Interactive Question Answering
Generalisation in Text-based Environments

Edan Toledo
University of Cape Town
Cape Town, South Africa
TLDEDA001@myuct.ac.za

ABSTRACT

Artificial intelligence has long aspired to create systems that can perform real-world activities and converse with humans using natural language. In the pursuit of this goal, *Interactive Question Answering* was proposed. Specifically for language comprehension, IQA within text-based environments is seen as a viable direction to train and evaluate these systems. This paper aims to investigate the effectiveness of a more human-like policy-based reinforcement learning approach and an environment dynamics model to promote generalisation in textual environments. We evaluate the policy-based agent performance and the use of an environment dynamics model on the QAit (Question Answering with interactive text) benchmark. The results produced indicate that policy-based reinforcement learning has significantly better generalisation capabilities than the value-based QAit baselines and that an environment dynamics model can be used to regularise and promote generalisation when access to multiple training environments is not possible.

CCS CONCEPTS

• **Reinforcement Learning → REINFORCE**; **Generalisation**; • **Natural Language Processing → Encoding Semantics**; **Question Answering**.

KEYWORDS

Question Answering Systems, Interactive Question Answering, Natural Language Processing, Reinforcement Learning, Text-based Games

1 INTRODUCTION

Throughout recent years, question answering (QA) systems have become increasingly helpful by providing users with the ability to find answers to questions posed in natural (or partially natural) language. These systems are used in a variety of ways [11] as the need to query large amounts of information becomes more prevalent. Despite advances in recent years, current QA systems often fail to generalise to unseen and out of domain information, thus limiting their use to data abundant fields. While attempts have been made to better generalisation capabilities [18, 28, 43], results have not seen significant improvement. Beyond the lack of generalisation, these systems are also restricted to declarative knowledge, thereby limiting their utility. These shortcomings have motivated research into *Interactive Question Answering* (IQA) as a viable solution. **IQA** refers to the task of answering questions about dynamic environments through the means of interaction. This involves creating a system to gather and interpret data, much like humans do. IQA exhibits much larger challenges than traditional QA due to an increase in system requirements.

IQA comprises of the following **4** main challenges:

- The system needs the ability to navigate through different environments.
- It needs to have sufficient comprehension ability to understand the environment and its dynamics.
- It must have the ability to interact with the environment and its objects intelligently.
- The system needs the ability to execute a series of actions conditioned on the question asked.

IQA focuses on learning the procedural knowledge of navigation and interaction in order to discover the information required to answer a given question, i.e. the declarative knowledge. Ultimately, the system must use and interpret the discovered knowledge to give the correct answer. Procedural knowledge comprises the sequences of actions required to perform some task. This greatly contrasts traditional methods, which almost exclusively focus on declarative knowledge. Traditional methods are mostly supervised learning tasks whereby existing Machine Reading Comprehension (MRC) datasets seemingly encourage models to simply learn shallow phrase and word matching between questions and knowledge sources. This is likely due to large amounts of textual overlap between questions and their answers. Trischler et al. [33] show how traditional models can easily exploit this overlap for significant performance, falsely leading to the belief that these models have true comprehension ability. Due to this, applications of these systems are limited to problems involving static, fully observed, and information sufficient documents. This is very unlike the necessary information-seeking behaviour that is required in answering a variety of natural questions [20].

General domain language comprehending systems has been a goal of artificial intelligence for a large part of its history [9]. The IQA task seeks to aid in the creation of these systems. The use of general domain QA systems would greatly benefit all domains consisting of low resources and insufficient quality training data. Beyond the benefit of applying IQA systems to static data in different domains lies the possibility of application in gathering procedural knowledge, which traditional systems are not capable of doing.

With such context in mind, Yuan et al. [41] proposed the Question Answering with interactive text (QAit) task, an IQA challenge using text-based environments. Classic text games, such as Zork, are often used in the attempt to create language comprehending agents [1, 8]. QAit utilises Microsoft TextWorld [12], a text-based game framework, to generate text game-like environments on the fly, providing ample opportunity for language learning. Agents interact with these text-game environments via natural language text commands in order to navigate through the world and gather the information required to answer a given question. QAit demonstrates the performance of popular value-based reinforcement learning methods showcasing the difficulty of the challenge. We hypothesise that a more human-like stochastic approach will not only learn faster but perform better in the QAit task.

To this end, we explore the effectiveness of policy-based methods within textual environments and the use of a predictive environment dynamics model in the creation of generalisable and language comprehending agents. Prior work has shown policy-based methods to have significantly better generalisation capabilities than their value-based counterparts [3, 5]. Additionally, the use of a predictive environment dynamics model as proposed by Pathak et al. [24] has been seen to further aid generalisation. Originally used to provide intrinsic motivation for agents with the goal of efficient exploration, Yao et al. [40] adapted this approach for text-based games to regularise the textual encoding and capture language semantics showing an improvement in single-game performance. The QAit baseline methods seek to learn and approximate the Q-function of the environment, which can be extremely challenging to learn in environments with large state and action spaces such as text environments. The above thus motivates the investigation of the proposed alternative methodologies.

We will compare a policy-based method, trained using the REINFORCE with baseline algorithm [39], and the use of a predictive environment dynamics model against QAit's baseline methods on the given QAit test set. For comparability, we make use of QAit's environment generation for training data. Three critical research problems are identified: Firstly, are policy-based methods more suitable for text-based environments due to a seemingly more human-like manner of decision making, i.e. non-deterministic. Secondly, do policy-based methods generalise better in text-based environments. Lastly, does using a predictive environment dynamics model as a regularisation technique improve text-based environment generalisation.

To address these research problems, we experimentally investigate the following research questions:

(1) How much impact does the policy-based method have on training and testing accuracy?
(2) How much faster/slower is the policy-based method's training process?
(3) How much data does the policy-based method require to reach the same level of performance seen in baselines?
(4) Does the environment dynamics model improve the baseline model's performances?
(5) Does the environment dynamics model along with the policy-based method increase performance further?

To answer each of these questions, we analyse the following comparisons between the policy-based/regularised agent and QAit baselines:

(1) Evaluation accuracy and sufficient information performance.
(2) Number of training episodes until convergence.
(3) Difference in training performance per 1000 episodes.
(4) Evaluation of accuracy and sufficient information performance of a DQN agent with an environment dynamics model.
(5) Evaluation of accuracy and sufficient information performance of a policy-based agent with an environment dynamics model.

The rest of the paper is structured as follows: Section 2 discusses the prerequisite knowledge for discussed concepts. This consists of what environments are, reinforcement learning core concepts and short explanations of the methods used as baselines. Section 3 discusses related work in text-based reinforcement learning, IQA and the use of environment dynamics models. Section 4 proceeds to explain the design and implementation of the agent architectures created and used. Section 5 discusses the experimental methodology used in evaluating new agent architectures. In section 6, all experiment results are presented and analysed. Lastly, section 7 summarises all key conclusions.

2 BACKGROUND

The following subsections are relevant and necessary concepts to understand for this paper. For sections 2.3 and 2.4, we refer to the learning rate as α.

2.1 Environments

Environments refer to the "world" an agent is situated in. Many different types of environments exist [4] for different tasks. The environment constructs and generates the simulated experience for reinforcement learning (RL) agents to learn how to perform tasks.

2.1.1 *TextWorld.*

TextWorld [12] is a sandbox environment that allows users to play text games interactively. It also provides generative capabilities to construct specific text-based games for different domains. Its generative processes allow users to fine-tune the difficulty, range, and language of created games. Additionally, TextWorld is used to research generalisation and transfer learning by constructing sets of different but related games. TextWorld has been used extensively in research involving text-based environments [2, 8, 15, 22, 41, 42] and is used in the QAit task. An example of a text environment created using TextWorld is CoinCollector [8]. This environment consists of a series of interconnected rooms with the single goal of finding and collecting coins. This environment aims to assess the exploratory nature of an agent.

2.2 Policies and Value Functions

Agents use policies to decide their actions at every time step t. A policy π is defined as a mapping of environment states $\mathcal{S}$ to the probabilities of selecting each possible action $a \in \mathcal{A}$ [30]. If an agent is following a policy π at time step t, the policy function $\pi(a|s)$ is the probability of performing the specific action $A_t = a$ given the agent is in state $S_t = s$.

Popular reinforcement learning algorithms such as Q learning or REINFORCE with baseline, in the process of finding the optimal policy (the policy that yields the highest cumulative reward), estimate the value function. The value function is a function of states or state-action pairs that estimate the *value* (expected cumulative reward to be received going forwards) of being in a specific state. Since the cumulative reward is dependant on future actions in future states, the value function is defined with respect to the policy the agent is following. The expected cumulative reward G_t can be defined as follows:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

where T is the terminal time step and γ is the discount factor (the weight of importance given to future rewards). This means formally the value function for any given policy is defined as:

$$V_\pi(s) = \mathbb{E}[G_t | S_t = s], \text{ for all } s \in \mathcal{S}$$

2.3 Value-Based Methods

Value-based methods are methods in which the agent tries to learn the value function of the environment it is situated in. The value function is extremely beneficial as it gives the agent information on what states it should be in. The agent can use the value function for action selection simply by acting greedily and choosing the action that will take it to the state with the highest value. This is called the greedy policy and is commonly how control is implemented in value-based methods (not including the possibility of exploration) [30]. Practically, most value-based methods try to learn the value of action-state pairs i.e Q values. The reason for this is that if the agent does not know the model/environment dynamics, it doesn't know which action will take it to the desired state, so the solution is to create a function that gives an estimate for the total expected cumulative reward if an agent takes a specific action in a specific state. The following are a few of the most popular value-based methods.

