Reinforcement Learning for Mapping Instructions to Actions with Reward Learning

Dipendra Misra
Dept. of Computer Science and Cornell Tech
Cornell University
New York, NY 10044
dkm@cs.cornell.edu

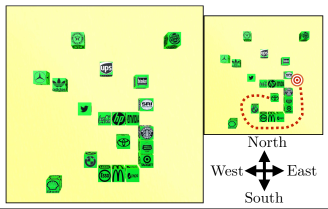
Yoav Artzi
Dept. of Computer Science and Cornell Tech
Cornell University
New York, NY 10044
yoav@cs.cornell.edu

Introduction

Natural language is an efficient medium for non-expert users to specify tasks for robotic agents. To effectively map natural language instructions to actions, a robotic agent must solve natural language, sensing, and planning problems. For example, consider the Blocks environment and instructions in Figure 1 (Bisk, Yuret, and Marcu 2016). The agent observes the environment as an RGB image using a camera sensor. Given the RGB input, the agent must recognize the blocks and their layout. To understand the instruction, the agent must identify the block to move (Toyota block) and the destination (just right of the SRI block). This requires solving semantic and grounding problems. For example, consider the topmost instruction in the figure. The agent needs to identify and ground the phrase *Toyota block* referring to the block to move. It must resolve and ground the phrase *SRI block* as a reference position, which is then modified by the spatial meaning recovered from *the same row as* or *first open space to the right of*, to identify the goal position. Finally, the agent needs to generate actions, for example moving the Toyota block around obstructing blocks.

Previous work assumed a symbolic environment representation (Chen and Mooney 2011; Artzi and Zettlemoyer 2013; Artzi, Das, and Petrov 2014; Misra et al. 2015; Mei, Bansal, and Walter 2016), or combined separately trained models to solve the different problems (Matuszek, Fox, and Koscher 2010; Tellex et al. 2011). We recently proposed a single-model approach for mapping instructions and visual observations to actions (Misra, Langford, and Artzi 2017). Our approach does not require intermediate representations, planning procedures, or training different models. Training relies on a reinforcement learning method approximated in a contextual bandit setting (Langford and Zhang 2007). During learning, the reward function requires access to the world state to evaluate task progress and completion. While this can be achieved by instrumenting the training environment (Levine et al. 2016), this is not always practical.

We address this limitation by learning a distance-based reward function (Popov et al. 2017) using distance learning, an efficient method for learning a distance function (Weinberger, Blitzer, and Saul 2006). Learning relies on simple statistics over execution states. Once learned, the reward function is computed using agent observations and does not require access to the world state. In this abstract, we describe our ongoing work towards this goal. We evaluate in the simulated Blocks environment (Bisk, Marcu, and Wong 2016), and empirically demonstrate the effectiveness of the learned reward. Our approach shows limited degradation in performance in comparison to a reward that has access to the world state, while outperforming supervised learning and common reinforcement learning methods.



Put the Toyota block in the same row as the SRI block, in the first open space to the right of the SRI block
Toyota to the immediate right of SRI, evenly aligned and slightly separated

Figure 1: Instructions in Blocks. The above instructions describe the same task. Given the observed RGB image of the start state (large image), our goal is to execute such instructions. In this task, the direct path to the target position is blocked. The agent must move the Toyota block around. The small image shows the target and an example path, which includes 34 steps.

Problem Description

Task Let $\mathcal{X}$ be the set of all *instructions*, $\mathcal{S}$ the set of all *world states*, and $\mathcal{A}$ the set of all *actions*. An instruction $\bar{x} \in \mathcal{X}$ is a sequence $\langle x_1, \ldots, x_n \rangle$, where each x_i is a token. The agent executes instructions by generating a sequence of actions, and indicates execution completion with the special action STOP. Action execution modifies the world state following a transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The execution $\bar{e}$ of an instruction $\bar{x}$ starting from s_1 is an m-length sequence $\langle (s_1, a_1), \ldots, (s_m, a_m) \rangle$, where $s_j \in \mathcal{S}$, $a_j \in \mathcal{A}$, $T(s_i, a_i) = s_{i+1}$ and $a_m = \texttt{STOP}$. In Blocks (Figure 1), a state specifies all block positions. For each action, the agent moves a block on the plane in one of four directions (north, south, east, or west). There are 20 blocks, and 81 possible actions at each step, including STOP. For example, the action TOYOTA-SOUTH moves the Toyota block one step south.

