Ask Before You Act

Generalising to Novel Environments by Asking Questions

Ross Murphy
ross.murphy.20
UCL ID 20102931

Sergey Mosesov
sergey.mosesov.13
UCL ID 20102931

Javier Leguina Peral
javier.peral.20
UCL ID 20150126

Thymo ter Doest
thymo.ter.doest.20
UCL ID 20102931

Abstract

Solving temporally-extended tasks is a challenge for most reinforcement learning (RL) algorithms [Jiang et al., 2019]. We investigate the ability of an RL agent to learn to ask natural language questions as a tool to understand its environment and achieve greater generalisation performance in novel, temporally-extended environments. We do this by endowing this agent with the ability of asking "yes-no" questions to an all-knowing Oracle. This allows the agent to obtain guidance regarding the task at hand, while limiting the access to new information. To study the emergence of such natural language questions in the context of temporally-extended tasks we first train our agent in a MiniGrid environment. We then transfer the trained agent to a different, harder environment. We observe a significant increase in generalisation performance compared to a baseline agent unable to ask questions. Through grounding its understanding of natural language in its environment, the agent can reason about the dynamics of its environment to the point that it can ask new, relevant questions when deployed in a novel environment.

1 Introduction

Deep Reinforcement Learning is a powerful framework for sequential decision making that has propelled agents to surpass human performance in many settings [van Hasselt et al., 2015, Stadie et al., 2015]. However, these methods suffer from many issues, including low sample efficiency, credit-assignment problem and struggle with hierarchical temporally-extended tasks [Minsky, 1961, Sutton and Barto, 2018]. Humans on the other hand, are able to quickly generalise to new environments by abstracting the structural similarities among various tasks [Spelke and Kinzler, 2007]. This indicates humans learn in a different manner to machines [Tsividis et al., 2017]. This generalisation capability of humans can be attributed to their possessing a set of priors about the world they inhabit, and these priors can be grounded in natural language. [Spelke and Kinzler, 2007, Luketina et al., 2019].

Furthermore, it has been hypothesised that, due to the need to encode information in the hierarchical and logical structure imposed by syntactically correct forms of expression, formal and natural languages allow us to encode abstractions and generalise [Gopnik and Meltzoff, 1987]. We extend this reasoning and we question whether jointly learning to navigate an environment together with the language required to express your knowledge about it leads to a deeper understanding of the world. In other words, by forcing an agent to encode its knowledge of the world in a way amenable to expression through natural language, we argue that such generalisation capabilities will emerge naturally.

A wide body of research exists which seeks to improve generalisation in RL using natural language. Examples abound of approaches which jointly train an agents on a language goal and an environment, where the overarching concept is that the agent learns to encode information on the structure of the world from natural language corpora (see [Luketina et al., 2019] and references therein). The idea being that the comprehension of language enables the agent to apply familiar heuristics to unfamiliar situations [Hermann et al., 2017].

However, the majority of these approaches use bespoke, human-composed corpora. It is difficult to see how this approach can scale in real-world applications. Instead, we advocate for giving agents the ability to ask natural language questions to an omniscient Oracle. By sampling from a vocabulary of relevant words, the agent must learn to generate syntactically correct questions parseable by the Oracle. Said Oracle then provides binary answers, to avoid the transmission of more information than desired. The agent then conditions its policy on the question-answer pair as well as an embedding of the history of previous state-question-answer-actions. To our knowledge, this is the first time an agent has been jointly trained to solve a task and do so by asking questions about its environment.

We wish to explore the following research questions:

1. Can an RL agent learn an effective joint policy over natural language questions and actions in its environment?
2. Will this joint policy lead to question asking be-

haviour which extracts information that is useful in solving the environment?

3. Can an RL agent generalise more effectively to novel environments through questions asking?

2 Related Work

2.1 Natural Language Annotations

An example of improving generalisation performance of an agent by augmenting it with language understanding can be seen in "RTFM" by [Zhong et al., 2019]. The agent is jointly trained on both language and environment goals, as it is passed a document which describes the dynamics of their environment in natural language. This paper uses human-generated templates, and then "procedurally" generates both the environment dynamics and descriptions of said dynamics by cycling through the possible combinations. With random ordering, the "number of unique documents exceeds 15 billion", and this serves to ensure the agent cannot just memorise an environment - description mapping. The quantity of potential documents is outstanding, however the model still relies on human-composed text at its heart.

A similar approach can be seen in [Branavan et al., 2014] where the authors show an agent can achieve greater performance in the game of "Civilization II" when it is jointly trained on playing the game and parsing the actual instruction manual for the game (which was naturally written by a human). [Branavan et al., 2009] present a RL method of learning a mapping from human-composed natural language instructions into sequences of executable actions, and show that (for well defined reward functions) this mapping can be learned despite having few if any labelled training examples (the agent learns solely via policy gradient methods back-propagating the reward). Our model has a very similar update method.

2.2 Transfer-Learning

A transfer-learning approach to the same problem can be seen in [Narasimhan et al., 2017]. The authors first ground an agent's understanding of natural language to the dynamics of an environment by passing the agent textual descriptions of the environment it was trained in. Greater generalisation performance is seen when subsequently training the agent in unseen environments with accompanying textual descriptions. A very similar approach is seen in [Harrison et al., 2017] where the authors train an encoder-decoder network to learn associations between natural language descriptions and state/action information. This learned model then guides an RL agent's exploration in unseen environments, making it more effective at learning therein.

