IQA
Literature Review

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

Abstract

This literature review aims to investigate the viability of a reinforcement learning based approach to question answering. We reviewed the state-of-the-art RL methodologies to examine the applicability to question answering specifically within the interactive question answering domain. We also assessed the viability of using text-based environments as a method of teaching language understanding. Evidence suggests that advanced exploration methods will be critical in the success of RL in IQA. Even though IQA can be framed as an RL problem, we would suggest that the integration of tried and tested RL, QA and NLP methods would be utilised for future optimal systems.

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

1 Introduction

Throughout recent years, question answering (QA) systems have become progressively more useful 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 [8] as the need to query large amounts of information becomes more prevalent. Despite advances in recent years, current QA systems often fail to generalize to unseen and out of domain information. While attempts have been made to better generalization capabilities [30, 56, 76] results have not seen significant improvement. These shortcomings have motivated research into training a system to *comprehend* the contextual semantics of a question in order to yield more accurate and relevant answers. Hence, current literature has proposed *interactive question answering* (IQA) as a viable solution. IQA refers to a system needing to interact with its environment to find answer to a question. A growing body of evidence has shown reinforcement learning coupled with text-based environments to be a promising avenue for training agents to gain contextual awareness and understanding thus achieving sufficient comprehension of their domain. This paper aims to provide a brief overview of relevant question answering systems, natural language processing concepts as well as a more in-depth review of reinforcement learning literature. Subsequently, it discusses the role of reinforcement learning within the question and interactive question answering domain while also investigating text-based environments as a viable avenue for IQA.

2 Question Answering Systems

Question answering (QA) is a field largely based in natural language processing and information retrieval [24]. It involves developing systems that respond to questions presented by humans in natural language. There are two types of question answering systems: open and closed domain [48]. Closed domain systems are only interested in answering questions that fit inside a strictly specified topic. Open-domain systems, on the other hand, are designed to answer questions on any subject. These systems normally require a large volume of publicly available data.

Question Answering systems can broadly be divided into two paradigms. Information Retrieval and Knowledge-Based [35].

2.1 Information Retrieval

Question answering systems based on information retrieval filters through large collections of documents and text on the internet (or in ontologies) [35] and applies ranking to find relevant passages that most likely contain the answer. Once these passages are found, neural reading comprehension algorithms are deployed to extract the answer from found passages [35].

2.2 Knowledge-Based

Knowledge-Base QA systems rely upon mapping natural language to some meaning representation (refer to section 3.2) which ultimately can be structured into data queries that can be executed on databases or knowledge bases [35].

2.3 Interactive Question Answering

Interactive question answering [19, 29, 74] refers to systems that can answer questions by interacting with the environment to find the answer. Interactive question answering can be seen as a combination of closed and open domain question answering systems without the need for extremely large datasets. Ultimately the goal of IQA is creating a system that is completely generalizable to any environment.

3 Natural Language Processing

Natural language processing (NLP) is regarded as the combined study of linguistics, computer science and machine learning. NLP concerns itself with *how* computers can interact with and *understand* natural human language. NLP is used in many fields for a vareity of tasks [39]. In the context of QA, most modern QA systems are said to integrate some form of NLP [24]. The following are a few relevant concepts within NLP:

3.1 Word Embeddings

Word embeddings are a numerical representations of words that are used in text-based computation [5, 17]. Word embeddings reduce the dimensionality of text-based problems so that computers can feasibly process text. These embeddings serve to capture relevant/useful information even in the absence of sentence structure [13]. Research has been done into the use of word embeddings to increase QA system performance [21]. By combining word embeddings with other features, performance increases and answers can extracted with greater ease [21]. The following is a high level overview of a popular word embedding technique.

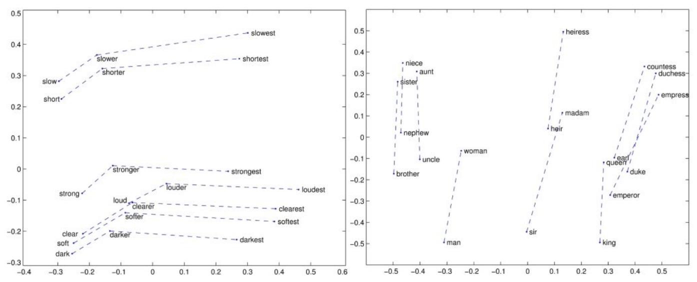

Figure 1. Example of Embedded Word Vector Relations [51]

3.1.1 Word2Vec.

Word2Vec [45] is one of the most popular methods of constructing embeddings for words in a vocabulary. The embeddings are constructed and learned by using a Skip Gram [44] model or a continuous bag-of-words model [45]. Both models use neural networks (NN) to learn a prediction task and apply the learned weights as a contextual numeric representation of a word. [45] show that Skip-Gram performs well with small data and can represent *rare/uncommon* words better whereas CBOW develops better representations for high-frequency words and has a faster training speed. The numeric vector representations of words produced by Word2Vec and other embedding techniques can capture syntactic and semantic information about words and their context [65]. It also places similar words and concepts closer together in vector space (refer to figure 1) which can be highly useful in text-related tasks.

