Task-Oriented Gameplay in Minecraft with Gated-Attention Networks

Ali Zaidi alikazim@stanford.edu
Stanford University / Microsoft AI and Research

Objectives and Results

- **Minecraft** is an enormously popular and highly customizable 3D grid-world game.
- We use Minecraft as a platform to examine **task-oriented language grounding**.
- **Task-oriented language grounding** refers to the challenge where autonomous agents must perform tasks specified by natural language instructions.
- We show that **Gated Attention Networks**, which combine image and text representations from the environment, perform very effectively on this problem.
- We use **policy and imitation learning** methods to effectively train our autonomous agent in Minecraft.

Our environment is based on the incredible platform Project Malmo [JHHB16], which provides a complete environment for AI experimentation on Minecraft.

Problem Formulation

Figure 1: An example of task-oriented grounding

State, Action and Policy

In task-oriented language grounding, the agent has two sources of information:

- The first is the *natural language instructions*, I.
- The second is the *first-person view of the environment*, M_t
- Let's denote by $s_t = \{I, M_t\}$ the state at each time step.
- The agent needs to extract semantically meaningful representations of the instructions and map it to the specifics of the scene dynamics.
- In particular, the goal of the agent is to learn an *optimal policy* $\pi(a_t|s_t)$, mapping observed states into optimal actions for the task.

Concretely, our goal is to find a policy that chooses an action in state s_t such that it maximizes the utility function (expected return):

$$J(\pi) = \mathbb{E}_{s_0,a_0,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].$$

Network Architecture and Policy Learning Methods

We use a single end-to-end architecture that combines the two sources of information, [CSP+17]. These two information sources are divided into two modules, a state processing module, which we describe next, and a policy learning module, which we describe below.

Convolutional Image Representations

As defined above, our states are defined by $s_t = \{I, M_t\}$ where M_t is the first-person view of the environment at iteration t. Each frame is fed through a convolutional neural network to create a convolutional representation of the state, $x_M = f(M_t, \theta_{\text{conv}}) \in \mathbb{R}^{d\times H\times W}$, where d denotes the number of feature maps, i..e, the intermediate representations in the convolutional network. This is shown in the upper module of Figure 2.

GRU Instructions Embeddings

Our natural language instructions, I, are processed using a Gated-Recurrent Unit (GRU), $x_I = f(I; \theta_{\text{GRU}})$, which we call our **instruction embeddings**. This is shown using the bottom module of the network architecture in Figure 2.

Attention and Hadamard Products

In addition, these instruction embeddings are passed through a fully-connected layer with a sigmoid activation function to create an attention vector, $a_I = h(x_I) \in \mathbb{R}^d$. Furthermore, each element of the attention vector is expanded into a $H \times W$ matrix to match the size of our state-image, which is then multiplied element-wise with the output of the CNN (i.e., we use the Hadamard product of the two embeddings):