While these papers both use natural language to improve the generalisation capabilities of RL agents, they differ from our model in that they both still rely on human-generated textual descriptions, rather than letting the agent generate its own NLC through conversing with an Oracle.

2.3 Meta-Learning

[Co-Reyes et al., 2018] frame a similar problem as one of meta-learning, where the RL agent goes through a meta-training procedure, in order to learn a procedure, to enable it to adapt to new tasks during meta-test time. In this paper, the agent learns to guide its policy via "iterative natural language corrections". This paper aligns closely with our model in that the temporal nature of the passing of natural language information to the agent is iterative, and the NLC is generated as the agent explores the new environment. This contrasts with the approaches seen in the 'natural language Annotations' section, where the NLC is passed to the agent in-one-go before it even begins in its new environment. In this paper the agent is continuously 'corrected' after each action, by receiving instructions such as 'Move closer to the pink block'. Our model has a similar setup in that the agent receives an Answer to a Question it asks, after each action. Our model is similar, but differs markedly in two key ways:

1. In our model, the onus is on the agent to ask about the pertinent information. The agent cannot just learn to receive natural language information, in our model it must learn to **ask** the right questions.
2. In our model, the agent receives information from an Oracle which can only respond 'True' or 'False'. This constrains the amount of 'extra' information the agent receives on each time-step to the the truth value of a query it itself has generated.

2.4 Learning by Asking (LBA)

The idea of Learning-by-Asking comes from [Misra et al., 2017]. The paper raises the key point that most visual-recognition models in the field of Computer Vision are 'passive' in that they are trained on a fixed corpus of data curated by humans, e.g. [Antol et al., 2015]. The authors propose Learning-by-Asking (LBA), which endows an agent with the agency to decide which questions it needs to ask of an Oracle. The authors find their model is both more sample efficient and better at generalising to novel test-time distributions on the CLEVR dataset [Johnson et al., 2016]. Again this has many parallels with the setting of real-world interactive learning. This element of agency is the crucial feature which we carry over and apply to the problem of "Language-Assisted RL" [Luketina et al., 2019].

The architecture of our Q-A RNN ties in with a related branch of research, 'Visual Question Generation' (VQG) [Mostafazadeh et al., 2016], a recent proposal as an alternative to image-captioning, in that we explicitly encourage (through the agent's loss) that the agent does ask Questions about items which do exist in the environment. We extend on this however and encourage our questions to not just be **relevant**, but also **informative**. We reinforce this by also explicitly including an entropy term in our loss function. Examples of using uncertainty measures to train networks can be seen in [Joshi et al., 2009].

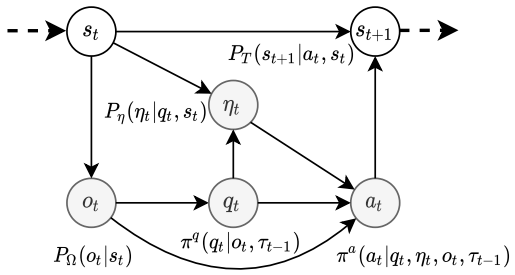

Figure 1: Graphical Model of POMDP. Dark-shading denotes observed variables, no-shading denotes latent variables.

3 Preliminaries

3.1 Reinforcement Learning Framework

Our RL setting can be formalised as an *augmented* Partially Observable Markov Decision Process (POMDP) [Sutton and Barto, 2018] (see Figure 1) defined by the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{Q}, \eta, \mathcal{A}, P_\Omega, P_\eta, P_T, R, \gamma)$, with $\mathcal{S}$ as the set of states, $\mathcal{O}$ as the set of observations, $\mathcal{Q}$ as the set of questions, η as the set of Oracle answers, $\mathcal{A}$ the set of actions, $P_\Omega : \mathcal{S} \times \mathcal{O} \to [0, 1]$ as the conditional observation probability function, $P_\eta : \mathcal{S} \times \mathcal{Q} \times \eta \to [0, 1]$ as the answer probability function, $P_T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ as the transition probability function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the reward function, and γ as the discount factor.

We want to find policy over actions, $\pi^a : \mathcal{A} \times \mathcal{O} \times \mathcal{Q} \times \eta \to [0, 1]$, that maximises the expected discounted return $\mathbb{E}_{s_0,a_0,\ldots}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$, where $a_t \sim \pi^a(a_t|o_t, q_t, \eta_t, \tau_{t-1})$, $s_t \sim P_T(s_{t+1}|s_t, a_t)$. Note that the policy over actions is conditioned on the current observation, question and answer, as well as τ_{t-1}, the history of past observations, question, answers and actions $\tau_{t-1} = (o_{:t-1}, q_{:t-1}, \eta_{:t-1}, a_{:t-1})$ up to time $t - 1$.

The main hypothesis of this paper is that, by introducing an intermediate policy over questions, $\pi^q : \mathcal{O} \times \mathcal{Q} \to [0, 1]$, $q_t \sim \pi^q(q_t|o_t, \tau_{t-1})$ we can query information from an Oracle in the form of answers η_t. We argue that the resulting policy π^a_ϕ will perform better that a baseline without such question asking module, not only due to having access to more information about the environment but also due the fact that by forcing an agent to encode its knowledge of the world in a way amenable to expression through natural language, further generalisation capabilities will emerge naturally.

