Self-Educated Language Agent with Hindsight Experience Replay for Instruction Following

Geoffrey Cideron, Mathieu Seurin
Univ. Lille, CNRS, Inria, UMR 9189 CRIStAL
{geoffrey.cideron,mathieu.seurin}@inria.fr

Florian Strub
DeepMind
fstrub@google.com

Olivier Pietquin
GoogleBrain
pietquin@google.com

Abstract

Language creates a compact representation of the world and allows the description of unlimited situations and objectives through compositionality. These properties make it a logical fit to guide the training of interactive agents as it could ease recurrent challenges in Reinforcement Learning such as sample-complexity, generalization, or multi-tasking. However, it remains an open-problem to relate language and RL in even simple instruction following scenarios. Current methods rely on expert demonstrations, auxiliary losses, or inductive biases in neural architectures. In this paper, we propose an orthogonal approach called Textual Hindsight Experience Replay (THER). THER learns an instruction generator without human demonstrations to replace goals in failed trajectories as in hindsight experience replay. We observe that this simple idea initiate a learning synergy between language acquisition and policy learning on instruction following tasks in the BabyAI environment.

1 Introduction

Our language has slowly evolved to communicate our intents, to state our objectives, or to describe complex situations [13]. It conveys information compactly by relying on composition and highlighting salient facts. Such properties are essential when developing interactive agents in complex environments. As language may express a vast diversity of goals and situations, it could alleviate the training of interactive agents over heterogeneous and composite tasks thanks to its intrinsic structure [16]. This property is all the more critical as classic Reinforcement Learning (RL) methods are facing generalization issues [8], and learning hierarchical and structured policies remains an open-problem [3, 14]. As recently advocated by Luketina et al. [16], language should thus be considered a first-class citizen to ease RL problem to improve generalization and sample efficiency.

Unfortunately, conditioning a policy on language also entails a supplementary difficulty as the agent needs to understand linguistic cues to alter its behavior. The agent thus needs to ground its language understanding by relating the words to its observations, actions, and rewards before being able to leverage the language structure [11, 10]. Once the linguistic symbols are grounded, the agent may then take advantage of language compositionality to condition its policy on new goals. It thus leads to the following questions: are we eventually making the reinforcement learning problem harder, or can we generate learning synergies between policy learning and language acquisition?

In this work, we use instruction following [16, 17, 25, 23, 18] as a natural testbed to examine this question. In this setting, the agent is given a textual description of its goal (e.g. "pick the red ball") and is rewarded when achieving it. The agent has thus to visually ground the language, i.e., linking and disentangling visual attributes (*shape*, *color*) from language description ("ball", "red") by using rewards to condition its policy toward task completion. On one side, the language compositionality allows for a high number of goals, and offer generalization opportunities; but on the other side, it dramatically complexifies the policy search space. Besides, instruction following is a notoriously hard RL problem since it has a sparse reward signal. In practice, the navigation and language grounding problems are often circumvented by warm-starting the policy with labeled trajectories [25, 1]. Although scalable, these approaches require numerous human demonstrations.Here, we want to jointly learn the navigation policy and language understanding from scratch without any human supervision. In a seminal work, Hermann et al. [10] successfully ground language instructions, but the authors had to use unsupervised losses and heavy curriculum to handle the sparse reward challenge.

In this paper, we take advantage of language compositionality to tackle the lack of reward signals. To do so, we extend Hindsight Experience Replay (HER) to language goals [2]. HER deals with the sparse reward problems in spatial scenario. It transforms unsuccessful trajectories into a successful one by redefining the policy goal *a posteriori*. It creates more episodes with positive rewards and a more diverse set of goals. However, it requires a mapping between the agent trajectory and the goal to substitute, which is not available when dealing with language tasks as pointed out by Chan et al. [5]. In this work, we introduce Textual Hindsight Experience Replay (THER), a training procedure where the agent jointly learns the language-goal mapping and the navigation policy by solely interacting with the environment. THER then tackles the sparse reward problem by relabeling language goals upon unsuccessful trajectories in a HER fashion. We evaluate our methods on the BabyAI world [6], showing clear improvement over RL baselines and highlighting the robustness of THER to noise.