2.3.1 Q-Learning. Q-learning [38] is a proposed method for agents to learn the optimal state-action value function directly instead of repeatedly performing policy evaluation and iteration. It is seen as a more sample efficient method of learning as it can reuse past experience. An example of the Q-learning update using one-step TD-learning is as follows:

$$Q(S_t, A_t) \mathrel{+}= \alpha(R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

Although Watkins et al. [38] have shown the convergence properties illustrating that Q-learning can be used effectively to solve Markov Decision Processes (MDPs), practically, these value-based methods suffer from poor convergence.

2.3.2 DQN. When modelled into MDPs, most modern problems have extremely large state spaces (potentially continuous state space). This is compounded when trying to learn the action-state pair value estimates as the number of states is multiplied by the number of actions. This high dimensionality makes traditional Q-learning for larger problems computationally infeasible. To solve for this, function approximation has been proposed. With modern advancements in deep learning, one type of function approximation used heavily in reinforcement learning are neural networks. DQNs (Deep Q Networks) [7] have been shown, with a few additions such as experience replay memory [21] and target networks, to solve large dimensional MDPs, such as Atari games, effectively. However, there are limitations to this approach. It was shown that non-linear function approximation, such as neural networks, can cause the Q-network to diverge [34]. DQN's also tend to overestimate the actual Q-value, which can eventually lead to sub-optimal policies [32]. Even though convergence is not theoretically guaranteed when using neural network function approximation and overestimation occurs, practically, we see successful results in applying the DQN algorithm to certain problems [7].

2.3.3 DDQN. DDQN (Double DQN) [35] is the proposed algorithm to solve the overestimation problem in DQNs. DDQN shows significant results in increasing stability and reliability of learning, reducing DQNs overoptimism as well as how this reduction in overoptimism allows the discovery of better policies [35].

2.3.4 Rainbow DQN. Rainbow DQN [19] is an extension to the original DQN algorithm that combines several improvements found over the years into a single agent. Rainbow DQN uses: DDQN to reduce the overestimation bias, Prioritised Experience Replay [25] to speed up learning, dueling networks [37], multi-step learning [29] for faster learning with suitably tuned hyper-parameters, Distributional reinforcement learning [10] instead of expected return, and noisy nets [14] for exploration. This extension shows improved performance in comparison to other methods.

2.4 Policy-Based Methods

Policy-based methods are methods in which the agent tries to learn the policy function, i.e $\pi(s|\theta)$ for $s \in \mathcal{S}$, directly. Any function can be used as long as it's differentiable. This can be advantageous as it can learn stochastic policies, whereas value-based methods are deterministic. It can also be used to learn continuous or high dimensional action spaces effectively and has better convergence properties. Policy gradient methods search for a local maximum in $J(\theta)$ by gradient ascent $\theta \longleftarrow \theta + \alpha\nabla_\theta J(\theta)$.

2.4.1 REINFORCE. REINFORCE [39] is a Monte Carlo method that updates the policy function's parameters directly using the policy gradient with respect to the objective function $J(\theta) = \mathbb{E}\big[\sum_{t=0}^{T} R_t|\pi_\theta\big]$. Sutton et al. [31] show that the gradient of the objective function to maximise expected total cumulative reward is $\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t)G_t$. This allows us to relatively easily calculate the gradient and update the policy's parameters directly using the REINFORCE update:

$$\theta_{t+1} = \theta_t + \alpha * G_t * \nabla_\theta \log \pi_\theta(A_t|S_t)$$

where α is the learning rate parameter. As per the shortcomings of Monte Carlo methods, REINFORCE suffers from high variance with a noisy gradient estimate and no clear credit assignment to positive or negative actions throughout the episode [30]. An easy way to improve REINFORCE is to reduce the variance of the empirical returns G_t by subtracting a baseline function $b(s)$ in the policy gradient. The baseline is regarded as a proxy for the true expected return. A popular option for the baseline function is the state value function $V(S_t)$. The new value after subtracting the baseline from G_t is defined as the advantage factor $Adv_t(S_t) = G_t - V(S_t)$. This changes the REINFORCE update to:

$$\theta_{t+1} = \theta_t + \alpha * Adv_t(s) * \nabla_\theta \log \pi_\theta(A_t|S_t)$$

This requires the REINFORCE agent to learn the value function alongside the policy and can introduce a bias as the cost of lowering variance. The value function is jointly learned by minimising the MSE loss:

$$MSE = \sum_{t=0}^{T-1}(G_t - V(S_t))^2$$

REINFORCE with baseline is similar to another policy-based method, the advantage actor-critic. The difference between the two is that the actor-critic makes use of $R_t + V(S_{t+1})$ instead of the Monte Carlo return G_t. This allows it to bootstrap experience but increases bias as a result.

3 RELATED WORK

3.1 Text-based Interactive Environments

The QAit (Question Answering with interactive text) [41] task proposes a novel text-based question answering problem whereby an agent must interact with a partially observable text-based environment to gather the declarative knowledge required to answer questions. QAit poses said questions about the location, existence and attributes of objects distributed throughout the environment. QAit produced and evaluated a set of baseline models on a created test-set of unseen environments and questions. This test-set is intended to be used as a benchmark for future research to evaluate an agents ability to comprehend language and generalise its action policy. The specifics of QAit are elaborated in section 4.1.

Another textual environment proposed is World of Bits [26]. World of Bits created a unique platform to teach agents to accomplish tasks via interaction with the internet. This is similar to QAit in that agents are situated in textual environments, but World of Bits does not generally require agents to gather information for their task, and the goal of an agent is to simply accomplish their task rather than using the information learned to answer a question.

3.2 Visual Interactive Question Answering

Interactive question answering is a new field of research without an extensive literature collection. Of the existing research - there is a large focus on visual question answering [13, 16]. Visual question answering focuses on teaching agents to interact in environments with visual inputs and answer questions based on these observations. These visual question answering environments mainly support simple navigation and camera movement actions that limit the agent's interaction with its surroundings. Visual question answering also poses the problem of interpreting different data modalities, which can make the creation of high performing agents very difficult as well as environment-specific. The works of Gordon et al. [16] and Das et al. [13] show poor performance increases above baselines, indicating the difficulty of this task. Both works use hierarchical reinforcement learning to learn all the different sub-policies required to answer questions in interactive environments.

3.3 Environment Dynamics in Text-Games

An agent understanding environment dynamics in text-based environments can be seen as having semantic comprehension. Yao et al. [40] show that most current reinforcement learning agents are not making use of language semantics in their learned representations. They further show that by eliminating semantics and using a fixed random non-semantic representation, agents can surprisingly memorise high score policies. Lastly, they investigate the use of an inverse environment dynamics model (inspired by the Intrinsic Curiosity Module [24]) whereby the agent receives two consecutive states as input and tries to predict the action that caused this transition. This inverse dynamics model is then used to regularise and promote semantic encoding into the learned state representation. Yao et al. investigate the learned semantic encoding by applying the representation to an entirely new environment and simply fine-tuning the agent's policy. There is a significant increase in performance compared to other models' learned representation indicating that a degree of language semantics is captured.

Table 1: Example of an agent interacting with an environment to gather knowledge and answer a question. [41]

Game	You find yourself in a backyard. You make out a patio table. You see a patio chair. It's stylish. In a corner you can see a BBQ. There's an open screen door leading south.
Quest	Is ghargh edible?
Agent	**>Qapla'**
Game	That's not a verb I recognize.
Agent	**>eat ghargh**
Game	You can't see any such thing.
Agent	**>go south**
Game	You've just entered a kitchen. You can see an open fridge here. It is empty! You see a closed oven. Look over there, it's a counter. On the counter you can see a diced potato, a red ghargh and a knife. There's an open door is leading north.
Agent	**>eat ghargh**
Game	You need to take the red ghargh first.
Agent	**>take red ghargh**
Game	You take the red ghargh from the counter.
Agent	**>eat ghargh**
Game	You eat the red ghargh. Not bad.
Agent	**>wait**
Agent	**>answer: yes**

4 DESIGN AND IMPLEMENTATION

4.1 Overview of QAit

4.1.1 Environment & Difficulty. We make use of QAit's modification and implementation of TextWorld to create text environments and associated questions dynamically. This is used to construct training sets consisting of a limited number of different environments. QAit aims to test an agent's language comprehension abilities using tasks that require an understanding of locality, existence, and attributes. All environments are generated by sampling from the world setting distribution (see Table 2), where environment configurations are distinguished into ***fixed*** map and ***random*** map categories. The fixed map category sees to the creation of environments consistently containing 6 unique rooms. In contrast, random map games draw from a uniform distribution to decide on the number of rooms to create.

