structure.

## 1 Introduction

Deep reinforcement learning (DRL) has enabled us to optimise control policies in MDPs with high-dimensional state and action spaces such as in game-play [Silver *et al.*, 2016] and robotics [Lillicrap *et al.*, 2016]. Two main hindrances in bringing deep reinforcement learning to the real world are the sample inefficiency and poor generalisation performance of current methods [Kirk *et al.*, 2021]. Amongst other approaches, including prior knowledge in the learning process of the agent promises to alleviate these obstacles and move reinforcement learning (RL) from a tabula rasa method to more human-like learning. Depending on the area of research, prior knowledge representations can vary from pretrained embeddings or weights [Devlin *et al.*, 2019] to symbolic knowledge representations such as logics [Vaezipoor *et al.*, 2021] and knowledge graphs (KGs) [Zhang *et al.*, 2020b]. While the former are easier to integrate into deep neural network-based algorithms, they lack specificity, abstractness, robustness and interpretability [van Harmelen and ten Teije, 2019].

One type of prior knowledge that is hard to obtain for purely data-driven methods is commonsense knowledge. Equipping reinforcement learning agents with commonsense or world knowledge is an important step towards improved

human-machine interaction [Akat *et al.*, 2020], as interesting interactions demand machines to access prior knowledge not learnable from experience. Commonsense games [Jiang *et al.*, 2020; Murugesan *et al.*, 2021] have emerged as a testbed for methods that aim at integrating commonsense knowledge into a RL agent. Prior work has focused on augmenting the state by extracting subparts of ConceptNet [Murugesan *et al.*, 2021]. Performance only improved when the extracted knowledge was tailored to the environment. Here, we focus on knowledge that is automatically extracted and should be useful across a range of commonsense games.

Humans abstract away from specific objects using classes, which allows them to learn behaviour at class-level and generalise to unseen objects [Yee, 2019]. Since commonsense games deal with real-world entities, we look at the problem of leveraging subclass knowledge from open-source KGs to improve sample efficiency and generalisation of an agent in commonsense games. We use subclass knowledge to formulate a state abstraction, that aggregates states depending on which classes are present in a given state. This state abstraction might not preserve all information necessary to act optimally in a state. Therefore, a method is needed that learns to integrate useful knowledge over a sequence of more and more fine-grained state representations. We show how a naive ensemble approach can fail to correctly integrate information from imperfectly abstracted states, and design a residual learning approach that is forced to learn the difference between policies over adjacent abstraction levels. The properties of both approaches are first studied in a toy setting where the effectiveness of class-based abstractions can be controlled. We then show that if a commonsense game is governed by class structure, the agent is more sample efficient and generalises better to unseen objects, outperforming embedding approaches and methods augmenting the state with subparts of ConceptNet. However, learning might be hampered, if the extracted class knowledge aggregates objects incorrectly. To summarise, our key contributions are:

- we use the subclass relationship from open-source KGs to formulate a state abstraction for commonsense games;
- we propose a residual learning approach that can be integrated with policy gradient algorithms to leverage imperfect state abstractions;
- we show that in environments with class structure our

\*Contact Author

method leads to more sample efficient learning and better generalisation to unseen objects.

## 2 Related Work

We introduce the resources available to include commonsense knowledge and the attempts that have been made by prior work to leverage these resources. Since the class knowledge is extracted from knowledge graphs, work on including KGs in deep neural network based architectures is discussed. The setting considered here also offers a new perspective on state abstraction in reinforcement learning.

**Commonsense KGs.** KGs store facts in form of entity-relation-entity triplets. Often a KG is constructed to capture either a general area of knowledge such as commonsense [Ilievski *et al.*, 2021], or more domain specific knowledge like the medical domain [Huang *et al.*, 2017]. While ConceptNet [Speer *et al.*, 2017] tries to represent all commonsense knowledge, others focus on specific parts such as cause and effect relations [Sap *et al.*, 2019]. Manually designed KGs [Miller, 1995] are less error prone, but provide less coverage and are more costly to design, making hybrid approaches popular [Speer *et al.*, 2017]. Here, we focus on WordNet [Miller, 1995], ConceptNet and DBpedia [Lehmann *et al.*, 2015] and study how the quality of their class knowledge affects our method. Representing KGs in vector form can be achieved via knowledge graph embedding techniques [Nickel and Kiela, 2017], where embeddings can be trained from scratch or word embeddings are finetuned [Speer and Lowry-Duda, 2017]. Hyperbolic embeddings [Nickel and Kiela, 2017] capture the hierarchical structure of WordNet given by the hypernym relation between two nouns and are investigated as an alternative method to include class prior knowledge.

