Pre-trained Word Embeddings for Goal-conditional Transfer Learning in Reinforcement Learning

Matthias Hutsebaut-Buysse[1] **Kevin Mets**[1] **Steven Latré**[1]

Abstract

Reinforcement learning (RL) algorithms typically start *tabula rasa*, without any prior knowledge of the environment, and without any prior skills. This however often leads to low sample efficiency, requiring a large amount of interaction with the environment. This is especially true in a lifelong learning setting, in which the agent needs to continually extend its capabilities. In this paper, we examine how a pre-trained task-independent word embedding can make a goal-conditional RL agent more sample efficient. We do this by facilitating transfer learning between different related tasks. We experimentally demonstrate our approach on a set of object navigation tasks.

1. Introduction

In order to build complex intelligent systems, an agent needs to be capable of re-using and adapting previously learned traits. This property is often called the *learning-to-learn* (Lake et al., 2017) ability of an agent.

This *learning-to-learn* approach is however in sharp contrast to how most RL approaches (Badia et al., 2020; Kapturowski et al., 2019; Schrittwieser et al., 2019) currently are capable of solving sequential decision-making problems. Current RL algorithms typically start *tabula rasa*, and do not re-use any knowledge previously learned in past tasks. These approaches are often very sample inefficient, requiring an unreasonable amount of interaction with the environment in order to learn new tasks.

A *learning-to-learn* approach could allow the agent to become more sample efficient, by allowing the agent to build upon what it already learned in past similar tasks. However, how to implement *learning-to-learn* in RL has remained mostly an open question. In supervised machine learning with neural networks, training performance on vision tasks can be significantly increased by re-using the initial layers of a previously trained neural network. These initial layers learn to recognize features which are mostly task-independent (Yosinski et al., 2014). Layers on top of these features learn to map combinations of the resulting features to the output labels.

Similar approaches have been used in RL (Taylor & Stone, 2009). Especially in *deep* RL, when working with high-dimensional inputs, it makes a lot of sense to re-use parts of the (learned) visual pipeline across different tasks (Chaplot et al., 2016).

However, mapping a high-dimensional input to a latent representation, is only part of the RL problem. In RL, the agent also needs to explore the environment in order to map actions to states. Such an action can consist of performing a single primitive action, such as *take one step forward*. However, exploration has been demonstrated (Jinnai et al., 2020; Eysenbach et al., 2019; Bacon et al., 2017) to be significantly faster when also utilizing temporal abstractions. These abstractions utilize multiple primitive actions, when exploring the environment (e.g. *walk to the garden*).

In our approach, we demonstrate that prior knowledge of a deep RL agent can be used as temporal abstractions in order to facilitate transfer learning to a novel previously unseen tasks. We do this by utilizing a goal-conditional agent. In this style, the RL agent receives a combination of the current state and a goal as its input. Assuming a finite set of possible goals, this goal is typically represented using a *one-hot* encoded vector. In this one-hot goal-space the distance has no meaning, as the distance between different goals is always the same.

We express the goal of the agent using natural language. We do this by using a task-independent pre-trained word embedding. This allows the agent to quickly link a new, previously unseen goal to what it has already learned from past tasks. We experimentally demonstrate that these kinds of pre-trained word goal-embeddings can be used to transfer knowledge in the form of temporal abstractions in a transfer learning settings.

[1] Department of Computer Science, University of Antwerp - imec. Correspondence to: Matthias Hutsebaut-Buysse <matthias.hutsebaut-buysse@uantwerpen.be>.

Accepted to the 1^{st} *Workshop on Language in Reinforcement Learning*, ICML 2020, Copyright 2020 by the author(s).

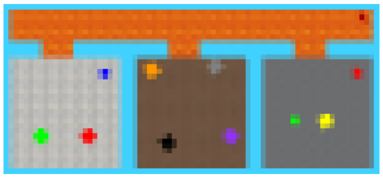

Figure 1. Top-down layout of the environment used in our experiments. The three rooms (bathroom, kitchen, bedroom) are connected through a long corridor.

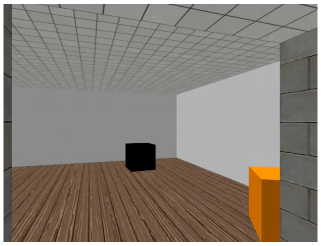

Figure 2. An example rendering of the viewpoint the agent receives as part of its state.

2. Related work

RL has been informed by natural language in various ways (Luketina et al., 2019). The majority of research has been conducted on how language instructions can be linked to actions (Chen & Mooney, 2011; Mei et al., 2016; Hermann et al., 2017). Additionally, language has been used as an instrument to communicate domain knowledge (Zhong et al., 2019), or assist by shaping the reward function (Bahdanau et al., 2019).