Table 2: Environment Generation - Fixed map worlds always have 6 rooms, whereas random map worlds sample uniformly to have between 2 to 12 rooms

	Fixed Map	Random Map
# Locations, N_r	6	$N_r \sim Uniform(2, 12)$
# Entities, N_e	$N_e \sim Uniform(3 \cdot N_r, 6 \cdot N_r)$	

.

Questions based on each environment are created on the fly as an agent plays a game, but the number of different worlds is set as an experimental parameter. In the original QAit paper, all agents are trained on datasets consisting of 1, 2, 10, 100, 500 created environments as well as an unlimited setting where different environments are created for each question, thereby theoretically not allowing an agent to see the same environment question pair twice. In this setting, more than 10^{40} different games can be created, indicating that an agent is unlikely to see the same environment again.

4.1.2 Question Types. There are three types of questions that the agent attempts to answer based on these generated worlds.

- **Location:** location type questions ask the whereabouts of objects situated within the world. An example of such a question is "Where is the can of soda?", where a suitable answer would be "fridge". The agent has to answer with the most relevant container of an object. This means that if an object is present within a container such as a fridge, the agent cannot report back the room the object is in, e.g. kitchen.
- **Existence:** existence type questions ask about the presence of objects situated within the world. An example of such a question is "is there a raw egg in the world?" where the answer would simply be yes or no.
- **Attribute:** attribute type questions, the most difficult of all three question types, ask about whether or not an object has a certain associated attribute. An example of such a question is "is apple edible", where the answer is also yes or no. This involves an agent executing a complicated series of steps to evaluate object attributes, e.g. the agent has to find the apple, pick it up, try to eat it and observe the outcome to answer. The QAit baselines arguably do not improve upon random baselines in this category, indicating the difficulty of this question type. Objects within the attribute question setting are given arbitrary and randomly made-up words to discourage agent memorisation of values, such as an apple always being edible (see Table 1).

4.1.3 Interaction. Since language generation can become intractable within a reinforcement learning setting, all text commands are triplets of the form action, modifier, object (e.g., open metallic gate). When there is no ambiguity present such as two different keys in a room, the environment understands commands without modifiers, e.g. pick key will result in picking up the "copper key" provided it is the only key in the room. At each game step, there are three lexicons that divide the vocabulary into actions, modifiers and objects. This reduces the size of the action space for each word in the command triplet compared to a sequential, freeform setting. The *wait* command indicates the agent wants to stop interaction and answer the question. An episode of experience terminates when the wait command is issued or upon reaching a maximum number of steps. For our experiments, we use a maximum of 50 steps. After environment interaction, the agent proceeds to answer the question. Table 3 gives an indication of the action space and observation space sampled over 10,000 games.

Table 3: Statistics of the QAit dataset. Numbers are averaged over 10,000 randomly sampled games. [41]

	Fixed Map	Random Map
Actions / Game	17	17
Modifiers / Game	18.5	17.7
Objects / Game	26.7	27.5
# Obs. Tokens	93.1	89.7

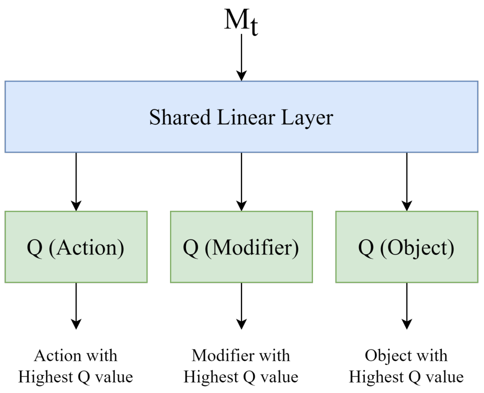

M_t
Shared Linear Layer
Q (Action)
Q (Modifier)
Q (Object)
Action with Highest Q value
Modifier with Highest Q value
Object with Highest Q value

Figure 1: Overview of QA-DQN Network Architecture

4.1.4 Rewards. The QAit environments have a shaped reward function to aid in learning. There are two types of rewards:

Exploration Rewards: An agent is rewarded each time it enters a previously unseen *state* where a *state*, in this case, is defined as a new physical location and/or new inventory status. This is done to promote exploration. Yuan et al. [8] show the benefit of using this exploration by counting method in text games.

Sufficient Information Rewards: For each question type, the reward function is heuristically crafted. A sufficient information reward is given at the end of an episode depending on the agent's actions throughout. These rewards are explained in section 5.1.2.

4.1.5 Evaluation. The QAit test set provides 500 held out games for both map types and all three question types. This testing set is used to benchmark the generalisation abilities of agents on all experiment configurations. This allows for models to be assessed in a reproducible manner and is analogous to supervised learning test sets.

4.1.6 Baseline Reinforcement Learning Agents. QAit provides five baselines - these are human, random, and three popular value-based reinforcement learning methods. The human baseline consists of results achieved by 21 human participants. The random baseline performs no interaction with the environment and simply samples answers from the potential answer pool. This is yes or no for existence type questions and all possible object names for location type questions. The reinforcement learning baselines are DQN, DDQN and Rainbow (see section 2.3 and figure 1). Each agent uses the same general training loop (see algorithm 1) with slightly altered parameters.

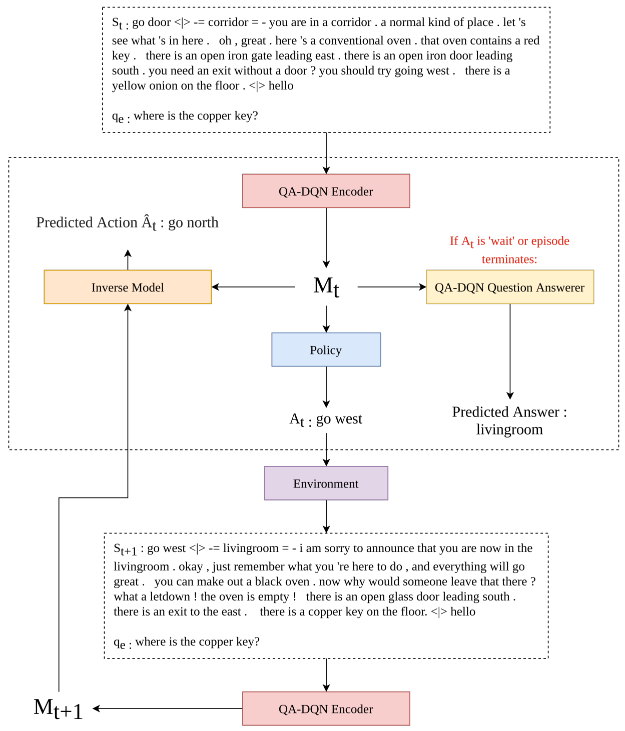
4.2 Agent Model

The following is an explanation of the architecture used by the agent (see Figure 2).

4.2.1 QA-DQN Encoder and Question Answerer. Due to the focus of this paper being on an investigation into an alternative to the value-based methods used, the agent still makes use of the RL baseline's architecture for textual encoding and question answering. The transformer-based [36] text encoder makes use of an embedding layer, two stacks of transformer blocks, one for encoding and the other for aggregation, and a final attention layer. The embedding

Algorithm 1: Training Loop

```
for episode e in training episodes do
    $q_e$ ←question about environment;
    $o_t$ ←starting observation;
    while agent has not said 'wait' or max number of steps T
      has not been reached do
        $M_t$ ←Encode ($o_t, q_e$);
        $a_t$ ←$\pi(M_t)$;
        $o_{t+1}, r_t$ ←environment step ($a_t$);
        save experience ($o_t, q_e, a_t, r_t, o_{t+1}$);
        if update frequency reached then
            update agent interaction model();
            update agent question answerer model();
        end
    end
    $predicted_e$ ←Answer Question ($M_T$);
    $answer_e$ ←correct answer;
    if agent ended up in correct state then
        save experience ($M_T, q_e, answer_e, predicted_e$);
    end
end
```


S_t : go door <|> -= corridor = - you are in a corridor . a normal kind of place . let 's see what 's in here . oh , great . here 's a conventional oven . that oven contains a red key . there is an open iron gate leading east . there is an open iron door leading south . you need an exit without a door ? you should try going west . there is a yellow onion on the floor . <|> hello

q_e : where is the copper key?

S_{t+1} : go west <|> -= livingroom = - i am sorry to announce that you are now in the livingroom . okay , just remember what you 're here to do , and everything will go great . you can make out a black oven . now why would someone leave that there ? what a letdown ! the oven is empty ! there is an open glass door leading south . there is an exit to the east . there is a copper key on the floor. <|> hello

q_e : where is the copper key?

