Learning Compositional Tasks from Language Instructions

Lajanugen Logeswaran¹, Wilka Carvalho², Honglak Lee¹,²

¹LG AI Research
²University of Michigan, Ann Arbor
llajan@lgresearch.ai

Abstract

The ability to combine learned knowledge and skills to solve novel tasks is a key aspect of generalization in humans that allows us to understand and perform tasks described by novel language utterances. While progress has been made in supervised learning settings, no work has yet studied compositional generalization of a reinforcement learning agent following natural language instructions in an embodied environment. We develop a set of tasks in a photo-realistic simulated kitchen environment that allow us to study the degree to which a behavioral policy captures the systematicity in language by studying its zero-shot generalization performance on held out natural language instructions. We show that our agent which leverages a novel additive action-value decomposition in tandem with attention-based subgoal prediction is able to exploit composition in text instructions to generalize to unseen tasks.

Introduction

Human language is characterized by systematic compositionality: one can combine known components – such as words or phrases – to produce novel linguistic combinations (Chomsky 2009). This is a key aspect of generalization in humans and enables us to understand and perform tasks specified by novel language utterances over familiar words or phrases. If you know what a “laptop” and a “fridge” are, you can easily understand how to perform the task “place the laptop in the fridge” even if you have never placed a laptop in a fridge.

Prior work studying the linguistic “systematicity” of neural networks have focused on sequence mapping tasks in a supervised learning setting (Lake and Baroni 2018; Lake 2019; Andreas 2019). In this work, we are interested in compositional generalization of a reinforcement learning agent following natural language instructions in an embodied environment. In particular, we explore the hypothesis that a language-conditioned reinforcement learning agent with a compositional inductive bias in its behavioral policy will exhibit systematic generalization to unobserved natural language instructions.

There has been a flurry of recent work on embodied learning tasks such as question answering (Gupta et al. 2017), vision-language navigation (Anderson et al. 2018) and object interaction (Shridhar et al. 2020; Carvalho et al. 2020; Singh et al. 2020; Corona et al. 2020; Blukis et al. 2022; Min et al. 2021) in embodied settings. In particular, the ALFRED task (Shridhar et al. 2020) studies agents that exploit detailed natural language instructions to generalize to novel instructions in novel environments at test time. Such existing benchmarks offer limited flexibility to study systematic generalization since (i) the benchmarks were not built for this purpose and it is unclear to what extent systematic generalization skills are required to solve the tasks and (ii) the tasks demand challenging reasoning skills such as visual recognition, planning over large number of time-steps and exploration in unseen layouts which makes it difficult to study compositional generalization ability in isolation.

In this work we develop a set of tasks in the AI2Thor virtual home environment (Kolve et al. 2017) which test the compositionality of embodied agents. In order to make
Despite these simplifications, agents still need to understand the instruction to figure out the sequence of object interactions that need to be performed and act over many time-steps with limited guidance. In order to successfully generalize at test time, an agent needs to learn to ground natural language instructions to temporally extended goal-oriented behaviors or “skills” in a compositional manner to perform novel tasks that are compositions of the tasks presented at train time. In addition, the agent needs to ground the actions and objects referenced in the text to actions and objects in the environment. We leverage this setting to develop and study a policy with an inductive bias for compositionality and show that this enables systematic generalization in the context of combining behavioral skills learned purely from reward without expert demonstrations.

We present an attention-based agent that learns to predict subgoals from language instructions via a learned attention mechanism. Our agent uses these subgoals with a novel policy parametrization which decomposes the action-value function in an additive fashion that enables estimating the action-value for novel object-interactions composed of objects and interactions experienced during training.

We show evidence that this parametrization facilitates exploiting the compositional nature of text instructions by showing systematic generalization to both unseen task descriptions and unseen tasks. We present an example in Figure 1, where the agent is able to systematically generalize the behavior “pickup up the knife” to “move the knife” and “cut the yellow apple” to “slice the apple”. Thanks to the additive inductive bias afforded by our action-value parametrization, it is able to compose these behaviors to perform the novel task “move the knife near the table and slice the apple” at test time.

