Opening the pod bay doors

building intelligent agents that can interpret, generate and learn from natural language

Jacob Andreas, MIT / Microsoft

web.mit.edu/jda/www / @jacobandreas
Following natural language instructions
Oh, okay. I'll invite Nicholas Kohn and Michelle Estes to the “Cram session”, and I'll put your meeting in City Center 2605. Does that look good to you?

Cram session
Today: 9:00 – 1:30 PM
 Nicholas Kohn; Michelle Estes;
City Center 2605

Yeah. And push back my one-on-one with Anjali to tomorrow.

https://www.youtube.com/watch?v=G_v5B_gYceM
Following natural language instructions
Instruction following: ingredients

Context
- Environment
- Actions

Data
- Instructions
- Supervision
Context: Environments & Actions

\[ S_0, S_1, S_3, S_4 \]
Figure 3: Compared with low-level visuomotor space, our panoramic action space (Sec. 3.3) allows the agents to have a complete perception of the scene, and to directly perform high-level actions.
We generate up to $K$ candidate routes for each instruction using this procedure, and rescore these routes ending in distinct states have predicted the highest-score routes for each state. States contain the follower's discrete location and heading (direction it is facing) in the visuomotor space represented by an encoding vector $v_i$. At each location, the agent can only move towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction restricted visual signal introduce challenges for instruction following. For example in Figure 3, the agent sees a sofa in the center of its view, and then perform a "go forward" action. This requires strong control to "turn left and go towards the sofa", the agent needs to perform a series of turning actions until it is higher-scoring row of each view angle and rank them according to Eq. 5. The encoding vectors $v_i$ uses low-level visuomotor control (such as turning left or right). The highest-scoring action predicted). The highest-scoring action is selected and expanded using Eq. 1. In addition to enabling pragmatic inference, this state-factored search procedure improves the performance of the follower model on its own (taking the candidate route with highest score under the model than the current best path to the end state). Since route scores are products of conditional probabilities, route scores are non-increasing, and this search procedure generates routes that do not pass through the same state twice—which we found to improve accuracy both for the base follower model and the pragmatic rescoring procedure.
We generate up to $K_{\text{TOP}}$ candidate routes for each instruction using this procedure, and rescore these routes ending in distinct states have predicted the highest-probability route (as scored by the follower model) found so far to be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed. Each view angle is represented by an encoding vector $v_i \in \mathbb{R}^{5 \times 36}$, with built-in mapping from low-level visuomotor control.

3.3 Panoramic Action Space

Since route scores are products of conditional probabilities, route scores are non-increasing, and since instructions typically describe acyclic routes, no remaining unexpanded routes. See Sec. B in the supplementary material for pseudocode.

As in Figure 3, we continue the context: Environments & Actions

- $s_0$: initial state
- $s_1$: state after $a_1$: go_forward
- $s_2$: state after $a_1$: go_forward and $a_2$: turn_right
- $s_3$: final state
- $S_0$: initial state
- $S_1$: state after $a_1$: go_forward
- $S_2$: state after $a_1$: go_forward and $a_2$: turn_right
- $S_3$: final state

Looks like an MDP!
We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized into 36 view angles (12 headings and 3 elevations with 30 degree intervals – in our implementation). The encoding vectors \( u \) and \( s \) and \( v \) of each view angle and action predicted. The highest-scoring route is used to make high-level decisions as to which navigable direction to go to next, with each navigable direction being a candidate route for each instruction using this procedure, and rescore these routes using each possible action from the state, producing routes to the neighboring states. For each of the agents to have a complete perception of the scene, and to directly perform high-level actions.

Since route scores are products of conditional probabilities, route scores are non-increasing, and since instructions typically describe acyclic routes, the performance of the follower model on its own (taking the candidate route with highest score under the model than the current best path to these routes), we allow re-expanding states if a higher-scoring route to that state is found. The highest-scoring route predicts the final state.

At each point in our state-factored search for searching and generating candidates in the follower environment, and whether the route has been completed (had the final state been reached). We continue the search procedure until a route has been completed (had the final state been reached) or there are no more successor states to expand.