$$M_{\text{GA}}(x_I, x_M) = M(h(x_I)) \odot x_M = M(a_I) \odot x_M$$

Gated Attention Network Architecture

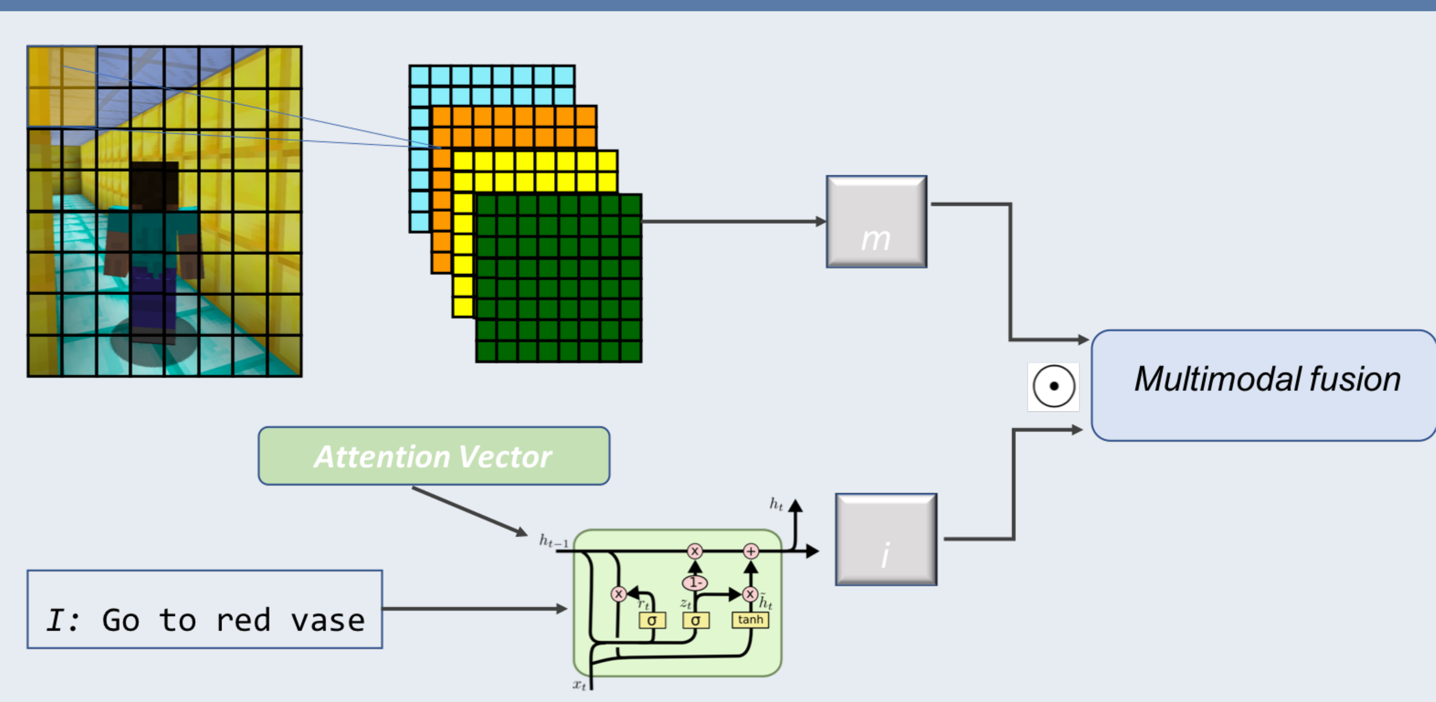

Figure 2: CNN-GRU network with attention and multimodal fusion

Policy Learning Module

The multimodal fusion unit, M_{GA} is provided to the policy learning module. We use imitation learning to learn the optimal policy. In particular, an oracle is implemented that finds the target object's location described by the natural language instructions. It then orients itself in the direction facing the longest corridor to the object and finds the shortest path through an $A^\star$ path to the object. The imitation learning module is trained with a single fully connected layer to estimate the policy function. Alternatively, we also utilize a Stein policy gradient, a particle based variational method that controls the gradient variance, [LW16].

Rather than optimizing for a single policy, Stein variational policy gradient searches for a distribution $q(\theta)$ to optimize the expected return using variational methods. Concretely, we formulate our variational optimization objective as

$$\max_q \left\{\mathbb{E}_q\left[J(\theta)\right] + \alpha \boldsymbol{H}(q)\right\},$$

where the second term provides entropy of the variational distribution q, and α is a "temperature" parameter that controls the rate of exploration. It's optimal value is given by

$$q(\theta) \propto \exp\left(\frac{1}{\alpha}J(\theta)\right) q_0(\theta).$$

This variational objective has the advantage of acting as a regularizer/control variate, and a fast particle based simulation mechanism.

Results

For experiments, we provide our agent with instructions to pick up specific items in a maze, with penalty-states that terminate the agent's life (i.e., falling down a a hole, or picking the wrong item). We examined our architecture at varying distances and with varying number of items which are shown in Table 1. The maximum reward is 1, and the numbers shown in the table are an average of 100 runs for each experiment. The experiments differ from how far the player's initial position is from the item to be retrieved, and from the number of occluding items there are in the player's path. Our initial results show that the model is fairly robust to the distance of the reward, but less robust when occluding items are placed in the agent's path.

	Number of items		
Distance	1	2	4
10	0.82	0.75	0.49
15	0.82	0.74	0.43
30	0.77	0.73	0.40
50	0.76	0.72	0.35
100	0.73	0.69	0.33

Table 1: Mean reward test results

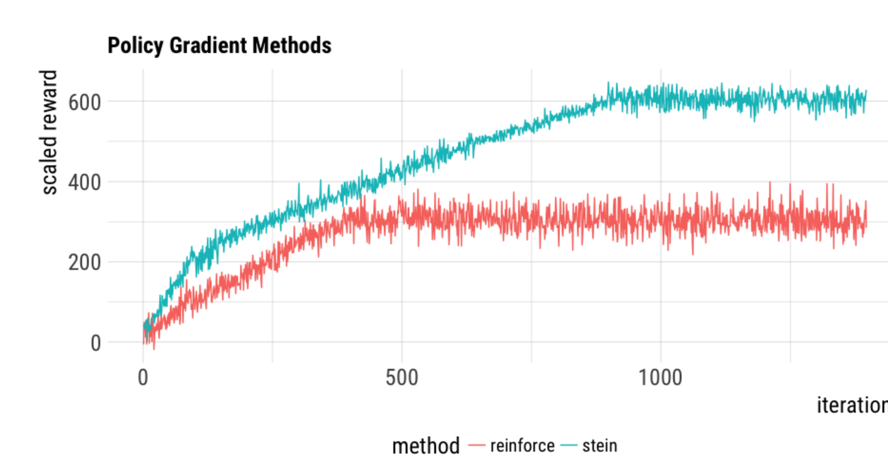

Figure 3: Learning Curves for SVPG and REINFORCE

Next Steps

- One-shot learning for objects not in training set.
- Comparisons with other policy gradient methods, such as *Asynchronous advantage acrtic critic, A3C algorithm* and the *Trust-region policy optimiation algorithm, TRPO*.
- Analysis of attention maps.

References

[CSP+17] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov, *Gated-attention architectures for task-oriented language grounding*, CoRR **abs/1706.07230** (2017).

[JHHB16] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell, *The malmo platform for artificial intelligence experimentation*, Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, 2016, pp. 4246–4247.

[LW16] Qiang Liu and Dilin Wang, *Stein variational gradient descent: A general purpose bayesian inference algorithm*, Advances in Neural Information Processing Systems 29 (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), Curran Associates, Inc., 2016, pp. 2378–2386.