2 Background And Notation

In reinforcement learning, an agent interacts with the environment to maximize its cumulative reward [22]. At each time step t, the agent is in a state $s_t \in \mathcal{S}$, where it selects an action $a_t \in \mathcal{A}$ according its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. It then receives a reward r_t from the environment's reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and moves to the next state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$. The quality of the policy is assessed by the Q-function defined by $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_t \gamma^t r(s_t,a_t)|s_0 = s, a_0 = a\right]$ for all (s,a) where $\gamma \in [0,1]$ is the discount factor. We define the optimal Q-value as $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, from which the optimal policy π^* is derived.

We here use Deep Q-learning (DQN) to approximate the optimal Q-function with neural networks and perform off-policy updates by sampling the transition (s_t, a_t, r_t, s_{t+1}) from a replay buffer [19].

In this article, we augment our environment with a goal space $\mathcal{G}$ which define a new reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$ and policy $\pi : \mathcal{S} \times \mathcal{G} \rightarrow \mathcal{A}$ by conditioning them on a goal descriptor $g \in \mathcal{G}$. Similarly, the Q-function is also conditioned on the goal, and it is referred to as Universal Value Function Approximator (UFVA) [20]. This approach allows learning holistic policies that generalize over goals in addition to states at the expense of complexifying the training process. In this paper, we explore how language can be used for structuring the goal space, and how it language composition ease generalization over unseen scenario in a UFVA setting.

Hindsight Experience Replay (HER) [2] is designed to increase the sample efficiency of off-policy RL algorithms such as DQN in the goal-conditioning setting. It reduces the sparse reward problem by taking advantage of failed trajectories, relabelling them with new goals. An expert then assigns the goal that was achieved by the agent when performing its trajectory.

Formally, HER assumes the existence of a predicate $f : \mathcal{S} \times \mathcal{G} \rightarrow \{0,1\}$ which encode whether the agent in a state s which satisfies the goal $f(s,g) = 1$. At any time step t, $r(s_t, a, g) = f(s_{t+1}, g)$

A goal g is drawn from the space $\mathcal{G}$ of goals at the beginning of an episode. At each time step t, the transition $(s_t, a_t, r_t, s_{t+1}, g)$ is stored in the DQN replay buffer, and at the end of an unsuccessful episode, an expert provides an additional goal g' that match the trajectory. New transitions $(s_t, a_t, r'_t, s_{t+1}, g')$ are thus added to the replay buffer for each time step t, where $r' = r(s_t, a_t, s_{t+1}, g')$. Finally, DQN is updated normally by sampling the replay buffer.

HER assumes that a mapping m between states s and goals g is given. In the original paper [2], this requirement is not restrictive as the goal space is a subset of the state space. Thus, the mapping m is straightforward since any state along the trajectory can be used as a substitution goal. In the general case, the goal space differs from the state space, and the mapping function is generally unknown. In the instruction following setting, there is no obvious mapping from visual states to linguistic cues. It thus requires expert intervention to provide a new language goal given the trajectory, which drastically reduces the interest of HER. Therefore, we here explore how to learn this mapping without any form of expert knowledge nor supervision.

3 Textual Hindsight Experience Replay

Textual Hindsight Experience Replay (THER) aims to learn a mapping that relates a trajectory to a potential goal in order to apply HER. The mapping function then relabels unsuccessful trajectories by predicting a substitute goal $\hat{g}$ as an expert would do, before appending it to the replay buffer. This mapping learning is performed alongside agent policy training.

Besides, we wish to discard any form of expert supervision to learn this mapping as it would reduce the practicability of the approach. Therefore, the core idea is to use environment signals to retrieve training mapping pairs. Instinctively, in the sparse reward setting, trajectories with positive reward encode ground-truth mapping pairs, while trajectories with negative rewards are negative samples. These cues are thus collected to train the mapping function for THER in a supervised fashion. We emphasize that such signals are inherent to the environment, and an external expert does not provide them. In the following, we only keep positive pairs in order to train a discriminative mapping model.