Note that, although expressed as a probability distribution the answer probability $P_\eta(\eta_t|q_t, s_t)$ is, in general, a deterministic function of the state and the question, given that questions are answered truthfully by the Oracle.

4 Oracle

The Oracle has full observability over the environment and is able to answer `True` or `False` to questions in relation to each object in the environment. The questions are posed as predicates such as "*red door is closed*" and "*green goal is north*".

$$
\begin{aligned}
\langle \textit{predicate} \rangle &::= \langle \textit{state predicate} \rangle \\
&\quad | \langle \textit{direction predicate} \rangle \\
\langle \textit{state predicate} \rangle &::= \langle \textit{noun phrase} \rangle \langle \textit{verb} \rangle \\
&\quad \langle \textit{state} \rangle \\
\langle \textit{direction predicate} \rangle &::= \langle \textit{noun phrase} \rangle \langle \textit{verb} \rangle \\
&\quad \langle \textit{direction} \rangle \\
\langle \textit{noun phrase} \rangle &::= \langle \textit{noun} \rangle | \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
\langle \textit{noun} \rangle &::= \texttt{door} \mid \texttt{wall} \mid \texttt{floor} \\
&\quad \mid \texttt{key} \mid \texttt{ball} \mid \texttt{box} \\
&\quad \mid \texttt{goal} \mid \texttt{lava} \mid \texttt{agent} \\
\langle \textit{verb} \rangle &::= \texttt{is} \\
\langle \textit{state} \rangle &::= \texttt{open} \mid \texttt{closed} \mid \texttt{locked} \\
\langle \textit{direction} \rangle &::= \texttt{north} \mid \texttt{south} \mid \texttt{east} \mid \texttt{west} \\
\langle \textit{adjective} \rangle &::= \texttt{red} \mid \texttt{green} \mid \texttt{blue} \\
&\quad \mid \texttt{purple} \mid \texttt{yellow} \mid \texttt{green}
\end{aligned}
$$

Figure 2: BNF Grammar

The Oracle is able to parse two types of predicates:

- Direction predicates: the relative position of an object to the agent.
- State predicates: the current state of an object in the environment.

A predicate is well posed if it adheres to the BNF grammar in Figure 2. The predicates can take one of four values which defines $P_\eta(\eta_t|q_t, s_t)$:

- `True`: the predicate is well posed, uniquely identifies an object present in the environment and correctly describes the state or direction of the object; the Oracle returns $\eta_t = [1, 1]$ and a reward $r^q_t = 0.2$,
- `False`: the predicate is well posed, uniquely identifies an object present in the environment but incorrectly describes the state or direction of the object; the Oracle returns $\eta_t = [0, 0]$ and a reward $r^q_t = 0.2$,
- `Undefined`: the predicate is well posed, but does not uniquely identify and object in the environment and hence has no truth value; the Oracle returns $\eta_t = [0, 1]$ and reward $r^q_t = 0$.
- `Syntax error`: the predicate is not well posed, i.e. it does not adhere to the BNF grammar. The Oracle returns $\eta_t = [1, 0]$ and rewards the agent with $r^q_t = -0.2$.

Note that the existence of an object in the current environment is not limited to the agent's perspective. In other words, the agent can ask questions related to objects not currently present in its observable neighbourhood. Through this, we aim to give the agent the ability to, potentially, learn and exploit a spatio-temporal knowledge representation of the objects it encounters by virtue of its ability to use memory. By asking questions about entities not currently in its sight, the agent could learn to associate the locality of said objects with

a perception of time, insofar as objects which are not currently seen –but whose mention leads to an answer from the Oracle that indicates their existence– either have been seen *somewhere before* or will, potentially, be seen *somewhere in the future*. Although the arousal of such behaviours is certainly unlikely at this stage, this does not preclude us from endowing the agent with the capability to do so, if only desideratively.

5 Ask Before You Act

We propose an architecture that leverages a dialogue between the agent and an Oracle to guide the agent's action policy. Our proposed model consists of a convolutional neural network [LeCun et al., 1998], a memory encoder, and two policies: a neural network question-asking policy parametrised by θ, $q_t \sim \pi^q_\theta(q_t|e^o_t, h^m_{t-1})$, and a neural network action policy parametrised by ϕ, $\pi^a_\phi(a_t|e^o_t, e^q_t, \eta_t, h^m_{t-1})$ (Figure 3). Note that the conditioning variables of the policies are no longer the general random variables o_t, q_t, τ_{t-1} but an neural embedding of such variables e^o_t, e^q_t, h^m_{t-1} respectively. This is done to retain only the most relevant features of the inputs.

At every time step t the agent receives a partial observation of the environment o_t, which is encoded via a convolutional neural network parametrised by ν to a latent lower-dimensional space $e^o_t = f_\nu(o_t)$. Together with this observation, the agent carries forward the hidden state of an LSTM network (the memory module, Figure 3) [Hochreiter and Schmidhuber, 1997], whose purpose is to encode the history of observations, questions, answers and actions up to time step $t-1$:

$$h^m_0 \sim \mathcal{N}(0, I) \tag{1a}$$

$$h^m_t = f_\mu(h^m_{t-1}, e^o_{t-1}, e^q_{t-1}, \eta_{t-1}, a_{t-1}), \tag{1b}$$

where f_μ is an LSTM network parametrised by μ. The concatenated vector $[e^o_t, h^m_{t-1}]$, is then passed to the question policy $\pi^q_\theta(q_t|e^o_t, h^m_{t-1})$.