**Commonsense Games.** To study the problem of integrating commonsense knowledge into an RL agent, commonsense games have recently been introduced [Jiang *et al.*, 2020; Murugesan *et al.*, 2021]. Prior methods have focused on leveraging knowledge graph embeddings [Jiang *et al.*, 2020] or augmenting the state representation via an extracted subpart of ConceptNet [Murugesan *et al.*, 2021]. While knowledge graph embeddings improve performance more than GloVe word embeddings [Pennington *et al.*, 2014], the knowledge graph they are based on is privileged game information. Extracting task-relevant knowledge from ConceptNet automatically is challenging. If the knowledge is manually specified, sample efficiency improves but heuristic extraction rules hamper learning. No method that learns to extract useful knowledge exists yet. The class knowledge we use here is not tailored to the environment, and therefore should hold across a range of environments.

**Integrating KGs into deep learning architectures.** The problem of integrating knowledge present in a knowledge graph into a learning algorithm based on deep neural networks has mostly been studied by the natural language community [Xie and Pu, 2021]. Use-cases include, but are not limited to, open-dialog [Moon *et al.*, 2019], task-oriented dialogue [Gou *et al.*, 2021] and story completion [Zhang *et al.*, 2020b].

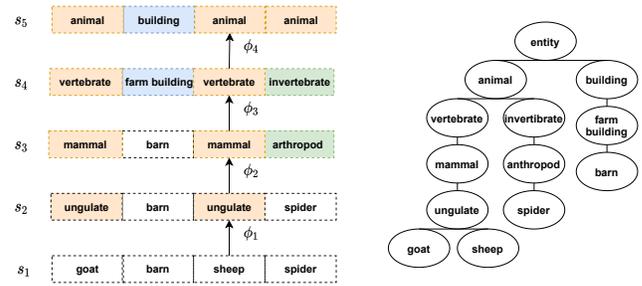


Figure 1: (Left) Visualisation of a state and its abstractions in Wordcraft based on classes extracted from WordNet. Colours indicate that an object is mapped to a class, where the same colour refers to the same class within an abstraction. (Right) Subgraph of the class tree that is used to determine the state abstractions.

Most methods are based on an attention mechanism over parts of the knowledge base [Moon *et al.*, 2019; Gou *et al.*, 2021]. Two key differences are that the knowledge graphs used are curated for the task, and therefore contain little noise. Additionally, most tasks are framed as supervised learning problems where annotation about correct reasoning patterns are given. Here, we extract knowledge from open-source knowledge graphs and have to deal with errors in the class structure due to problems with entity reconciliation and incompleteness of knowledge.

**State abstraction in RL.** State abstraction aims to partition the state space of a base Markov Decision Process (MDP) into abstract states to reduce the complexity of the state space on which a policy is learnt [Li *et al.*, 2006]. Different criteria for aggregating states have been proposed [Givan *et al.*, 2003]. They guarantee that an optimal policy learnt for the abstract MDP remains optimal for the base MDP. To leverage state abstraction an aggregation function has to be learned [Zhang *et al.*, 2020a], which either needs additional samples or is performed on-policy leading to a potential collapse of the aggregation function [Kemertas and Aumentado-Armstrong, 2021]. The case in which an approximate state abstraction is given as prior knowledge has not been looked at yet. The abstraction given here must not satisfy any consistency criteria and can consist of multiple abstraction levels. A method that is capable of integrating useful knowledge from each abstraction level is needed.

## 3 Problem Setting

Reinforcement learning enables us to learn optimal behaviour in an MDP  $M = (S, A, R, T, \gamma)$  with state space  $S$ , action space  $A$ , reward function  $R : S \times A \rightarrow \mathbb{R}$ , discount factor  $\gamma$  and transition function  $T : S \times A \rightarrow \Delta S$ , where  $\Delta S$  represents the set of probability distributions over the space  $S$ . The goal is to learn from experience a policy  $\pi : S \rightarrow \Delta A$  that optimises the objective:

$$J(\pi) = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] = \mathbb{E}_{\pi} [G], \quad (1)$$

where  $G$  is the discounted return. A state abstraction function  $\phi : S \rightarrow S'$  aggregates states into abstract states with

the goal to reduce the complexity of the state space. Given an arbitrary weighting function  $w : S \rightarrow [0, 1]$  s.t.  $\forall s' \in S', \sum_{s \in \phi^{-1}(s')} w(s) = 1$ , one can define an abstract reward function  $R'$  and transition function  $T'$  on the abstract state space  $S'$ :

$$R'(s', a) = \sum_{s \in \phi^{-1}(s')} w(s)R(s, a) \quad (2)$$

$$T'(\bar{s}'|s', a) = \sum_{\bar{s} \in \phi^{-1}(\bar{s}')} \sum_{s \in \phi^{-1}(s')} w(s)T(\bar{s}|s, a), \quad (3)$$

