Leveraging Sentiment Analysis for Sparse Reward Reinforcement Learning in Text-Based Games

A Project Report

submitted by

AMEET S DESHPANDE

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2019

THESIS CERTIFICATE

This is to certify that the thesis entitled **Leveraging Sentiment Analysis for Sparse Reward Reinforcement Learning in Text-Based Games**, submitted by **Ameet S Deshpande** (**CS15B001**), to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 3rd May, 2019

ACKNOWLEDGEMENTS

I want to thank Prof. Balaraman Ravindran for both introducing me to Reinforcement Learning and encouraging me to carry out research. He motivated me to choose a problem statement close to my heart and provided constant guidance. Special thanks are due to Prof. Karthik Narasimhan of Princeton University who co-advised me on this project. His passion for this problem and his willingness to help at any hour were crucial to the completion of this project. I would also like to thank Prof. Sutanu Chakraborti and Prof. Mitesh Khapra for introducing me to the fields of Natural Language Processing and Deep Learning which were crucial ingredients of the thesis. I am grateful to my parents, wingmates, and my friend Briti for keeping me motivated even when the project was not producing the desired results.

ABSTRACT

KEYWORDS: Sparse Rewards, Deep Q-Network, Sentiment Analysis

With the advent of Neural Networks, researchers are increasingly using Reinforcement Learning (RL) for applications which have high dimensional state representations, like images. Combining the representational power of neural networks with classical RL algorithms like policy gradients and Q-Learning engenders control mechanisms in complex environments. While RL agents like Asynchronous Advantage Actor-Critic and Deep-Q Networks have achieved superhuman performance in perceptual environments like Atari, there has been a dearth of studies in text domains. Large vocabulary sets in Text-Based environments make them high dimensional by design, and several challenges like partial-observability, sparse rewards, and exponentially large action spaces complicate the problem further. As a result, even state-of-the-art agents fail to achieve performance close to humans on simple games. Thus, Text-Based games can serve as a great playground to test agents or algorithms.

This thesis aims to alleviate issues associated with sparse rewards obtained during the game. Traditional methods which use state descriptions to extract only a state representation ignore the feedback which is inherently present. Descriptions like "Good Job! You ate the food" indicate progress, and descriptions like "You entered a new room" indicate exploration. Positive and negative cues like these can be converted to rewards using sentiment analysis. This converts the sparse reward problem to a dense one which is easier to solve. Furthermore, this can enable “Reinforcement Learning without rewards” where the agent can learn entirely from these intrinsic sentiment rewards. This framework is similar to Intrinsic Motivation, where the environment does not necessarily provide the rewards, but the agent analyzes and realizes them by itself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
RL	Reinforcement Learning
NLP	Natural Language Processing
DP	Dynamic Programming
MC	Monte Carlo
TD	Temporal Difference
OOV	Out Of Vocabulary
IMRL	Intrinsically Motivated Reinforcement Learning
MDP	Markov Decision Process
SA	Sentiment Analysis
DD	Diverse Density
LSTM	Long Short Term Memory
DQN	Deep Q-Network

NOTATION

α	Learning Rate
γ	Discounting Factor
s	State
a	Action
r_e	Extrinsic Reward
r_i	Intrinsic Reward
G	Return
λ_i	Weights for Intrinsic Rewards

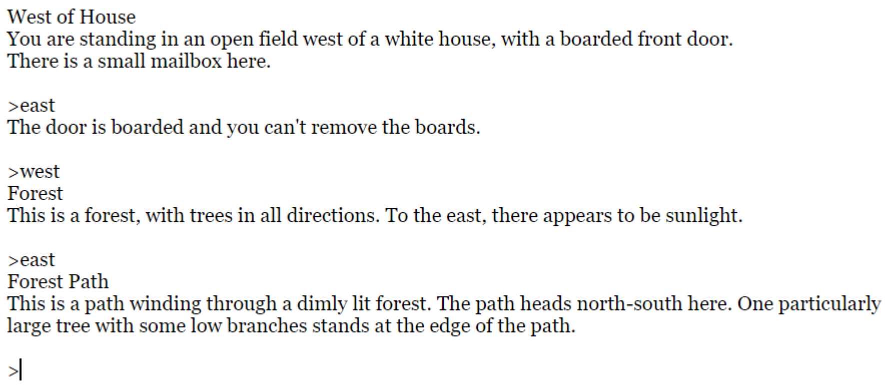
CHAPTER 1

Introduction

With Deep Learning and Reinforcement Learning being used in tandem, complicated control problems can feasibly be solved using RL algorithms. Games ranging from Atari (Mnih *et al.* [2015]) to Go (Silver *et al.* [2016]) have been solved with superhuman performance. There have also been a lot of studies in learning policies for robots (Kober *et al.* [2013]). Given this success in perceptual tasks, it is only natural to use RL for solving NLP problems.

1.1 Text-Based Games

Text-Based Games offer the perfect environments to test RL methods on NLP tasks. At every time step, the Environment provides an observation of the state. Hereon, that observation is called the state itself (we distinguish the two in Chapter 2). Figure 1.1 shows an example of Zork I game being played.



```
West of House
You are standing in an open field west of a white house, with a boarded front door.
There is a small mailbox here.

>east
The door is boarded and you can't remove the boards.

>west
Forest
This is a forest, with trees in all directions. To the east, there appears to be sunlight.

>east
Forest Path
This is a path winding through a dimly lit forest. The path heads north-south here. One particularly
large tree with some low branches stands at the edge of the path.

>|
```

Figure 1.1: Example descriptions from Zork I

1.1.1 Description of Games

At every discrete time step t, a state description (s_t) represented by text is output by the game engine. The text is generally of variable length. The agent is expected to take action (a_t), which is a text command. The language and the vocabulary of the state descriptions and actions may be different. Not all actions are permissible in a given state. For example, the action "Take the spoon" does not make sense when the description is "You are in a vacant room." Such actions do not change the state, and the game engine outputs a message (s_{t+1}) which indicates the same. As with other RL environments, a reward (r_{t+1}) accompanies the next state.

1.1.2 Types of Games

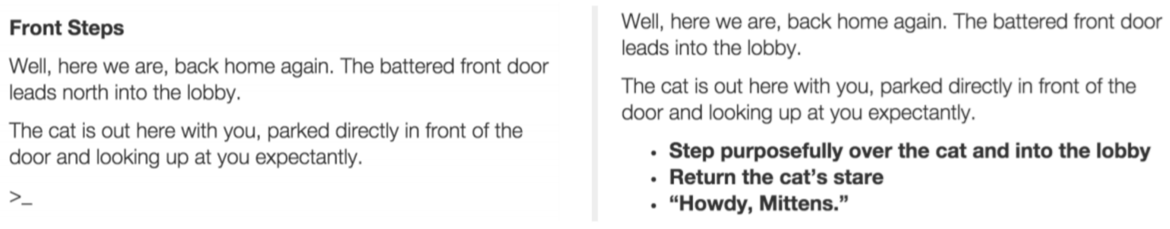
There are mainly three different types of games (He *et al.* [2015])

1. In **Parser-Based Games**, the agent's commands are freeform. These could be phrases or sentences
2. In **Choice-Based Games** the agent chooses one of the options presented to it
3. In **Hypertext-Based Games**, the agent chooses one of the hyperlinks present in the text

In this thesis, we focus on Parser-Based Games as they are the most general of the three. Figures 1.2 and 1.3 illustrate the same.