Figure 2: Overview of our Agent Architecture

layer consists of word-level and character-level embeddings that are aggregated together to produce a vector representation for each token in the text. Word-level embeddings are initialised by the 300-dimensional fastText [23] trained on Common Crawl and kept fixed throughout training. A series of convolutional, linear and highway

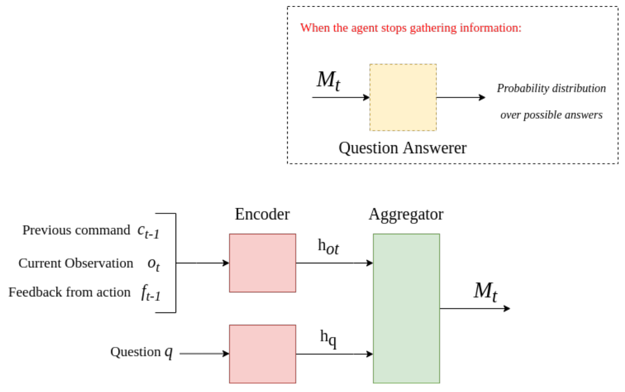
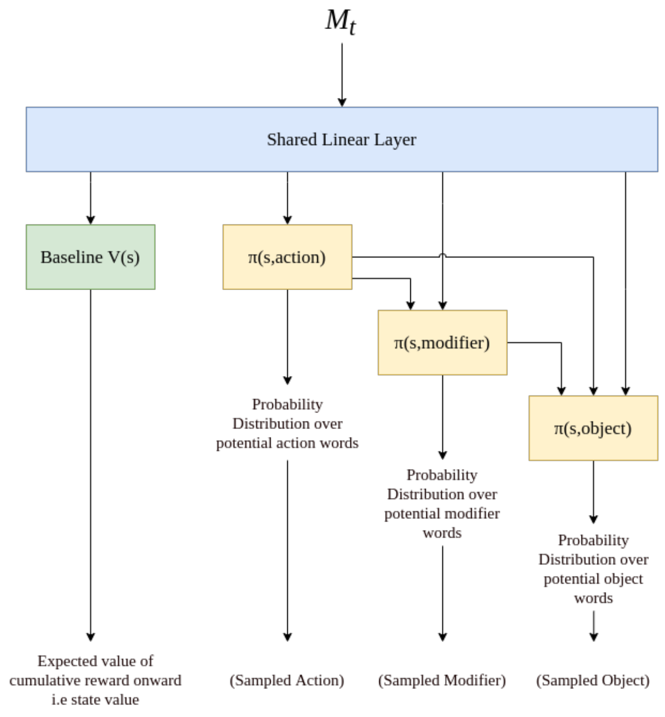

When the agent stops gathering information:
M_t
Probability distribution over possible answers
Question Answerer
Encoder
Aggregator
Previous command c_t-1
Current Observation o_t
Feedback from action f_t-1
h_ot
Question q
h_q
M_t

Figure 3: Overview of QA-DQN Encoder Architecture [41]

network [6] layers are used to aggregate the word and character embeddings. These aggregated embeddings are then input into the transformer blocks. The final output of the transformer blocks is an encoded sequence that is used as input for the agent's policy and baseline functions as well as the question answerer. The question answerer appends an additional stack of aggregation transformer blocks to compute the answer from the given state representation, i.e. the output of the encoder. Further details of implementation can be found in the Yuan et al. QAit paper [41]. At each game step, the current game observation and question are processed and merged together to produce the final state representation that is used by the agent. This gives the agent context of the question at each time step so the agent cannot forget the goal. Figure 3 shows a high-level overview of the architecture.

4.2.2 REINFORCE with Baseline. The policy-based method aims to adapt the existing QAit architecture to improve performance and learning efficiency. The new agent makes use of the existing text encoding architecture, outlined in section 4.2.1, to learn a policy directly as well as the state-value function of the environment. The agent consists of four main components: the QA-DQN encoder, the policy function, the value function and the QA-DQN question answerer. The policy is a parameterised neural network responsible for all interactive decisions. This means it needs to receive an observation from the environment and output the text command it deems best to do in that situation in order to maximise reward, i.e. answer the question. The policy consists of a shared linear layer and three separate output layers. The shared linear layer takes the max pooled output of the text encoder M_t and outputs a fixed-size representation. Each consecutive output layer is conditioned on the previous output along with the shared linear layer (See Figure 4). We did this in an attempt to create more legible commands since consecutive words should be aware of their priors. Each output layer represents the policy for each word in the command triplet (Action, Modifier, Object). Each output gives a probability distribution over the vocab to represent the probabilities with which a word should be selected. Each word in the command triplet is then sampled from the outputted probability distributions to form the command at each game step. This more closely emulates how a human would speak compared to value-based methods, which act greedily and are deterministic. The baseline aims to approximate the state-value function to aid in the training of the policy. The baseline network also makes



M_t
Shared Linear Layer
Baseline V(s)
π(s,action)
π(s,modifier)
π(s,object)
Probability Distribution over potential action words
Probability Distribution over potential modifier words
Probability Distribution over potential object words
Expected value of cumulative reward onward i.e state value
(Sampled Action)
(Sampled Modifier)
(Sampled Object)

Figure 4: Overview of Policy and Baseline Architecture

use of the policy's shared linear layer as input to produce the state value $V(s)$. This is done to regularise the shared linear layer to a common representation to ideally aid the policy and value function in generalisation [27]. The REINFORCE with baseline algorithm is what is used to update model weights. REINFORCE uses Monte-Carlo episodic sample returns G_t (refer to 2.2) to update the baseline and calculate the advantage factor with which to update the policy. Beyond the classic REINFORCE with baseline implementation, each policy's entropy (for each word in the command triplet) is used in the loss function. This is to incentivise more stochastic policies to be learnt whilst maximising reward. This is done in order to promote exploration and generalisation. This slightly altered REINFORCE update is shown in algorithm 2. As discussed in section 2.4.1, the loss functions used for the policy and baseline are:

$$\mathcal{L}_a = \frac{1}{T} \sum_{t=0}^{T} Adv_t * \log \pi(a_t)$$

$$\mathcal{L}_c = \frac{1}{2T} \sum_{t=0}^{T} (Adv_t)^2$$

4.3 Learned Environment Dynamics

The learned encoder representations of textual state observations are only optimised for the policy and baseline loss functions. This can lead to severely overfitted and specific representations that are not useful for any environments beyond the training games. This learned representation can be completely absent of any language semantics, which is seen as important for an agent's ability to apply its knowledge to new environments. In text games, a semantic rich representation is proposed as a way to increase generalisation

Algorithm 2: REINFORCE Update

```
// Note : $\pi(A_t) = Probability$ of action $A$ at time-step $t$
for time step $t$ in episode trajectory $(S_t, A_t, R_t)$ do
    $G_t \leftarrow \sum_{k=t}^{T} R_t$;
    $Adv_t \leftarrow G_t - V(S_t)$;
    $act, mod, obj = A_t$;
    $\mathcal{L}_{a_t} = (\pi(act) + \pi(mod) + \pi(obj)) * Adv_t$;
    $\mathcal{L}_{c_t} = (Adv_t)^2$;
    $\mathcal{L}_{e_t} = Entropy(\pi(act)) + Entropy(\pi(mod)) +$
      $Entropy(\pi(obj))$;
end
$\mathcal{L} = \frac{1}{T} \sum_{k=0}^{T} (\mathcal{L}_{a_t} + \mathcal{L}_{c_t} - \beta * \mathcal{L}_{e_t})$;
Calculate Gradients;
Perform one step of gradient descent;
```

performance as well as prevent overfitting. Yao et al. [40] show that an inverse dynamics model can be used to regularise Q values, promote encoding of action-relevant observations, and provide a form of intrinsic motivation for exploration. A further evaluation of the transfer learning capabilities of semantic rich representations show benefits for generalisation. In this paper, the environment dynamics model is used to improve generalisation capability and illustrate its effectiveness as a regularisation technique. Inspired by Pathak et al. [24], the environment dynamics model consists of a forward and inverse dynamics model. The forward dynamics model takes in a state representation and an action to predict the next state representation. This attempts to model an agent's predictive understanding of the world. The model consists of two linear layers and makes use of MSE loss. The inverse dynamics model uses two consecutive state representations and attempts to predict the action that caused the state transition (see Figure 5). This attempts to model an understanding of world dynamics. Since each command can be seen as a triplet of (Action, Modifier, Object), the inverse model has three separate neural decoders that attempt to predict each word in the triplet. Each decoder uses cross-entropy loss upon which are summed $\mathcal{L}_{inverse} = \sum_{i=1}^{3} \mathcal{L}_i$. The losses of the forward and inverse models are joined with a weighted addition and added to the joint policy-baseline loss:

$$\mathcal{L} = \lambda(\mathcal{L}_a + \mathcal{L}_c) + (1 - \beta) * \mathcal{L}_{inverse} + \beta * \mathcal{L}_{forward}$$

Upon initial experimentation, we proceed only to make use of the inverse model and loss, as seen in Yao et al. [40], to regularise the transformers state representation. Differing from Yao et al., we do not make use of the inverse loss as an intrinsic reward as it seemed to inhibit training strongly. We hypothesise that this occurs by creating too much noise in the reward function due to different environments containing slightly altered dynamics, ultimately preventing the loss from decreasing sufficiently. Furthermore, we do believe that there is a benefit to the use of the forward model, but this requires further experimentation.