3.2 Semantic Parsing

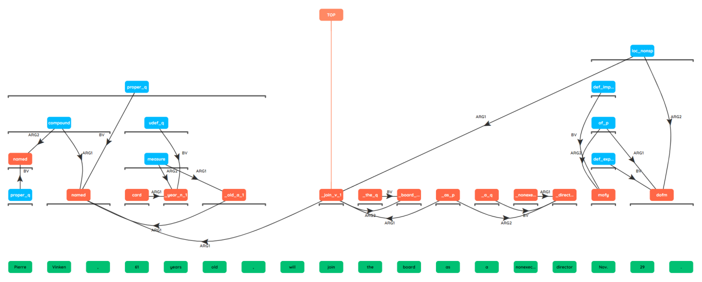

Figure 2. Example of Semantic Graph

Semantic parsers try to express natural language in a logical form (semantic representation) that a computer can understand and potentially act upon [38]. Semantic parsing is used in a variety of domains such as question answering [6, 34], machine translation [2] and code generation [52]. See figure 2 for an example of a type of semantic representation - a semantic graph.

Semantic parsing has been applied in the question-answering domain as a way to convert natural language questions into actionable queries against a knowledge graph or database [33, 72].

3.3 Recurrent Neural Networks

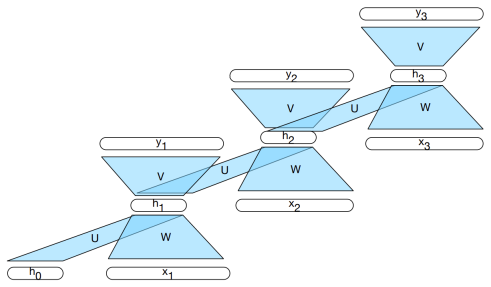

Figure 3. RNN unrolled in time [35]

Any network featuring a cycle inside its network connections is referred to as a recurrent neural network (RNN) [23]. That is any network in which the value of a unit is directly or indirectly determined by the network's previous outputs. RNN's use this cyclical nature as a way to interpret temporal changes. RNNs are used as the building block in building sequence-to-sequence [57] models.

3.3.1 Encoding.

Sequence-to-sequence models are models in which the input and output can be variable length. Sequence-to-sequence models have many uses, such as machine translation [57], but can also be used to encode [15] the context or semantic meaning of text into an n-dimensional vector. The *encoder* (see figure 3), which is a stack of RNNs, process each word (embedding) sequentially using the current word x_i and a representation of the previous word/context h_{i-1} to compute an output y_i and new representation h_i. The last hidden vector h_n of a sequence is the entire sentence encoded into vector space with relevant information from all the words in the sequence. Encoders are valuable as an effective way to capture contextual information in a vector representation. [47, 73, 74] show the extensive use of encoding models in text-based settings as a way to represent, potentially, entire paragraphs into a single representative vector.

4 Reinforcement Learning

Reinforcement learning is a sub-field of machine learning that concerns itself with teaching an agent how to map environmental situations to actions so as to maximise the cumulative reward given by the environment i.e optimal control. The goal for the agent is to learn on its own and discover which actions lead, ultimately, to better outcomes (higher rewards).

4.1 Preliminary Concepts

4.1.1 Markov Decision Processes.

Markov decision processes (MDP) are mathematical frameworks that can be used to describe an environment in a reinforcement learning problem. Almost all problems that can be solved by RL can be modelled into an MDP. Since

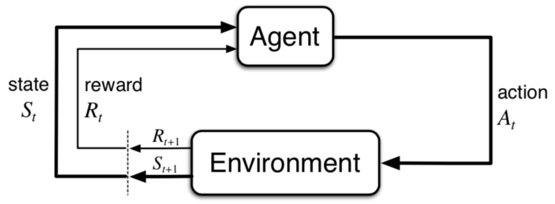

Figure 4. Agent's interaction with environment

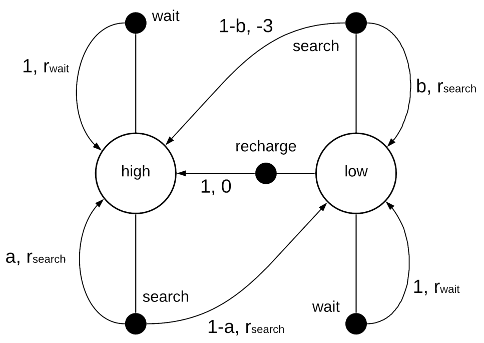

Figure 5. Example of a MDP

the reinforcement learning problem is mathematically idealized by Markov Decision Processes, one can construct precise theoretical statements about these problems using MDPs [60].

Figure 5 shows an example of a MDP. The circles represent the states and the arrows represent transitions with specific probabilities. As can be seen, states can offer more than one potential transition. An agent will transition from state to state with a certain probability depending on the action/transition it decides to take. Figure 4 shows the agent's interaction with the environment in a MDP. The agent and environment (MDP) interact at each step in a sequence of discrete time steps. At every step the agent receives an observation (some representation) of the state $S_t \in \mathcal{S}$, and selects an action $A_t \in \mathcal{A}(s)$ based on this observation. Once the agent has executed its action, one time step later, the agent will receive a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ due to its action and the state it ends up in. This reward can be positive or negative depending on the outcome of the action. The agent now finds itself in a new state $S_{t+1} \in \mathcal{S}$ and the cycle repeats until a terminal state is reached.

MDPs have the Markov property meaning that the future is independent of the past given the present i.e $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, ..., S_t]$. Intuitively this means that every state must include the information about all aspects of the past agent–environment interaction that makes a difference for the future, meaning that all actions can be accurately decided solely based upon the state the agent is in.

An important part of the application of reinforcement learning to domains such as question answering is how the MDP is represented. [12] show a way of formulating the question answering (In an IR setting) task into an MDP upon which RL agents can attempt to solve. States are defined by the current status of the answer space. At each iteration, a sentence, from the document where an answer can be found, is added to the answer pool. Actions are then defined as selecting a sentence from the remaining sentences that are not present in the extracted summary thus far. Results show that by simply reformulating the problem into an MDP and applying RL, significant improvements upon the baseline models can be found.