**Figure 3:** Compared with low-level visuomotor space, our panoramic action space (Sec. 3.3) allows the agents to have a complete perception of the scene, and to directly perform high-level actions.
As shown in Figure 3, in our panoramic representation, the agent first “looks around” and perceives a 360-degree panoramic view of its surrounding scene from its current location, which is discretized about high-level actions, using a panoramic action space with panoramic representation, converted to “turn left and go towards the sofa”, the agent needs to perform a series of turning actions until it sees a sofa in the center of its view, and then perform a “go forward” action. This requires strong skills of planning and memorization of visual inputs. While a possible way to address this challenge is to learn a hierarchical policy such as in \[ \] uses low-level visuomotor control (such as turning left or right), in our work we directly allow the agent to reason using each possible action from the state, producing routes to the neighboring states. For each of these routes ending in distinct states have predicted the candidate routes for each instruction using this procedure, and rescore using Eq. 1. In addition to enabling pragmatic inference, this state-factored search procedure improves the performance of the follower model on its own (taking the candidate route with highest score under the instruction: … Turn left and go towards the sofa …).

**Instructions**

Go forward, then turn to face right.
As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a 360-degree panoramic view of its surrounding scene from its current location, which is discretized about high-level actions, using a panoramic action space with panoramic representation, converted into 36 view angles (12 headings and 5 elevations with 30 degree intervals – in our implementation). At each location, the agent can only move towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction turn left and go towards the sofa, the agent needs to perform a series of turning actions until it sees a sofa in the center of its view, and then perform a "go forward" action. This requires strong skills of planning and memorization of visual inputs. While a possible way to address this challenge is to "turn left" and go towards this direction!

### 3.3 Panoramic Action Space

The encoding vectors $r_i$ and $s_i$ are stored as the best route to instructions found to improve accuracy both for the base follower model and the pragmatic rescoring procedure, since route scores are products of conditional probabilities, route scores are non-increasing, and the agents to have a complete perception of the scene, and to directly perform high-level actions. Since route scores are non-increasing, and so this search procedure generates routes that do not pass through the same state twice—which we approximate, and so we allow re-expanding states if a higher-scoring route to that state is found.

The encoding vectors $r_i$ is stored as the best route to instructions...

**Go forward, then turn to face right.**

**Find the sofa.**
We generate up to $j$ each view angle a 360-degree panoramic view of its surrounding scene from its current location, which is discretized.

Skill learning in navigation involves learning a hierarchical policy such as in a sequence-to-sequence agent, which is represented by an encoding vector $v_i$ of each view angle and $s_j$ is stored as the best route to the target state $s$.

Figure 3: Compared with low-level visuomotor space, our panoramic action space (Sec. 3.3) allows the agents to have a complete perception of the scene, and to directly perform high-level actions.

For example in Figure 3, we see a sofa in the center of its view, and then perform a "go forward" action. This requires strong skills of planning and memorization of visual inputs. While a possible way to address this challenge is to "turn left and go towards the sofa", the agent needs to perform a series of turning actions until it sees a sofa in the center of its view, and then perform a "go forward" action. This requires strong skills of planning and memorization of visual inputs.

Since route scores are products of conditional probabilities, route scores are non-increasing, and no remaining unexpanded routes. See Sec. B in the supplementary material for pseudocode.
Supervision

Find the sofa.
Find the sofa.
Supervision

Find the sofa.
Go forward, then turn to face right.