Formally, THER is composed of a dataset D of $\langle s, g \rangle$ pairs, a replay buffer R and a parametrized mapping model $m_{\boldsymbol{w}}$. For each episode, a goal g is picked, and the agent generates transitions $(s_t, a_t, r_t, s_{t+1}, g)$ that are appended to the replay buffer R. The Q-function parameters are updated with an off-policy algorithms by sampling minibatches from D. Upon episode termination, if the goal is achieved, i.e. $f(s_T, g) = 1$, the $\langle s_T, g \rangle$ pair is appended to the dataset D. If the goal is not achieved, a substitute goal is sampled from the mapping model[1] $m_{\boldsymbol{w}}(s_T) = \hat{g}'$ and the additional transitions $\{(s_t, a_t, r_t, s_{t+1}, \hat{g}')\}_{t=0}^{T}$ are added to the replay buffer. At regular intervals, the mapping model $m_{\boldsymbol{w}}$ is optimized to predict the goal g given the trajectory τ by sampling mini-batches from D. algorithm 1 summarizes our approach. Noticeably, THER can be extended to partially observable environments by replacing the predicate function $f(s, g)$ by $f(\tau, g)$, i.e., the completion of a goal depends on the full trajectory rather than one state.

4 Experiments

Using THER in instruction following THER assumes that learning to ground language is more sample-efficient than learning jointly to ground language and navigating via vanilla reinforcement learning. The learned mapping then would alleviate the task's dual difficulty by densifying rewards and generating a diverse set of goals. We are testing this hypothesis in the context of instruction following [16] as it requires the agent to ground attributes to their visual aspect, and to navigate towards the objective.

Models In this experiment, THER is composed of two separate models. The instruction generator is a neural network outputting a sequence of words given the final state of a trajectory. It is trained by gradient descent using a cross-entropy loss on the dataset D collected as described in section 3. We train a DQN network following [19] with a dueling head [24], double q-learning [9], and a uniform replay buffer. The network receives a tuple $< s, g >$ as input and output an action corresponding to the argmax over states-actions values $Q(s, a)$. We use ϵ-greedy exploration with decaying epsilon (More details at Appendix A).

Environments We experiment our approach on a grid world environment called MiniGrid [6]. This environment offers a variety of instruction-following tasks using a synthetic language for grounded language learning. We use a 10x10 grid with 10 objects randomly located in the room. Each object has 4 attributes (shade, size, color, and type) inducing a total of 300 different objects. The agent has four actions {forward, left, right, pick}, and it can only see the 7x7 grid in front of it. For each episode, one object's attributes are randomly picked as a goal, and the text generator translates it in synthetic language as detailed in Appendix B, e.g., "Go get a tiny light blue ball." The agent is then

[1]The mapping model can be utilized upon an accuracy criterion is reached to avoid random goal sampling.

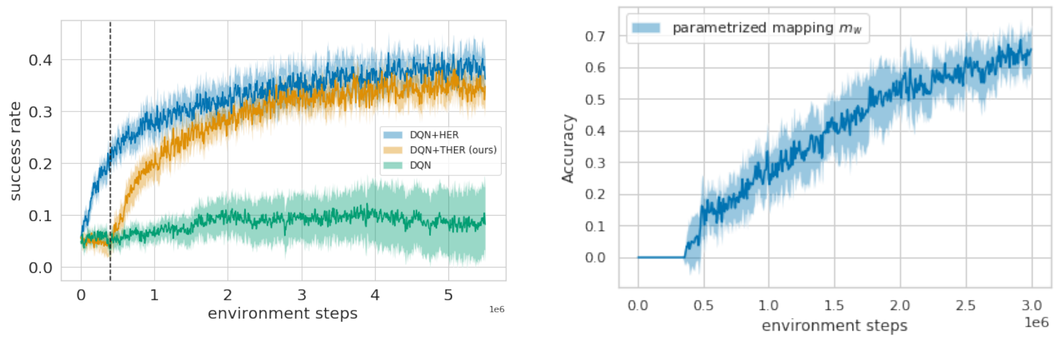

Figure 1: **Left**: learning curves for DQN, DQN+HER, DQN+THER in a 10x10 gridworld with 10 objects each composed with 4 attributes. The instruction generator is used after the vertical bar. **Right**: the mapping accuracy for the prediction of instructions. $m_{\boldsymbol{w}}$ starts being trained after collecting 1000 positive trajectories. The results are averaged over 5 seeds and the shaded area corresponds to one standard deviation.

rewarded when picking one object matching the goal description, which ends the episode; otherwise, the episode terminates after 40 steps or after picking an incorrect object.