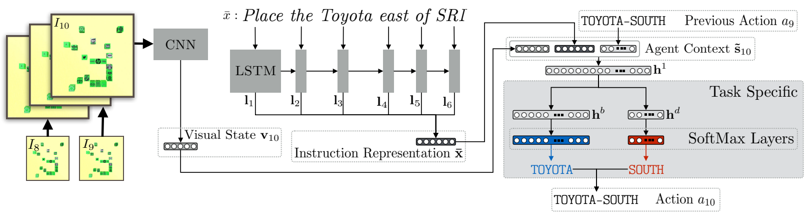

$\bar{x}$: Place the Toyota east of SRI
I_{10}
I_8
I_9
CNN
LSTM
$\mathbf{l}_1$ $\mathbf{l}_2$ $\mathbf{l}_3$ $\mathbf{l}_4$ $\mathbf{l}_5$ $\mathbf{l}_6$
Visual State $\mathbf{v}_{10}$
Instruction Representation $\bar{\mathbf{x}}$
TOYOTA-SOUTH Previous Action a_9
Agent Context $\tilde{\mathbf{s}}_{10}$
$\mathbf{h}^1$
Task Specific
$\mathbf{h}^b$
$\mathbf{h}^d$
SoftMax Layers
TOYOTA
SOUTH
TOYOTA-SOUTH Action a_{10}

Figure 2: Illustration of the policy architecture showing the 10th step in the execution of *Place the Toyota east of SRI* starting from the state in Figure 1. The inputs are the instruction $\bar{x}$, the current state image I_{10}, previous state images (I_8 and I_9), and the previous action a_9. The text and images are embedded with a recurrent neural network LSTM (Hochreiter and Schmidhuber 1997) and a convolutional neural network CNN (LeCun et al. 1998). The action is selected with a multi-layer perceptron.

Data We train and evaluate on Blocks. The data set includes $16,767$ natural language instructions with a vocabulary of $1,426$. The mean instruction length is 15.27, and the mean number of actions 15.4. When compared to common data sets (MacMahon, Stankiewics, and Kuipers 2006; Matuszek et al. 2012; Misra et al. 2015), the instructions are longer, have a larger vocabulary, and require a larger number of actions (Misra, Langford, and Artzi 2017, Table 1).

Single-Model Approach

Model The agent observes the world state using a camera sensor. Given a world state s, the agent observes an RGB image $I \in \mathcal{I}$ generated by the function $\text{IMG}(s)$. We distinguish between the world state s and the *agent context* $\tilde{s}$, which includes the instruction, the image observation $\text{IMG}(s)$, images of previous states and the previous action. To map instructions to actions, the agent reasons about the agent context to generate a sequence of actions. At each step, the agent generates a single action. We model the agent with a neural network policy. At each step j, the network takes as input the current agent context $\tilde{s}_j$, and predicts the next action to execute a_j. Figure 2 illustrates the policy network. For full details see Misra, Langford, and Artzi (2017).

Learning We estimate the policy parameters using reinforcement learning in a contextual bandit setting. In a contextual bandit setting, maximizing the immediate reward suffices and provides stronger theoretical guarantees than unconstrained reinforcement learning (Agarwal et al. 2014).

Reward with an Instrumented Environment

In Misra, Langford, and Artzi (2017), we define a simple task-completion reward computed from the world state. To leverage demonstrations of the desired system behavior, we use reward shaping (Ng, Harada, and Russell 1999; Wiewiora, Cottrell, and Elkan 2003). Computing this reward requires instrumenting the training environment, a challenging engineering task in complex domains.