4.1.2 Policies and Value Functions.

Agent's use policies to decide their actions at every time step. A policy π is defined as a mapping of environment states $\mathcal{S}$ to the probabilities of selecting each possible action $\mathcal{A}$ [60]. If an agent is following a policy π at time t, the policy function $\pi(a|s)$ is the probability of performing the specific action $A_t = a$ given the agent is in state $S_t = s$.

A large proportion of reinforcement learning algorithms, 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 action-state 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 discount factor (the weight of importance given to immediate and future rewards). This means formally the value function for any given policy is defined as:

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

The bellman equation for the value function decomposes the value function into two parts, the immediate reward and the discounted value of successor states. This can be formulated as :

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma * v(S_{t+1})|S_t = s]$$

The bellman equation illustrates an important recursive property that is used for all methods that involve estimating a value function.

4.1.3 Prediction and Control.

In reinforcement learning, there are two separate goals: Prediction - where the goal of the agent is to evaluate how well a given policy performs. An example of this is to simply find the value function of an MDP given some policy. The other goal, which is usually the focus of many, is Control - where the goal of the agent is to find the optimal policy (best possible performing policy) to some MDP [60].

4.2 Environments

Environments refer to the "world" an agent is situated in. Many different types of environments exist [9] for different tasks. The environment constructs the simulated experience for RL agents to learn how to perform some task.

4.2.1 TextWorld.

TextWorld [18] is a sandbox environment that allows you to play text games interactively. It also provides generative capabilities to easily construct specific text-based games for different domains. Its generative processes allow it to fine-tune the difficulty, range, and language of created games. TextWorld could also be used to research generalization and transfer learning by constructing sets of different but related games. TextWorld has been used extensively in research involving text-based environments [1, 26, 42, 73–75].

4.3 Model Free

Model-free methods are those of which the agent has no internal representation of the MDP/Environment it is in. This means the agent has no knowledge of the MDP transition or reward dynamics and cannot use this information in the process of learning.

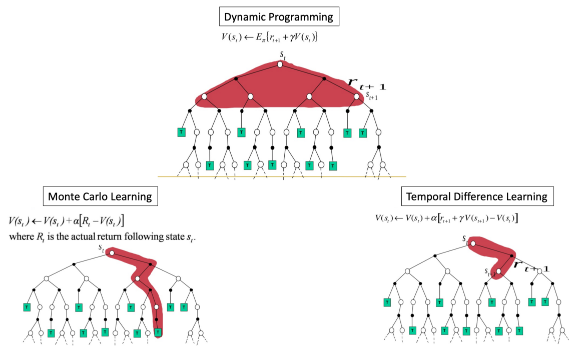
Dynamic Programming
$V(s_t) \leftarrow E_\pi\{r_{t+1} + \gamma V(s_t)\}$
s_t
r_{t+1}
s_{t+1}
T
Monte Carlo Learning
$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)]$
where R_t is the actual return following state s_t.
Temporal Difference Learning
$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$

Figure 6. Dynamic Programming, Monte Carlo, and Temporal Difference Backup illustrated

4.3.1 Monte Carlo.

Monte Carlo methods are one form of model-free methods that directly learn from experience. Monte Carlo methods have the assumption that all episodes will terminate and that experience is divided into episodes (sequence of time steps starting in a start state and ending in a terminal state). The agent simulates episodes of experience and learns directly from this experience thereby not requiring any knowledge of how the environment functions. In the learning process, an agent generates episodes of experience e.g ($S_0, A_0, R_1, S_1, A_1, R_2, ...S_{T-1}, A_{T-1}, R_T$) and uses this history to learn a value function or policy function. This means Monte Carlo methods have to simulate entire episodes of experience before learning can take place. Due to this, Monte Carlo cannot be applied to continuous MDPs. What distinguishes Monte Carlo apart from classic dynamic programming and temporal difference learning, is that Monte Carlo learning does not employ bootstrapping. Hence the estimate for one state does not build upon the estimate of any other state therefore Monte Carlo methods have a lower bias compared to other methods but this is traded off with a higher variance as episode trajectories can be highly different [60]. An example of a Monte Carlo update for a value function is as follows:

$$V(S_t) \longleftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

where G_t represents the cumulative value seen from time t onward and alpha represents the step size.

4.3.2 Temporal Difference.

Temporal difference methods are the same as Monte Carlo methods in that they directly learn from experience without the need of a model but unlike Monte Carlo, they don't need to wait for episode termination to learn, they bootstrap from the sample trajectory [60]. TD methods make use of every individual time step to learn. At time $t+1$, the experience of a single time step (in the case of one-step TD-learning) is used to update either the agent's policy or value function. An example of a one-step TD update for the value function is as follows:

$$V(S_t) \longleftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

This is updating the agent's current state value estimate in the direction of the successive state's value plus the immediate reward it has seen. Intuitively this is essentially a combination of dynamic programming and Monte Carlo ideas, where the agent updates its estimates based on other learned estimates and does this from raw experience. Since TD methods bootstrap, the variance between state updates is much lower than its Monte Carlo counterparts but the trade-off is in bias as values are updated based on other estimates instead of experienced trajectories. This needs to be kept in mind when deciding upon which type of learning to use.

4.3.3 Value-Based Methods.

Value-based methods are methods in which the agent tries to learn the value function of the MDP it is situated in. The value function is extremely valuable 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 usually how control is implemented in value-based methods (not including the possibility of exploration) [60]. Practically, most value-based methods try to learn the value of action-state pairs. The idea behind this, is that if the agent does not know the model 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. This action-state expected reward value is called the Q-value. The following are a few of the most popular value-based methods.