**Related Work**

**Compositional Generalization** Compositionality is an important aspect of generalization: decomposing a learning problem into components that the agent knows how to address. An agent with a compositional understanding of its environment and its own skills can more easily combine these skills to solve tasks never observed during training. Prior work has studied compositional generalization in sequence mapping tasks. Benchmarks such as SCAN (Lake and Baroni 2018) and gSCAN (Ruis et al. 2020) study translating synthetic text descriptions to an action sequence (e.g. jump twice → JUMP JUMP). gSCAN couples SCAN instances with entities in a grid environment and solving a task requires grounding the text and entities similar to our work. Prior approaches for these benchmarks impose compositional inductive biases in models by augmenting models with memory (Lake 2019) and data augmentation (Andreas 2019; Shaw et al. 2020). Lake (2019) proposed a memory-augmented sequence-to-sequence model and a meta-learning algorithm to enable models to recognize the compositional structure of instances. Andreas (2019) introduce a data augmentation approach where they synthesize new examples by combining phrases from known examples to provide a compositional inductive bias to models. Shaw et al. (2020) demonstrate a grammar induction method and combine it with a pre-trained T5 model (Raffel et al. 2019) for semantic parsing tasks. In this work we use attention mechanisms and introduce a novel policy parameterization to impose compositional inductive biases.

**Text-based Embodied Control** There is rich literature on instruction following agents (Chen and Mooney 2011; Tellex et al. 2011; Mei, Bansal, and Walter 2016; Fried, Andreas, and Klein 2017; Suhr et al. 2019). Advances in photo-realistic simulation environments such as DeepMind Lab (Beattie et al. 2016) and AI2Thor (Kolve et al. 2017) have driven recent progress in embodied agents that learn from text instructions. Chaplot et al. (2018) consider a simple navigation task where an agent has to move to an object specified by a set of attributes such as shape and color. They propose the gated attention model to generalize compositionally in the attribute space. Hill et al. (2019) consider systematic generalization in 2D and 3D environments with synthetic text instructions. Compared to these work, we consider object interaction tasks in a photo realistic simulated environment with human-authored language instructions.

ALFRED (Shridhar et al. 2020) is a recently proposed benchmark which couples tasks in the AI2Thor environment with text instructions. The dataset comes with expert demonstrations and human-written text descriptions of tasks. Prior approaches to ALFRED train policies using imitation learning (Singh et al. 2020; Corona et al. 2020). More recently, policies based on spatial semantic map building have received more interest (Blukis et al. 2022; Min et al. 2021).

In contrast to the ALFRED learning setup, we consider a simplified scenario of learning compositional skills from high-level task descriptions. We further do not assume access to expert task demonstrations. These assumptions allow us to focus on compositional generalization to zero-shot tasks, which is not the main goal of the ALFRED benchmark. However, the approach presented here can potentially be applicable to ALFRED when combined with learning from demonstrations.

**Hierarchical Reinforcement Learning** Learning to directly map perceptions to low-level action sequences can be challenging. An alternative hierarchical approach is to first come up with a sequence of subtasks, which can be considered as high-level actions (Andreas, Klein, and Levine 2017; Zhu et al. 2017). Each of those subtasks can then be realized using low-level actions. Our policy has an implicit hierarchical structure where latent subgoals are represented as text embeddings using attention. Language was used as an abstraction for the high-level policy in Jiang et al. (2019a) for object rearrangement tasks based on the CLEVR engine (Johnson et al. 2017).

Finally, generalization to unseen instructions has been
Figure 2: Approach Overview. We perform attention over the text instruction to construct an embedding $t_{sg}$ that represents the current subgoal. The text embedding subgoal $t_{sg}$ attends to scene object embeddings to construct an object subgoal representation $v_{sg}$. An MLP takes $t_{sg}$, $v_{sg}$ and observation features $e_{obs}$ as input and predicts state-action values $Q(s, a)$. The entire model is trained end-to-end using Q-learning. See text for details.

considered in prior work such as Oh et al. (2017); Lynch and Sermanet (2020), although compositional generalization is not their main focus.

**Problem**

We consider an embodied agent acting in a kitchen environment to solve basic tasks from language instructions (See Fig. 5 for an example task). At the beginning of an episode the agent receives a text instruction $\tau$. Our goal is to learn a policy $\pi(a|s, \tau); a \in A, s \in S$ that predicts actions in order to complete tasks. The agent state $s$ is partially observable – it receives an egocentric observation $o_{s}$ of the scene. We further assume that an oracle object recognition model provides the object ids for objects in the egocentric observation.