1.1.3 Challenges in Text-Based Games

Apart from the general challenges in RL environments, Text-Based Games pose the following problems which need to be tackled. (Côté *et al.* [2018])


Front Steps

Well, here we are, back home again. The battered front door leads north into the lobby.

The cat is out here with you, parked directly in front of the door and looking up at you expectantly.

>_

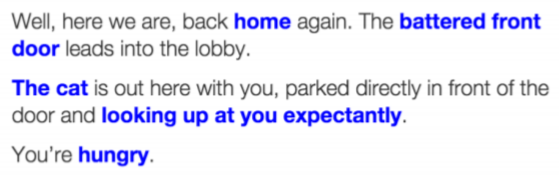
Well, here we are, back home again. The battered front door leads into the lobby.

The cat is out here with you, parked directly in front of the door and looking up at you expectantly.

- **Step purposefully over the cat and into the lobby**
- **Return the cat's stare**
- **"Howdy, Mittens."**

(a) Parser-based (b) Choiced-based

Figure 1.2: Parser-Based and Choice-Based


Well, here we are, back **home** again. The **battered front door** leads into the lobby.

The cat is out here with you, parked directly in front of the door and **looking up at you expectantly**.

You're **hungry**.

(c) Hypertext-based

Figure 1.3: Hypertext-Based

1. **Sparse Rewards:** While this is a problem common to many RL environments, the problem of sparse rewards is severe with Text-based Games. A positive reward is given only if an agent successfully completes a game and no reward is given anywhere else.

2. **Partial Observability:** When the game engine uses text to describe the underlying state, not everything is described. At times some details are left out. It is also possible that different observations can be used for the same states at different points in time. To ensure correct learning, the agent needs to learn a mapping between observations and states.

3. **Large State Space:** Since the state descriptions in Text-Based Games can be of variable length and a sequence of any number of words from the vocabulary, the state space is exponentially large. To alleviate this, an LSTM (Hochreiter and Schmidhuber [1997]) is used to construct a fixed dimensional representation, but there is some loss of information.

4. **Large Action Space:** Similar to the previous problem, the action space can be any sequence of words. Additionally, most of the actions are not even valid in a given state. Thus an agent needs to learn how to pick an action in this very sparse space.

5. **Domain Knowledge:** Text-Based Games often assume some information as they are ultimately designed for humans. For example, a knife is to be used to cut, and a pan is to be used for cooking. While RL agents can mine these associations, there is a combinatorial exploration with respect to the possible

associations when the number of objects increases. Domain Knowledge in such cases is of paramount importance.

1.2 Problem Statement

This thesis aims to tackle the problem of sparse rewards in RL. Rather than modifying the algorithm, the aim is to convert the problem to a dense reward environment which can facilitate improved performance.

The aim is to use the state descriptions to extract rewards in the domain of Text-Based games. For example, in a text-based game, if taking action "Go Right" resulted in the description, "Good Job! You have reached a new room", it indicates that the agent can give itself a positive reward, thus allowing the agent to get dense rewards along the way. Similarly, in a dialogue generation task, if the user says "No, that is not what I asked for", the agent can give itself a negative reward because the user seems irritated.

The motivation stems from the fact that text descriptions are often opinionated. After performing an action, not only does the game engine provide the next state, but also supplies a clue about how the performance of the previous action was. Existing methods do not utilize this extra signal. Using Sentiment Analysis will allow the agent to interpret the state descriptions and give rewards to *itself* as and when needed. This is one step closer to a framework we call "Reinforcement Learning without Rewards" where the agent only receives state descriptions and needs to use domain-knowledge or existing information to mine the rewards.

1.3 Contributions

In this thesis, we introduce the idea of using the state descriptions to extract rewards and thus potentially learning without a reward signal.

In Chapter 1 we present the problem definition and an introduction to Text-Based Games. In Chapter 2 we present the Background material needed to assimilate the ideas in the thesis. Chapter 3 discusses the related work and studies which motivated us to pursue this idea. In Chapter 4 we present three methods, namely, Sentiment Analysis Method, Diverse Density Method, and Validity Detector Method. In Chapter 5 we analyze the advantages and disadvantages of the methods and provide empirical results on the games.

While our results are meek, we feel that our experiments show promise. We provide a proof-of-concept to use the Sentiment Analysis method and list down the requirements for it to work well. We hope that the community will build on this idea of eliminating the need for a reward signal when state descriptions have that hidden information.

CHAPTER 2

Background

2.1 Reinforcement Learning

In this thesis, we consider the standard Reinforcement Learning (RL) setting where an Agent acts in an Environment $\mathcal{E}$. Typically, the Environment is considered to be a Markov Decision Process (MDP). The Agent takes actions (a_t) and interacts with the Environment at discrete time steps after receiving a representation of the current state it is in (s_t). This is also often accompanied by a reward r_t which the Environment provides. The agent aims to learn a policy (π), which is a mapping from states to action probabilities, such that the Return is maximum. In this thesis, the Return (G_t) is defined as the discounted expected reward. $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. γ is called the *discounting factor* and controls how far-sighted the agent is.

The MDP $\mathcal{M}$ is thus characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ which represent the State, Action, Reward sets, and the transition probabilities, $\mathcal{P}(s_{t+1}, r_t | s_t, a_t)$.

The Value Function ($V_\pi(s)$) is the expected return from a state s when following the policy π. Similarly, the Action-Value function ($Q_\pi(s, a)$) is the expected return when the action a is taken in state s and policy π is followed thereafter. $V_*(s)$ and $Q_*(s, a)$ represent the optimal value function and the optimal action-value function respectively. Figure 2.1 depicts the RL framework.