5.1 Question Policy

The agent's question policy is responsible for generating relevant questions given the current observation and the hidden state of the memory embedding. The inclusion of the trajectory embedding allows the agent to ask questions with a larger time horizon. By retaining information from previous observations, questions-answer pairs and actions, the agent need not ask the same questions when in similar states thus being able to explore more efficiently and generalise better.

More specifically, the question policy consists of a pre-trained generative language model. The question q_t distribution is modelled according to

$$\pi^q_\theta(q_t|e^o_t, h^m_{t-1}) = \pi^q_\theta\left(w_{1:N}|e^o_t, h^m_{t-1}\right) \tag{2}$$

$$= \prod_{i=1}^{N} p_\theta\left(w_t \mid w_{1:t-1}, e^o_t, h^m_{t-1}\right) \tag{3}$$

with w_i as the individual tokens. The language model consists of a word embedding layer, LSTM layer and a fully connected layer. The word embedding layer is a mapping from the vocabulary (which consists of all valid tokens in the BNF grammar) to a latent space using $e^{w_i} = g_\theta(w_i)$.

The model is pre-trained on a corpus containing all possible phrases which adhere to the BNF grammar Figure 2. We pre-train the model in a supervised manner on this corpus using cross-entropy loss. The resulting pre-trained model is then used to initialise the question policy π^q_θ which will be fine-tuned using the agent model updates (see Section 5.3) with the word embedding layer fixed.

At every time step t the question policy $\pi^q_\theta(q_t|e^o_t, h^m_{t-1})$ takes as its initial hidden state the concatenated $h^{w_0}_t = [e^o_t, h^m_{t-1}]$, and as input the embedding vector corresponding to the <sos> token $w_0 = g_\theta(\text{<sos>})$. We denote $h^{w_0}_t$ as the initial hidden state of the π^q_θ at the (environment) time step t. The hidden state $h^{w_i}_t$ is computed recursively as

$$h^{w_i}_t = f_\theta(w_i, h^{w_{i-1}}_t) \tag{4}$$

where $f_\theta(\cdot)$ is a single pass through the embedding layer and LSTM cell. We then pass the hidden state $h^{w_i}_t$ through a single fully connected layer and softmax layer after which we sample w_i the i'th token of the question. This process continues until the <eos> token is sampled. Then the question q_t is the concatenation of the sampled tokens without the <sos> and <eos> tokens $q_t = [w_1, \ldots, w_{N-1}]$.

The resulting question q_t is passed to the Oracle, which returns a tuple (η_t, r^q_t), where η_t is the answer to said question and r^q_t is a reward (see Section 4).

5.2 Action Policy

The resulting answer η_t from the Oracle is concatenated with the last hidden state from the question policy π^q_θ, denoted h^q_t and to an embedding of the question e^q_t. The question embedding e^q_t is computed by mean pooling the word embeddings $e^{w_i}_t$. Together with the embedding of the current observation e^o_t and the memory hidden state h^m_{t-1}, this conforms the input to the action policy π^a_ϕ (see Figure 3).

As mentioned before, the action policy π^a_ϕ consists of a neural network parametrised by ϕ, with a softmax layer over the number of environment actions as its final layer. The action at each time step is sampled from this distribution $a_t \sim \pi^a_\phi(a_t|e^o_t, e^q_t, \eta_t, h^m_t)$.

Finally, the observation embedding, question embedding, answer and action are passed as the input to the memory module, which encodes it and outputs the encoded hidden state for the next time step.

5.3 Update

Action Loss. The action policy is trained using a PPO style loss:

$$\mathcal{L}^a_t(\phi) = \left[\mathcal{L}^{clip}_t(\phi) - c_1\mathcal{L}^{vf}_t(\phi) + c_2 H[\pi^a_\phi]\right] \tag{5a}$$

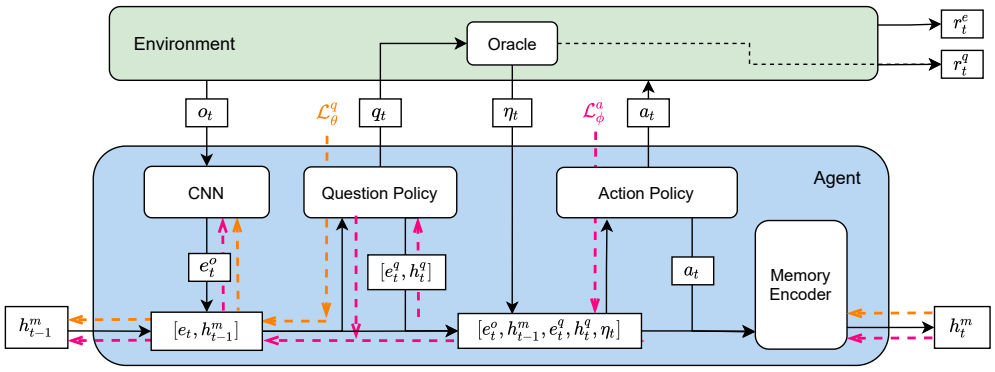

Figure 3: Model architecture with gradient flow.