The action space consists of navigation and object interaction actions $A = A_{nav} \cup A_{int}$. There are 8 navigation actions $A_{nav} = \{move\ forward,\ move\ back,\ move\ left,\ move\ right,\ turn\ left,\ turn\ right,\ look\ up,\ look\ down\}$. Interaction actions $A_{int} = B \times I$ are specified using an interaction $b \in B$ and an object id $o \in I$ where $B = \{pickup,\ place,\ slice,\ toggle\ on,\ toggle\ off,\ turn\ on,\ turn\ off\}$ and $I$ is a pre-defined set of identifiers of objects that are available to the agent for interaction in the current observation.

The agent receives a positive reward for successfully completing a task. It also receives a small negative reward for every time-step. In addition, we also assume that every correct object interaction receives a positive reward. In addition to providing a denser learning signal, the rewards are also used to identify subgoals as described in the next section. In practice such dense rewards may be unavailable, but this is outside the scope of our study and left as future work.

**Approach**

We approach the problem by considering a task $\tau$ to be composed of subgoals $g_1, ..., g_n$, where each subgoal $g_i$ involves navigating to a particular object and interacting with it. For example, the task *place an apple on the table* involves finding the apple and picking it up, followed by navigating to the table and putting down the apple, which can be considered to be the two subgoals for executing the task. Each object interaction required to complete the task thus corresponds to a subgoal. Since every subgoal completion receives a positive reward, the number of subgoals completed at every time-step $N_{sg}$ is known to the agent. The subgoals themselves are not known to the agent – we use attention on the text instruction to compute a latent subgoal representation.

**Text Subgoal Inference**

Given instruction $\tau$ composed of the tokens $(w_1, ..., w_n)$, we obtain the corresponding token embeddings $E = (e_1, ..., e_n)$ and use an RNN to encode the instruction to obtain a sequence of contextualized token representations $H = (h_1, ..., h_n)$. We compute a text subgoal $t_{sg}$ for a given time-step by computing attention on the instruction using $N_{sg}$ as query where $N_{sg}$ is a vector representation of $N_{sg}$. This is shown in Eq. (1) ($Q, K, V$ are learnable parameter matrices).

$$t_{sg} = Attention(query = N_{sg}, keys = values = H)$$

$$= \sum_{h \in H} \frac{\exp((Q(s))^\top (K(h)))}{\sum_{h' \in H} \exp((Q(s))^\top (K(h')))} V_h$$

We expect the attention to focus on words in the instruction relevant to executing the current subgoal. For instance, if the agent is expected to interact with an apple, the attention module could learn to focus on the word ‘apple’.

**Cross-modal Reasoning**

Given the text subgoal $t_{sg}$, we use an attention mechanism to reason about objects in the scene within some distance from...
the agent. This helps the agent understand if objects of interest relevant to the subgoal are present nearby. Let the set of nearby scene objects be \( O = \{ o^1, \ldots, o^n \} \), where the \( o^i \) is an object id assigned by an oracle. The \( o^i \)s can thus be treated as indexes into an embedding table that produces object embeddings \( O_e = \{ o^1_e, \ldots, o^n_e \} \). The cross-modal attention is given by Eq. (4) where the text subgoal attends to the scene object embeddings \( Q', K', V' \) are learnable parameter matrices. We augment the scene objects embeddings \( O_e \) with an additional learned embedding \( o^{a-o-b} \) which is expected to absorb any probability mass not assigned to scene objects \( O'_e = O_e \cup o^{a-o-b} \). The attention produces an object subgoal embedding \( v_{sg} \).

\[
v_{sg} = \text{Attention}(query = t_{sg}, keys = values = O'_e) \tag{3}
\]

\[
= \sum_{o_e \in O'_e} \alpha_{o_e} \sum_{o'_e \in O'_e} \exp((Q't_{sg})^T (K'o'_e)) V'o_e \tag{4}
\]

### Policy Learning

We use a deep Q-learning algorithm to train a policy (Mnih et al. 2013), where a neural network is trained to approximate the state-action value function \( Q(s, a) \). Given the current observation, text subgoal and object subgoal, the state-action value for a navigation action \( a \in \mathcal{A}_{nav} \) is given by Eq. (5), where \( f_{nav} \) is an MLP (multi-layer perceptron) and \( e_{obs} = f_{CNN}(obs) \) is a feature vector of the observation image computed using a CNN encoder.

\[
Q_{nav}(s, a) = f_{nav}(a|e_{obs}, t_{sg}, v_{sg}) \tag{5}
\]