to obtain an abstract MDP  $M' = (S', A, R', T', \gamma)$ . If the abstraction  $\phi$  satisfies consistency criteria (see section 2, State abstraction in RL) a policy learned over  $M'$  allows for optimal behaviour in  $M$ , which we reference from now on as base MDP [Li *et al.*, 2006]. Here, we assume that we are given abstraction functions  $\phi_1, \dots, \phi_n$  with  $\phi_i : S_{i-1} \rightarrow S_i$ , where  $S_0$  corresponds to the state space of the base MDP. Since  $\phi_i$  must not necessarily satisfy any consistency criteria, learning a policy over one of the corresponding abstract MDPs  $M_i$  can result in a non-optimal policy. The goal is to learn a policy  $\pi$  or action value function  $Q$ , that takes as input a hierarchy of state abstractions  $s = (s_1, \dots, s_n)$ . Here, we want to make use of the more abstract states  $s_2, \dots, s_n$  for more sample efficient learning and better generalisation.

## 4 Methodology

The method can be separated into two components: (i) constructing the abstraction functions  $\phi_1, \dots, \phi_n$  from the subclass relationships in open source knowledge graphs; (ii) learning a policy over the hierarchy of abstract states  $s = (s_1, \dots, s_n)$  given the abstraction functions.

**Constructing the abstraction functions  $\phi_i$ .** A state in a commonsense game features real world entities and their relations, which can be modelled as a set, sequence or graph of entities. The idea is to replace each entity with its superclass, so that states that contain objects with the same superclass are aggregated into the same abstract state. Let  $E$  be the vocabulary of symbols that can appear in any of the abstract states  $s_i$ , i.e.  $s_i = \{e_1, \dots, e_k | e_l \in E\}$ . The symbols that refer to real-world objects are denoted by  $O \subseteq E$  and  $C_{tree}$  represents their class tree. A class tree is a rooted tree in which the leaves are objects and the parent of each node is its superclass. The root is a generic entity class of which every object/class is a subclass (see Appendix A for an example)<sup>1</sup>. To help define  $\phi_i$ , we introduce an entity based abstraction  $\phi_i^E : E \rightarrow E$ . Let  $C_k$  represent objects/classes with depth  $k$  in  $C_{tree}$  and  $L$  be the depth of  $C_{tree}$ , then we can define  $\phi_i^E$  and  $\phi_i$ :

$$\phi_i^E(e) = \begin{cases} \text{Pa}(e), & \text{if } e \in C_{L+1-i} \\ e, & \text{else,} \end{cases} \quad (4)$$

$$\phi_i(s) = \{\phi_i^E(e) | e \in s\}, \quad (5)$$

where  $\text{Pa}(e)$  are the parents of entity  $e$  in the class tree  $C_{tree}$ . This abstraction process is visualised in Figure 1. In practice,

<sup>1</sup>Supplementary material at <https://arxiv.org/pdf/2201.12126.pdf>.

we need to be able to extract the set of relevant objects from the game state and construct a class tree from open-source KGs. If the game state is not a set of entities but rather text, we use spaCy<sup>2</sup> to extract all nouns as the set of objects. The class tree is extracted from either DBpedia, ConceptNet or WordNet. For the detailed algorithms of each KG extraction, we refer to Appendix A. Here, we discuss some of the caveats that arise when extracting class trees from open-source KGs, and how to tackle them. Class tree can become imbalanced, i.e. the depths of the leaves, representing the objects, differs (Figure 2). As each additional layer with a small number of classes adds computational overhead but provides little abstraction, we collapse layers depending on their contribution towards abstraction (Figure 2). While in DBpedia or WordNet the found entities are mapped to a unique superclass, entities in ConceptNet are associated with multiple superclasses. To handle the case of multiple superclasses, each entity is mapped to the set of all  $i$ -step superclasses for  $i = 1, \dots, n$ . To obtain representations for these class sets, the embeddings of each element of the set are averaged.

**Learning policies over a hierarchy of abstract states.** Since prior methods in commonsense games are policy gradient-based, we will focus on this class of algorithms, while providing a similar analysis for value-based methods in Appendix B. First, we look at a naive method to learn a policy in our setting, discuss its potential weaknesses and then propose a novel gradient update to overcome these weaknesses.