4.3.4 Q-Learning.

Q-learning [69] is a way for agent's to learn the optimal state-action value function directly instead of repeatedly performing policy evaluation and iteration. An example of the the Q-learning update using one-step TD-learning is as follows:

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

Although [69] has shown the convergence properties illustrating that Q-learning can be used effectively to solve MDP's, practically these value based methods suffer from poor convergence.

Pure value-based methods aren't as commonplace in RL-based question answering systems but have been used successfully. [12] use Q-Learning to train an agent to extract answers to complex question from document passages. This shows that even with potential convergence issues, pure value-based approaches can attain significant performance on question answering tasks.

4.3.5 DQN.
Most modern problems when modelled into MDPs 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. The solution to this problem is function approximation. With the modern advancements in deep learning, one type of function approximation that is used heavily in modern RL are neural networks. DQNs (Deep Q Networks) [46] have been shown, with a few additions such as experience replay memory [41] and target network, to be able to solve large dimensional MDPs, such as Atari games, effectively. However there are limitations with this approach, it was shown that non-linear function approximation, such as neural networks, can cause the Q-network to diverge [63]. DQN's also tend to overestimate the actual Q-value which can eventually lead to sub-optimal policies [62]. 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 [27, 46, 74].

4.3.6 DDQN.
Double DQN [64] is the proposed algorithm to solve the overestimation problem in DQNs. Double DQN 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 [64].

4.3.7 Rainbow DQN.
Rainbow DQN [32] 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 [53] to speed up learning, dueling networks [68], multi-step learning [58] for faster learning with suitably tuned hyper-parameter, Distributional reinforcement learning [4] instead of expected return, and noisy nets [25] for exploration. This extension shows improved performance in comparison to other methods.

[47] illustrate that DQN methods can be successful in accomplishing text-based game tasks illustrating a sufficient semantic interpretation of language. [74] taking a similar approach compare the performance of DQN [46], DDQN [64], and Rainbow DQN [32] algorithms in a text-based interactive question answering setting. The performance difference between these methodologies is small indicating a different approach might be needed to increase results greatly.

4.3.8 Policy Based Methods.
Policy-based methods are methods in which the agent tries to learn the policy function directly. This can be advantageous as it is able to learn stochastic policies whereas value-based methods are deterministic. It can also be used to effectively learn continuous or high dimensional action spaces and has better convergence properties. Policy gradient methods search for a local maximum in $J(\theta)$ by gradient ascent $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$. The following are a few of the most popular policy-based methods.

4.3.9 REINFORCE.
REINFORCE [70] is a Monte Carlo method that updates the policy functions parameters directly using the policy gradient with respect to $J(\theta) = \mathbb{E}\big[\sum_{t=0}^{T-1} R_{t+1}|\pi_\theta\big]$. [61] shows that the gradient of the objective function to maximize 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:

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

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 [60].

REINFORCE has seen success in the question-answering domain both with the use of knowledge graphs and sentence extraction from corpora [16, 20, 66, 71] where as pure value-based methods haven't. Possible reasons for this are that in these question answering tasks the action space can be very large due to graph or document complexity. This can make it hard for value-based methods such as DQN [46] to converge and learn effective policies. Another reason, particularly in the knowledge graph paradigm, is that stochastic policies, unlike DQN's deterministic policies, might perform better by reducing the likelihood of an agent getting stuck in the knowledge graph traversal process.

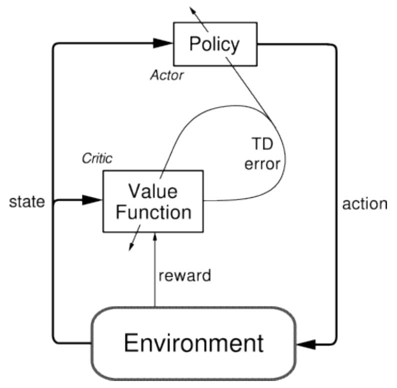

Figure 7. Actor-Critic Architecture

4.3.10 Actor-Critic.
Actor-Critic methods [40] are a combination of policy-based and value-based methods with the goal of combining each method's strong points. Actor-Critic methods consist of two models: The critic which updates some value function e.g($Q(S_t, A_t)$) (intuitively, it tells the actor how good the action was) and The actor which decides which action to take as well as updating the policy parameters in the direction suggested by the critic. This allows us to bootstrap and reduces variance.

Since actor-critic ideally merges the strong points of both value and policy-based methods, it can be assumed that the application of these methods can significantly improve upon prior methods. [67] demonstrate the potential benefit of using an actor-critic approach to IR question answering.

4.4 Sparse Reward Environments

Most real-world scenarios when modelled as MDPs have very sparse or deceptive extrinsic rewards, this means it can be very hard for an agent to learn which actions led to a positive or negative outcome. Previous methods discussed show trouble in learning high performing policies in these sparsely rewarded environments e.g Montezuma's Revenge [46, 64]. This problem can be compounded when the state-action space is very large.

Question answering, especially interactive question answering, form sparsely rewarded environments as long sequences of actions may be required before any reward is given making the task even harder for reinforcement learning to solve. This indicates the need for advanced reward shaping and exploration techniques.

4.4.1 Reward Shaping. Reward shaping [43] is a commonly used technique [7] to deal with sparsely rewarded environments. It involves enhancing the primary extrinsic reward given by the environment with additional reward features to aid the agent in the learning process. Although reward shaping has been successful in practice, there are a few limitations with this method. The additional reward features often have to be hand-crafted by domain experts for the approach to be successful. This also introduces a human bias that could inhibit the agent from learning true optimal policies that humans would never think of.