The state-action values for interaction actions \( a = (b, o) \in \mathcal{B} \times I \) can be analogously modeled as in Eq. (6). We found it helpful to decompose the state-action value in an additive fashion over an action score \( f_{int}^b \) and an object score \( f_{int}^o \) as in Eq. (7). Intuitively, \( f_{int}^b \) learns to model action preferences, whereas \( f_{int}^o \) learns to ground text goals to physical objects. In addition to sharing parameters across actions and objects, this decomposition allows us to model state-action values of object interactions not experienced during training, as long as the specific interaction and the object were encountered. Unless specified otherwise we use the decomposed value function \( Q_{int}^{add} \) in our experiments.

\[
Q_{int}^{full}(s, a) = f_{int}(a|e_{obs}, t_{sg}, v_{sg}) \tag{6}
\]

\[
Q_{int}^{add}(s, a) = f_{int}^b(b|e_{obs}, t_{sg}, v_{sg}) + f_{int}^o(o|t_{sg}) \tag{7}
\]

where \( a = (b, o) \in \mathcal{B} \times I \).

In summary, the state-action value function is modeled as in Eq. (8).

\[
Q(s, a) = \begin{cases} 
Q_{nav}(s, a); & a \in \mathcal{A}_{nav} \\
Q_{int}(s, a); & a \in \mathcal{A}_{int} 
\end{cases} \tag{8}
\]

The overall model (see Fig. 2 for an illustration) including parameters of the subgoal inference (Eq. 1) and cross-modal reasoning (Eq. 4) components, as well as the MLPs in Eqs. (5) and (7) are trained end-to-end using a double-DQN algorithm (Van Hasselt, Guez, and Silver 2016). Once the model has been trained we construct a greedy policy by choosing actions with the highest state-action values for inference.

### Experiments

#### Tasks and Dataset

We use the AI2Thor (Kolve et al. 2017) environment as a testbed for our experiments. While there exist prior benchmarks that couple language instructions with embodied environments such as ALFRED (Shridhar et al. 2020), they were not designed to study compositional generalization. We thus construct a new task setup that allows us to flexibly vary tasks and object arguments. We consider the following task types in our experiments,

- **pickup x**: Find and pick up object \( x \)
- **place x in y**: Find and pick up object \( x \), followed by navigating towards \( y \) and placing it.
- **slice x with y**: Secure cutting instrument \( y \), find object \( x \) and perform the slice action on it.

<table>
<thead>
<tr>
<th>Task type</th>
<th>Task descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pick up pot</strong></td>
<td>Go to the stove and pick up the pot.</td>
</tr>
<tr>
<td></td>
<td>Pick up the pot on the bottom right burner on the stove.</td>
</tr>
<tr>
<td><strong>place spoon in pan</strong></td>
<td>get spoon on counter near salt shaker and put it away in pan near stove.</td>
</tr>
<tr>
<td></td>
<td>Pick up the spoon from the table near the salt shaker and move it to the pan on the counter by the sink.</td>
</tr>
<tr>
<td><strong>place knife in plate</strong></td>
<td>Pick up a knife from the left side of the kitchen, near the trashcan, and move it to the plate.</td>
</tr>
<tr>
<td></td>
<td>get knife near sink and put on opposite side of stove on dark plate</td>
</tr>
<tr>
<td><strong>slice bread with knife</strong></td>
<td>Pick the knife and slice the bread.</td>
</tr>
<tr>
<td></td>
<td>Take the knife with the yellow handle from the counter by the sink and use it to cut horizontal slices out of the loaf of bread on the white table.</td>
</tr>
<tr>
<td><strong>slice lettuce with butterknife</strong></td>
<td>Using the silver butter knife you have in your hand, cut up the lettuce into thin slices.</td>
</tr>
<tr>
<td></td>
<td>Pick up the butter knife on the counter and cut the lettuce on the table.</td>
</tr>
</tbody>
</table>

Table 1: Example task types and corresponding task descriptions. Note that the task descriptions are used for training and testing agents. The task types are not known to the agents.
We use Amazon Mechanical Turk to collect natural language descriptions of tasks for training and evaluation. A turker is shown key observation frames during the execution of a particular task and is asked to describe in a sentence how they would describe the task to a robot. Turkers were instructed to do their best to correctly identify task relevant objects. But often descriptions from the turkers incorrectly identify objects such as identifying a potato as an avocado. Such descriptions were manually fixed so that the correct object identities are mentioned in the instructions. We collected 5 natural language descriptions each for 35 task types that include pickup, place and slice tasks. The descriptions consist of 170 unique tokens and have an average length of 12 tokens. Table 1 shows example descriptions collected for some task types.