Baselines We want to asses if the agent benefits from learning an instruction generator and use it to substitute goal as done in HER. We denote this approach DQN+THER. We compare our approach to DQN without goal substitution (called DQN) and DQN with goal substitution from a perfect mapping provided by an external expert (called DQN+HER). Therefore, DQN is the expected lower bound as it receives the least amount of rewards, whereas DQN+HER is the upper bound as the learned mapping can not outperform the expert. Note that we only start using the parametrized mapping function after collecting 1000 positive trajectories.

Results In Figure 1 (left), we show the success rate of the benchmarked algorithms per environment steps. We first observe that DQN does not manage to learn a good policy, and its performance remains close to a random policy. On the other side, DQN+HER quickly manages to pick the correct object 35% of the time. Finally, DQN+THER success rates start increasing as soon as we use the mapping function, to rapidly perform nearly as well as DQN+HER. Figure 1 (right) shows the performance accuracy of the mapping generator by environment steps (We use a parser to assess if all four attributes are correct). We observe a steady improvement of the accuracy during training before reaching 65% accuracy after 3M steps.

Discussion We first note that even when the mapping accuracy is 20%, the policy success rate starts increasing, and DQN+THER becomes nearly as good as DQN+HER despite having a maximum mapping accuracy of 65%. This observation suggests that HER is robust to noisy attributes. As the instruction contains four attributes, one may argue that one attribute is at least correct $20\%^{1/4} = 67\%$[2] of the time given a mapping accuracy of 20%, which would ease the training. However, the agent does not know which attributes are correct, and yet, it learns to ground language attributes by disentangling noisy goals, which fails without THER.

Secondly, we observe that the number of positive trajectories needed to learn a non-random mapping $m_{\boldsymbol{w}}$ is lower than the number of positive trajectories needed to obtain a valid policy with DQN (even after 3M environment steps the policy is close to random). This suggests that it is easier to learn the linguistic mapping function than learning the navigation policy.

As a result, there is a virtuous circle that arises: as soon as the mapping is correct, the agent success rate increases, providing additional ground-truth mapping pairs, increasing the mapping accuracy, increasing the quality of substitute goals, which increase policy accuracy. As a result, there is a natural synergy that occurs between language grounding and navigation policy.

[2]if we assume that the error is uniform over the attributes

5 Conclusion

We introduce Textual Hindsight Experience Replay (THER) as an extension to HER for language. We define a protocol to learn a mapping function to relabel unsuccessful trajectories with predicted consistent language instructions. We show that THER nearly matches HER performances despite only relying on signals from the environment. We provide empirical evidence that THER manages to alleviate the instruction following task by jointly learning language grounding and navigation policy with training synergies. THER has light underlying assumptions, it is complementary to other instruction following methods, and it does not require human data; making it valuable for large-scale experiments.

References

[1] Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proc. of CVPR*.

[2] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Proc. of NIPS*.

[3] Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77.

[4] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. of NIPS*.

[5] Chan, H., Wu, Y., Kiros, J., Fidler, S., and Ba, J. (2018). Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *1st Workshop on Goal Specifications for Reinforcement Learning, Workshop held jointly at ICML, IJCAI, AAMAS*.

[6] Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). BabyAI: First steps towards grounded language learning with a human in the loop. In *Proc. of ICLR*.

[7] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc. of NIPS Workshop on Deep Learning*.

[8] Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2018). Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*.

[9] Hasselt, H. v., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proc. of AAAI*.

[10] 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. (2017). Grounded language learning in a simulated 3d world.

[11] Kiela, D., Bulat, L., Vero, A. L., and Clark, S. (2016). Virtual embodiment: A scalable long-term strategy for artificial intelligence research. *arXiv preprint arXiv:1610.07432*.