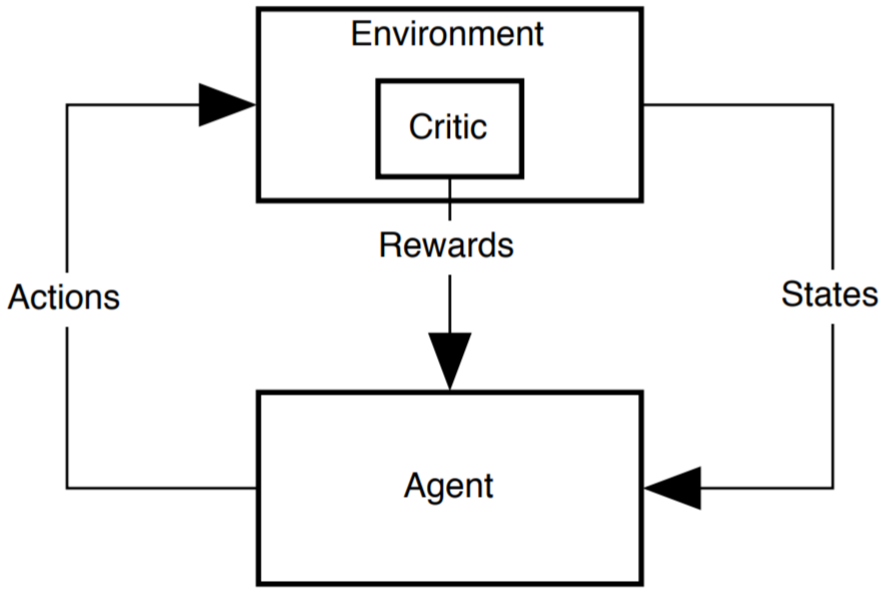


Figure 2.1: RL Framework

2.1.1 Dynamic Programming and Monte-Carlo Methods

Given all the information about the Environment (MDP), which includes the transition probabilities and the rewards, it is quite straight-forward to use Dynamic Programming (DP) methods to arrive at optimal policies. Bellman Optimality equation naturally reduces to a DP solution, but each sweep is very expensive, spanning multiple states for every iteration. Further, the two stages of policy evaluation and policy improvement make it a costly ordeal.

Monte-Carlo (MC) methods rely on spawning multiple rollouts from states and using the discounted expected reward as the value estimates. However, if the trajectories are long, this method might take a long time to converge. The update can be viewed as the following.

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

2.1.2 Q-Learning

Temporal-Difference (TD) Methods, on the other hand, perform *bootstrapping*, which allows one estimate to depend on another estimate. These can be viewed as a hybrid between DP and MC methods (Sutton and Barto [2018]). The 1-step TD-error is usually defined as the following.

$$R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

As a result, the update rule is the following.

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

Q-Learning is one type of TD method for which the update rule is as follows.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Instead of picking an action according to a policy, Q-Learning uses the action which gives the maximum action-value estimate. This also makes it an off-policy method because the max operation allows the usage of trajectories generated by any policy.

2.1.3 Deep Q-Network

Mnih *et al.* [2015] was one of the first studies to ensure stable learning when function approximation and Q-Learning is combined. They used a Convolutional Neural

Network (Krizhevsky *et al.* [2012]) to convert raw pixels of the image to a lower dimensional representation, which was considered as the state representation. Function approximation is useful when the dimensionality of the raw representation is high and tabular methods tend to be expensive. While convergence for Q-Learning can be proved, no convergence results exist for Deep Q-Networks. A few implementations are important to ensure stable learning.

- **Replay Memory** ensures that consecutive updates are not correlated. While this increases the memory requirements, it also paves the way for usage of techniques like Prioritized Experience Replay (Schaul *et al.* [2015]).
- **Target Networks** are used to decrease the correlation between the target and the estimate.

The following is the loss function for iteration i, with θ_i^- representing the parameters of the target network.

$$L_i(\theta_i) =_{s,a,r,s' \sim U(D)} [(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2]$$

2.1.4 Partially Observable Markov Decision Processes (POMDPs)

Until now, we have assumed that in an MDP we observe the state at each time step. For example, if an image based game is being played, then the image, or multiple frames are viewed at each time step. There are however many cases when the state cannot be observed completely, and a different space called the observation space is required. For an intuitive example, if the same image based game had half, it's screen covered, that is partial observability.

The new observation space can be considered as the states, and the same algorithms can be used, but the environment is not Markov any more. In such cases, a belief state (Kaelbling *et al.* [1998]) is maintained, which induces a probability distribution over all the possible states.

Partial Observability is very common in Text-Based games because even if the agent is present in some room, not everything in the room is described at all points of time. However, most studies ignore the partial-observability and hope that the function approximator learns how to handle multiple observations of the same state. In this thesis, we proceed with the same approach of directly applying algorithms to the text/state descriptions.

2.2 Sentiment Analysis

Sentiment Analysis (SA) is an NLP sub-field which tries to extract opinions, emotions, and sentiments from the text. With the explosion in the number of documents on the web, there is no dearth of opinionated articles, making SA very important. It can be viewed as a one number summary of the polarity of the article.

Typically, there are two different types of methods which can perform SA (Medhat *et al.* [2014])

1. **Lexicon-based approach** uses either the dictionary to extract opinionated words, or uses corpus statistics like term frequency or word co-occurrence.
2. **Machine Learning based approach** uses features extracted from the text to learn models like Logistic Regression, SVM or Neural Networks to output the sentiment.

There are three different approaches to SA (Zhang *et al.* [2018])

1. **Document level SA** classifies the whole document into positive, negative or neutral.

2. **Sentence Level SA** is more granular and performs a two-step classification. Subjectivity classification first decides if a sentence is opinionated, and then each sentence is classified as positive or negative.
3. **Aspect-based SA** aims to extract sentiments about different aspects of entities present in the text.

Often, word embeddings are used to represent the words in Machine Learning based approaches as they already encode some kind of polarity. SA is used as a black-box in this thesis.

CHAPTER 3

Related Work

3.1 Reinforcement Learning in Text-Based Games

One of the first models to use RL to solve Text-Based games was the LSTM-DQN Model (Narasimhan *et al.* [2015]). An LSTM is used to process the text, and the fixed dimensional output is used as the state representation. The state representation s is constructed by taking the mean of the output vectors at each step. This is called the representation generator (ϕ_R).

The action scorer (ϕ_A) module calculates the Q values. Since the model only considers commands of the form (*action, object*), the following formulae gives the final Q value, where o represents the object and a represents the action.

$$Q(s, ao) = \frac{Q(s, a) + Q(s, o)}{2}$$

The architecture can handle a fixed number of actions, like the DQN (Mnih *et al.* [2015]).

Replay memory is used to store all the experiences. A variant of Prioritized Experience Replay (Schaul *et al.* [2015]) is used where transitions which have positive rewards are chosen a larger fraction of time as compared to transitions with no or negative rewards.