5 EXPERIMENT METHODOLOGY

5.1 Evaluation Metrics

The following are the evaluation metrics used in order to evaluate different aspects of performance.

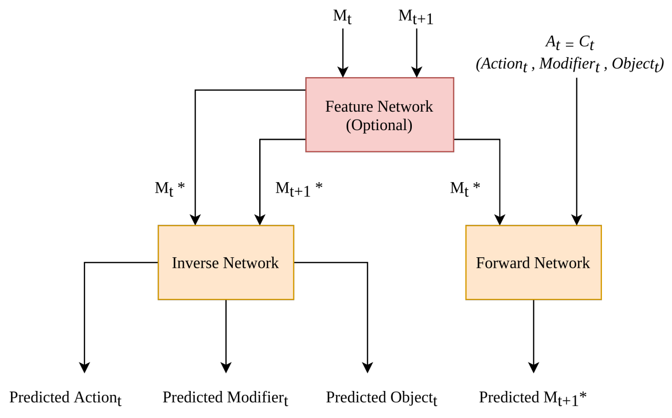


Figure 5: Overview of Environment-Dynamics Architecture - M_t* is the transformed representation - since this transformation is optional if not used $M_t* = M_t$

5.1.1 Accuracy. Accuracy refers to the proportion of correctly answered questions and is deemed the most important metric since, ultimately, the goal of IQA is to answer a question. Other metrics used alongside accuracy to evaluate question-answering performance are precision, recall and F1 score. However, these take lower precedence due to the well-balanced answer distribution of the test set and are not reported. The distribution of answers in the QAit evaluation set is presented in the appendix (see Table 10, 11 and 12).

5.1.2 Sufficient Information. Sufficient information is a metric used to evaluate the amount of information gathered by the agent and whether or not the information was sufficient to answer the question [41]. It is also used as part of the reward function (see section 4.1.4). This is a metric to evaluate the performance of the navigation and interaction required to answer a given question. The sufficient information score is calculated when the agent decides to stop the interaction and answer the question. For each question type, the sufficient information score is calculated as follows:

- Location: A score of 1 is given if, when the agent decides to stop the interaction, the entity mentioned in the question is present in the final observation. This indicates the agent has witnessed the information it needs to answer the question successfully. If the mentioned entity is not present in the final observation then a score of 0 is given.
- Existence: If the answer to the question is ***yes*** then a score of 1 is given if the entity mentioned in the question is present in the final observation. If the answer to the question is ***no***, then a score between 0 and 1 is given proportional to the amount of exploration coverage of the environment the agent has performed. Intuitively this can be seen as a confidence score - if the agent witnesses the entity, it is 100% confident of its existence; otherwise, until it explores the entire environment, it is not 100% confident.
- Attribute: Attribute questions have a set of heuristics defined to verify each attribute and assign a score of sufficient information. The set of heuristics can be seen in the appendix (see Table 9). Each attribute has specific commands that need to be executed for sufficient information to be gathered. This also depends on the agent being in certain states for these outcomes to be observed correctly, e.g. an agent needs to be holding an object to try to eat the object.

5.1.3 Episodes Experienced. A measure for sample efficiency is the number of training episodes required to achieve a certain level of performance. Sample efficiency is crucial when computing resources and time is limited. The number of episodes the agent has seen dictates how much experience was available for an agent to make use of and learn from.

5.2 Experiments

Three primary experiments were performed under four different environment configurations. The primary experiments were:

- REINFORCE with Baseline
- REINFORCE with Baseline using Environment Dynamics Model
- DQN using Environment Dynamics Model

The environment configurations were:

- Random/Fixed Map with 1 game
- Random/Fixed Map with 500 games

Each experiment was trained on a single GPU and took approximately 3 - 8 days, depending on the question type and experiment. For single-game experiments, policy-based agents were trained for one hundred thousand episodes, and DQN with semantic regularisation were trained for 200 000 episodes. This is because the policy-based methods converge much earlier. For the 500 game experiments, all agents were trained for 200 000 episodes.

6 RESULTS AND DISCUSSION

Table 4: Agent performance on fixed map zero-shot test games when trained on 1 game and 500 games settings. Note Att. and Exi. are binary questions with expected accuracy of 0.5.

	Fixed		
Model	**Loc.**	**Exi.**	**Att.**
Random	0.027	0.497	0.496
1 game			
DQN	0.122 (0.160)	0.628 (0.124)	0.500 (0.035)
DDQN	0.156 (0.178)	0.624 (0.148)	0.498 (0.033)
Rainbow	0.164 (0.178)	0.616 (0.083)	**0.516 (0.039)**
DQN w/ Semantics	0.152 (0.158)	0.624 (0.122)	0.490 (0.063)
Policy-based	0.168 (0.172)	0.584 (0.217)	0.514 (0.060)
Policy-based w/ Semantics	**0.182 (0.184)**	**0.630 (0.228)**	0.494 (0.068)
500 games			
DQN	0.224 (0.244)	0.674 (0.279)	**0.534(0.014)**
DDQN	0.218 (0.228)	0.626 (0.213)	0.508 (0.026)
Rainbow	0.190 (0.196)	0.656(0.207)	0.496 (0.029)
Policy-based	**0.948 (0.958)**	**0.948 (0.892)**	0.466 (0.045)
Policy-based w/ Semantics	0.748 (0.768)	0.932 (0.872)	0.506 (0.044)

Table 5: Agent performance on random map zero-shot test games when trained the 500 games setting. Note Att. and Exi. are binary questions with expected accuracy of 0.5.

Random			
Model	**Loc.**	**Exi.**	**Att.**
Random	0.034	0.5	0.499
500 games			
DQN	0.204 (0.216)	0.678 (0.214)	0.530 (0.017)
DDQN	0.222 (0.246)	0.656 (0.188)	0.486 (0.023)
Rainbow	0.172 (0.178)	0.678 (0.191)	0.494 (0.017)
Policy-based	**0.570 (0.588)**	0.836 (0.560)	**0.534 (0.044)**
Policy-based w/ Semantics	0.440 (0.458)	**0.868 (0.672)**	0.514 (0.039)

6.1 Accuracy and Sufficient Information Results

6.1.1 Policy-based method. Table 4 and 5 give the QAit test set results for all experiments as well as show the baseline models' performance. Table 6 and 7 shown in the appendix give the full results including training performance. It is clear that a policy-based method outperforms value-based methods both in training and testing performance. When training on one game, the policy-based agent overfits more than the baselines by achieving 100% accuracy on all three question types along with high sufficient information scores - we hypothesise that this is due to the policy-based agent making more efficient use of data, thereby in the same training period overfitting to a greater extent. Despite this, we can see that the testing performance of the policy-based method is still arguably better than the baseline methods by showing similar question accuracy performances but a significant increase in sufficient information scores. This indicates the policy itself is performing better, with respect to navigation and interaction, but the overfitted QA model inhibits performance when trying to answer questions.

The 500 games experiment shows a better indication of the performance increase compared to the baselines by achieving significantly higher results. For location type questions in the fixed map setting, the policy-based agent achieves an accuracy of 94.8% on the test-set. This is a large improvement over QAit's state-of-the-art result of 22.4%. We see similar scale improvements in existence type questions for the fixed map setting where the policy-based model achieved an accuracy of 94.8%. This is compared to QAit's 67.4% accuracy. This comparison doesn't capture the true performance increase, which can be seen by looking at sufficient information score where we see a dramatic improvement in navigation and interaction. The policy-based method scores 0.892 compared to QAit's best score of 0.279. The results of the random map setting are also a large improvement over the baselines, albeit not as stark as in fixed maps - we believe this is because of the more difficult nature of random map type games as well as having almost double the possible answers (see Table 10). Interestingly, in the fixed map setting, the results of the attribute type questions show worse QA accuracy but greater sufficient information score. As one can see for all attribute type questions, neither the policy-based model nor the baselines achieve results much higher than the random baseline in terms of QA accuracy. By looking at the sufficient information score, we can see that neither model truly ends up in the states they should be in therefore these results are most likely due to chance.

We see that a policy-based method is more capable of learning in a text-based environment indicated by its ability to completely learn the single-game training data. The 500 games results shows that the stochastic nature of the policy-based method affords it enough flexibility to learn more generalised policies that can be learned to jointly perform in numerous training environments. Lastly, the 500 games results also show that the learnt policy does not learn specific environmental information but environment agnostic behaviour.