[12] Kingma, D. P. and Ba, J. L. (2015). Adam: Amethod for stochastic optimization. In *Proc. of ICLR*.

[13] Kirby, S., Tamariz, M., Cornish, H., and Smith, K. (2015). Compression and communication in the cultural evolution of linguistic structure. *Cognition*, 141:87–102.

[14] Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proc. of NIPS*.

[15] LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*.

[16] Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language. In *Proc. of IJCAI*.

[17] Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. (2018). Mapping instructions to actions in 3d environments with visual goal prediction. In *Proc. of EMNLP*, pages 2667–2678.

[18] Misra, D., Langford, J., and Artzi, Y. (2017). Mapping instructions and visual observations to actions with reinforcement learning. In *Proc. of EMNLP*, pages 1004–1015.

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

[20] Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *Proc. of ICML*.

[21] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

[22] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

[23] Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.-F., Wang, W. Y., and Zhang, L. (2019). Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proc. of CVPR*.

[24] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proc. of ICML*.

[25] Zang, X., Pokle, A., Vázquez, M., Chen, K., Niebles, J. C., Soto, A., and Savarese, S. (2018). Translating navigation instructions in natural language to a high-level plan for behavioral robot navigation. In *Proc. of EMNLP*.

A Appendix A: Training details

A.1 Instruction Generator architecture

The encoder is a convolution neural network [15] with ReLU activation functions after each layer for processing an observation of dimension 7x7x3. It is composed of: 16 of 2x2 kernel, followed a max pooling 2D of size 2x2 then 32 of 2x2 kernel, and finally 256 of 2x2 kernel. For the convolutional layers and the pooling layer, the stride is always equal to 1.

The decoder is a recurrent neural network composed of gated recurrent units (GRU) [7] that outputs the instruction word by word. The convolutional layers extract relevant information from the image and compress it in a latent representation. This latent representation is then used as the initial hidden state of a GRU layer. The initial input of the GRU layer is the token *BEGIN*. At each time step, the GRU layer outputs a distribution over words as in Seq2Seq [21], we can use as the input of the next step the word the is the most likely. When the token *END* is chosen the prediction stops. We use a word embedding of dimension 128 and an instruction embedding of dimension 256. The number of words in the dictionary is equal to 27.

The training is done with cross-entropy between the distribution of probabilities over words predicted by the model and the true word. A technique called teacher forcing is employed to accelerate and to stabilize the learning. Teacher forcing refers to using the ground truth for the next input of the GRU instead of using the last predicted word [4].

The learning rate is fixed to 10^{-4} and the network is trained using the Adam optimizer with default parameters [12] and a regularization of 10^{-6} over all parameters.

A.2 DQN Architecture

To deal with the partial observability of the environment (described in section 4) the state corresponds to the last 4 frames stacked as in [19].

Visual Encoder: Each frame is encoded by a convolutional neural network and then passed through a LSTM. Each layer is followed by a ReLU activation function. The convolutional neural network is composed of: 16 of 2x2 kernel, followed a max pooling 2D of size 2x2 then 32 of 2x2 kernel, and finally 64 of 2x2 kernel. For the convolutional layers and the pooling layer, the stride is always equal to 1. The LSTM has an input and hidden size of 64. The last LSTM hidden state h_t corresponds to visual embedding.

Instruction Encoding: First each word is embedded with an embedding of size 32. Instructions are encoded word by word using a GRU. The GRU has an input size of 32 which corresponds to the word embedding size and a hidden size of 128 which corresponds to the instruction embedding size. The last GRU hidden state h_t is kept as the instruction embedding.

We concatenate the visual embedding and the instruction embedding and add fully connected layers in the same fashion as in the dueling architecture [24] on top to compute the Q-values for each action.

B Appendix B: Environment details

A screenshot of the environment is provided in Figure 2.

The state of the environment does not correspond to a RBG image but to channels encoding different info about each cell (color, type etc...). More details are available at gym-minigrid.

The synthetic language used for instructions is composed in three parts. First, a clause like *get a* (all the clauses are displayed below). Then, a series of attributes randomly ordered describing the object and lastly, the type of the object. An example of instruction is *Go fetch a tiny dark red ball*. One object can be described by a maximum of 4 attributes: shade, size, color, and type. They have modalities going from 2 to 6.