A simple approach to learning a policy  $\pi$  over  $s$  is to take an ensemble approach by having a network with separate parameters for each abstraction level to predict logits, that are then summed up and converted via the softmax operator into a final policy  $\pi$ . Let  $s_{i,t}$  denote the abstract state on the  $i$ -th level at timestep  $t$ , then  $\pi$  is computed via:

$$\pi(a_t | s_t) = \text{Softmax} \left( \sum_{i=1}^n \text{NN}_{\theta_i}(s_{i,t}) \right), \quad (6)$$

where  $\text{NN}_{\theta_i}$  is a neural network processing the abstract state  $s_i$  parameterised by  $\theta_i$ . This policy can then be trained via any policy gradient algorithm [Schulman *et al.*, 2015; Mnih *et al.*, 2016; Espeholt *et al.*, 2018]. From here on, we will refer to this approach as sum-method.

There is no mechanism that forces the sum approach to learn on the most abstract level possible, potentially leading to worse generalisation to unseen objects. At train time making all predictions solely based on the lowest level (ignoring all higher levels) can be a solution that maximises discounted return, though it will not generalise well to unseen objects. To circumvent this problem, we adapt the policy gradient so that the parameters  $\theta_i$  at each abstraction level are optimised to approximate an optimal policy for the  $i$ -th abstraction level given the computed logits on abstraction level  $i - 1$ . Let  $s_{i,t}^n = (s_{i,t}, \dots, s_{n,t})$  denote the hierarchy of abstract states at timestep  $t$  down to the  $i$ -th level. Define the policy on the  $i$ -th level as

$$\pi_i(a | s_{i,t}^n) = \text{Softmax} \left( \sum_{k=i}^n \text{NN}_{\theta_k}(s_{k,t}) \right). \quad (7)$$

<sup>2</sup><https://spacy.io/>

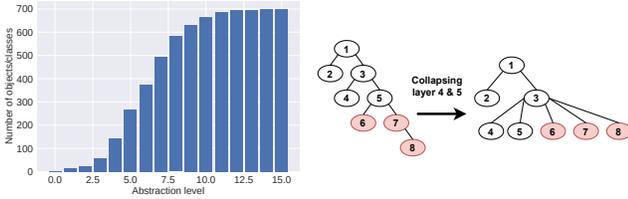


Figure 2: (Left) The number of different objects/classes that can appear for each state abstraction level in the Wordcraft environment with the superclass relation extracted from Wordnet. (Right) Visualisation of collapsing two layers in a class tree.

and notice that  $\pi = \pi_1$ . To obtain a policy gradient expression that contains the abstract policies  $\pi_i$ , we write  $\pi$  as a product of abstract policies:

$$\pi(a|s_t) = \left( \prod_{i=1}^{n-1} \frac{\pi_i(a|s_{i,t}^n)}{\pi_{i+1}(a|s_{i+1,t}^n)} \right) \pi_n(a|s_{n,t}). \quad (8)$$

and plug it into the policy gradient expression for an episodic task with discounted return  $G$ :

$$\nabla_{\theta} J(\theta) = \sum_{i=1}^n \mathbb{E}_{\pi} \left[ \sum_{t=1}^T \nabla_{\theta} \log \left( \frac{\pi_i(a|s_{i,t}^n)}{\pi_{i+1}(a|s_{i+1,t}^n)} \right) G \right], \quad (9)$$

where  $\pi_{n+1} \equiv 1$ . Notice that in Equation 9, the gradient of the parameters  $\theta_i$  depend on the values of all policies of the level equal or lower than  $i$ . The idea is to take the gradient for each abstraction level  $i$  only with respect to  $\theta_i$  and not the full set of parameters  $\theta$ . This optimises the parameters  $\theta_i$  not with respect to their effect on the overall policy, but their effect on the abstract policy on level  $i$ . The residual policy gradient is given by:

$$\nabla_{\theta} J_{res}(\theta) = \sum_{i=1}^n \mathbb{E}_{\pi} \left[ \sum_{t=1}^T \nabla_{\theta_i} \log(\pi_i(a|s_{i,t}^n)) G \right]. \quad (10)$$

Each element of the first sum resembles the policy gradient loss of a policy over the abstract state  $s_i$ . However, the sampled trajectories are from the overall policy  $\pi$  and not the policy  $\pi_i$  and the policy  $\pi_i$  inherits a bias from previous layers in form of logit-levels. We refer to the method based on the update in Equation 10 as residual approach. An advantage of the residual and sum approach is that the computation of the logits from each layer can be done in parallel. Any sequential processing of levels would have a prohibitively large computational overhead.

## 5 Experimental Evaluation

Our methodology is based on the assumption that abstraction via class knowledge extracted from open-source KGs is useful in commonsense game environments. This must not necessarily be true. It is first sensible to study the workings of our methodology in an idealised setting, where we control whether and how abstraction is useful for generalisation and sample efficiency. Then, we evaluate the method on two

commonsense games, namely a variant of Textworld Commonsense [Murugesan *et al.*, 2021] and Wordcraft<sup>3</sup>.