$$\mathcal{L}_t^{clip} = \hat{A}_t^{GAE} \min\left[r_t^e(\phi), \text{clip}(r_t^e(\phi), 1 \pm \epsilon)\right] \quad (5b)$$

$$\mathcal{L}_t^{vf} = \mathcal{L}_H\left(V_\phi(s_t) - V_t^{target}\right) = \mathcal{L}_H(\delta_t), \quad (5c)$$

where $r_t^e(\phi)$ is the probability ratio

$$r_t^e(\phi) = \frac{\pi_\phi^a(a_t|e_t^o, e_t^q, \eta_t, h_t^m)}{\pi_{\phi_{old}}^a(a_t|e_t^o, e_t^q, \eta_t, h_t^m)}, \quad (6)$$

so $r_t^e(\phi_{old}) = 1$; $\hat{A}_t^{GAE}$ is the generalised advantage function

$$\hat{A}_t^{GAE} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \quad (7)$$

with $\delta = r_t^e + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ being the temporal-difference error [Sutton and Barto, 2018] and $\mathcal{L}_{H_1}$ denotes the smooth L1 loss or Huber loss for $a = 1$. [Huber, 1964]. Moreover, $\text{clip}(r_t^e(\phi), 1 \pm \epsilon)$ saturates the ratio $r_t^e(\phi)$ to be in the interval $[1 - \epsilon, 1 + \epsilon]$. $H[\pi_\phi^a]$ is an entropy regulariser which encourages higher entropy in the policy distribution, increasing the extent to which the agent explores the environment. The constants c_1 and c_2 are hyperparameters that weight the relative importance of the different losses (see Table 2).

Although not shown in Figure 3, the value estimate $V_\phi(s_t)$ is generated by a neural network built on top of the policy network, with the softmax layer replaced by a fully connected layer from the number of actions to a scalar.

Question Loss. The loss applied to the question policy is a modified version of REINFORCE [Sutton and Barto, 2018] with entropy regularisation:

$$\mathcal{L}_t^q(\theta) = (c_3 r_t^q + c_4 G_t) \ln \pi_\theta^q(q_t|e_t^o, h_{t-1}^m) + c_5 H[\pi_\theta^q], \quad (8)$$

where c_3, c_4, c_5 are hyperparameters 2 and G_t is episode reward at time step t [Sutton and Barto, 2018]:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k^e. \quad (9)$$

Note that this value can be computed since we only perform gradient steps at at the end of each episode.

The purpose of this update is twofold: through r_t^q we enforce that the questions are syntactically correct and address objects present in the current environment (see Section 4); through G_t we increase the episode return associated to the questions which influenced the policy in such a way that the question policy maximises the expected return the environment.

Note that the entropy term in the question loss leads to increased entropy in the policy distribution over words and thus encouraging the agent to ask more diverse questions.

Backpropagation During the course of an episode, transitions are stored in a buffer. At the end of every episode these transitions are used to calculate the losses, which are combined into

$$\mathcal{L}(\phi, \theta) = \mathcal{L}_t^q(\theta) + \mathcal{L}_t^a(\phi). \quad (10)$$

A single gradient step then is taken on $-\mathcal{L}(\phi, \theta)$ using ADAM as an optimiser with a fixed learning rate α (See Table 2) [Kingma and Ba, 2014]. The data buffer is emptied after each episode.

Finally, since the model trains end-to-end, through the above losses we also update the parameters ν of the CNN. The parameters μ of the memory LSTM are also updated with this loss, which is backpropagated through time until the first transition of the episode. See Figure 3 for a diagram of the gradient flow overlaid onto the model.

6 Experiments

For experiments we use the MiniGrid environment [Chevalier-Boisvert et al., 2018]. This is a series of fast, lightweight grid world environments. More specifically, we use the so-called MultiRoom set of environments, in which an agent needs to traverse a set of rooms separated by doors in order to get to the model. These environments provide two main features which make them attractive for our purposes: first, they have a reduced set of objects present (walls, coloured doors and a goal) which allows for easy language acquisition; and they are procedurally generated, which can stop the agents from simply memorising the environment. For

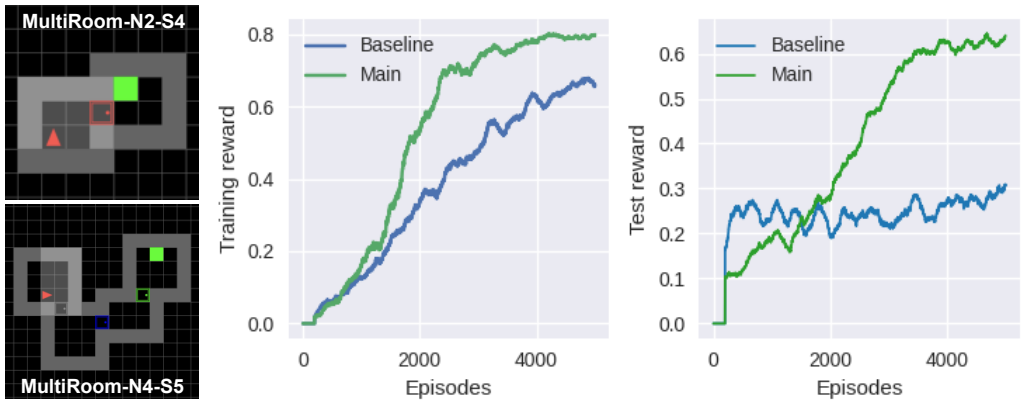

Figure 4: Top left: Sample of the training environment. Bottom left: Sample of the testing environment. Left graph: 100 episode moving average reward during training, averaged over 7 seeds. Right graph: 100 episode moving average reward during testing, averaged over 7 seeds.