Reward Learning with Metric Learning

We adopt an inverse reinforcement learning (Ng and Russell 2000, IRL) approach and learn a reward function that is computed from the agent context. The original reward function is based on comparing world state and computing distances between them. Therefore, we cast the reward learning problem as learning a distant metric between world states, as observed in the agent context. Formally, our goal is to learn a distance function $d_\theta : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ with parameters θ. We define $d_\theta(I_1, I_2) = \|\phi_\theta(I_1) - \phi_\theta(I_2)\|_2$, where ϕ is a convolutional neural network with parameters θ. To learn the distance function, we assume access to a dataset of image triplets $\{(I_a^{(n)}, I_+^{(n)}, I_-^{(n)})\}_{n=1}^N$, where the state of $I_+^{(n)}$ is closer to state of $I_a^{(n)}$ than the state of $I_-^{(n)}$. We minimize the triplet loss (Weinberger, Blitzer, and Saul 2006):

$$L = \frac{1}{N}\sum_{n=1}^{N} \max\{0, d_\theta(I_a^{(n)}, I_+^{(n)}) - d_\theta(I_a^{(n)}, I_-^{(n)}) + 1\} \ .$$

To generate triplets we assume access to executions with progress meta data. The progress data includes for each execution step if it is closer to the final state than the previous step or further. A state may be further from the final state than the previous if the agent was moving further to, for example, go around an obstacle. We generate triplets for each pair of adjacent states and the final execution state. The image of the final state is I_a, and the images of the next state and the current state are used as I_+ and I_-, or vice versa, based on the progress meta data. The reward function is then defined as a potential difference $R(I, a, I') = d_\theta(I_g, I) - d_\theta(I_g, I')$ where I, I' and I_g are the images of the current, next, and goal states.

Algorithm	Distance Error
Demonstrations	0.35
RANDOM	15.3
SUPERVISED	4.65
REINFORCE (Sutton et al. 1999)	5.57
DQN (Mnih et al. 2013)	6.04
Our approach	
w/instrumented environment	3.60
w/learned reward	4.07

Table 1: Mean error development results.

Results Table 1 shows current development results. We measure execution error as the distance between the final and goal states, normalized by the block size. We report the mean error of following the demonstrations, random behavior (RANDOM), supervised learning (SUPERVISED), two reinforcement learning baselines (REINFORCE and DQN), our contextual bandit approach with instrumented training environment, and our approach with a learned reward.

Conclusion

We use metric learning to induce a reward function for learning to map instructions to actions. The reward function does not require the world state, and enables reinforcement learning without instrumenting the training environment. Our contextual bandit learning approach is designed for a few-samples regime. When the number of samples is unbounded, the drawbacks observed in this scenario for optimizing longer term reward do not hold.

References

Agarwal, A.; Hsu, D. J.; Kale, S.; Langford, J.; Li, L.; and Schapire, R. E. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the International Conference on Machine Learning*.

Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics* 1:49–62.

Artzi, Y.; Das, D.; and Petrov, S. 2014. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Bisk, Y.; Marcu, D.; and Wong, W. 2016. Towards a dataset for human computer communication via grounded language acquisition. In *Proceedings of the AAAI Workshop on Symbiotic Cognitive Systems*.

Bisk, Y.; Yuret, D.; and Marcu, D. 2016. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Chen, D. L., and Mooney, R. J. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9.

Langford, J., and Zhang, T. 2007. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17.

MacMahon, M.; Stankiewics, B.; and Kuipers, B. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Matuszek, C.; Herbst, E.; Zettlemoyer, L. S.; and Fox, D. 2012. Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Matuszek, C.; Fox, D.; and Koscher, K. 2010. Following directions using statistical machine translation. In *Proceedings of the international conference on Human-robot interaction*.

Mei, H.; Bansal, M.; and Walter, R. M. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Misra, K. D.; Tao, K.; Liang, P.; and Saxena, A. 2015. Environment-driven lexicon induction for high-level instructions. 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)*.

Misra, D.; Langford, J.; and Artzi, Y. 2017. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Ng, A. Y.; Harada, D.; and Russell, S. J. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*.

Popov, I.; Heess, N.; Lillicrap, T.; Hafner, R.; Barth-Maron, G.; Vecerik, M.; Lampe, T.; Tassa, Y.; Erez, T.; and Riedmiller, M. 2017. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M.; Banerjee, A.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.

Weinberger, K. Q.; Blitzer, J.; and Saul, L. K. 2006. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, 1473–1480.

Wiewiora, E.; Cottrell, G. W.; and Elkan, C. 2003. Principled methods for advising reinforcement learning agents. In *Proceedings of the International Conference on Machine Learning*.