**Toy environment.** We start with a rooted tree, where each node is represented via a random embedding. The leaves of the tree represent the states of a base MDP. Each depth level of the tree represents one abstraction level. Every inner node of the tree represents an abstract state that aggregates the states of its children. The optimal action for each leaf (base state) is determined by first fixing an abstraction level  $l$  and randomly sampling one of five possible actions for each abstract state on that level. Then, the optimal action for a leaf is given by the sampled action for its corresponding abstract state on level  $l$ . The time horizon is one step, i.e. for each episode a leaf state is sampled, and if the agent chooses the correct action, it receives a reward of one. We test on new leaves with unseen random embeddings, but keep the same abstraction hierarchy and the same pattern of optimal actions. The sum and residual approach are compared to a policy trained only given the base states and a policy given only the optimal abstract states (oracle). To study the effect of noise in the abstraction, we replace the optimal action as determined by the optimal state with noise probability  $\sigma$  (noise setting) or ensure that the abstraction step only aggregates a single state (ambiguity setting). All policies are trained via REINFORCE [Williams, 1992] with a value function as baseline and entropy regularisation. More details on the chosen trees and the policy network architecture can be found in Appendix C.

**Textworld Commonsense.** In text-based games, the state and action space are given as text. In Textworld Commonsense (TWC), an agent is located in a household and has to put objects in their correct commonsense location, i.e. the location one would expect these objects to be in a normal household. The agent receives a reward of one for each object that is put into the correct location. The agent is evaluated by the achieved normalised reward and the number of steps needed to solve the environment, where 50 steps is the maximum number of steps allowed. To make abstraction necessary, we use a large number of objects per class and increase the number of games the agent sees during training from 5 to 90. This raises the exposure to different objects at training time. The agent is evaluated on a validation set where it encounters previously seen objects and a test set where it does not. The difficulty level of a game is determined by the number of rooms, the number of objects to move and the number of distractors (objects already in the correct location), which are here two, two and three respectively. To study the effect of inaccuracies in the extracted class trees from WordNet, ConceptNet and DBpedia, we compare them to a manual aggregation of objects based on their reward and transition behaviour. Murugesan *et al.* [2021] benchmark different architectures that have been proposed to solve text-based games [He *et al.*, 2016]. Here, we focus on their proposed method that makes use of a recurrent neural network architecture and numberbatch embeddings [Speer and Lowry-Duda, 2017], which performed best in terms of normalised reward

<sup>3</sup>Code can be found at <https://github.com/NikeHop/CSRL>.

and number of steps needed. For more details on the architecture and the learning algorithm, we refer to the initial paper. As baselines, we choose: (i) adding class information via hyperbolic embeddings trained on WordNet [Nickel and Kiela, 2017] by concatenating them to numberbatch embeddings of objects; (ii) extracting the *LocatedAt* relation from ConceptNet for game objects, encoding it via graph attention and combining it with the textual embedding [Murugesan *et al.*, 2021].

**Wordcraft.** An agent is given a goal entity and ingredient entities, and has to combine the ingredient entities to obtain the goal entity. A set of recipes determines which combination of input objects leads to which output entity. One can create different settings depending on the number of distractors (irrelevant entities in the set of recipe entities) and the set of recipes available at training time. Generalisation to unseen recipes guarantees that during training not all recipes are available, and generalisation to unseen goals ensures that at test time the goal entity has not been encountered before as a goal entity. Jiang *et al.* [2020] trained a policy using the IMPALA algorithm [Espeholt *et al.*, 2018]. We retain the training algorithm and the multi-head attention architecture [Vaswani *et al.*, 2017] used for the policy network to train our policy components  $NN_{\theta_1}, \dots, NN_{\theta_n}$ .

## 6 Results

After checking whether the results in the ideal environment are as expected, we discuss results in TWC and Wordcraft.

### 6.1 Toy Environment

From Figure 3 top, one can see that the sum and residual approach are both more sample efficient than the baseline without abstraction; on par with the sample efficiency of the oracle. The same holds for the generalisation performance. As the base method faces unseen random embeddings at test time, there is no possibility for generalisation. In the noise setting (Figure 3 middle), the policy trained on the abstract state reaches its performance limit at the percentage of states whose action is determined via their abstract state; the base, residual and sum method reach instead optimal performance. The achieved reward at generalisation time decreases for both the sum and residual method, but the sum approach suffers more from overfitting to the noise at training time when compared to the residual approach. In the case of ambiguous abstraction, we see that the residual approach outperforms the sum approach. This can be explained by the residual gradient update, which forces the residual method to learn everything on the most abstract level, while the sum approach distributes the contribution to the final policy over the abstraction layers. At test time, the sum approach puts too much emphasis on the uninformative base state causing the policy to take wrong actions.