Similar to our work, natural language has also been used to transfer knowledge. For example in (Narasimhan et al., 2018) a method is proposed to transfer knowledge between different environments. In previous work (Hutsebaut-Buysse et al., 2020), we proposed a method to train a custom goal word embedding based on transfer performance.

3. Object navigation task setting

In this paper, we are concerned with the problem of object navigation. In a single instance of this problem, the agent is randomly spawned in a corridor, and needs to navigate towards an up-front specified object in the environment. The episode is considered successful if the agent has positioned itself near the goal object in a maximum of 500 steps.

In order to solve this problem, the agent does not have access to a map of the environment, and only needs to rely on RGB sensory input.

For our experiments, we use a custom designed level in the *MiniWorld* (Chevalier-Boisvert, 2018) benchmarking environment. Figure 1 shows the layout of the used environment. Figure 2 renders an example viewpoint of the agent.

Our designed level mimics a small domestic apartment. The layout consists of three rooms connected through a corridor. Each room contains a number of objects in fixed positions:

- **Bathroom**: shower, bathtub, toilet
- **Kitchen**: stove, toaster, table, microwave
- **Bedroom**: bed, wardrobe, nightstand

Objects are represented with spheres and cubes in different arbitrarily chosen colors. For example, in Figure 2, the black box represents the *table* object.

After taking an action, the agent receives a reward which is equal to the improvement of the distance to the goal object. A *slack* penalty of -0.01 is added to the reward, in order to force the agent to move. A bonus reward of 10 is awarded when reaching the goal object.

4. Method

4.1. Goal-encoding

In order for a RL agent to be capable of executing multiple tasks, the required task can be specified to the agent using a goal-vector. In our problem setting, this goal-vector should correspond with the object the agent needs to navigate to.

Typically, in order to encode different goals, a discrete *one-hot* encoding is used. Unfortunately, when using such a vector, the number of goals should be known in advance, as it is not straightforward to alter a neural network which depends on this vector.

However, in a lifelong learning setting (Silver et al., 2013) we would like the agent to be capable of learning to navigate to new goals, without having to explicitly define the number of goals in advance. In order to support this, we propose to encode goals using a pre-trained word embedding.

Such a model is trained (Mikolov et al., 2013) by taking as input a large corpus of texts, and outputs a vector space. Words that appear in similar contexts, are trained to be also close to each other in the output vector space. We reason that this prior knowledge can be of great use in a multi-task object navigation task, and that goals closer in word vector space, will also transfer better between different RL policies.

The pre-trained model we use (Honnibal & Montani, 2017) is trained on the OntoNotes 5 (Weischedel et al., 2013) dataset. This dataset contains a large set of different type of documents, and is not linked in any way with our task setting. The resulting model is capable of expressing a goal description with continuous vectors of size 300.

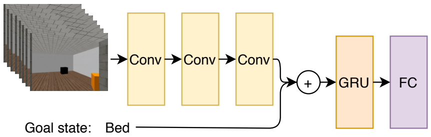

Figure 3. Goal-conditional architecture.

4.2. Training architecture

In order to allow our agent to solve object navigation tasks, we use a standard DRQN architecture (Hausknecht & Stone, 2015). We use the *recurrent* flavor (with sequence-length 8) of the DQN algorithm (Mnih et al., 2015), because the current state does not contain enough information for the agent to successfully navigate the environment. The goal-vector is concatenated with the visual perception part of our architecture. This architecture is displayed in Figure 3.

4.3. Transfer

In order for our lifelong learning agent to be capable of transferring knowledge from one task to another task, we propose to adapt the ϵ-greedy exploration scheme (Watkins, 1989). In this scheme, the agent takes a random action ϵ-percent of the time, instead of greedily following the current policy π. This allows the agent to explore (potentially better) actions, it would normally not take under the current policy. This ϵ value is typically decayed during training as the agent becomes more confident in its policy.

We propose to instead of purely taking random exploratory actions in order to navigate to a new goal (e.g. *bathtub*), to also explore actions which would correspond to the action the agent would take if it would be provided with a different goal-vector which the agent already has mastered before (e.g. *shower*).

However, how can the agent know which goal-vector will transfer best to satisfy the new unseen goal? We propose to solve this question by measuring the cosine similarity of the unseen goal object and the mastered goal objects in their word embedding space. As these embeddings are trained to put words which are often related to each other close to each other in the vector-space, we reason that goals close in this space will most commonly also be located in similar positions in typical building layouts.