Find the sofa.
Instruction following: formally

**Context**

States $S$
Actions $A$
Transitions $T: S \times A \rightarrow S$

**Data**

Instruction $X$
Demo $Y$
Reward $R$

Goal: find a policy $S \times X \rightarrow A$
Move into the living room. Go forward then face the sofa.

```
go_forward turn_left turn_left go_forward turn_right```

As machine translation

Move into the living room. Go forward then face the sofa.

\[\text{go\_forward\ turn\_left\ turn\_left\ go\_forward\ turn\_right}\]
Move into the living room. Go forward then face the sofa.

as machine translation

Each view angle into 36 view angles (12 headings and 3 elevations with 30 degree intervals – in our implementation). We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

Since route scores are products of conditional probabilities, route scores are non-increasing, and so we allow re-expanding states if a higher-scoring route to that state is found. We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a new panoramic view of its surroundings, which includes the location of the sofa and other navigable directions. The agent then uses this panoramic representation to plan its next action, which is to turn left and go towards the sofa.

The sequence-to-sequence agent in each state. States contain the follower's discrete location and heading (direction it is facing) in the visuomotor space.

As machine translation

Each view angle into 36 view angles (12 headings and 3 elevations with 30 degree intervals – in our implementation). We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

Since route scores are products of conditional probabilities, route scores are non-increasing, and so we allow re-expanding states if a higher-scoring route to that state is found. We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a new panoramic view of its surroundings, which includes the location of the sofa and other navigable directions. The agent then uses this panoramic representation to plan its next action, which is to turn left and go towards the sofa.

The sequence-to-sequence agent in each state. States contain the follower's discrete location and heading (direction it is facing) in the visuomotor space.

As machine translation

Each view angle into 36 view angles (12 headings and 3 elevations with 30 degree intervals – in our implementation). We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

Since route scores are products of conditional probabilities, route scores are non-increasing, and so we allow re-expanding states if a higher-scoring route to that state is found. We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a new panoramic view of its surroundings, which includes the location of the sofa and other navigable directions. The agent then uses this panoramic representation to plan its next action, which is to turn left and go towards the sofa.

The sequence-to-sequence agent in each state. States contain the follower's discrete location and heading (direction it is facing) in the visuomotor space.

As machine translation

Each view angle into 36 view angles (12 headings and 3 elevations with 30 degree intervals – in our implementation). We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

Since route scores are products of conditional probabilities, route scores are non-increasing, and so we allow re-expanding states if a higher-scoring route to that state is found. We generate up to 360-degree panoramic view of its surrounding scene from its current location, which is discretized by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control makes high-level decisions as to which navigable direction to go to next, with each navigable direction towards a few navigable directions (provided by the navigation environment) as other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to turn left and go towards the sofa, the agent needs to perform a series of turning actions until it turns right.

As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a new panoramic view of its surroundings, which includes the location of the sofa and other navigable directions. The agent then uses this panoramic representation to plan its next action, which is to turn left and go towards the sofa.

The sequence-to-sequence agent in each state. States contain the follower's discrete location and heading (direction it is facing) in the visuomotor space.
Approach 1: predicting action sequences
From instructions to actions

Go through the door and end facing into the next room.

turn_right
turn_left
go_forward
stop
Go through the door and end facing into the next room.

Start

end

go_forward

stop
From instructions to actions

Key idea: solve this like a normal MDP, with the instruction as part of the state observation.
From instructions to actions

Training

$$\max_{\theta} \ p(action \mid text, state; \theta)$$

$$\max_{\theta} \ E_{state \mid \theta} \ R(action \mid state)$$

Evaluation

$$\max_{action} \ p(action \mid text, state; \theta)$$
Go through the door and end facing into the next room.
Go through the door and end facing into the next room.

Are we there yet?

turn_right
turn_left
go_forward
stop
Go through the door and end facing into the next room.

Are we there yet?

turn_right
turn_left
go_forward
stop
Are we there yet?

Key idea: make the state space track both "reading state" and physical state.
Augmented state spaces

Environment states $S_e$
Environment actions $A_e$

Reading states $S_r$
Reading actions $A_r$