4.5 Exploration

Exploration is important for an agent to discover new and promising states that can lead to more optimal policies. It is especially important in sparsely rewarded environments when a large sequence of specific actions might need to be done to achieve some reward. Traditional exploration methods such as epsilon-greedy [60] and entropy regularisation [70] have shown success in inducing novel behaviours in agents but these methods are not sufficient for the exploration needed in many environments [49].

Text-based games are, fundamentally, partially observable Markov decision processes [36] since the true environment state is never observed and only described by text. This means to act optimally an agent must keep track of its observations in some manner. The importance of efficient exploration is becoming more apparent as research grows. [73] show the benefit of employing more advanced exploration methods in text-based games as well as how these exploration methods help agents generalise better to unseen environments. The following are some more recently created exploration methods:

4.5.1 Noisy Networks. Noisy networks [25] are a simple and efficient way of inducing random exploratory behaviour that allows for the agent to decide for itself when and how much it wants to explore. Noisy networks are neural networks where the weights and biases are affected by some parametric function of noise. Using noisy networks in deep reinforcement learning consists of making the last layer of the network, e.g Q-Function or Value-Function, a noisy network. This induces randomness in the values produced thereby affecting the policy. These parameters of the parametric noise function are controlled by the agent and optimised in the learning process. This allows a form of context to be given to the exploration task. [25] shows how this method of exploration greatly increases performance in certain environments but not all, this shows how noisy networks are a problem specific strategy.

4.5.2 Curiosity. For many sparsely rewarded environments, shaping the reward function is not possible. Using random exploration methods rely upon the agent stumbling onto the goal state by chance. This can be practically impossible in large environments and will result in failure to learn. Curiosity or intrinsic motivation can be seen as a new way of learning which requires no extrinsic rewards from the environment. It is a controlled form of exploration. The two most popular formulations of intrinsic reward can be grouped as follows: The first class of methods try to encourage the agent to explore states it hasn't seen before. [3] have shown great results in very sparsely rewarded environments, such as Montezuma's Revenge, using such exploration with DQN. The second class of methods focus on encouraging the agent to take actions that lower the error in the agent's ability to predict the consequences of its actions. Intuitively this aims to increase the agent's knowledge about the environment [14, 54, 55]. Measuring the novelty i.e how different a state is from a previous state, or building an internal environmental model to predict the next state can be a difficult problem in high dimensional state spaces. This is compounded with environment stochasticity and noise which ultimately makes intrinsic reward calculation difficult.

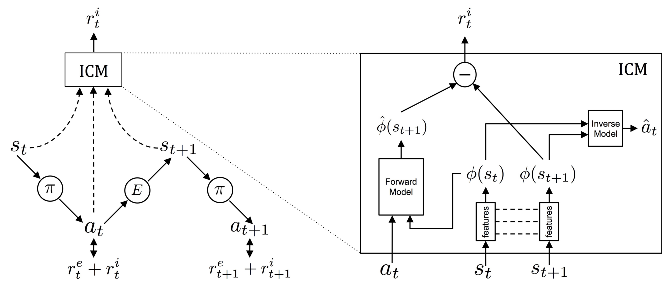

Figure 8. Intrinsic Curiosity Module [50]

4.5.3 ICM. [50] have derived a method, belonging to the second class, to avoid these difficulties by using a model that only predicts the environmental changes that are caused because of the agent's actions or those that affect the agent. It works by transforming the agent's observation into a feature space where only the relevant information is represented. The agent itself learns the feature space using a neural network trained on an inverse task of predicting the agent's action given the current and successive state. By training on the inverse task, the model's features embedding space only concerns itself with factors that affect the agent. The feature space of the inverse model is then used to train a new neural network to predict the next state's feature representation given the current state's features and action. The agent is then supplied with the prediction error as an intrinsic reward. Refer to figure 8 for an illustration. The intrinsic reward is summed with the extrinsic reward, such as $r_t = r_t^e + \beta r_t^i$ where β balances the exploration and exploitation, to ultimately solve the task given. [50] shows how curiosity greatly improves exploration as well as acts as a mechanism for agents to learn skills that potentially help in future scenarios. This exploration helps in solving very sparse reward settings. [50] also shows that fine-tuning an agent that has been trained with curiosity and extrinsic rewards is more beneficial (learns faster and achieves higher scores) than training from scratch for new unseen environments. This shows how curiosity helps agents generalise to new environments. This method however could fail in learning a sufficient feature representation of the environment thereby leading to poor performance.

4.5.4 Random Network Distillation.

Random network distillation [11] is an exploration method that introduces a new approach for the prediction problem. Two neural networks are created: one that is fixed where weights are randomized called the target network and one that is trained on data collected by the agent called the predictor. The target network simply receives the agent's observation as input and outputs a fixed random embedding of the state. The predictor network receives an observation and also outputs a random embedding of the state. The predictor network is then trained to minimize the difference between its output and the fixed target output. This process essentially "distils" the predictor network into a trained one. The prediction error between the two networks is expected to be larger for novel states that are more dissimilar to the states the predictor network has seen and been trained on. This prediction error forms part of the intrinsic reward as a way to encourage moving to novel states. One drawback to this method is the risk of using a powerful optimization algorithm for training the predictor. The algorithm might train the predictor network to perfectly mimic the target thereby reducing all intrinsic reward for the agent to zero.