Experiments are performed on two games called the *Home World* and *Fantasy*

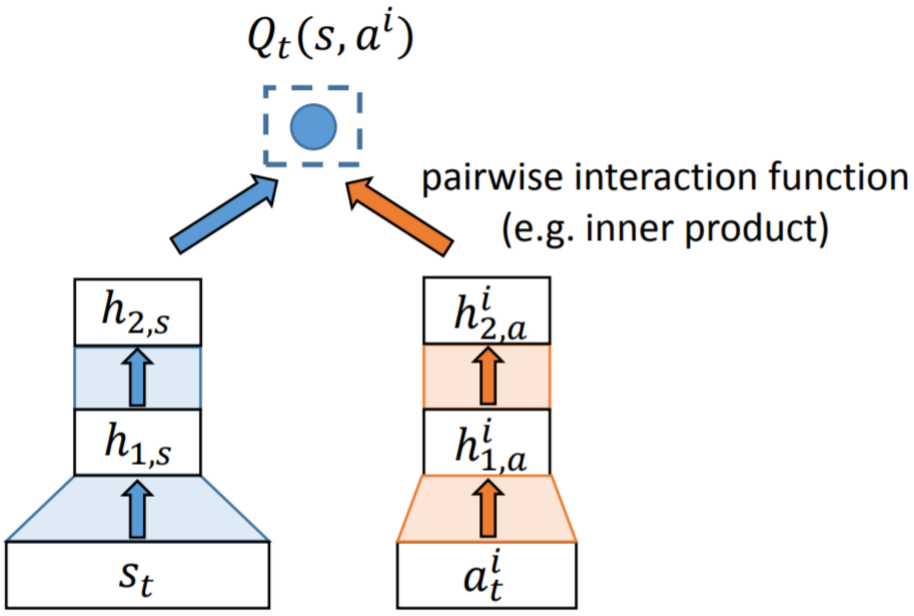


Figure 3.1: DRRN

World with a vocabulary size of 84 and 1340 words respectively. This is much smaller than usual Text-Based Games and the games in our training set. While the agent solves both the games completely, these games are not very hard to solve, given the small number of states that are present.

The LSTM-DQN is a standard architecture used for Text-Based Games. A different architecture called Deep Reinforcement Relevance Network (DRRN) (He *et al.* [2015]) solves the problem of a fixed number of actions. While LSTM-DQN needs to output the action-values of all the actions, DRRN uses separate embeddings for the state space and action space. Two networks are used to process the state and action descriptions, and a pairwise interaction function is used to combine them to get the final value. The advantage of this type of method is that, in theory, it can calculate the action-values of any number of actions, as the interaction function is usually a simple inner product. Figure 3.1 shows the architecture.

There have been multiple studies that aim to solve the challenges talked about in Chapter 1. In Yuan *et al.* [2018], a recurrent agent is proposed to solve Text-

Based Games. LSTM-DQN has the disadvantage of not being able to handle Partial-Observability, and using an LSTM as the representation generator (ϕ_R) will allow the agent to condition its actions on previous states as well. This work is an extension of the Recurrent Deep Q-Network (Hausknecht and Stone [2015]) for the Natural Language Domain. They also use a count-based episodic intrinsic reward much like in Bellemare *et al.* [2016] which improves exploration.

A lot of work has focused on reducing the large action spaces which are common in most Text-Based Games. In Fulda *et al.* [2017], the agent learns to choose affordances. Common-sense knowledge tells that there are certain verb-noun pairs which make sense. For example, you can "Sing Song", but you can't "Sing Door". Utilizing this kind of knowledge allows effective pruning of the action space, thus reducing the complexity of training and improving performance.

Along similar lines, Haroush *et al.* [2018] learns *how not to act* by using an Action Elimination Network (AEN). AEN uses a classifier and previous trajectories to predict which actions will be valid or invalid based on the game engine response. But the agent uses extra information about the validity of actions from the emulator, which is generally not available.

Ammanabrolu and Riedl [2018] learn graph representations for each state which help score actions based on how relevant or valid they are. This scoring module is a very simple NLP module which is not a part of the training procedure.

Tao *et al.* [2018] use an interesting approach to produce actions using generative models. They claim that this model has the potential of producing novel commands that cannot be predicted by other models.

3.2 Inducing Rewards from Text

Some work closely related to ours try to use Natural Language goal specifications to induce a reward function. In Squire *et al.*, Natural Language is used to specify goals. A Reward function is then induced based on the objects that are present in the given state. Applications of this study are however limited by the fact that it uses an Object-Oriented MDP (OOMDP) (Diuk *et al.* [2008]) in which all the objects present in the state need to be specified. Along similar lines, Williams *et al.* [2018] use a Combinatory Categorical Grammar (CCG) to convert commands into reward functions that can be used by the OOMDP.

Krening *et al.* [2017] uses Sentiment Analysis to categorize human reactions into advice and warnings. These are then used in an object-focused way to learn policies which utilize natural language advice without any state information.

3.3 Unsupervised Rewards

Perhaps the work most closely related to ours in spirit would be Sermanet *et al.* [2016]. In that work, a robot infers a reward function based on the current image input. A form of Imitation Learning (Ross and Bagnell [2010]) is used to learn correct trajectories, and intermediate states in successful trajectories are identified. When a robot is in action, based on the intermediate state it is in, it can reward itself if those intermediate states have been seen on successful trajectories before. An example of this is, say a robot needs to learn how to pour water in a glass. Based on the image it is receiving, if its hand is close to the glass, it can give itself a high reward, and if its hand is far away and tilting in some awkward direction, it can give itself a negative reward. In essence, the robot can reward itself, and a possibly sparse reward task with respect to the environment is converted to a dense reward task.

3.4 Intrinsic Motivation

Intrinsic motivation is the will to do something based on something internal rather than some extrinsic property. In the Reinforcement Learning domain, it is the will to act or learn not based on external rewards, but something else entirely. A common example is that of a baby whacking random objects not because it is getting any rewards (in fact it might get punished because it might hurt), but because it is internally curious. To quantify this notion, the baby (agent) has an internal reward mechanism which enables it to perform tasks which might seem futile from an external perspective. Intrinsically Motivated Reinforcement Learning (IMRL) (Chentanez *et al.* [2005] and Şimşek and Barto [2006]) is the sub-domain which focuses on this type of learning.

The concept of "Salient Events" is often used to determine intrinsic rewards. When an agent feels that it has encountered a salient event, it can reward itself. In the case of the baby, objects moving and a sense of control on the objects may be salient events. In the case of a music composer, periodic sounds may be salient events. The concept of intrinsic rewards is depicted in Figure 3.2.

IMRL is related to this thesis because the Critic in the Internal Environment can be viewed as the Sentiment Analysis Engine. The internal critic analyzes the state descriptions which are the sensations and can assign itself rewards, hence learning without any external rewards. The Critic can thus also be considered as Domain Knowledge.