Table 1: Table summarising the results obtained for generalisation (Figure 4) and noisy Oracle (Figure 6) experiments. Results show mean ± std. dev. calculated over 7 environment seeds.

Model	Train	Test
Baseline	0.737 ± 0.114	0.202 ± 0.239
Main	**0.803 ± 0.011**	**0.711 ± 0.135**
Main (Rand Oracle)	N/A	0.565 ± 0.207
FiLM	0.770 ± 0.067	0.504 ± 0.277
FiLM (Rand Oracle)	N/A	0.258 ± 0.331

these environments, the reward is calculated as:

$$r_t^e = \begin{cases} 0 & \text{if} \quad t \neq T \\ 1 - \frac{9}{10}\frac{T}{T_{max}} & \text{otherwise} \end{cases} \tag{11}$$

where $T_{max} = 20S_{room}$, which depends on the maximum possible size of the rooms in the environment. Finally, all reported results are averaged over 7 different environment seeds.

6.1 Baseline

For the experiments we use a PPO baseline model [Schulman et al., 2017], which also uses Generalised Advantage Estimation (GAE) [Schulman et al., 2018]. To make the baseline comparable to our model, we also add a convolutional neural network and memory module with the same architectures to the main model. The only difference between the baseline and the main model lies in the existence of a question policy and its interaction with the Oracle. As above, the model is trained end-to-end, with a single gradient step done in batch for each episode using ADAM. The learning rate and other hyperparameters can be seen in Table 2.

6.2 Generalising to Unseen Environments

We train both the baseline and the main agent on the `MiniGrid-MultiRoom-N2-S4-v0` environment until convergence. This environment consists of two rooms with maximum size of $S_{room} = 4$ separated by a locked door. The agent must open and pass through the door in order to reach the goal located in the other room, as shown in Figure 4. The layout of the environment is randomly generated at every episode so the agent cannot simply memorise a sequence of steps to the goal.

We subsequently deploy both agents on a novel environment the larger, and more complex `MiniGrid-MultiRoom-N4-S5-v0` environment. This environment comprises four rooms with maximum size of $S_{room} = 5$ separated by locked doors, with the goal located in the final room (Figure 3). Both environments are procedurally generated with varying door colours and room locations. When testing the agents, we still perform gradient updates, as to allow the models to adapt to the new environment. This follows the curriculum learning setting, evaluating the agents ability to transfer learn to a harder environment [Bengio et al., 2009].

The only difference between the train and testing regimes lies in the reward given by the Oracle. During testing, the agent is only penalised for syntactically incorrect premises, but receives no reward otherwise i.e. we no longer include a positive reward for uniquely identifying an object in the environment.

We wish to evaluate the agent's ability to ground its understanding of natural language in the train environment, and carry over this knowledge to the unseen environment without further reinforcement for correctly identifying objects in the new environment. All other experiments follow the same methodology outlined above.

The mean rewards across all runs are recorded in Figure 4. Full results can be seen in Table 1. We observe that the main agent outperforms the baseline both during training and testing. These results indicate that the question asking module in concert with the Oracle's answers allow the main agent to generalise more effectively to the unseen environment.

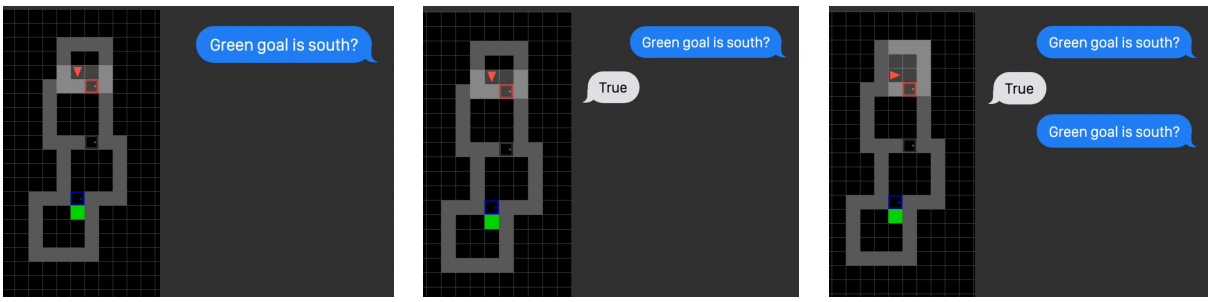

Figure 5: Qualitative demonstration of the agent behaviour in the test environment.

6.3 Qualitative Assessment

When qualitatively assessing the dialogue generated between the agent and the Oracle, we noticed that the majority of the agent's questions referred to the "*green goal*". See Figure 5 for a qualitative demonstration of said behaviour. This is the expected behaviour, when one considers that the agent is penalised during training for asking about objects which are not present in its environment. Thus, the agent learns to ask about as the one object which is present across all MiniGrid environments and will incur in the smallest penalisation.