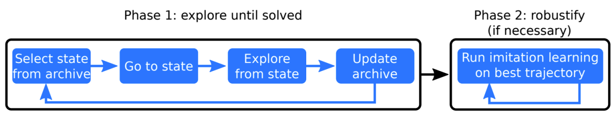
Phase 1: explore until solved
Select state from archive
Go to state
Explore from state
Update archive
Phase 2: robustify (if necessary)
Run imitation learning on best trajectory

Figure 9. Go-Explore Algorithm [22]

4.5.5 Go-Explore.

Go-Explore [22] is the current state of the art performing algorithm on the classic sparse reward environment of *Montezuma's Revenge **and** Pitfall*. Go-Explore is an exploration approach that does not use intrinsic motivation, instead it does the following: the first phase of the algorithm is to explore the state space by keeping an archive of different game states (called 'cells' in this context) and the trajectories that lead to them. The agent selects a cell from its archive in some probabilistic manner biased towards newer states, it uses its trajectory knowledge to travel back to that state and proceeds to explore from there. For all cells visited in the process, if the new trajectory is better, store the new trajectory as a way to reach that cell. Go-Explore essentially remembers and returns to promising states for exploration. This avoids the problem of detachment and derailment that intrinsic motivation methods may suffer from. Detachment refers to the concept that an agent using intrinsic motivation might get separated from high intrinsic reward areas because the intrinsic reward at the border of a new area has been consumed by the agent thereby making it difficult for the agent to find its way back to explore a new section. Derailment refers to when an agent discovers a promising state and it would be desirable to return to that state to explore from there, then derailment might occur. The longer, more complicated, and exact a series of actions must be to achieve a previously identified high intrinsic reward state, the more the stochastic nature of the policies will "derail" the agent and prevent them from returning. The second phase evaluates if the first phase solutions are not robust to noise and if necessary will robustify them into a neural network with imitation learning. Even though Go-Explore boasts SOTA results, the algorithm has many drawbacks. The environment must be able to reset back to a specific state or the environment must be deterministic so that the trajectories stored correctly take the agent back to the state it wants to go to. The Go-Explore algorithm also has no evidence of being able to generalize.

[42] show incredible results in applying the go-explore exploration method to text-based games. It is also demonstrated that agents using go-explore adapted better to unseen environments than their counterparts by solving over half of the unseen environments in the test set.

4.6 Model-Based

Model-based methods are methods in which the environment dynamics such as reward and transition functions are learned by (or given to) the agent and modelled into some representation that the agent can use to choose the optimal action. Model-based learning differs from model-free learning in that instead of trying to maximise the reward received, the agent tries to find the optimal trajectory with the lowest cost according to some defined cost function. Models can help agents learn faster and find better policies in fewer interactions with the environments (i.e it is sample efficient) [37]. Models can help agents with the ability to predict the future and this can be very beneficial in certain situations [59].

Model-based methods is still a relatively new research area and the application of model-based methods in the question-answering domain is yet to be investigated. This indicates the potential of a new approach to question answering. The following is a recent significant method in model-based reinforcement learning.

4.6.1 Recurrent World Models.

World models [31] are a representation of spatial and temporal features of a world. World models have a variety of uses. They can be used as internal low dimensional representations for control and action selection. They can also be used to generate synthetic data to train an agent thereby increasing sample efficiency in the learning process. World models also give an agent the ability to plan. [31] demonstrates a new way of developing world models which is the combination of the former two uses where the agent generates synthetic experience in the low

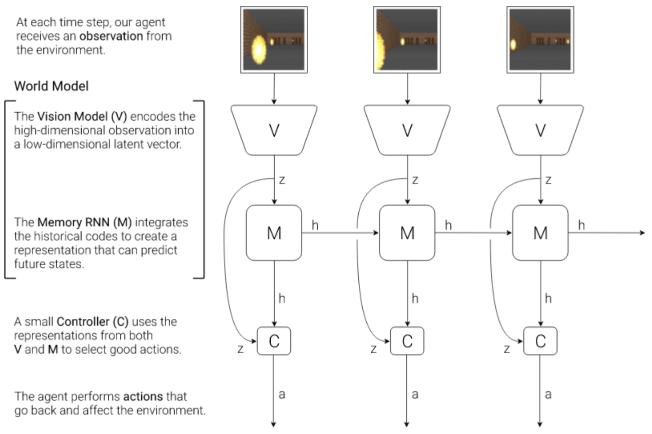
At each time step, our agent receives an observation from the environment.
World Model
The Vision Model (V) encodes the high-dimensional observation into a low-dimensional latent vector.
The Memory RNN (M) integrates the historical codes to create a representation that can predict future states.
A small Controller (C) uses the representations from both V and M to select good actions.
The agent performs actions that go back and affect the environment.
V
M
C
z
h
a

Figure 10. Recurrent World Model Overview [31]

dimensional internal representation to learn a policy and apply it to solve a task. This new method to train world models is done using large RNNs in an unsupervised manner that capture a compressed spatial and temporal representation of the real environment. [31] also develop a new agent architecture to use these world models. Refer to figure 10 for an illustration of how the world model and agent interact. This method shows it is possible to develop a world model in an unsupervised fashion, e.g using random rollout trajectories as training data, that can be used for training an agent's policy for an environment it has never actually seen. This gives the training process incredible sample efficiency. This technique achieves SOTA results in the CarRacing-v0 open AI gym environment.

5 Reinforcement Learning in Question Answering

Recent literature [10, 12, 20, 28, 67] has shown promise in applying RL to the question answering domain. The following is a brief overview and discussion of previously mentioned literature.

5.1 Knowledge Base Applications

[20] illustrates RL's capability of knowledge base traversal and relationship inference for question answering. Using REINFORCE, (refer to section 4.3.9), an agent is trained to navigate large knowledge graphs conditioned on only the question. This requires the agent to infer new relations between two nodes as a way to answer the question. Comparative results show the method is the same if not better than it's counterparts on benchmark datasets.