6.1.2 Environment Dynamics. Using the environment dynamics model to promote semantic encoding and prevent overfitting appears to be successful in certain circumstances. As can be seen in the single-game setting, the use of the environment dynamics model helped the policy-based agent achieve the highest results on the test-set (see Table 4) even though the training performance was the same. This indicates the learned representation of state captures more generally applicable knowledge that aids in zero-shot performance. A small follow-up experiment was performed where the weights of the text encoder were frozen for a semantically regularised agent and a standard policy-based agent after being trained on the single-game setting. The agents have only their policy and baseline functions fine-tuned on new environments. The semantic agent achieved much higher scores consistently from the start indicating the semantic encoding does, in fact, improve generalisation, at least when access to multiple training environments is not available. This result is supported by Yao et al.'s findings on the transfer learning capability of semantic regularisation [40]. As the number of training games increases - the effect of the semantic regularisation decreases and becomes less necessary. The results seen when applying the environment dynamics model on 500 training games does not seem to increase performance and simply slows down learning. We believe this to be due to an agent being required to learn some form of semantics if it wishes to perform well on a large number of different environments in its training set, thereby making the regularisation unnecessary. When applying the semantic regularisation to the DQN model trained in the single-game setting, we see improved results for location type questions, but worse results for existence and attribute. This seems to suggest that the environment dynamics model has a greater effect on the policy-based method or that longer training times are required to see the benefit.

Since policy-based methods can greatly overfit in the single-game setting, an environment dynamics model is seen to be a helpful regularisation technique when access to multiple training environments is not possible. Conversely, when access is possible, the environment dynamics model inhibits learning and performance, thereby adding no observable benefit.

6.2 Sample Efficiency Results

Using the training curves seen in figure 6 (as well as 7 and 9 - seen in appendix) as examples of the general performance, we see that the policy-based method performs better in training and testing and achieves much higher performance using much less data than all the baseline methods, indicating a higher sample efficiency in this domain. After 30 000 episodes of training in the 500 games setting, the REINFORCE agent achieves results on par with the baselines. This is significant as the baselines were trained for over 200 000

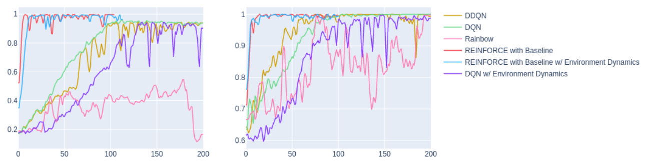


Figure 6: Training accuracy curve of single-game setting. Location and existence on the left and right respectively.

episodes - indicating that the REINFORCE agent has a greatly improved sample efficiency. Furthermore, the gradient of improvement is extremely steep. This is mirrored in the single-game setting where the REINFORCE agent achieves the baselines' results in less than 15 000 episodes (see Figure 6). This is a notable result since, in general, off-policy methods, such as the QAit baselines, are more sample efficient than on-policy methods, e.g. REINFORCE. Additionally, the REINFORCE agent only updates weights once per episode, whereas the baselines update their weights every 20 steps, which can potentially be twice or three times an episode. This indicates that learning a policy in the QAit environment is an easier task than approximating the Q-function. The use of the environment dynamics model seems to lower sample efficiency for both the DQN and policy-based methods. This is believed to be due to the extra difficulty of learning game semantics thereby requiring more data points.

The improvement in sample efficiency indicates that policy-based methods are more suited to text-based environments. This increase also alleviates the computational resources required at scale, thereby aiding the feasibility of future research since less training data is needed.

6.3 Effect of Increasing Training Games

A trend seen in the original QAit paper [41] is seen here as well. As the number of training games increases, so does the policy-based agent's ability to generalise to new unseen environments. The effect of increasing training games seems to increase the policy-based agents' generalisation capability at a faster rate than the baseline methods as seen by performance difference between the single-game and 500 games settings.

6.4 Replicability

All experiments are completely and easily reproducible. The code will be open-sourced and available on GitHub. Each experiment makes use of a specific random seed. This seed is given in the available configuration file. The configuration file is set to reproduce experiments apart from the necessary changes required for each run, such as enabling the REINFORCE with Baseline algorithm. All experiment models are also saved and available to load and use. The models will be available on the GitHub repository. The hyper-parameters used for the policy-based agent and environment dynamics model are given in table 8 and the configuration file.

7 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

The QAit codebase and baselines provided by Yuan et al. [41] has been made available using an MIT License, allowing for unrestricted use. Since QAit is thus open-source, there are no foreseeable legal issues. Due to the lack of human or animal subjects in conjunction with having no privacy breaching experiments or data collection, we see that there are no associated ethical concerns. In terms of professional issues, project members will follow the Open Source Software guidelines described by Grodzinsky et al. [17] and ensure that the use of QAit will be done in such an appropriate professional and ethical manner.

8 LIMITATIONS

A large limitation for this paper was computational resources and available time for training models. Additionally, due to these computational and temporal limitations, only a select few of the deemed important game configurations were used in experiments. Lastly, hyper-parameter tuning was only minimally performed meaning the results shown could be sub-optimal.

9 CONCLUSION

This research presented the advantages of using a policy-based method in textual environments, specifically the QAit task, as well as the potential for a new regularisation technique to aid in textual generalisation. More specifically, we investigated the differences in training, sample efficiency and evaluation performances of the REINFORCE with baseline algorithm compared to three popular value-based reinforcement learning baselines: DQN, DDQN and Rainbow. We further investigated the use of a semantic regularisation technique to improve the baseline and policy methods performance. The results produced strongly suggest that policy-based reinforcement learning methods are not only more suited for textual domains due to their training sample efficiency and performance, but they also possess generalisation capabilities beyond their value-based counterparts. Secondly, the results also show that the use of an environment dynamics model for encoding semantics is a viable regularisation technique when access to more than one training game is not possible.

10 FUTURE WORK

Curiosity: A large problem with the traditional Intrinsic Curiosity Module (ICM) [24] approach is that it was not designed for multiple or procedural training environments. An issue that can occur is an agent never learns the outcome of a state-action pair due to environments having different world structures, thereby always providing intrinsic reward - this defeats the purpose of the ICM by not allowing an agent to learn the dynamics of the world correctly. A new way of providing intrinsic motivation in procedural environments would provide great benefit to the IQA task in text-based environments as rewards are sparse and exploration can be hard, as seen in attribute type questions. Furthermore, the use of the forward dynamics model in the process of semantic regularisation should be investigated.

Architecture: The current architecture used is very complex and could potentially be replaced with a much simpler end-to-end transformer based architecture. An investigation into removing the complexity would be useful in identifying the most important architectural aspects required for the IQA task. This would speed up learning as well as require less computational resources.

11 ACKNOWLEDGEMENTS

A huge thank you to both Dr Buys and Dr Shock for their invaluable help throughout this research project. Thank you as well to the UCT HPC team for their support. Lastly, thank you to Dr Yuan and the rest of the QAit team for their initial research as well as providing us with their training logs.

REFERENCES

[1] Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3557–3565. https://doi.org/10.18653/v1/N19-1358

[2] Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3557–3565. https://doi.org/10.18653/v1/N19-1358

[3] Charles Packer an. 2018. Assessing Generalization in Deep Reinforcement Learning. *ArXiv preprint* abs/1810.12282 (2018). https://arxiv.org/abs/1810.12282

[4] Greg Brockman an. 2016. OpenAI Gym. *ArXiv preprint* abs/1606.01540 (2016). https://arxiv.org/abs/1606.01540

[5] Jesse Farebrother an. 2018. Generalization and Regularization in DQN. *ArXiv preprint* abs/1810.00123 (2018). https://arxiv.org/abs/1810.00123

[6] Rupesh Kumar Srivastava an. 2015. Highway Networks. *ArXiv preprint* abs/1505.00387 (2015). https://arxiv.org/abs/1505.00387

[7] Volodymyr Mnih an. 2013. Playing Atari with Deep Reinforcement Learning. *ArXiv preprint* abs/1312.5602 (2013). https://arxiv.org/abs/1312.5602

[8] Xingdi Yuan an. 2018. Counting to Explore and Generalize in Text-based Games. *ArXiv preprint* abs/1806.11525 (2018). https://arxiv.org/abs/1806.11525

[9] Yolanda Gil an. 2019. A 20-Year Community Roadmap for Artificial Intelligence Research i. *ArXiv preprint* abs/1908.02624 (2019). https://arxiv.org/abs/1908.02624

[10] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 449–458. http://proceedings.mlr.press/v70/bellemare17a.html

[11] Abdelghani Bouziane, Djelloul Bouchiha, Noureddine Doumi, and Mimoun Malki. 2015. Question Answering Systems: Survey and Trends. *Procedia Computer Science* 73 (2015), 366–375. https://doi.org/10.1016/j.procs.2015.12.005 International Conference on Advanced Wireless Information and Communication Technologies (AWICT 2015).

[12] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*. Springer, 41–75.