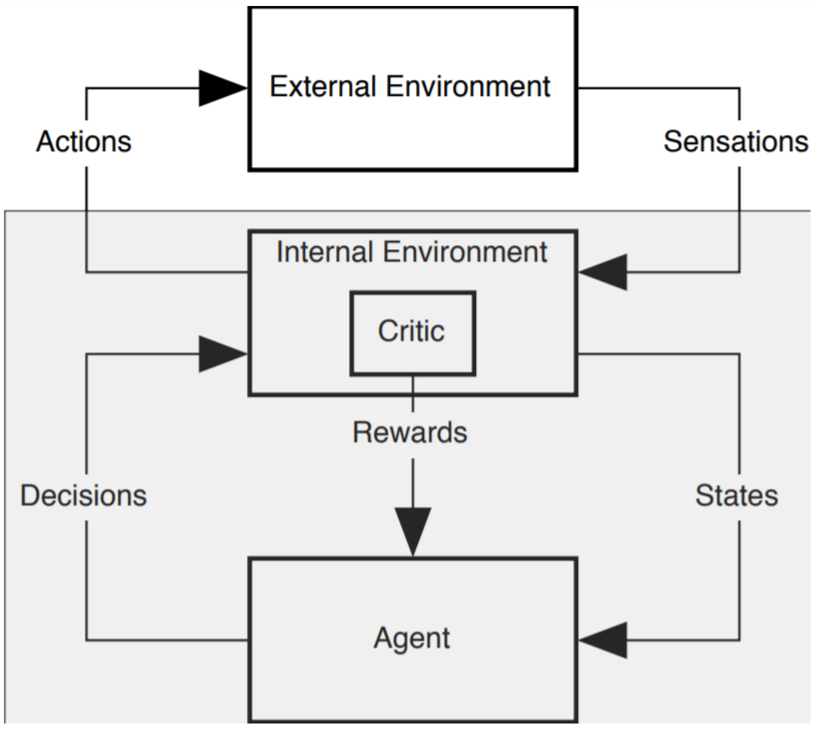


Figure 3.2: IMRL

CHAPTER 4

Method

This chapter discusses different methods which use some variant of Sentiment Analysis to introduce dense rewards. As mentioned in Chapter 3, the rewards can be treated as intrinsic rewards. These rewards are inferred by an internal critic, which in our case is a form of Sentiment Analyzer. The following are a few characteristics which will help differentiate the methods.

Table 4.1: Characteristics of different methods

Type of Sentiment Analyzer
Knowledge of Internal Critic

4.1 Using Sentiment Analysis

This method uses different Sentiment Analysis (SA) techniques to deduce an intrinsic reward from the text. The SA engine is not a part of the training procedure, and thus might not be suitable for games which have many Out-of-vocabulary (OOV) words. The saliency of a state description is determined by the sentiment score. To ensure that it is comparable across games, a baseline b is introduced per game. Some games are inherently encouraging, which means that the average sentiment of the game is high, while other games might be neutral. Subtracting the baseline $(r_i - b)$ ensures normalization. Figure 4.1 depicts the model.

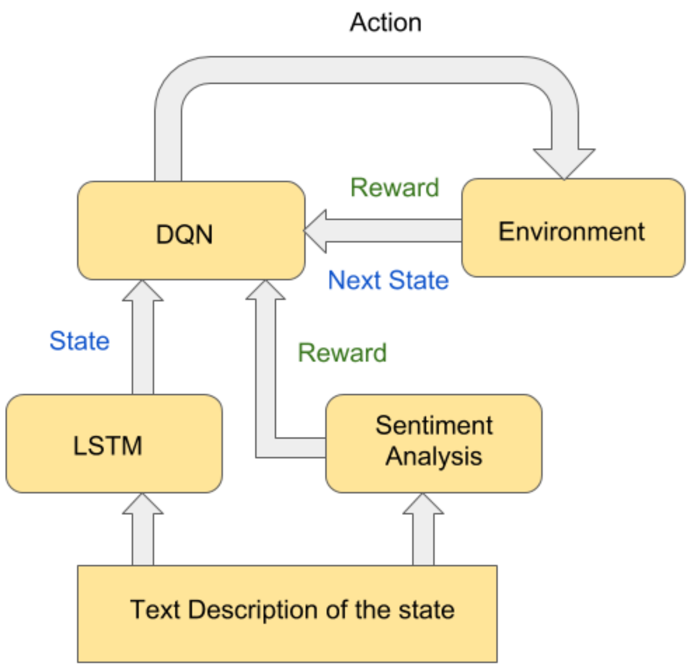


Figure 4.1: Sentiment Analysis Model

We use the Sentiment Analysis model mentioned in Socher *et al.* [2013] because of the availability of code and its performance. We will refer to this SA method as `StanfordNLP` in this thesis. There are five different classes, (- - | - | 0 | + | ++) which stand for very negative, negative, neutral and so on. Sentiment Classification is on a sentence basis. To obtain the scores for the whole paragraph, there are multiple averaging techniques.

1. Take an average of all the sentence scores
2. Take a weighted average with more weight given to more recent sentences
3. Take the min/max of the sentence scores

We use the second technique to obtain the scores. The `StanfordNLP` method uses a DL model called Recursive Neural Tensor Network (RNTN), and thus takes word order into account, unlike other methods which rely on the polarity of individual words. Multiple rule-based SA methods from open-source implementations found here (`http://tiny.cc/w7iy5y`) cannot detect the sentiments of the text descriptions.

The following are the characteristics of the method.

Table 4.2: Sentiment Analysis Method

Type of Sentiment Analyzer	`StanfordNLP`
Knowledge of Internal Critic	External

4.2 Diverse Density

Motivated by a study which finds sub-goals in an RL environment using Diverse Density (McGovern and Barto [2001]), we propose the Diverse Density method. In this study, the aim is to find bottleneck states which serve as sub-goals, and then use them in an options framework (Sutton *et al.* [1999]). To find bottleneck states, positive and negative bags are created, which correspond to successful and unsuccessful trajectories, and the problem is cast as a Multiple-Instance learning problem. The following is the equation used to find concepts, which in this case are states. *DD* represents the diverse density.

$$DD(t) = \prod_{1 \leq i \leq n} Pr(t|B_i^+) \prod_{1 \leq i \leq m} (1 - Pr(t|B_i^-))$$

The state with the highest Diverse Density is picked as the bottleneck state. Some heuristics need to be used neglect states near the start or goal states as they tend to have high diverse densities. The intuitive idea is to pick states which are present in positive instances and not present in negative instances.

We modify the DD method to learn an SA model by using state descriptions from unsuccessful trajectories as negative instances and successful trajectories as positive instances. This method works only when successful trajectories can be obtained, either by using a random agent or by learning a vanilla LSTM-DQN model. Over-sampling is used to negate the imbalance between successful and unsuccessful episodes. We use a word based SA method instead of a more sophisticated approach for the following reasons.

