Robotic Instruction for Multiple Agents via Natural Language

Isaac Peterson
isaac.peterson@usu.edu

Kaiden McMillen
email@domain

Braxton Geary
braxton.geary@usu.edu

Abstract

In this project, we explore the application of reinforcement learning (RL) to train an agent capable of navigating a 2D grid environment and identifying geometric shapes with distinct colors. The agent recieves instruction via natural langauge and then the agent moves one space at a time across the grid, using RL techniques to learn an optimal policy for efficiently locating and identifying shapes such as green triangles and red squares. We define the task as a partially observable Markov decision process (POMDP), where the agent's observations are limited to the grid space it occupies. Our approach involves implementing Reinforcement Learning to teach the agent to maximize rewards by minimizing the time and steps required to find and correctly identify a shape. The results demonstrate how reinforcement learning can be applied to shape recognition and navigation tasks, with potential applications in robotics, search-and-rescue missions, and autonomous systems. If time permits, we want to further explore what modifications need to be made in order to command a fleet of agents to accomplish tasks in an optimal manner.

1 Introduction

As technologies continue to advance, the integration of robots into every day life is continuing to increase. As humans rely mostly on speech for communication and instruction, it is essential that robots are also developed to understand and decipher language, executing commands effectively and efficiently.

There are a myriad of challenges that one confronts when attempting to solve this problem. First and foremost, establishing an interface between the spoken command and correct execution of that command. To properly execute the spoken command, the agent needs to have an accurate understanding of its environment, which in the real world can become extremely complex, and the connection between the environment and the spoken language. Initial attempts were made by utilizing more logic based methods to help the robot understand the task that needs to be solved (Liu, 2016). To mitigate the challenges that come with real world interpretation, many researchers defaulted to simulations to instruct agents in a more structured world.

Video games form a natural challenge for agents, with clear tasks to complete in a very structured world. (Chaplot et al., 2017) used the environment in the video game DOOM™to train their agent to follow natural language instructions, with other researchers using similiary techniques in Minecraft™(Tessler et al.), and even developing their own virtual environments for natural language task completion (Anderson et al., 2017) (Wang et al., 2024).

Our goal is to use the simplified environment in Minigrid (Chevalier-Boisvert et al., 2023) to explore the process of natural langauge task completion, with more research on the multi-agent natural language task completion problem. Where instead of instructing a single agent to complete a task, we instruct a fleet of agents and explore the challenges that come with the increased number of agents along solutions to address these challenges.

2 Related Work

Most of the work in this project will be based on the work done by (Chaplot et al., 2017). They utilize a combination of large language models and reinforcement learning to help an agent understand specific instructions to navigate in the game environment of doom. Others utilized the world of minecraft to help agents complete tasks, with less of a focus on instruction and more on generalized learning rather than interpreting language (Oh et al., 2017), (Tessler et al.). Combining the implementation of (Chaplot et al., 2017) with the

Minigrid environment (Chevalier-Boisvert et al., 2023), (Chevalier-Boisvert et al., 2018) we can further explore the challenging problem of robotic instruction with large language models, with the added challenge of addressing multiple agents.

3 Methods

The Minigrid environment is available on a public repository in github. Which will form the foundation for the environment we will use for the work that we will present in this research. Also foundational to our research is the State Processing Module as shown in Figure 1. Our modified state processing module to use the Minigrid environment and handle multiple agents is shown in Figure 2.

Using notation as described in (Chaplot et al., 2017) with slight modification to address our problem, we consider multiple agents interacting with an episodic environment Σ, at the beginning of each episode, each agent recieves the same natural language instruction L, which in our case is a description of the task within the Minigrid environment. At each timestep the agent recieves information about the 8 surrounding blocks around the agent in the environment E_t. The episode ends when the agents complete their task or some prefixed maximum time length is met. Let $s_t = (E_t, L)$ represent the state at every timestep, the goal is for the agents to learn the optimal policy $\pi(a_t|s_t)$, which maps the observed states in Minigrid to the optimal next action.

We will also use the vector representation of the instruction along with a single attention layer to convert the instruction L into a machine interpretable vector $x_L = f_a(L)$. We will use a multilayer perceptron (MLP) to interpret the grid environment, where e is a vector of length 8 x n representing the 8 blocks surrounding the agent, and n being equal to the number of agents, and the result after being passed through the MLP is $x_E = f_{mlp}(e)$. We concatenate these two vectors together to get our output vector $X = [x_L||x_E]$ which we pass onto our reinforcement learning model to update our policy $\pi(a_t|s_t)$.

4 Expected Experiments

Since reinforcement learning is an online learning model, we only require the environment as the data to be explored and used to train our model. As mentioned previously we will be using the Minigrid environment to accomplish this task. However, we will have to handcraft some scenarios and reward for the multi-agent portion of this research. Typically the minigrid environment was used for a single agent to complete levels by collecting a key and then exiting to the next level via a green square.

The minigrid environment provides many building blocks for reinforcement learning tasks, such as a reward for each action. We will have to update this reward to handle multiple agents. For example, if we had two agents completing the level in the minimum number of steps, the optimal policy would be for one agent to grab the key, while the other agent waits by the exit until the key is retrieved. We will need to customize the environment in Minigrid so that the agents can learn this type of behavior. In order to establish baselines for our model performance, we will determine the optimal timesteps to complete each task and compare that with the actions taken by the agents. We can even incorporate this into their reward at the end of the epsiode so the agents learn to complete each level in a more optimal manner.