5.2 Interactive Question Answering

Interactive question answering [19, 29, 74], which requires an agent to explore its environment to gather the necessary knowledge to answer a given question, can be seen as a natural extension to the question answering challenge. The question of whether or not a system can be trained to have the ability of knowing how to find and reason for answers in dynamic and changing environments. Unlike navigating knowledge graphs to find the answer [20], interactive question answering requires an agent to understand natural language concepts within the given questions as well as display an understanding of how to find the answer via exploration, interaction and interpretation of the environment it is situated in. Information about unseen environments couldn't have been memorized and rewards are very sparse. Both [19, 29] explore interactive question-answering in 3D virtual environments giving large importance to agent's visually interpreting the scene. Both approaches break down an agent's responsibilities into separately trainable models e.g navigation, vision, language. Each sub-model relays their information and interpretation of what do to ultimately succeed in answering the question given.

[74] investigates the interactive question answering domain within an entirely text-based environment [18], seemingly requiring the agent to have an understanding of the language it operates in. TextWorld [18] is used as an environment to procedurally create different text-based games. In conjunction with these games - questions are created. The agent then has to navigate the text game to find the answer to the given question. [74] use a value-based approach (refer to section 4.3.3) specifically Q-learning. The agent is also trained on changing environments to discourage memorization and over-fitting as well as encourage actual language understanding. A core idea is that the ability of the agent to learn a language as a way to represent its environment gives hope that a more generalizable policy will also be learned.

The results of the recent IQA literature [19, 29, 74] show how interactive question answering is a challenging task that leaves much room for improvement.

6 Conclusion

To conclude, reinforcement learning methods have extensive literature expanding the field into different directions. Its been illustrated that new advances in RL show promise in not only traditional question answering systems but specifically the field of interactive question answering. The lacking performative results in the interactive question answering domain show room for improvement and potential for research. The ability for question answering systems to generalise to new and unseen environments has not been shown to be adequate which further reinforces the idea of teaching agents or models the skills to fundamentally find the answer to a given question where the answer is not explicitly represented in some structured manner. Whilst ultimately interactive question answering has been framed to be a reinforcement learning problem, elements from traditional QA systems and natural language processing play a large role in the potential success to be had.

References

[1] Prithviraj Ammanabrolu and Mark O. Riedl. 2018. Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. *CoRR* abs/1812.01628 (2018). arXiv:1812.01628 http://arxiv.org/abs/1812.01628

[2] Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic Parsing as Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 47–52. https://www.aclweb.org/anthology/P13-2009

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf

[4] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. *CoRR* abs/1707.06887 (2017). arXiv:1707.06887 http://arxiv.org/abs/1707.06887

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3, null (March 2003), 1137–1155.

[6] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1533–1544. https://www.aclweb.org/anthology/D13-1160

[7] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR* abs/1912.06680 (2019). arXiv:1912.06680 http://arxiv.org/abs/1912.06680

[8] 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).

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[10] Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. *CoRR* abs/1705.07830 (2017). arXiv:1705.07830 http://arxiv.org/abs/1705.07830

[11] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. 2018. Exploration by Random Network Distillation. *CoRR* abs/1810.12894 (2018). arXiv:1810.12894 http://arxiv.org/abs/1810.12894

[12] Yllias Chali, Sadid A. Hasan, and Mustapha Mojahid. 2015. A reinforcement learning formulation to the complex question answering problem. *Information Processing Management* 51, 3 (2015), 252–272. https://doi.org/10.1016/j.ipm.2015.01.002

[13] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. 2013. The Expressive Power of Word Embeddings. *CoRR* abs/1301.3226 (2013). arXiv:1301.3226 http://arxiv.org/abs/1301.3226

[14] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2005. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou (Eds.), Vol. 17. MIT Press. https://proceedings.neurips.cc/paper/2004/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf

[15] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014). arXiv:1406.1078 http://arxiv.org/abs/1406.1078

[16] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-Fine Question Answering for Long Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 209–220. https://doi.org/10.18653/v1/P17-1020

[17] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 160–167. https://doi.org/10.1145/1390156.1390177

[18] 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.

[19] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[20] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alexander J. Smola, and Andrew McCallum. 2017. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. *CoRR* abs/1711.05851 (2017). arXiv:1711.05851 http://arxiv.org/abs/1711.05851

[21] Dimitris Dimitriadis and Grigorios Tsoumakas. 2019. Word embeddings and external resources for answer processing in biomedical factoid question answering. *Journal of Biomedical Informatics* 92 (2019), 103118. https://doi.org/10.1016/j.jbi.2019.103118

[22] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *CoRR* abs/1901.10995 (2019). arXiv:1901.10995 http://arxiv.org/abs/1901.10995

[23] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179–211. https://doi.org/10.1016/0364-0213(90)90002-E

[24] Olivier Ferret, Brigitte Grau, Martine Hurault-Plantet, Gabriel Illouz, Christian Jacquemin, Laura Monceaux, and A Vilnat. 2002. How NLP can improve question answering. *KO Knowledge Organization* 29, 3-4 (2002), 135–155.