### 6.2 Textworld Commonsense

The results suggest that both the sum and residual approach are able to leverage the abstract states. It remains an open question whether this transfers to more complex environments with a longer time-horizon, where the abstraction is derived by extracting knowledge from an open-source KG.

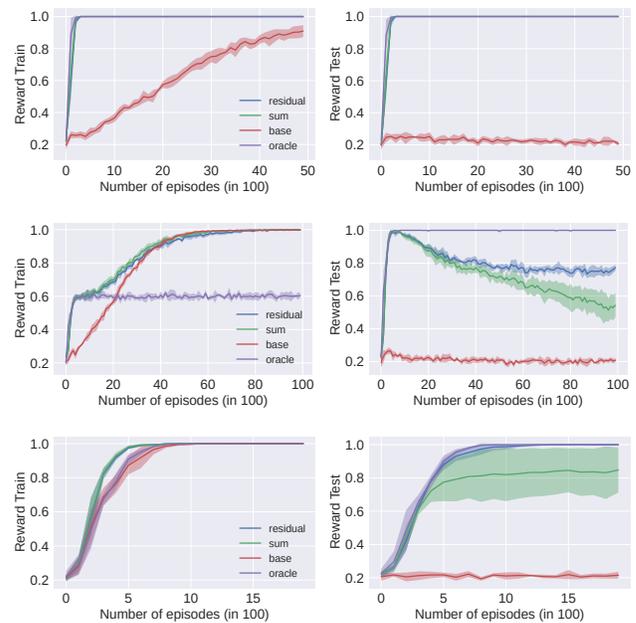


Figure 3: Training and generalisation results in the toy environment. The abstract state perfectly determines the action to take in the base state (top). Noise is added to this relation, so that in 50% of the time the optimal action is determined randomly beforehand (middle). In the ambiguous setting, not every abstract state has multiple substates (bottom). Experiments are run over five seeds.

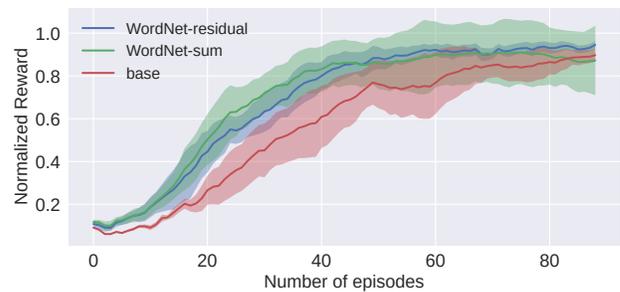


Figure 4: Training performance of the baseline (base), the sum and residual abstraction with knowledge extracted from WordNet in the difficult environment. Experiments are run over ten seeds.

**Sample efficiency.** From Figure 4, one can see that the residual and sum approach learn with less samples compared to the base approach. The peak mean difference is 20 episodes from a total of 100 training episodes. While the sum approach learns towards the beginning of training as efficiently, it collapses for some random seeds towards the end of training, resulting in higher volatility of the achieved normalised reward. This is similar to what was previously observed in experiments in the noisy setting of the ideal environment. That the performance of the sum approach collapses for certain seeds suggests, that the learned behaviour for more abstract states depends on random initialisation of weights. This stability issue is not present for the residual ap-

Method	Reward		Steps	
	Valid.	Test	Valid.	Test
Base	0.91 (0.04)	0.85 (0.05)	25.98 (3.14)	29.36 (3.42)
Base-H	0.83 (0.06)	0.75 (0.14)	30.59 (5.39)	33.92 (6.08)
Base-L	0.90 (0.04)	0.86 (0.06)	24.83 (2.31)	28.71 (3.46)
M-R	0.96 (0.03)	0.96 (0.02)	21.26 (2.65)	20.00 (1.57)
M-S	<b>0.97 (0.02)</b>	<b>0.96 (0.02)</b>	<b>20.95 (1.38)</b>	<b>19.55 (1.69)</b>
W-R	<b>0.93 (0.02)</b>	<b>0.94 (0.02)</b>	<b>23.25 (1.46)</b>	<b>24.04 (1.75)</b>
W-S	0.88 (0.10)	0.87 (0.17)	25.69 (4.78)	26.21 (6.81)

Table 1: Generalisation results for the easy and difficult level w.r.t the validation set and the test set, in terms of mean number of steps taken (standard deviation in brackets) for each type of knowledge graph (M=manual, W=WordNet) and each approach (R=residual, S=Sum). Base refers to the baseline with no abstraction, Base-H refers to baseline with hyperbolic embeddings and Base-L for baseline added *LocatedAt* relations. In bold, the best performing method without manually specified knowledge. We highlight in red the conditions in which the manual class graph outperforms the other methods. Experiments are run over ten seeds.