5 Teamwork

The design of the framework as shown in Figure 2 makes it easy to deligate the tasks into three separate stages for the project.

5.1 Environment Setup

Kaiden McMillen

5.2 State Processing Module

Isaac Peterson will setup the state processing module. This includes handling the state information from the Minigrid environment and the input instruction L, passing them through their appropriate models and creating the output vector X to be processed by the reinforcement learning model.

5.3 Multi-Agent Reinforcement Learning

Braxton Geary will develop the Reinforcement Learning (RL) module for our agent using a Deep Q-Network (DQN) integrated with an Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al., 2016). The architecture is comprised of an initial fully connected layer, followed by an LSTM layer, and a concluding fully connected layer, allowing it to process inputs from the State Processing Module. The module outputs a tuple with a scalar critic value and an array of action probabilities. Where the Actor and Critic share the

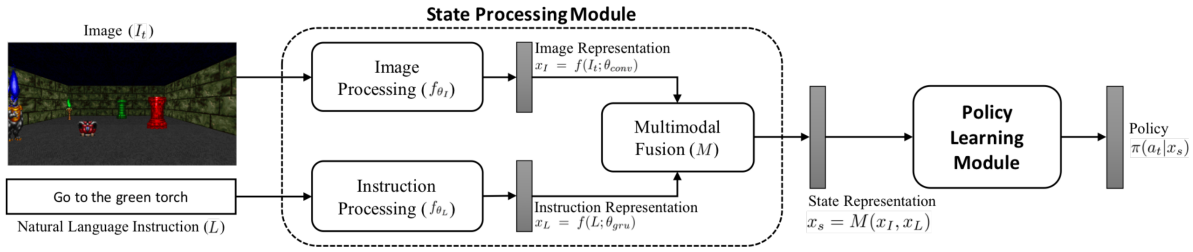

Figure 1: State processing method as developed by (Chaplot et al., 2017).

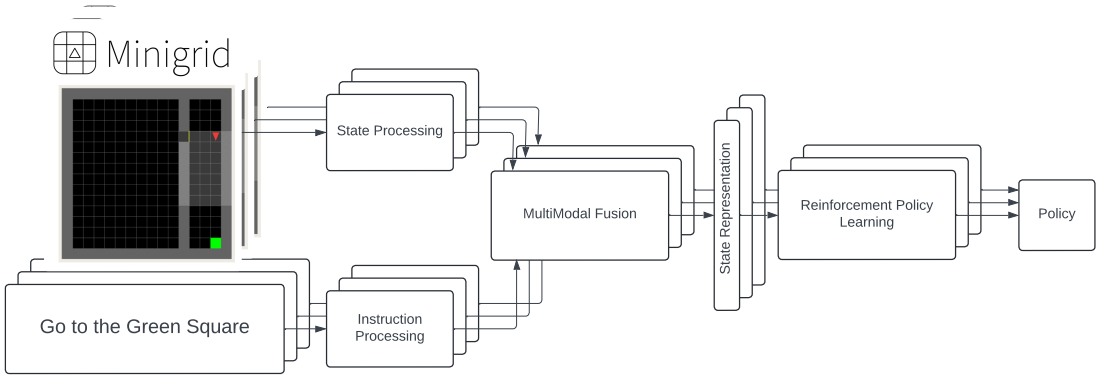

Figure 2: Our approach to solving the multi-agent natural language task-solving problem, showing layers for each agent.

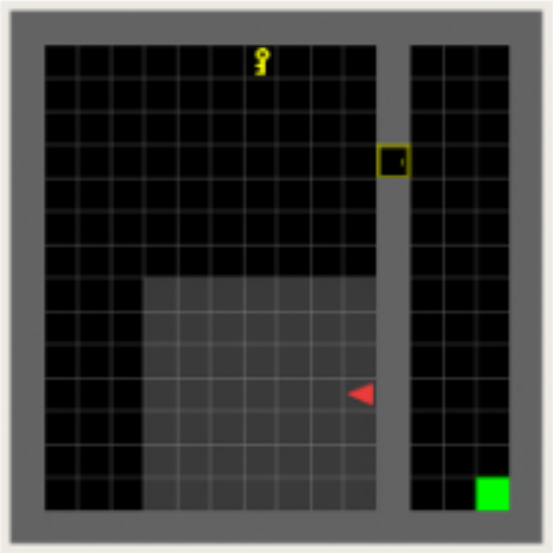

Figure 3: Example of a minigrid level.

same neural network, the actor's output undergoes a softmax operation to yield action probabilities, a_t. Given that the Minigrid environment limits certain actions, we will apply a negative infinity mask to the logits for these unavailable actions, ensuring they remain unselected by the agent. To further enhance stability, we will also incorporate a Proximal Policy Optimization (PPO) approach, which will strengthen the model's reliability and effectiveness across diverse scenarios.

References

2016. *Natural-Language-Instructed Industrial Task Execution*, volume Volume 1B: 36th Computers and Information in Engineering Conference of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2017. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2017. Gated-attention architectures for task-oriented language grounding.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft.

Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, Peizhou Cao, Wenye Yu, Zichao Ye, Jialun Li, Junfeng Long, Zirui Wang, Huiling Wang, Ying Zhao, Zhongying Tu, Yu Qiao, Dahua Lin, and Jiangmiao Pang. 2024. Grutopia: Dream general robots in a city at scale.