**Short Horizon Tasks** The pickup tasks are used for evaluating multi-task and zero-shot generalization with seen and unseen descriptions of task types. We use 10 pickup task types \( \text{pickup} \ X \) where \( X \in \{\text{apple}, \text{bread}, \text{tomato}, \text{potato}, \text{lettuce}, \text{spoon}, \text{butter knife}, \text{plate}, \text{pot}\} \). These tasks are used for evaluating generalization to seen and unseen descriptions of known short-horizon task types. They are also used in generalization to longer horizon tasks as described later in this section.

**Longer Horizon Tasks** The place and slice tasks are used for evaluating generalization to longer-horizon tasks. Table 2 shows tasks used for training and evaluation. In addition to multitask generalization, we use these tasks to study zero-shot compositional generalization to unseen task descriptions. The unseen descriptions can correspond to task types that were encountered during training (i.e., seen task types), similar to the pickup tasks. A more challenging generalization scenario is to generalize to text descriptions of task types not encountered during training (i.e., unseen task types).

We consider two types of tasks in the latter scenario. The \( \text{obj-obj} \) setting examines the ability of the agent to generalize to tasks composed of unseen combinations of objects. For instance, in the test task type \( \text{place potato in plate} \), the relevant objects \( \text{potato, plate} \) were encountered during training in tasks such as \( \text{place potato in pot} \) and \( \text{place apple in plate} \).

The \( \text{task-obj} \) setting is a harder generalization problem where the agent is expected to generalize to unseen combinations of tasks and objects. For the test task type \( \text{slice lettuce with knife} \), the object \( \text{lettuce} \) was never observed in the context of a \( \text{slice} \) task during training. However, the agent has access to \( \text{pickup} \) tasks and is expected to learn to interact with lettuce by using the \( \text{pickup lettuce} \) task. This can be challenging because the agent was only taught how to pick up lettuce, and did not learn to associate lettuce with slice tasks.

The seen task types in Table 2 were designed such that each object argument appears in multiple task types. Furthermore, when choosing object arguments for a given task type, we prioritized objects that appear in as many tasks as possible. For instance, in the pickup and place tasks setup, the objects were plate, pan, pot, spoon, etc. where each ob-

---

**Table 2: Task types used for training and testing on place and slice tasks.**

**Seen task types**
- place apple in plate
- place butterknife in plate slice lettuce with knife
- place apple in pan
- place potato in pan
- place potato in plate
- place butterknife in pot
- place potato in pot
- place slice bread with butterknife

**Unseen task types (obj-obj setting)**
- place potato in plate slice potato with knife
- place apple in pan slice tomato with butterknife
- place spoon in pot

**Unseen task types (task-obj setting)**
- place knife in plate slice lettuce with knife
- place knife in pan slice lettuce with butterknife
- place knife in pot

---

**Baselines**

We compare the proposed approach against the following baselines.

**RNN** In this baseline we replace the attentional model with an RNN that produces an embedding of the text instruction. While this model can potentially work for unseen instructions, we examine if the encoding effectively captures the compositional information present in the instructions.

**Gated Attention** We consider the gated attention architecture from Chaplot et al. (2018). This architecture combines the instruction representation with the visual observation using a gated attention operation. The fused representation is fed to an MLP which models the state-action values.

All models and baselines are trained using the DDQN (Double deep Q-learning) algorithm (Van Hasselt, Guez, and Silver 2016).

---

**Hyperparameters**

Word embeddings and the RNN have representation size 32. Objects are represented by embeddings of size 32 from an
Table 3: Task success rates of models under different generalization settings. Models are evaluated on seen/unseen descriptions of seen tasks and on unseen descriptions of unseen tasks. For unseen tasks, we further evaluate under unseen combinations of objects as well as unseen combinations of tasks and objects. Best numbers are boldfaced.