We question whether this behaviour is a result of the language grounding or a result of the statistical phenomena outlined above, especially given that in some instances the agent does act logically following certain questions about the location of the green goal, whereas in others, we fail to see a significant correlation between the questions and the actions.

More specifically, in certain episodes we observed a directly interpretable correspondence between the question-answer pairs and actions. For instance, the agent may ask "*green goal is south?*", receive the answer `True` and turn south.

However the agent would sometimes appears to ignore the question-answer pairs when taking the immediate action. Although one would expect the influence of the question-answer pair to manifest itself immediately after receiving an answer, it is unclear whether the agent entirely disregards the new information or if the QA pairs are being encoded in the memory module and are acted upon at a later time step. Nevertheless, the promising quantitative results seem to indicate that the information is being used at some point.

6.4 Ablations & Extensions

6.4.1 FiLM

Given the success of [Zhong et al., 2019], we also experimented with an alternative model architecture incorporating FiLM layers [Perez et al., 2017]. Rather than passing the encoding of the QA pairs to the policy and value networks we instead condition the observation on the QA pairs using a FiLMed CNN network [Perez et al., 2017]. The CNN network is extended with five additional Residual Convolution Layers (ResBlocks). The FiLM layer cosists of a fully connected linear layer which takes the encoding of the QA pairs as an input, and outputs the parameters of an affine-transformation which is applied to the activations of the ResBlocks.

This addition did not significantly improve the performance of our model. We hypothesise this is because our GridWorld environments are not sufficiently visually complex, so the benefit of the FiLM architecture is slight. In other words, exploration of the environment via the visual domain seems an easier option than the exploration through the "information domain", due to , perhaps, a not-too-reduced partial observability. Nonetheless, we included this model in Figure 6, and we publish our corresponding code in our GitHub repository. [1]

6.4.2 Random Oracle

We again train our main, FiLM, and baseline agents on the `MiniGrid-MultiRoom-N2-S4-v0` and test them in the `MiniGrid-MultiRoom-N4-S5-v0`, as before. However, in this ablation, during testing we also test the agents with a "confused" Oracle, which responds to the agent with random answers.

We compare how the test performance of the agents differs when the Oracle is responding with useful answers versus random noise. These results are shown in Figure 6 and are also in Table 1.

We note that both the main and the FiLM extended agents both see superior performance in the test environment when receiving correct answers from the Oracle, versus when they receive random answers. We interpret these results as indicating that the agents are succeeding, to some extent, in internalising the information which they receive by asking questions, in order to understand their novel environment.

6.4.3 Mutual Information Regularisation

Given the observed weak conditioning of the actions on the questions for some training instances, we considered how we could enforce a stronger conditioning between QA pairs and the subsequent actions.

We formalise this objective by performing a maximisation of the mutual information between an action a_t and its preceding question q_t

[1] `www.github.com/ser-ge/ask_before_you_act`

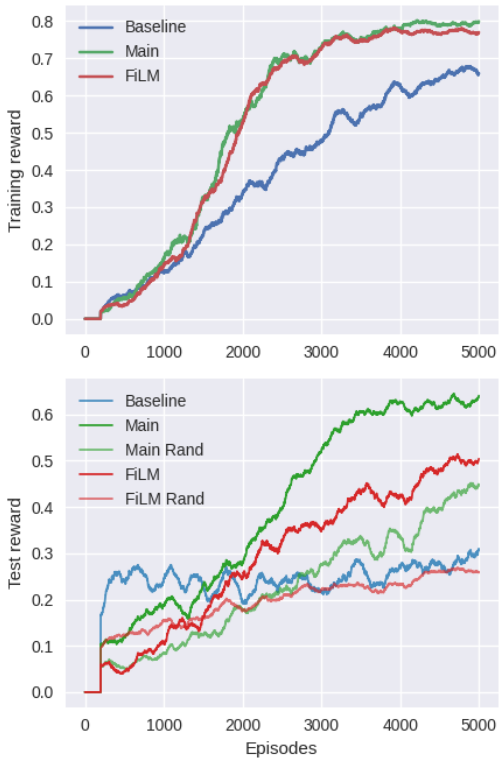

Figure 6: Top Plot: Training main, main agent, and FiLM agent on `MiniGrid-MultiRoom-N2-S4-v0`. Bottom Plot: Testing both main and main agent on `MiniGrid-MultiRoom-N4-S5-v0`.

$$I(a_t; q_t) = H(a_t) - H(a_t|q_t) \tag{12}$$

$$= \sum_{\mathcal{A}} p(a_t) \log p(a_t) - p(a_t|q_t) \log p(a_t|q_t) \tag{13}$$

where the other conditioning variables are implied for ease of notation. The mutual information as expressed in (12) quantifies how much can be known about a_t once q_t is observed. Note that since the answer η_t is a deterministic function of the question and the state given by the environment, we cannot treat it as a variable in our model, and so we make it implicit in our notation.

We see that the RHS of expression (12) is simply the entropy of the action policy, as given by our model $H[\pi^a_\phi]$, quantity which is also used for the loss functions in Section 5.3. However, we do not know the marginal probability of a_t, which is required for the LHS of expression (13).