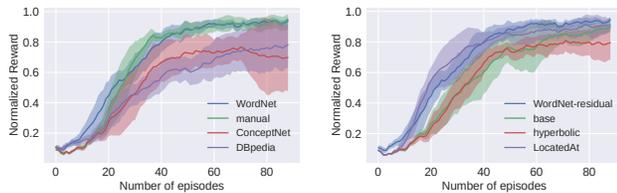


Figure 5: Training performance in the difficult environment using the residual approach when the class knowledge comes from different knowledge graphs trained (left) and when the baseline is augmented with hyperbolic embeddings or the *LocatedAt* relation from ConceptNet (right). Experiments are run over ten seeds.

proach. Figure 5 (right) shows that adding hyperbolic embeddings does not improve sample efficiency and even decreases performance towards the end of training. Adding the *LocatedAt* relations improves sample efficiency to a similar degree as adding class abstractions.

**Generalisation.** Table 1 shows that, without any additional knowledge the performance of the baseline, measured via mean normalised reward and mean number of steps, drops when faced with unseen objects at test time. Adding hyperbolic embeddings does not alleviate that problem but rather hampers generalisation performance on the validation and test set. When the *LocatedAt* relation from ConceptNet is added to the state representation the drop in performance is reduced marginally. Given a manually defined class abstraction, generalisation to unseen objects is possible without any drop in performance for the residual and the sum approach. This remains true for the residual approach when the manually defined class knowledge is exchanged with the WordNet class knowledge. The sum approach based on WordNet performs worse on the validation and training set due to the collapse of performance at training time.

**KG ablation.** From Figure 5 and Table 1 it is evident that for the residual approach, the noise and additional abstraction layers introduced by moving from the manually defined

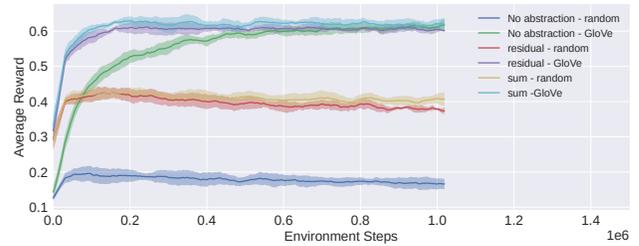


Figure 6: Generalisation results for the Wordcraft environment with respect to unseen goal entities. Experiments are run over ten seeds.

classes to the classes given by WordNet, only cause a small drop in generalisation performance and no additional sample inefficiency. The abstractions derived from DBpedia and ConceptNet hamper learning particularly for the residual approach (Figure 5). By investigating the DBpedia abstraction, we notice that many entities are not resolved correctly. Therefore objects with completely different semantics get aggregated. The poor performance from the ConceptNet abstraction hints at a problem with averaging class embeddings over multiple superclasses. Although two objects may have an overlap in their set of superclasses, the resulting embeddings could still differ heavily due to the non-overlapping classes.

### 6.3 Wordcraft

Figure 6 shows the generalisation performance in the Wordcraft environment. For the case of random embeddings, abstraction can help to improve generalisation results. Replacing random embeddings with GloVe embeddings improves generalisation beyond the class abstraction, and combining class abstraction with GloVe embeddings does not result in any additional benefit. Looking at the recipe structure in Wordcraft, objects are combined based on their semantic similarity, which can be better captured via word embeddings rather than through classes. Although no generalisation gains can be identified, adding abstraction in the presence of GloVe embeddings leads to improved sample efficiency. A more detailed analysis discussing the difference in generalisation between class abstraction and pretrained embeddings in the Wordcraft environment can be found in Appendix ??.

## 7 Conclusion

We show how class knowledge, extracted from open-source KGs, can be leveraged to learn behaviour for classes instead of individual objects in commonsense games. To force the RL agent to make use of class knowledge even in the presence of noise, we propose a novel residual policy gradient update based on an ensemble learning approach. If the extracted class structure approximates relevant classes in the environment, the sample efficiency and generalisation performance to unseen objects are improved. Future work could look at other settings where imperfect prior knowledge is given and whether this can be leveraged for learning. Finally, KGs contain more semantic information than classes, which future work could try to leverage for an RL agent in commonsense games.

## Acknowledgements

This research was (partially) funded by the Hybrid Intelligence Center, a 10-year programme funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, <https://hybrid-intelligence-centre.nl>.