[25] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. 2017. Noisy Networks for Exploration. *CoRR* abs/1706.10295 (2017). arXiv:1706.10295 http://arxiv.org/abs/1706.10295

[26] Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural Approaches to Conversational AI. *CoRR* abs/1809.08267 (2018). arXiv:1809.08267 http://arxiv.org/abs/1809.08267

[27] Ziming Gao, Yuan Gao, Yi Hu, Zhengyong Jiang, and Jionglong Su. 2020. Application of Deep Q-Network in Portfolio Management. In *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. 268–275. https://doi.org/10.1109/ICBDA49040.2020.9101333

[28] Fréderic Godin, Anjishnu Kumar, and Arpit Mittal. 2019. Using Ternary Rewards to Reason over Knowledge Graphs with Deep Reinforcement Learning. *CoRR* abs/1902.10236 (2019). arXiv:1902.10236 http://arxiv.org/abs/1902.10236

[29] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual Question Answering in Interactive Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2020. Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases. *CoRR* abs/2011.07743 (2020). arXiv:2011.07743 https://arxiv.org/abs/2011.07743

[31] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. *CoRR* abs/1809.01999 (2018). arXiv:1809.01999 http://arxiv.org/abs/1809.01999

[32] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. *CoRR* abs/1710.02298 (2017). arXiv:1710.02298 http://arxiv.org/abs/

1710.02298

[33] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 963–973. https://doi.org/10.18653/v1/P17-1089

[34] Robin Jia and Percy Liang. 2016. Data Recombination for Neural Semantic Parsing. *CoRR* abs/1606.03622 (2016). arXiv:1606.03622 http://arxiv.org/abs/1606.03622

[35] Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition.* Pearson Prentice Hall, Upper Saddle River, N.J. http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y

[36] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[37] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, George Tucker, and Henryk Michalewski. 2019. Model-Based Reinforcement Learning for Atari. *CoRR* abs/1903.00374 (2019). arXiv:1903.00374 http://arxiv.org/abs/1903.00374

[38] Aishwarya Kamath and Rajarshi Das. 2018. A Survey on Semantic Parsing. *CoRR* abs/1812.00978 (2018). arXiv:1812.00978 http://arxiv.org/abs/1812.00978

[39] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2017. Natural Language Processing: State of The Art, Current Trends and Challenges. *CoRR* abs/1708.05148 (2017). arXiv:1708.05148 http://arxiv.org/abs/1708.05148

[40] Vijay R. Konda and John N. Tsitsiklis. 2003. OnActor-Critic Algorithms. *SIAM Journal on Control and Optimization* 42, 4 (Jan. 2003), 1143–1166. https://doi.org/10.1137/s0363012901385691

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

[42] 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. *CoRR* abs/2001.08868 (2020). arXiv:2001.08868 https://arxiv.org/abs/2001.08868

[43] Maja J Mataric. 1994. Reward Functions for Accelerated Learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 181–189.

[44] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1301.3781

[45] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013). arXiv:1310.4546 http://arxiv.org/abs/1310.4546

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[47] Karthik Narasimhan, Tejas D. Kulkarni, and Regina Barzilay. 2015. Language Understanding for Text-based Games Using Deep Reinforcement Learning. *CoRR* abs/1506.08941 (2015). arXiv:1506.08941 http://arxiv.org/abs/1506.08941

[48] Bolanle Ojokoh, Emmanuel Adebisi, and and. 2019. A Review of Question Answering Systems. *Journal of Web Engineering* 17, 8 (2019), 717–758. https://doi.org/10.13052/jwe1540-9589.1785

[49] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. 2019. Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research* 20, 124 (2019), 1–62. http://jmlr.org/papers/v20/18-339.html

[50] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. *CoRR* abs/1705.05363 (2017). arXiv:1705.05363 http://arxiv.org/abs/1705.05363

[51] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[52] Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. *CoRR* abs/1704.07535 (2017). arXiv:1704.07535 http://arxiv.org/abs/1704.07535

[53] 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

[54] Jürgen Schmidhuber. 1991. A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers.

[55] Jürgen Schmidhuber. 2010. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development* 2, 3 (2010), 230–247. https://doi.org/10.1109/TAMD.2010.2056368

[56] 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

[57] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *CoRR* abs/1409.3215 (2014). arXiv:1409.3215 http://arxiv.org/abs/1409.3215

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

[59] Richard S. Sutton. 1990. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 216–224.

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

[61] 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

[62] 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.

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

[64] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR* abs/1509.06461 (2015). arXiv:1509.06461 http://arxiv.org/abs/1509.06461

[65] Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Talukdar. 2019. Incorporating Syntactic and Semantic Information in Word Embeddings using Graph Convolutional Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3308–3318. https://doi.org/10.18653/v1/P19-1320

[66] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. Rlt;supgt;3lt;/supgt;: Reinforced Ranker-Reader for Open-Domain Question Answering. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://ojs.aaai.org/index.php/AAAI/article/view/12053

[67] Yu Wang and Hongxia Jin. 2019. A Deep Reinforcement Learning Based Multi-Step Coarse to Fine Question Answering (MSCQA) System. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 7224–7232. https://doi.org/10.1609/aaai.v33i01.33017224

[68] Ziyu Wang, Nando de Freitas, and Marc Lanctot. 2015. Dueling Network Architectures for Deep Reinforcement Learning. *CoRR* abs/1511.06581 (2015).

arXiv:1511.06581 http://arxiv.org/abs/1511.06581

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

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

[71] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. *CoRR* abs/1707.06690 (2017). arXiv:1707.06690 http://arxiv.org/abs/1707.06690

[72] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 1321–1331. https://doi.org/10.3115/v1/P15-1128

[73] Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew J. Hausknecht, and Adam Trischler. 2018. Counting to Explore and Generalize in Text-based Games. *CoRR* abs/1806.11525 (2018). arXiv:1806.11525 http://arxiv.org/abs/1806.11525

[74] 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

[75] Xingdi Yuan, Jie Fu, Marc-Alexandre Côté, Yi Tay, Christopher J. Pal, and Adam Trischler. 2019. Interactive Machine Comprehension with Information Seeking Agents. *CoRR* abs/1908.10449 (2019). arXiv:1908.10449 http://arxiv.org/abs/1908.10449

[76] 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