1. There is not enough training data to use supervised learning methods like neural networks
2. Text-Based games have very few adversarial examples like double negations. Phrases like "not not good" are generally not present.
3. Parser-Based games do not have a large number of synonyms. Even if they do, increasing the number of trajectories should handle the problem quite easily

We use n-grams to generate short phrases which appear in positive trajectories and not in negative trajectories. We use a modified Naive-Bayes approach and take the ratio $DD = \frac{P(w|Successful)}{P(w|Un-successful)}$ into account. Phrases with a high DD but low counts are removed. The final intrinsic reward will be $(r_{SA} + \lambda_1 r_{DD})$, thus taking both game-related and external sentiments into account.

We, however, concede that a major limitation of this method is the fact that it needs a few successful trajectories to work, and this might not be possible in many games by using a random agent. To test the usefulness of the method, we use walkthroughs to generate positive trajectories. The following table lists the characteristics.

Table 4.3: Diverse Density Method

Type of Sentiment Analyzer	`Naive-Bayes`
Knowledge of Internal Critic	Internal

4.3 Validity Detector

The third method uses a `Validity Detector` to assess the validity of an action in a particular state. We borrow ideas from Hausknecht *et al.* [2019] and use three different classes to decide the validity of an action.

- Unrecognized action
- Recognized action but failed
- Recognized action and successful

Note that we take into consideration only the success or failure of the action, and not if it was part of a successful trajectory. Unlike the other two methods which introduce rewards based on how high the value of the state is, this method gives exploration bonuses which improves convergence time and overcomes sub-optimal solutions.

The classifier is a FastText Classifier (Joulin *et al.* [2016]) trained on a manually generated dataset. Note that this is not a Deep-Learning classifier and uses mostly linear and bag-of-words models. Apart from the dataset used by Hausknecht *et al.* [2019], we add some successful and unsuccessful responses from our training games and train a classifier on the same.

Table 4.4: Validity Detector Method

Type of Sentiment Analyzer	`Validity Detector`
Knowledge of Internal Critic	Internal and External

CHAPTER 5

Experiments and Discussion

5.1 Environments and Models

5.1.1 Games

We use a subset of train games provided as part of the TextWorld Framework (Côté *et al.* [2018]) which allows both creation and execution of Text-Based Games. Experimentation with the more famous `z5` games is harder because of the following reasons.

1. RL agents cannot still achieve good performance on these games and it will be hard to see if modifications to LSTM-DQN are indeed helpful
2. The games are not related to each other in any way, have a completely different vocabulary and are of varying difficulty levels
3. The action spaces are typically very different. Synonyms are often unrecognized, giving a false notion that the agent is not learning

The advantages of using our training set are that the games can be arranged based on difficulty levels, and the action space is more free-form than a lot of `z5` games. Also, a large number of games (1000) allows us to give statistically significant results.

The games are cooking-based games where recipes have to be prepared. There are 3 categories, ones in which the agent needs to prepare 1 recipe, 2 recipes, and so on.As expected, categories 2 and 3 are harder. We perform analysis on each sub-category to see the performance improvements.

5.1.2 Evaluation Metric

We experiment on the three different sub-categories and provide individual scores. Each game has a different maximum that can be achieved. The score achieved by the agent is normalized by the maximum score, and the average is taken. An example is shown below.

Table 5.1: Evaluation Metric Example

Agent Score	Maximum Score
1	3
2	4
2	2

$$Score = \frac{\frac{1}{3} + \frac{2}{4} + \frac{2}{2}}{3}$$

5.1.3 Model

A few changes are made to the LSTM-DQN model (Narasimhan *et al.* [2015]) so that commands with more than two words can be generated. In the original implementation, the average of the Q-values of the action and object are taken.

$$Q(s, ao) = \frac{Q(s, a) + Q(s, o)}{2}$$

But this approach has limitations. Instead, a specific template is chosen which best suits the games in question.

$$verb \quad adjective \quad noun \quad preposition \quad adjective - 2 \quad noun - 2$$

The preposition is determined by using a preposition map which depends on the verb that is predicted. Some examples are given below.

$$take : from, \quad chop : with, \quad slice : with, \quad insert : into, \quad put : on$$

Five words corresponding to the template are predicted first.

1. If the verb is either `inventory` or `look`, then that is considered as the action
2. If the verb is not present in the preposition map, `(verb, adjective, noun)` is considered as the action
3. If the verb is present in the map, `(verb, adjective, noun, adjective-2, noun-2)` is considered as the action
4. If any adjective is the end-of-line (`<EOL>`) token, it is ignored

When the prediction is being made for the *verb*, a *verb mask* is used which ensures that the Q-values for all non-verbs are set to the minimum possible value. The same procedure is followed for adjectives and nouns. Figure 5.1 depicts the model used.

5.2 Sentiment Analysis Method

We use the `StanfordNLP` method and compare it against LSTM-DQN. Some examples of sentiment scores are given below.

Table 5.2: Sentiment Analysis Examples

Sentence	**Class**
But the thing is empty, unfortunately	Negative
Adding the meal to your inventory	Negative
You eat the meal. Not bad.	Positive

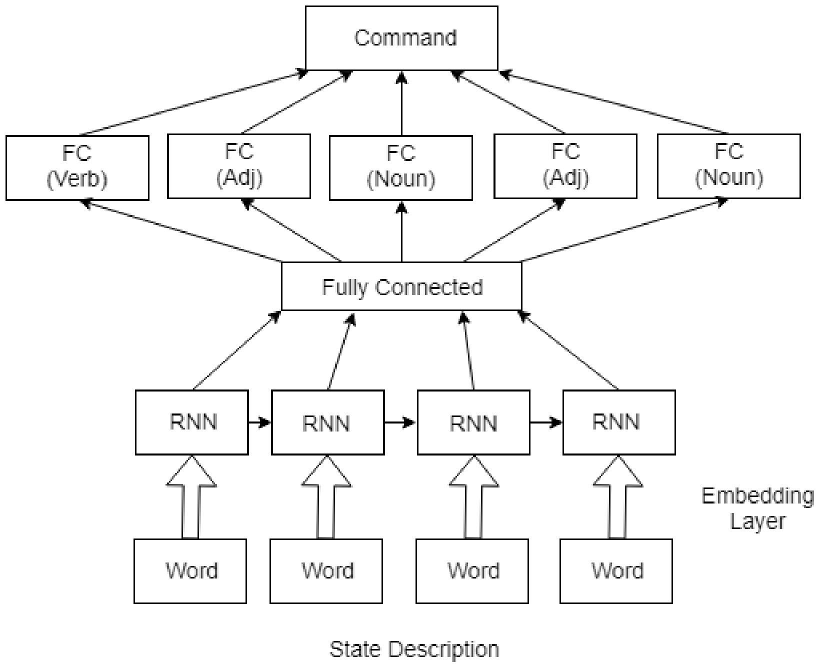


Figure 5.1: Model

While the first and second sentences show that the SA method has captured a positive sentiment, there are often examples like the second sentence where the SA method might predict the wrong class.