Augmented state spaces

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$

Augmented state spaces

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$
Augmented state spaces

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$

Go forward then face the sofa.

Go forward then face the sofa.

Go forward then face the sofa.

Go forward then face the sofa.
Augmented state spaces

States \( S = S_e \times S_r \)

Actions \( A = A_e \cup A_r \)

Transitions \( T: S \times A \rightarrow S \)

Goal: find a policy \( S \times X \rightarrow A \)
Augmented state spaces: training

Training

\[
\max p(action \mid text, state; \theta)
\]
\[
\max E_{state} \mid \theta R(action \mid state)
\]

Evaluation

\[
\max p(action \mid text, state; \theta)_{action}
\]

[Branavan et al., ACL ’09]
Our goal is to predict a sequence of actions. We consider both immediate reward, which provides feedback after each action, and delayed reward that produces a positive value after the final action only if the task was completed successfully. We will also demonstrate how manually annotated action sequences can be incorporated into the reward.

Training state transitions, are deterministic. For each step, the figure shows the words selected by the action, along with the corresponding system response. During training, we are provided with a set of states and actions visited while in the environment state transitions, and consequently mapping state transitions according to the transition distribution, and a history of states and actions visited while in the environment.

During test, actions are selected to maximize the expected reward, which does not provide feedback until the last action. For example, the expected reward will yield the best actions.

We model the policy distribution as a function of states and actions and a set of features, giving us the flexibility to incorporate a diverse range of features. Under this representation, the possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence). The possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence). The possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence).

The possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence). The possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence). The possible next actions is defined by enumerating sub-spans of the current sentence (i.e., unused words in the current sentence).
clear the two long columns, and then the row
Augmented state spaces: better training

Training

$$\max \ p(action \mid text, state; \theta)$$

$$\max \ \mathbb{E}_{state \mid \theta} R(action \mid state)$$

Evaluation

$$\max_{action} \ p(action \mid text, state; \theta)$$
Learning the reading state

Move into the living room. Go forward then face the sofa.

go_forward	turn_left	turn_left
go_forward	turn_right
Learning the reading state

Move into the living room. Go forward then face the sofa.

go_forward turn_left turn_left face turn_right
Learning the reading state

Key idea: move “reading state” into the hidden state of an RNN.

[Mei et al., AAAI ’16]
Learning the reading state

Training

\[
\max \quad p(action \mid text, state; \theta)
\]

\[
\max \quad E_{state \mid \theta} R(action \mid state)
\]

Evaluation

\[
\max_{action} \quad p(action \mid text, state; \theta)
\]
Approach 2: predicting constraints
Find a table next to a chair.

go_forward go_forward turn_left go_forward turn_left
Actions, goals, constraints

[Find] [a table] [next to] [a chair].

go_forward go_forward turn_left go_forward turn_left
Actions, goals, constraints

[Find] [a table] [next to] [a chair].
Actions, goals, constraints

[Find] [a table] [next to] [a chair].

Find a table next to a chair.

![Diagram of actions, goals, constraints with an image of a room showing a table and a chair.]

Note: The image shows a room with a table and a chair, with arrows indicating the actions of finding a table next to a chair.
Actions, goals, constraints

Key idea: predict constraints rather than action sequences, and let a planner do the rest of the work.
[Find] [a table] [next to] [a chair].
[Find] [a table] [next to] [a chair].

Predicting constraints
Predicting constraints

[Find] [a table] [next to] [a chair].

x6?

x5?

x3

x3

x6

x5

x1

x2
Predicting constraints

[Find] [a table] [next to] [a chair].

x6? adj x5?
Predicting constraints

[Find] [a table] [next to] [a chair].

x6?  adj  x5?

x1, x2, x3, x5, x6
Predicting constraints

[Find] [a table] [next to] [a chair].
Predicting constraints

[Find] [a table] [next to] [a chair].

x1 x2 x3 x4 x5 x6
Predicting constraints

[Find] [a table] [next to] [a chair].

obj?   rel?   obj?

x1     x2     x5

x3     x3     x6

x5
Learning a constraint parser

\[ \max_{\theta} p(labels \mid text, graph; \theta) \]

[Find] [a table] [next to] [a chair].

\[ x_6? \text{ adj } x_5? \]
Inferring constraints

\[
\max \ p(labels \mid text, graph; \theta)
\]

\textit{Find} [a table] [next to] [a chair].

\[
\begin{align*}
\text{x6?} & \quad \text{adj} & \quad \text{x5?} \\
\end{align*}
\]
Inferring constraints

\[
\max \ p(labels \mid text, graph; \theta)
\]

\[\text{[Put]} \ [\text{the cup}] \ [\text{on}] \ [\text{the table}]\].

[Tellex et al., NCAI ’11]
Logical constraint languages

\[
\max_\theta p(\text{constraint} \mid \text{text}; \theta) \quad \max_{\text{constraint}} p(\text{constraint} \mid \text{text}; \theta)
\]

Find a table next to a chair.

\[
\text{at}(x_1) \text{ table}(x_1) \text{ next_to}(x_1, x_2) \text{ chair}(x_2)