[13] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied Question Answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 1–10. https://doi.org/10.1109/CVPR.2018.00008

[14] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. 2018. Noisy Networks For Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rywHCPkAW

[15] Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural Approaches to Conversational AI. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 1371–1374. https://doi.org/10.1145/3209978.3210183

[16] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual Question Answering in Interactive Environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 4089–4098. https://doi.org/10.1109/CVPR.2018.00430

[17] F. Grodzinsky, K. Miller, and M. J. Wolf. 2003. Ethical issues in open source software. *J. Inf. Commun. Ethics Soc.* 1 (2003), 193–205.

[18] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. *Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases*. Association for Computing Machinery, New York, NY, USA, 3477–3488. https://doi.org/10.1145/3442381.3449992

[19] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3215–3222. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204

[20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466. https://doi.org/10.1162/tacl_a_00276

[21] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8, 3-4 (1992), 293–321. https://doi.org/10.1007/bf00992699

[22] Andrea Madotto, Mahdi Namazifar, Joost Huizinga, Piero Molino, Adrien Ecoffet, Huaixiu Zheng, Alexandros Papangelis, Dian Yu, Chandra Khatri, and Gökhan Tür. 2020. Exploration Based Language Learning for Text-Based Games. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 1488–1494. https://doi.org/10.24963/ijcai.2020/207

[23] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Miyazaki, Japan. https://aclanthology.org/L18-1008

[24] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2778–2787. http://proceedings.mlr.press/v70/pathak17a.html

[25] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.05952

[26] Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3135–3144. http://proceedings.mlr.press/v70/shi17a.html

[27] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (Oct. 2017), 354–. http://dx.doi.org/10.1038/nature24270

[28] Dan Su, Yan Xu, Genta Indra Winata, Peng Xu, Hyeondey Kim, Zihan Liu, and Pascale Fung. 2019. Generalizing Question Answering System with Pre-trained Language Model Fine-tuning. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, Hong Kong, China, 203–211. https://doi.org/10.18653/v1/D19-5827

[29] Richard S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3, 1 (1988), 9–44. https://doi.org/10.1007/bf00115009

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

[31] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press. https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

[32] Sebastian Thrun and Anton Schwartz. 1993. Issues in Using Function Approximation for Reinforcement Learning. In *In Proceedings of the Fourth Connectionist Models Summer School*. Erlbaum.

[33] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A Machine Comprehension Dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Vancouver, Canada, 191–200. https://doi.org/10.18653/v1/W17-2623

[34] John N. Tsitsiklis and Benjamin Van Roy. 1997. *An analysis of temporal-difference learning with function approximation*. Technical Report. IEEE Transactions on Automatic Control.

[35] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, Dale

Schuurmans and Michael P. Wellman (Eds.). AAAI Press, 2094–2100. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[37] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 1995–2003. http://proceedings.mlr.press/v48/wangf16.html

[38] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292. https://doi.org/10.1007/bf00992698

[39] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3-4 (1992), 229–256. https://doi.org/10.1007/bf00992696

[40] Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and Acting while Blindfolded: The Need for Semantics in Text Game Agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3097–3102. https://doi.org/10.18653/v1/2021.naacl-main.247

[41] Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Chris Pal, Yoshua Bengio, and Adam Trischler. 2019. Interactive Language Learning by Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2796–2813. https://doi.org/10.18653/v1/D19-1280

[42] Xingdi Yuan, Jie Fu, Marc-Alexandre Côté, Yi Tay, Chris Pal, and Adam Trischler. 2020. Interactive Machine Comprehension with Information Seeking Agents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2325–2338. https://doi.org/10.18653/v1/2020.acl-main.211

[43] Wenfei Zhang, Zhixin Suo, Ming Gao, Hongzhi Lu, Guoqin Lun, and Xuhua Zhang. 2021. Research on Enhancing Generalization Ability of Question Answering System by Retelling Technology. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. 982–985. https://doi.org/10.1109/ICBAIE52039.2021.9390073

A FULL RESULTS

Table 6 and 7 show training and testing performances of all models.

B HYPER-PARAMETERS

The hyper-parameters used in experiments are shown in table 8.

C HEURISTICS FOR ATTRIBUTE QUESTIONS

The heuristics used to derive sufficient information scores are given in table 9. Each attribute shows the command, starting state and score given a passing or failing outcome.

D QAIT TEST SET ANSWER DISTRIBUTION

The QAit test set's answer distribution is shown in tables 10, 11, and 12.

E TRAINING CURVES

Accuracy and sufficient information training curves for fixed and random 500 games experiments and single game experiments are shown in figure 7,8, 9, and 10.

Table 6: Results of Fixed Map Experiments - Accuracy is shown in black - Sufficient Information is shown in blue (refer to section 5.1)

Fixed						
Model	**Location**		**Existence**		**Attribute**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Random	-	0.027	-	0.497	-	0.496
1 game						
DQN	0.972 (0.972)	0.122 (0.160)	1.000 (0.881)	0.628 (0.124)	1.000 (0.049)	0.500 (0.035)
DDQN	0.960 (0.960)	0.156 (0.178)	1.000 (0.647)	0.624 (0.148)	1.000 (0.023)	0.498 (0.033)
Rainbow	0.562 (0.562)	0.164 (0.178)	1.000 (0.187)	0.616 (0.083)	1.000 (0.049)	**0.516 (0.039)**
DQN w/ Semantics	0.962 (0.962)	0.152 (0.158)	1.000 (0.876)	0.624 (0.122)	0.998 (0.203)	0.490 (0.063)
Policy-based	1.000 (1.000)	0.168 (0.172)	1.000 (0.933)	0.584 (0.217)	1.000 (0.216)	0.514 (0.060)
Policy-based w/ Semantics	1.000 (1.000)	**0.182 (0.184)**	1.000 (0.932)	**0.630 (0.228)**	0.948 (0.197)	0.494 (0.068)
500 games						
DQN	0.430 (0.430)	0.224 (0.244)	0.742 (0.136)	0.674 (0.279)	0.700 (0.015)	**0.534 (0.014)**
DDQN	0.406 (0.406)	0.218 (0.228)	0.734 (0.173)	0.626 (0.213)	0.714 (0.021)	0.508 (0.026)
Rainbow	0.358 (0.358)	0.190 (0.196)	0.768 (0.187)	0.656 (0.207)	0.736 (0.032)	0.496 (0.029)
Policy-based	0.990 (0.990)	**0.948 (0.958)**	0.964 (0.916)	**0.948 (0.892)**	0.748 (0.048)	0.466 (0.045)
Policy-based w/ Semantics	0.886 (0.888)	0.748 (0.768)	0.958 (0.917)	0.932 (0.872)	0.580 (0.043)	0.506 (0.044)

Table 7: Results of Random Map Experiments - Accuracy is shown in black - Sufficient Information is shown in blue (refer to section 5.1)

Random						
Model	**Location**		**Existence**		**Attribute**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Random	-	0.034	-	0.5	-	0.499
500 games						
DQN	0.430 (0.430)	0.204 (0.216)	0.752 (0.162)	0.678 (0.214)	0.678 (0.019)	0.530 (0.017)
DDQN	0.458 (0.458)	0.222 (0.246)	0.754 (0.158)	0.656 (0.188)	0.716 (0.024)	0.486 (0.023)
Rainbow	0.370 (0.370)	0.172 (0.178)	0.748 (0.275)	0.678 (0.191)	0.636 (0.020)	0.494 (0.017)
Policy-based	0.818 (0.818)	**0.570 (0.588)**	0.866 (0.628)	0.836 (0.560)	0.754 (0.045)	**0.534 (0.044)**
Policy-based w/ Semantics	0.820 (0.820)	0.440 (0.458)	0.852 (0.675)	**0.868 (0.672)**	0.594 (0.042)	0.514 (0.039)

Table 8: Hyper-parameter set up for REINFORCE agent and ICM semantics regularization

Parameter	**Value**
Entropy Coefficient	0.05
Learning Rate	0.00025
Optimizer	Adam
Max Number of Steps	50
Random Seed	42
ICM Beta	0
ICM Lambda	1
ICM Hidden Size	128

Table 9: Heuristic conditions for determining whether the agent has enough information to answer a given attribute question. We use "object" to refer to the object mentioned in the question. Words in italics represent placeholders that can be replaced by any object from the environment that has the appropriate attribute (e.g. carrot could be used as a cuttable). Pass and Fail columns represent how much reward the agent will receive given the corresponding command's outcome (resp. success or failure). [41]

Attribute	Command	State	Pass	Fail	Explanation
sharp	cut cuttable	holding (cuttable) & uncut (cuttable) & holding (object)	1	1	Trying to cut something cuttable that hasn't been cut yet while holding the object.
	take object	reachable(object)	0	1	Sharp objects should be portable.
cuttable	cut object	holding (object) & holding (sharp)	1	1	Trying to cut the object while holding something sharp.
	take object	reachable (object)	0	1	Cuttable object should be portable.
edible	eat object	holding (object)	1	1	Trying to eat the object.
	take object	reachable (object)	0	1	Edible objects should be portable.
drinkable	drink object	holding (object)	1	1	Trying to drink the object.
	take object	reachable (object)	0	1	Drinkable objects should be portable.
holder	-	on (portable, object)	1	0	Observing object(s) on a supporter.
		in (portable, object)	1	0	Observing object(s) inside a container.
	take object	reachable (object)	1	0	Holder objects should not be portable.
portable	-	holding (object)	1	0	Holding the object means it is portable.
	take object	reachable (object)	1	1	Portable objects can be taken.
heat_source	cook cookable	holding (cookable) & uncooked (cookable) & reachable (object)	1	1	Trying to cook something cookable that hasn't been cooked yet while being next to the object.
	take object	reachable (object)	1	0	Heat source objects should not be portable.
cookable	cook object	holding (object) & reachable (heat_source)	1	1	Trying to cook the object while being next to a heat source.
	take object	reachable(object)	0	1	Cookable objects should be portable.
openable	open object	reachable (object) & closed (object)	1	1	Trying to open the closed object.
	close object	reachable (object) & open (object)	1	1	Trying to close the open object.