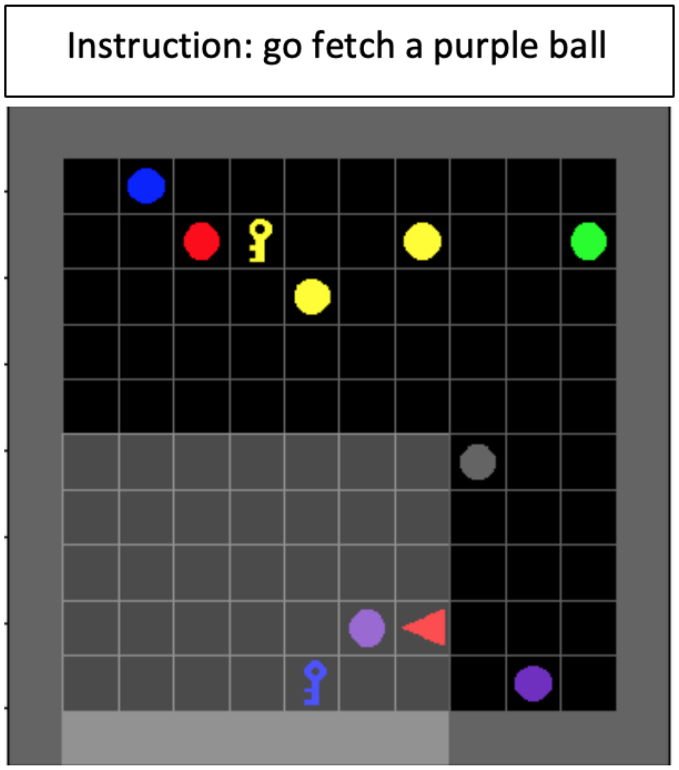

Figure 2: A Screenshot of the environment. The agent only sees the light gray area. It should be noted that in is this example, object only have 2 attributes (color and type) but in our experiments, 4 attributes are being used.

The environment used is a grid-world of variable size containing objects. Objects can be described using up to 4 attributes. The list of all attributes is the following:

- **Shapes** *key*, *ball*
- **Colors** *red*, *green*, *blue*, *purple*, *yellow*, *grey*
- **Shades** *very_dark*, *dark*, *neutral*, *light*, *very_light*
- **Sizes** *tiny*, *small*, *medium*, *large*, *giant*

The five possible clauses are:

- *get a*
- *go fetch a*
- *go get a*
- *fetch a*
- *you must fetch a*

C THER Algorithm detailed

Algorithm 1: Textual Hindsight Experience Replay (THER)

```
Given:
    • an off-policy RL algorithm (e.g. DQN) $\mathbb{A}$
    • a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$.
Initialize $\mathbb{A}$ , replay buffer $R$, dataset $D$ of $\langle instruction, state \rangle$, Instruction Generator $m_{\boldsymbol{w}}$;
for episode=1,M do
    Sample a goal $g$ and an initial state $s_0$;
    $t = 0$;
    repeat
        Execute an action $a_t$ chosen from the behavioral policy $\mathbb{A}$: $a_t \leftarrow \pi(s_t||g)$;
        Observe a reward $r_t = r(s_t, a_t, g)$ and a new state $s_{t+1}$;
        Store the transition $(s_t, a_t, r_t, s_{t+1}, g)$ in $R$;
        Update Q-network parameters using the policy $\mathbb{A}$ and sampled minibatches from $R$;
        $t = t + 1$;
    until episode ends;
    if $f(s_t, g) = 1$ then
        Store the pair $\langle s_t, g \rangle$ in $D$;
        Update $m$-network parameters by sampling minibatches from $D$;
    end
    else
        if $m$ accuracy is high enough then
            Sample $\hat{g}' = m_{\boldsymbol{w}}(s_t)$;
            Replace $g$ by $\hat{g}'$ in the transitions of the last episode and set $\hat{r} = r(s_t, a_t, \hat{g}')$.
        end
    end
end
```