## References

- [Akat *et al.*, 2020] Z. Akat, D. Balliet, M. de Rijke, and F. Dignum *et al.* A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence. *Computer*, 53(8):18–28, 2020.
- [Devlin *et al.*, 2019] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- [Espeholt *et al.*, 2018] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, and V. Mnih *et al.* Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, 2018.
- [Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003.
- [Gou *et al.*, 2021] Y. Gou, Y. Lei, L. Liu, Y. Dai, and C. Shen. Contextualize knowledge bases with transformer for end-to-end task-oriented dialogue systems. In *EMNLP*, 2021.
- [He *et al.*, 2016] J. He, J. Chen, X. He, J. Gao, and L. Li *et al.* Deep reinforcement learning with a natural language action space. In *ACL*, 2016.
- [Huang *et al.*, 2017] Z. Huang, J. Yang, F. v. Harmelen, and Q. Hu. Constructing knowledge graphs of depression. In *International Conference on Health Information Science*. Springer, 2017.
- [Ilievski *et al.*, 2021] F. Ilievski, P. A. Szekely, and B. Zhang. CSKG: the commonsense knowledge graph. In *ESWC*, 2021.
- [Jiang *et al.*, 2020] M. Jiang, J. Luketina, N. Nardelli, P. Minervini, and P. HS Torr *et al.* Wordcraft: An environment for benchmarking commonsense agents. *arXiv preprint arXiv:2007.09185*, 2020.
- [Kemertas and Aumentado-Armstrong, 2021] M. Kemertas and T. Aumentado-Armstrong. Towards robust bisimulation metric learning. *CoRR*, 2021.
- [Kirk *et al.*, 2021] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.
- [Lehmann *et al.*, 2015] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, and D. Kontokostas *et al.* Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 2015.
- [Li *et al.*, 2006] L. Li, T. J Walsh, and M. L Littman. Towards a unified theory of state abstraction for MDPs. *ISAIM*, 4:5, 2006.
- [Lillicrap *et al.*, 2016] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, and *et al.* Continuous control with deep reinforcement learning. In *ICLR*, 2016.
- [Miller, 1995] G. A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995.
- [Mnih *et al.*, 2016] V. Mnih, A. P. Badia, M. Mirza, A. Graves, and T. P. Lillicrap *et al.* Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.
- [Moon *et al.*, 2019] S. Moon, P. Shah, A. Kumar, and R. Subba. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *ACL*, 2019.
- [Murugesan *et al.*, 2021] K. Murugesan, M. Atzeni, P. Kapanipathi, P. Shukla, and S. Kumaravel *et al.* Text-based RL agents with commonsense knowledge: New challenges, environments and baselines. In *AAAI*, 2021.
- [Nickel and Kiela, 2017] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *NeurIPS*, 2017.
- [Pennington *et al.*, 2014] J. Pennington, R. Socher, and C. D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.
- [Sap *et al.*, 2019] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, and N. Lourie *et al.* ATOMIC: an atlas of machine commonsense for if-then reasoning. In *AAAI*, 2019.
- [Schulman *et al.*, 2015] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.
- [Silver *et al.*, 2016] D. Silver, A. Huang, C. J. Maddison, A. Guez, and L. Sifre *et al.* Mastering the game of go with deep neural networks and tree search. *Nat.*, 2016.
- [Speer and Lowry-Duda, 2017] R. Speer and J. Lowry-Duda. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. In *SemEval@ACL*, 2017.
- [Speer *et al.*, 2017] R. Speer, J. Chin, and C. Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, 2017.
- [Vaezipoor *et al.*, 2021] P. Vaezipoor, A. C. Li, R. T. Icarte, and S. A. McIlraith. Lt2action: Generalizing LTL instructions for multi-task RL. In *ICML*, 2021.
- [van Harmelen and ten Teije, 2019] F. van Harmelen and A. ten Teije. A boxology of design patterns for hybrid learning and reasoning systems. In *BNAIC*, 2019.
- [Vaswani *et al.*, 2017] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, and L. Jones *et al.* Attention is all you need. In *NeurIPS*, 2017.
- [Williams, 1992] R. J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.
- [Xie and Pu, 2021] Y. Xie and P. Pu. How commonsense knowledge helps with natural language tasks: A survey of recent resources and methodologies. *CoRR*, 2021.
- [Yee, 2019] Eiling Yee. Abstraction and concepts: when, how, where, what and why? *Language, Cognition and Neuroscience*, 34(10):1257–1265, 2019.
- [Zhang *et al.*, 2020a] A. Zhang, C. Lyle, S. Sodhani, A. Filos, and M. Kwiatkowska *et al.* Invariant causal prediction for block-MDPs. In *ICML*, 2020.
- [Zhang *et al.*, 2020b] M. Zhang, K. Ye, R. Hwa, and A. Kovashka. Story completion with explicit modeling of commonsense knowledge. In *CVPR*, 2020.