To compute this quantity, we would want to marginalise q_t over the joint probability of q_t and a_t

$$p(a_t) = \sum_{\mathcal{Q}} p(a_t, q_t) = \sum_{\mathcal{Q}} p(a_t|q_t)p(q_t), \tag{14}$$

where $p(q_t)$ can be estimated from the softmax distributions over tokens given by the LSTM question policy, and $p(a_t|q_t)$ is just a forward pass of the action policy.

We can approximate $p(a_t)$ by performing N number of forward passes on the question policy and computing for each question the joint probability over tokens as per expression (3). We then pass each of the questions to the Oracle, which returns N different answers η_t. Given each of the N QA pairs, we can now run the policy forward pass to obtain N values of $p(a_t|q_t)$. By multiplying each $p(a_t|q_t)$ with its respective distribution over actions $p(a_t)$ and summing over the N samples, we can obtain an estimate of $p(a_t)$. It remains to calculate the entropy of the resulting distribution as in (13).

Finally, by incorporating the mutual information $I(a_t; q_t)$ into the loss of the model, we can force this term to be maximal, relative to the other terms in the loss function. In principle, this could improve the explainability of the actions in light of the questions.

7 Conclusion

We proposed *Ask Before you Act*, a problem setting in which an RL agent must learn a joint policy over actions and natural language questions posed to an all knowing Oracle, to obtain information that would assist it in solving its environment. We investigate the ability of the main agent to successfully query useful information by evaluating its ability to generalise to unseen harder environments having grounded its natural language understanding in the training environment. The main agent performs well in this curriculum setting, learning a joint policy over action and questions which significantly outperforms the baseline in the harder environment. In order to evaluate the usefulness of the Oracle's answers to the agent we performed ablations with a random Oracle in the test environment. We observed a material reduction in performance with a random Oracle, indicating that the agent successfully leverages the ability to query information through natural language.

Despite promising quantitative results the qualitative results were inconclusive. We observed that at times the agents' behaviour did not correspond directly to the dialogue with the Oracle; or that the agent would fixate on asking questions about the same object (i.e. green goal). Further work may focus on more challenging generalisations settings with no object permanence across environments. In addition one may wish to conduct experiments with a tighter coupling of action policy to the questions through mutual information regularisation which would allow for a more robust assessment of the role of dialogue in solving the environment.

References

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *CoRR*, abs/1906.07343, 2019. URL `http://arxiv.org/abs/1906.07343`.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015. URL http://arxiv.org/abs/1507.00814.

Marvin Minsky. steps.pdf. https://courses.csail.mit.edu/6.803/pdf/steps.pdf, 1961. (Accessed on 05/31/2021).

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

Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1):89–96, 2007. doi: https://doi.org/10.1111/j.1467-7687.2007.00569.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-7687.2007.00569.x.

Pedro Tsividis, Thomas Pouncy, Jaqueline Xu, Joshua Tenenbaum, and Samuel Gershman. Human learning in atari, 2017. URL https://www.aaai.org/ocs/index.php/SSS/SSS17/paper/view/15280.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob N. Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *CoRR*, abs/1906.03926, 2019. URL http://arxiv.org/abs/1906.03926.

Gopnik and Meltzoff. The development of categorization in the second year and its relation to other cognitive and linguistic developments on jstor. https://www.jstor.org/stable/1130692?seq=1, 1987. (Accessed on 05/30/2021).

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. *CoRR*, abs/1706.06551, 2017. URL http://arxiv.org/abs/1706.06551.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: generalising to novel environment dynamics via reading. *CoRR*, abs/1910.08210, 2019. URL http://arxiv.org/abs/1910.08210.

S. R. K. Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. *CoRR*, abs/1401.5390, 2014. URL http://arxiv.org/abs/1401.5390.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Suntec, Singapore, aug 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1010.

Karthik Narasimhan, Regina Barzilay, and Tommi S. Jaakkola. Deep transfer in reinforcement learning by language grounding. *CoRR*, abs/1708.00133, 2017. URL http://arxiv.org/abs/1708.00133.

Brent Harrison, Upol Ehsan, and Mark O. Riedl. Guiding reinforcement learning exploration using natural language. *CoRR*, abs/1707.08616, 2017. URL http://arxiv.org/abs/1707.08616.

John D. Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine. Guiding policies with language via meta-learning. *CoRR*, abs/1811.07882, 2018. URL http://arxiv.org/abs/1811.07882.

Ishan Misra, Ross B. Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. Learning by asking questions. *CoRR*, abs/1712.01238, 2017. URL http://arxiv.org/abs/1712.01238.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015. URL http://arxiv.org/abs/1505.00468.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Larry Zitnick, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. *CoRR*, abs/1603.06059, 2016. URL http://arxiv.org/abs/1603.06059.

Ajay J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379, 2009.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. lecun-98.pdf. http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf, 1998. (Accessed on 05/30/2021).

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017. URL http://arxiv.org/abs/1709.07871.

A Appendix

A.1 Hyperparameters

We performed grid search over the hyperparameters set out in Table 2 individually for each model.

Table 2: Hyperparameters: these hyperparameters were selected through grid search using the WandB service.

Agent	Main	FiLM	Baseline
α	0.0005	0.0001	0.001
ϵ	0.2	0.15	0.2
γ	0.99	0.99	0.99
λ	0.95	0.95	0.95
c_1	1	1	1
c_2	0.1	0.1	0.1
c_3	0.25	0.25	N/A
c_4	1	1	N/A
c_5	0.2	0.5	N/A