Intuitively using knowledge from a prior object goal allows the agent to use this knowledge as a form of temporal abstraction, which corresponds to navigating to the room the object can most likely be found.

It however remains essential that the agent keeps doing enough exploration, especially in states close to the prior goal object. We propose to introduce a sampling rate hyper-parameter α in order to balance the trade-off between biased sampling from the prior policy, and random exploration.

In summary the policy of our agent when tasked with reaching goal z, word embedding $\mathcal{M}$ and prior goals $\omega_{0...i} \in \Omega$ looks as follows:

- $P(1-\epsilon)$: take greedy action $\pi(s_t, z)$
- $P(\epsilon * \alpha)$: sample action from $\pi(s, \omega)$ with $w = argmax_w(cos(M(z), M(w))$
- $P(\epsilon * (1-\alpha))$: take random action

5. Experiments and results

Experiments are terminated after reaching a success rate of 0.95 on the last 100.000 steps (and only minimal exploration $\epsilon = 0.01$ is done). In all experiments ϵ is linearly decayed over 1M steps, and we use an experience replay buffer of size 500.000.

5.1. Using language goal-vector vs one-hot goal-vector

In our first experiment, we examine the impact of the goal-vector on the training performance when training a goal-conditional agent on a set of four different goals.

The results of our experiments presented in Figure 4 give an indication that directly specifying the goal object using the word embedding ($\mathbb{R}^{300}$) has no significant negative effect over using a one-hot goal object encoding ($\mathbb{R}^{10}$). There also seems to be an interesting relation that using the word goal descriptions has a slightly positive effect of exploration, and using the one-hot encoding seems to work better when the policy is almost ($\epsilon = 0.01$) completely greedy (after 1M timesteps).

Using the goal word embedding for our lifelong learning agent is ideal, as we do not need to specify the amount of possible goal objects up front. The word goal embedding allows us to input a large amount of goals (the used model has 20k unique vectors).

5.2. Initial training on limited goal sets

We would like our lifelong agent to be capable of navigating to as many goals as possible. In order to do so, we could train our agent on a large set of goals. However, research has demonstrated (Narvekar et al., 2020) that using a carefully selected task curriculum often leads to better results.

We plotted the results of training our agent using different sizes of goal sets, in Figure 5. These results demonstrate that larger sets of goals are significantly harder to train. This finding supports our claim that a lifelong learning agent significantly benefits from first learning a small sub-set of goals, and gradually expanding its capabilities through transfer learning.

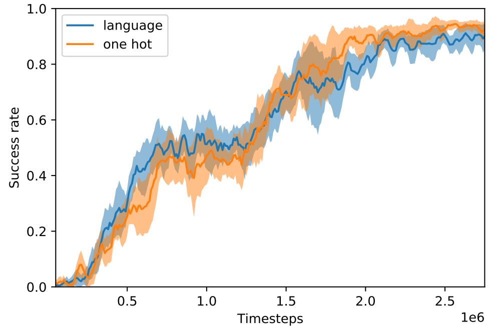

Figure 4. Comparing one-hot encoding vs language goal-vector on a set of 4 goals (results are averaged over 3 runs).

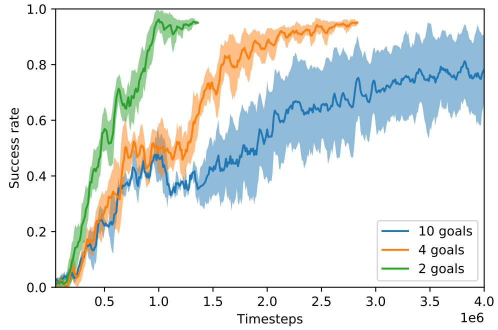

Figure 5. Comparing training performance on different sizes of goal object sets (results are averaged over 3 runs).

5.3. Transfer to new objects using prior policy

In our final experiment, we allow the agent to transfer knowledge from one goal object to a different unseen goal object using the transfer mechanism described in Section 4.3.

We start with a policy which has been trained to reliably reach four goal objects in the environment (*shower, toilet, bed* and *toaster*). In this experiment we test the transfer capability of our algorithm in order to learn to reach a new goal object *bathtub* using a prior sampling-rate of $\alpha = 0.2$. The new policy is randomly initialized.

Our preliminary results, plotted in Figure 7, demonstrate that the goal object that transfers best to the new unseen goal object (*bathtub*) is *shower*, which is also the goal object that is closest in language space (Figure 6). The performance when using the second closest goal (*toilet*) in language space also performs similarly.

Unrelated goals such as *bed* and *toaster* hinder the agent, steering the agent to the wrong room (*kitchen*) and we observe a negative transfer effect compared to just learning to navigate to the goal without any prior knowledge.