\]
Logical constraint languages

\[
\max_{\theta} \ p(\text{constraint} \mid \text{text}; \ \theta) \quad \quad \max_{\text{constraint}} \ p(\text{constraint} \mid \text{text}; \ \theta)
\]

Find a table next to a chair.

\[
\text{at}(\ x_1\ ) \ \text{table}(\ x_1\ ) \ \text{next\_to}(\ x_1,\ x_2\ ) \ \text{chair}(\ x_2\ )
\]
Logical constraint languages

![Diagram of a map environment with logical constraints and examples of logical forms.]

- (a) chair \( \lambda x. \text{chair}(x) \)
- (b) hall \( \lambda x. \text{hall}(x) \)
- (c) the chair \( \hat{v}.x. \text{chair}(x) \)
- (d) you \( \text{you} \)
- (e) blue hall \( \lambda x. \text{hall}(x) \land \text{blue}(x) \)
- (f) chair in the intersection \( \lambda x. \text{chair}(x) \land \text{intersect}(i.y. \text{junction}(y), x) \)
- (g) in front of you \( \lambda x. \text{in\_front\_of}(you, x) \)

[Artzi et al., TACL '13]
Constraints without logic

Find a table next to a chair.

- go_forward
- turn_left
- turn_left
- go_forward
- turn_right
Key idea: use freeform learned potential functions rather than symbolic constraints
As shown in Figure 3, in our panoramic representation, the agent first "looks around" and perceives a sofa in the center of its view, and then perform a "go forward" action. This requires strong and restricted visual signal introduce challenges for instruction following. For example in Figure 3, by 30 degrees, and only perceives frontal visual sensory input. Such fine-grained visuomotor control

We generate up to \( K \) candidate routes for each instruction using this procedure, and rescore these routes with constraints without logic. We continue the search for the best route, which has not yet been expanded (had successors produced), is selected and expanded. Constraints without logic

Since route scores are products of conditional probabilities, route scores are non-increasing, and no remaining unexpanded routes. See Sec. B in the supplementary material for pseudocode.

Constraints without logic
Find a table next to a chair.

go_forward turn_left turn_left go_forward turn_right
**3.3 Panoramic Action Space**

At each point in our state-factored search for searching and generating candidates in the follower model, we store the highest-probability route (as scored by the follower model) found so far to approximate, and so we allow re-expanding states if a higher-scoring route to that state is found. Since route scores are products of conditional probabilities, route scores are non-increasing, and so this search procedure generates routes that do not pass through the same state twice—which we approximate, and so we allow re-expanding states if a higher-scoring route to that state is found.

Since instructions typically describe acyclic routes, we generate up to 5 candidate routes for each instruction using this procedure, and rescore these routes ending in distinct states have predicted the highest-scoring route, which has not yet been expanded (had successors produced), is selected and expanded into 36 view angles (12 headings of each view angle and 5 elevations with 30 degree intervals – in our implementation).

The sequence-to-sequence agent in [13] uses low-level visuomotor control (such as turning left or right action, or there are other directions can be physically obstructed (e.g. blocked by a table). Here, in our action space the agent only needs to make high-level decisions as to which navigable direction to go to next, with each navigable direction

**Constraints without logic**

Find a table next to a chair.
Clear the columns,
then the row
Clear the columns, then the row

(no "column"!)

Constraints without logic
Figure 9: (a-c) Visualizations of tasks from the ISI Language Grounding dataset (Bisk et al., 2016) and our model’s value map predictions. The agentive block and goal location are outlined in red for visibility. (d) The MSE of the value map prediction as a function of a subgoal’s ordering in an overall task. The model performs better on subgoals later in a task despite the subgoals being treated completely independently during both training and testing.

A controllable agent, whereas the ISI set allows multiple blocks to be moved. We therefore modify the ISI setup using an oracle to determine which block is given agency during each step. This allows us to retain the linguistic variability of the dataset while overcoming the mismatch in task setup. The states are discretized to a 13 × 13 map and the instructions are lemmatized.

Performance on the modified ISI dataset is reported in Table 4 and representative visualizations are shown in Figure 9. Our model outperforms both baselines by a greater margin in policy quality than on our own dataset.

Misra et al. (2017) also use this dataset and report results in part by determining the minimum distance between an agent and a goal during an evaluation lasting $N$ steps. This evaluation metric is therefore dependent on this timeout parameter $N$. Because we discretized the state space so as to be able to represent it as a grid of embeddings, the notion of a single step has been changed and direct comparison limited to $N$ steps is ill-defined. Hence, due to modifications in the task setup, we cannot compare directly to the results in Misra et al. (2017).

Understanding grounding evaluation