Table 10: Answer distribution for location type questions

Map Type:	Fixed	Random
pantry	68	39
livingroom	87	34
shed	89	44
inventory	27	32
corridor	79	41
bedroom	75	32
driveway	75	38
street	-	38
bathroom	-	46
supermarket	-	29
garden	-	38
backyard	-	51
driveway	-	38

Table 11: Answer distribution for existence type questions.

Map Type:	Fixed	Random
yes	252	237
no	248	263

Table 12: Answer distribution for attribute type questions.

Map Type:	Fixed	Random
yes	242	236
no	258	264

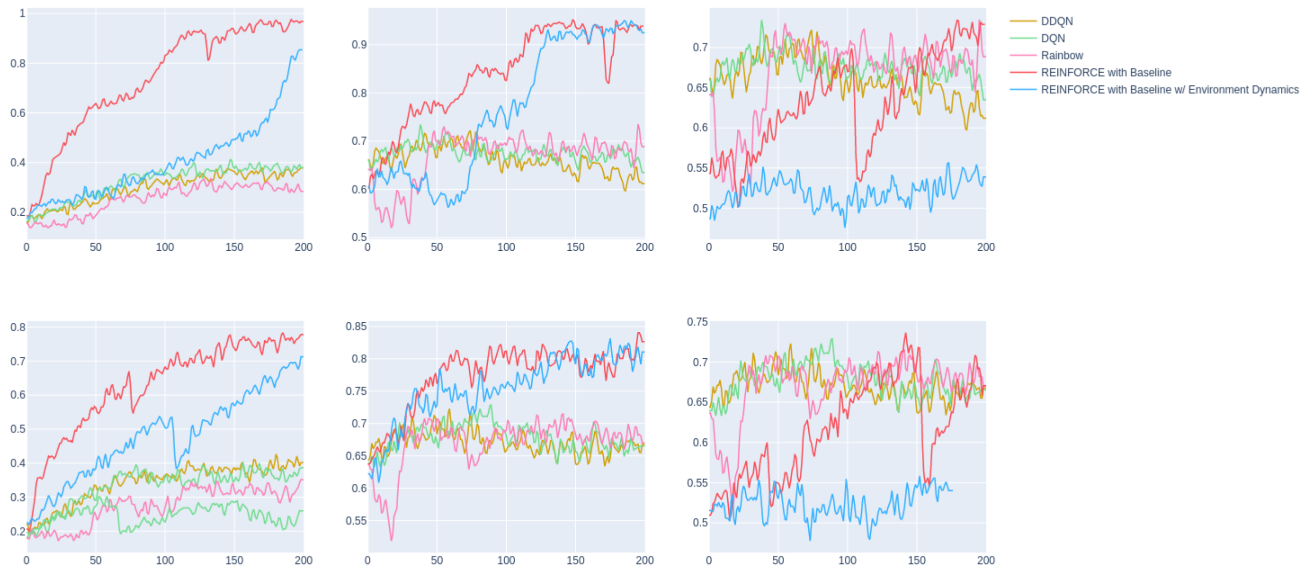


Figure 7: Training accuracy curves of policy-based agent and baselines on 500 games. Top row shows fixed map games and bottom row shows random map games. Each column shows a different question type - location - existence - attribute

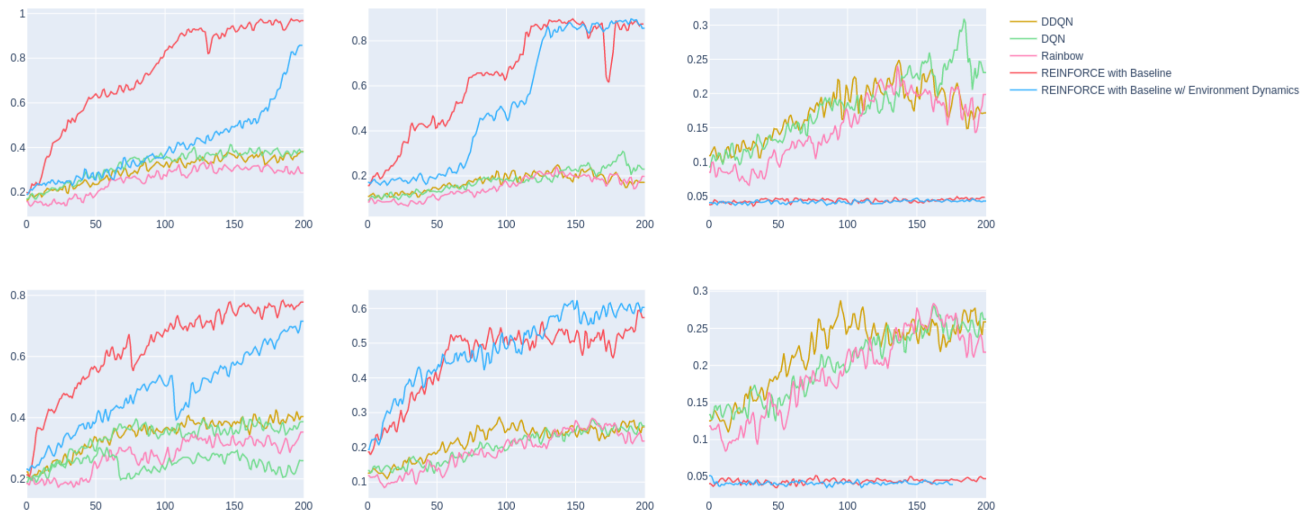


Figure 8: Sufficient Information curves of policy-based agent and baselines on 500 games. Top row shows fixed map games and bottom row shows random map games. Each column shows a different question type - location - existence - attribute

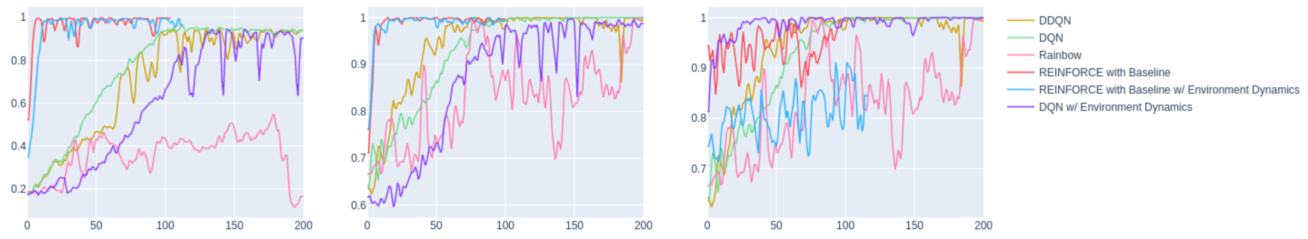


Figure 9: Training accuracy curves of policy-based agent and baselines on a single game setting. Each column shows a different question type - location - existence - attribute

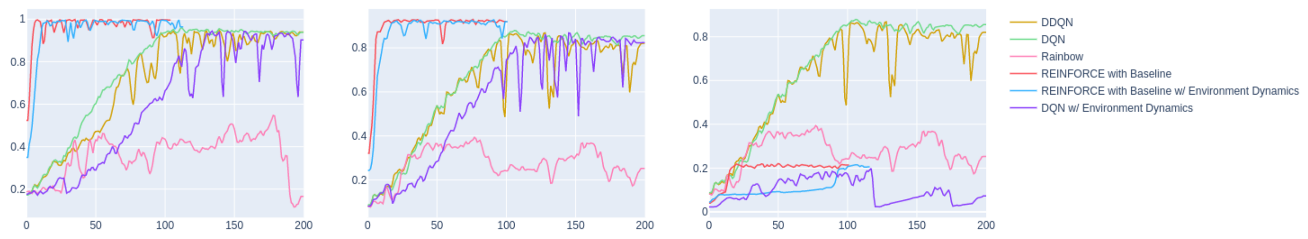


Figure 10: Sufficient Information curves of policy-based agent and baselines on a single game setting. Each column shows a different question type - location - existence - attribute