The total reward that the model receives is $(r_{ext} + \lambda_1 r_i)$, and we choose $\lambda_1 = 0.2$, after tuning it using the values $[0.05, 0.1, 0.2, 0.5, 1]$.

The following are the scores on different games.

Table 5.3: Sentiment Analysis Method Scores

Category	**LSTM-DQN**	**SA**
Recipe 1	0.17	0.18
Recipe 2	0.11	0.11
Recipe 3	0.10	0.10

As can be seen, there is a minimal performance improvement in the case of Recipe 1 games, which is attributed to the agent successfully solving a few games in which it was not able to achieve positive rewards. This shows that adding intrinsic rewards helps. But the fact that there is no score change in the other two categories is dismal.

Whenever a very positive step is taken in the direction of the goal, a positive reward is issued by the Environment. To check how our method performs when the rewards are really sparse, we remove all such rewards and give a '+1' only when the agent reaches the goal state. In this case, the following are the scores.

Table 5.4: Sentiment Analysis Method Scores (with modification)

Category	**LSTM-DQN**	**SA**
Recipe 1	0.13	0.13
Recipe 2	0.07	0.08
Recipe 3	0.07	0.08

On further analysis, we found that when the rewards on the way are removed, the performance of LSTM-DQN degrades on all the tasks. But the performance in a few games for Recipe 2 and Recipe 3 are higher. This is because, after finishing one recipe, a positive sentiment state description is given, which allows the SA agent to reward itself, ensuring the agent moves on to the next recipe.

However, there is no score difference between the two methods on Recipe 1 tasks, which means that even though our method is picking positive/negative sentiments, the rewards are not enough to offset the performance. There is promise in this method; we could accurately point out cases when giving a positive reward helped the agent take the right action after completing a recipe.

We feel that the failure of this method is because the Sentiment Analysis method uses external knowledge and is often not able to recognize the sentiments of state descriptions which would be considered positive by human observers.

5.3 Diverse Density Method

We use a modified Naive-Bayes method coupled with oversampling to identify words with strong positive sentiments. The following are phrases with the most positive sentiments after removing stopwords. We also remove colors from the list of words because they are disentangled factors (Higgins *et al.* [2018]).

Table 5.5: Phrases with positive sentiments

Phrases
Your inventory
The meal
You dice
Eat the meal

The total reward is thus given by $(r_{ext} + \lambda_1 \times (\lambda_2 r_{sa} + r_{dd}))$, where r_{sa} and r_{dd} correspond to rewards given by the SA and the DD method respectively. We use the same λ_1 from the previous section, and $\lambda_2 = 0$ gave the best results. Again, we remove the intermediate rewards for comparison. The following are the results on the games.

Table 5.6: Diverse Density Method (with modification)

Category	**LSTM-DQN**	**DD**
Recipe 1	0.13	0.16
Recipe 2	0.07	0.08
Recipe 3	0.07	0.08

As can be seen, there is an improvement in performance on all three categories. While this method is unrealistic because it uses walkthroughs to generate positive trajectories, it validates the fact that a Sentiment Analyzer with correct domain knowledge might be useful.

It is interesting to note that all the performance improvement in Recipe 2 and Recipe 3 is because of the agent completing the first recipe in games where it was failing before. The score improvement is not because of the completion of the whole game in cases where it was partially completed. This might be because words in state representations temporally closer to the initial state are ranked higher than other words.

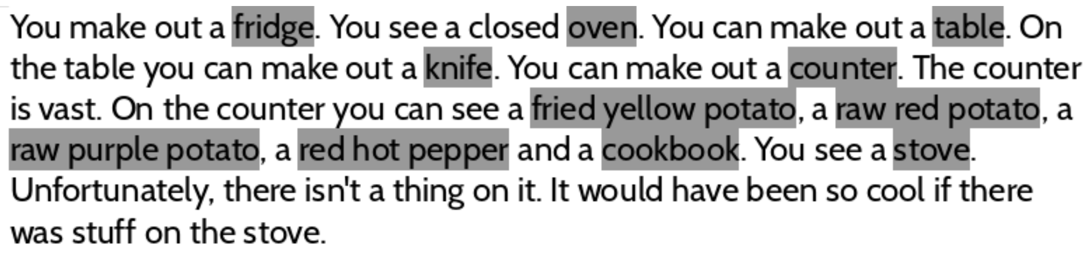
5.4 Validity Detector Method

As described in Chapter 4, we add intrinsic rewards based on the validity of the action, and not necessarily the presence in successful trajectories. The results are dismal though.

Table 5.7: Validity Detector Method Scores

Category	**LSTM-DQN**	**Validity Detector**
Recipe 1	0.17	0.17
Recipe 2	0.11	0.11
Recipe 3	0.10	0.10

The performance actually degrades on some games as well. We speculate that this behavior is because of the large number of entities present in the states of the games we considered. In typical games, there are not many objects, and performing a valid action on them will more often than not mean that the agent is one step closer to the goal. However, in the games we consider, there are many redundant objects, and performing valid actions on them gives an intrinsic reward bonus without actually taking the agent any closer to the goal. An example is presented in Figure 5.2. The entities are highlighted in grey.



You make out a fridge. You see a closed oven. You can make out a table. On the table you can make out a knife. You can make out a counter. The counter is vast. On the counter you can see a fried yellow potato, a raw red potato, a raw purple potato, a red hot pepper and a cookbook. You see a stove. Unfortunately, there isn't a thing on it. It would have been so cool if there was stuff on the stove.

Figure 5.2: Entities in State Descriptions

CHAPTER 6

Conclusion and Future Work

From our experiments, the partial success of the Diverse Density method shows that state descriptions can indeed provide more information than previously thought. Just like a robot which can perceive its surroundings and infer if it is doing well or not by itself, agents in Text-Based games can use Sentiment Analysis to infer usefulness of the state.

Even though the DD method offers only meek results, it shows a lot of promise because it corroborates the fact that choosing the correct words which represent sentiments can improve performance. This is a step in the direction of Intrinsically Motivated Reinforcement Learning.

We hope to extend the DD method by making it a continual learning loop. Instead of training in two phases, one being the sentiment analysis learning and the other being policy learning, we can execute them in tandem, ensuring that both the modules improve as learning progresses.

Another direction we wish to explore is the exploitation of domain knowledge. Sentiment Analysis methods without any domain knowledge are not useful, so having a method which can both acquire domain knowledge from the game it is playing, and can also use the knowledge it has acquired from other games, can help in a variety of ways.

- Domain Knowledge can facilitate Transfer Learning
- It can help eliminate actions, thus reducing the large action space to a much smaller set of valid actions

- It can help in the prediction of the effects of the actions, thus enabling the use of Curiosity-Driven Exploration (Pathak *et al.* [2017]) and World Models (Ha and Schmidhuber [2018])

We hypothesize that domain knowledge is crucial to solving Text-Based games and creating methods which can utilize both knowledge and data (trajectories) will be key to solving these games.