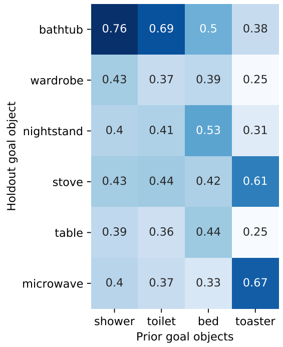

Figure 6. Cosine similarity of holdout goal objects and prior goal objects in the word embedding

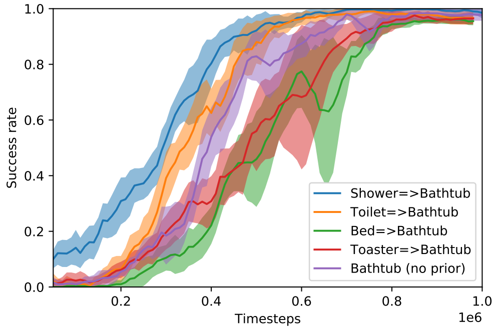

Figure 7. Comparison of using different prior goals in order to learn how to reach a new unseen goal object (*bathtub*) (results are averaged over 3 runs).

6. Conclusion

In this paper we presented our preliminary ideas on how natural language can assist a RL agent in a lifelong learning setting.

Our approach consists of training the agent on small sets of goals, directly inputting the goal descriptions in natural language. We utilize similarity of descriptions of seen and unseen goal objects in natural language in order to decide how to transfer existing knowledge to novel tasks. In order to transfer knowledge, we propose a simple, but effective transfer mechanism.

We support our method with preliminary results in a 3D simulated domestic environment. In future work we propose to further examine the impact of different language models, utilize more complex floor layouts, and we would like to study more complex prior goal selection schemes (e.g. use different prior goals weighted by their similarity with the current goal).

Acknowledgements

This research received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme.

References

Bacon, P.-L., Harb, J., and Precup, D. The Option-Critic Architecture. In *AAAI17*, 2017.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. Agent57: Outperforming the Atari Human Benchmark. 2020.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, A., Kohli, P., and Grefenstette, E. Learning to Understand Goal Specifications by Modelling Reward. In *ICLR19*, 2019.

Chaplot, D. S., Sathyendra, K. M., Lample, G., and Salakhutdinov, R. Transfer Deep Reinforcement Learning in 3D Environments: An Empirical Study. In *NIPS Deep Reinforcemente Leaning Workshop*, 2016.

Chen, D. L. and Mooney, R. J. Learning to Interpret Natural Language Navigation Instructions from Observation. In *AAAI11*, pp. 7, 2011.

Chevalier-Boisvert, M. gym-miniworld environment for openai gym. https://github.com/maximecb/gym-miniworld, 2018.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is All You Need: Learning Skills without a Reward Function. 2019.

Hausknecht, M. and Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. 2015.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., and Blunsom, P. Grounded Language Learning in a Simulated 3D World. 2017.

Honnibal, M. and Montani, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Hutsebaut-Buysse, M., Mets, K., and Latré, S. Fast Task-Adaptation for tasks labeled using Natural Language in Reinforcement Learning. In *ESANN2020*, 2020.

Jinnai, Y., Park, J. W., Machado, M. C., and Konidaris, G. Exploration in Reinforcement Learning with Deep Covering Options. In *ICLR2020*, pp. 13, 2020.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent Experience Replay in Distributed Reinforcement Learning. 2019.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017. ISSN 0140-525X, 1469-1825. doi: 10.1017/S0140525X16001837.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A Survey of Reinforcement Learning Informed by Natural Language. In *IJCAI19*, 2019.

Mei, H., Bansal, M., and Walter, M. R. Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences. In *AAAI16*, 2016.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. 518(7540):529–533, 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236.

Narasimhan, K., Barzilay, R., and Jaakkola, T. Grounding Language for Transfer in Deep Reinforcement Learning. 63, 2018.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. 2020.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. 2019.

Silver, D. L., Yang, Q., and Li, L. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *AAAI13*, 2013.

Taylor, M. E. and Stone, P. Transfer Learning for Reinforcement Learning Domains: A Survey. 10, 2009.

Watkins, C. J. C. H. Learning from delayed rewards. 1989.

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., El-Bachouti, M., Belvin, R., and Houston, A. OntoNotes: A Large Training Corpus for Enhanced Processing. 2013.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *NIPS14*, 2014.

Zhong, V., Rocktäschel, T., and Grefenstette, E. RTFM: Generalising to Novel Environment Dynamics via Reading. In *ICLR20*, 2019.