<table>
<thead>
<tr>
<th>Model</th>
<th>Seen tasks</th>
<th></th>
<th>Unseen tasks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Seen descriptions</td>
<td>Unseen descriptions</td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>0.65</td>
<td>0.65</td>
<td>0.26</td>
<td>0.13</td>
</tr>
<tr>
<td>Gated Attention</td>
<td>0.92</td>
<td>0.85</td>
<td>0.66</td>
<td>0.34</td>
</tr>
<tr>
<td>Ours (a) $Q_{nav} + Q_{add}^{obj}$ (no cross modal)</td>
<td>0.89</td>
<td>0.76</td>
<td>0.84</td>
<td>0.77</td>
</tr>
<tr>
<td>Ours (b) $Q_{nav} + Q_{null}^{obj}$ (with cross modal)</td>
<td>0.93</td>
<td>0.85</td>
<td>0.44</td>
<td>0.34</td>
</tr>
<tr>
<td>Ours (c) $Q_{nav} + Q_{add}^{int}$ (with cross modal)</td>
<td><strong>0.95</strong></td>
<td><strong>0.87</strong></td>
<td><strong>0.94</strong></td>
<td><strong>0.91</strong></td>
</tr>
</tbody>
</table>

Figure 3: Visualizing task attention for pickup tasks. Words in darker shades received higher attention probabilities.

The agent is first pre-trained on the pickup tasks as described in the previous section, and then fine-tuned on descriptions of seen tasks in Table 2. Figure 4 compares the learning progress of agents trained from scratch and an agent that has been pre-trained on the pickup tasks. The pre-trained agent learns twice as fast compared to the agent trained from scratch and achieves perfect success rate on training task descriptions.

**Generalization**  Table 3 shows the average task completion success rate of models under different generalization scenarios. We report performance on seen/unseen descriptions of seen task types as well as (unseen) descriptions of unseen task types from Table 2. The RNN and Gated Attention baselines are limited by the fact that the text instruction is represented using the same encoding across all time-steps, which has limited ability to capture compositional information. The inductive bias of Gated Attention enables better performance, but it has difficulty generalizing to unseen tasks. The attention based model outperforms these baselines, which indicates that the attention mechanism helps exploit compositional information in the instruction better than a fixed encoding.

In addition to better performance, the attention model has the advantage of being more interpretable. Figure 5 shows...
Instruction attention

Subgoal 1: bring the potato from the table to the plate on the right of the oven.

Subgoal 2: bring the potato from the table to the plate on the right of the oven.

Instruction attention

Subgoal 1: pick up the butter knife on the counter, and horizontally slice the lettuce.

Subgoal 2: pick up the butter knife on the counter, and horizontally slice the lettuce.

Figure 5: Agent’s observation at different time-steps while performing a place task and a slice task. The attention distribution in the text goal inference component while executing each subgoal is also given below the agent observations.

Ablations

We perform ablations to study the impact of cross-modal reasoning and decomposing the value function in an additive fashion.

Cross-modal Reasoning

We examine model performance without the cross-modal reasoning component. In this case, the MLPs in Equations (5) and (7) only receive the text subgoal and observation encoding as inputs and the visual subgoal $v_{sg}$ is omitted. This is shown as $Q_{\text{nav}} + Q^{\text{add}}_{\text{int}}$ (no cross-modal) in Table 3. From rows (a) and (c) in Table 3 it is clear that the cross-modal reasoning components helps ground text in scene objects and enables better generalization across all settings.

Interaction Q-values

We examine the benefit of decomposing the value function approximation of interaction actions in an additive fashion in $Q^{\text{add}}_{\text{int}}$ (Equation (7)). We compare it against $Q^{\text{full}}_{\text{int}}$ (Equation (6)), which treats each (interaction, object) pair as a separate atomic action. Comparing rows (b), (c) in Table 3 we see that the additive decomposition is crucial for generalization to unseen tasks.

Conclusion

In this work we proposed attention based agents that can exploit the compositional nature of language instructions to generalize to unseen tasks. The policy mimics a hierarchical process where a text embedding obtained via attention represents the subgoal to be executed and the policy network executes the low level actions. The proposed method performs strongly against baselines on a testbed we created based on a photorealistic simulated environment and provides some interpretability.

Compared to existing benchmarks such as ALFRED we made simplifying assumptions such as oracle visual recognition, relatively short horizon tasks and generalization within single kitchen layout which allows us to focus on compositional generalization in embodied settings. However, the ideas presented here can potentially be combined with curriculum learning and learning from human demonstrations to perform complex tasks that require planning over hundreds of time-steps such as in the ALFRED setting, and we leave this to future work.
References