Another direction we want to extend our work in is to consider the usage of multiple templates. While the games with which we experimented in this thesis required only 1 template, there are other games which might require very different commands. In such a case, we could have action-value estimates over different templates, which would in turn choose the command. This setup is very similar to Hierarchical-DQN (Kulkarni *et al.* [2016]) with the key difference being that the options in this setup will pick only one action. There is thus limited temporal abstraction.

Another natural extension of our work is to use RL for Dialogue Generation (Li *et al.* [2016]). If the RL agent opts for a Human-in-the-loop learning paradigm, it gets constant human feedback with respect to how the user is feeling at that moment. Since human-generated texts are usually opinionated, our method can not only use them as state descriptions but also as reward signals, thus accelerating learning.

APPENDIX A

Hyperparameters

Table A.1: Hyperparameters

Name	**Value**
Discount Factor	0.5
Replay Memory Priority Fraction	0.25
Batch Size	16
Epochs	500
Steps per episode	50
Optimizer	Adam
Learning Rate	0.001
Embedding Size	64
LSTM Hidden State Dimension	192
Fully Connected Layer Dimension	128
Random Seed	42

Training for 1 game takes about 14-32 minutes on an 8-core machine with an NVIDIA Tesla P4 Graphics Card.

REFERENCES

Ammanabrolu, P. and **M. O. Riedl** (2018). Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*.

Bellemare, M., **S. Srinivasan**, **G. Ostrovski**, **T. Schaul**, **D. Saxton**, and **R. Munos**, Unifying count-based exploration and intrinsic motivation. *In Advances in Neural Information Processing Systems*. 2016.

Chentanez, N., **A. G. Barto**, and **S. P. Singh**, Intrinsically motivated reinforcement learning. *In Advances in neural information processing systems*. 2005.

Côté, M.-A., **Á. Kádár**, **X. Yuan**, **B. Kybartas**, **T. Barnes**, **E. Fine**, **J. Moore**, **M. Hausknecht**, **L. E. Asri**, **M. Adada**, *et al.* (2018). Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Diuk, C., **A. Cohen**, and **M. L. Littman**, An object-oriented representation for efficient reinforcement learning. *In Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

Fulda, N., **D. Ricks**, **B. Murdoch**, and **D. Wingate** (2017). What can you do with a rock? affordance extraction via word embeddings. *arXiv preprint arXiv:1703.03429*.

Ha, D. and **J. Schmidhuber** (2018). World models. *arXiv preprint arXiv:1803.10122*.

Haroush, M., **T. Zahavy**, **D. J. Mankowitz**, and **S. Mannor** (2018). Learning how not to act in text-based games.

Hausknecht, M., **R. Loynd**, **G. Yang**, **A. Swaminathan**, and **J. D. Williams** (2019). Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*.

Hausknecht, M. and **P. Stone**, Deep recurrent q-learning for partially observable mdps. *In 2015 AAAI Fall Symposium Series*. 2015.

He, J., **J. Chen**, **X. He**, **J. Gao**, **L. Li**, **L. Deng**, and **M. Ostendorf** (2015). Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*.

Higgins, I., **D. Amos**, **D. Pfau**, **S. Racaniere**, **L. Matthey**, **D. Rezende**, and **A. Lerchner** (2018). Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*.

Hochreiter, S. and **J. Schmidhuber** (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

Joulin, A., **E. Grave**, **P. Bojanowski**, and **T. Mikolov** (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kaelbling, L. P., **M. L. Littman**, and **A. R. Cassandra** (1998). Planning and acting in partially observable stochastic domains. *Artificial intelligence*, **101**(1-2), 99–134.

Kober, J., **J. A. Bagnell**, and **J. Peters** (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, **32**(11), 1238–1274.

Krening, S., **B. Harrison**, **K. M. Feigh**, **C. L. Isbell**, **M. Riedl**, and **A. Thomaz** (2017). Learning from explanations using sentiment and advice in rl. *IEEE Transactions on Cognitive and Developmental Systems*, **9**(1), 44–55.

Krizhevsky, A., **I. Sutskever**, and **G. E. Hinton**, Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*. 2012.

Kulkarni, T. D., **K. Narasimhan**, **A. Saeedi**, and **J. Tenenbaum**, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *In Advances in neural information processing systems*. 2016.

Li, J., **W. Monroe**, **A. Ritter**, **M. Galley**, **J. Gao**, and **D. Jurafsky** (2016). Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

McGovern, A. and **A. G. Barto** (2001). Automatic discovery of subgoals in reinforcement learning using diverse density.

Medhat, W., **A. Hassan**, and **H. Korashy** (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, **5**(4), 1093–1113.

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

Narasimhan, K., **T. Kulkarni**, and **R. Barzilay** (2015). Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Pathak, D., **P. Agrawal**, **A. A. Efros**, and **T. Darrell**, Curiosity-driven exploration by self-supervised prediction. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.

Ross, S. and **D. Bagnell**, Efficient reductions for imitation learning. *In Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010.

Schaul, T., **J. Quan**, **I. Antonoglou**, and **D. Silver** (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Sermanet, P., **K. Xu**, and **S. Levine** (2016). Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*.

Silver, D., **A. Huang**, **C. J. Maddison**, **A. Guez**, **L. Sifre**, **G. Van Den Driessche**, **J. Schrittwieser**, **I. Antonoglou**, **V. Panneershelvam**, **M. Lanctot**, *et al.* (2016). Mastering the game of go with deep neural networks and tree search. *nature*, **529**(7587), 484.

Şimşek, Ö. and **A. G. Barto**, An intrinsic reward mechanism for efficient exploration. *In Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

Socher, R., **A. Perelygin**, **J. Wu**, **J. Chuang**, **C. D. Manning**, **A. Ng**, and **C. Potts**, Recursive deep models for semantic compositionality over a sentiment treebank. *In Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013.

Squire, S., **S. Tellex**, **D. Arumugam**, and **L. Yang** (). Grounding english commands to reward functions.

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

Sutton, R. S., **D. Precup**, and **S. Singh** (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, **112**(1-2), 181–211.

Tao, R. Y., **M.-A. Côté**, **X. Yuan**, and **L. E. Asri** (2018). Towards solving text-based games by producing adaptive action spaces. *arXiv preprint arXiv:1812.00855*.

Williams, E. C., **N. Gopalan**, **M. Rhee**, and **S. Tellex**, Learning to parse natural language to grounded reward functions with weak supervision. *In 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

Yuan, X., **M.-A. Côté**, **A. Sordoni**, **R. Laroche**, **R. T. d. Combes**, **M. Hausknecht**, and **A. Trischler** (2018). Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.

Zhang, L., **S. Wang**, and **B. Liu** (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **8**(4), e1253.