An interesting finding in our analysis was that the difficulty of the language interpretation task is a function of the stage in task execution (Figure 9(d)). In the ISI Language Grounding set (Bisk et al., 2016), each individual instruction (describing where to move a particular block) is a subgoal in a larger task (such as constructing a circle with all of the blocks). The value maps predicted for subgoals occurring later in a task are more accurate than those occurring early in the task. It is likely that the language plays a less crucial role in specifying the subgoal position in the task.

When a model is available and the states are not overwhelmingly high-dimensional, policy quality is a useful metric that is independent of this type of parameter. As such, it is our default metric here. However, estimating policy quality for environments substantially larger than those investigated here is a challenge in itself.

[Janner et al., TACL ’18]
Our toolkit so far
Instruction following

Act in complex environments
  With expressive policies that condition on instructions and observations

Track progress over time
  In the underlying state space or RNN state

Plan ahead and reason about outcomes
  With a symbolic planner or learned cost function
What else can we do?
Application: instruction generation
Instruction following

Move into the living room. Go forward then face the sofa.

go_forward turn_left turn_left go_forward turn_right
Instruction following generation

*Move into the living room. Go forward then face the sofa.*

go_forward turn_left turn_left go_forward turn_right
Prediction action sequences

find a sofa

go_forward turn_left turn_left go_forward turn_right
Instruction generation

Key idea: a good instruction gets readers to their goal with high probability (whatever the training data says!)
Instruction generation

Max posterior probability

\[
\max_{text} \ p(text \mid plan; \theta)
\]

(“how do people describe this?”)
Instruction generation

Max posterior probability

\[
\max_{\text{text}} p(\text{text} \mid \text{plan}; \theta)
\]

(“how do people describe this?”)

Min Bayes risk

\[
\max_{\text{text}} p(\text{plan} \mid \text{text}; \theta)
\]

(“how do I make people do this?”)
Reasoning about outcomes

$$\max_{\text{text}} p(\text{plan} \mid \text{text}; \theta)$$

I will make a turn.
Reasoning about outcomes

\[
\max_{\text{text}} \ p(\text{plan} \mid \text{text}; \theta)
\]

I will make a turn.

Listener
Reasoning about outcomes

$$\max_{\text{text}} p(\text{plan} \mid \text{text}; \theta)$$

I will go straight through.

Listener
Reasoning about outcomes

\[
\max_{text} \ p(plan \mid text; \theta)
\]

I will turn left at the brick intersection.

Listener

[Fried et al., NAACL ’18]
Reasoning about belief

I will turn left at the brick intersection.

[Frank & Goodman, Trends in Cog. Sci. ’12]
**speaker**: Walk past the dining room table and chairs and wait there.

**listener**: Walk past the dining room table and chairs and take a right into the living room. Stop once you are on the rug.

**human**: Turn right and walk through the kitchen. Go right into the living room and stop by the rug.
Application: machine teaching
Instructions as scaffolds for RL
Instructions as parameter-tying schemes
Instructions as parameter tying schemes

Environment states $S_e$
Environment actions $A_e$

Reading states $S_e$
Reading actions $A_e$

Figure 3: Compared with low-level visuomotor space, our panoramic action space (Sec. 3.3) allows agents to have a complete perception of the scene, and to directly perform high-level actions.
Instructions as parameter-tying schemes

- Go north, go north, go east, go south
- Go north, go north, go west
- Go north, go east, go north, ...
Go north.

Go east.

Go north.

[Andreas et al., ICML '17]
Learning interactively from corrections
Conditioning on the past

Push the chair against the wall.

go_forward grasp turn_left go_forward release
Conditioning on the past

Push the chair against the wall.

No, the red chair.

turn_left grasp go_forward go_forward go_forward release
Push the chair against the wall.

No, the red chair.

Now a little to the left.
Conditioning on the past

Key idea: learn to solve problems interactively by conditioning on the whole history of instructions.

[Co-Reyes et al., ICLR ’19]
Touch cyan block.

Move closer to magenta block.

Move a lot up.

Move a little up.
Learning with latent language
Language learning as pertaining
Structured exploration
Structured exploration

reach the heart

$R = -1$
Structured exploration

reach the heart

east of the gold star

go to the east of the heart

R = -1

R = 0

R = 3
Structured exploration

Language learning

go east of the heart

Reinforcement learning

[Andreas et al., NAACL ’19]
Structured few-shot learning

examples

<table>
<thead>
<tr>
<th>emboldens</th>
<th>emboldecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>kisses</td>
<td>kisses</td>
</tr>
<tr>
<td>loneliness</td>
<td>locelicens</td>
</tr>
<tr>
<td>vein</td>
<td>veic</td>
</tr>
<tr>
<td>dogtrot</td>
<td>dogtrot</td>
</tr>
</tbody>
</table>

change any n to a c

pred. description

Figure 6: Example predictions for string editing.

Policy Search

The previous two sections examined supervised settings where the learning signal comes from few examples but is readily accessible. In this section, we move to a set of reinforcement learning problems, where the learning signal is instead sparse and time-consuming to obtain. We evaluate on a collection of 2-D treasure hunting tasks. These tasks require the agent to discover a rule that determines the location of buried treasure in a large collection of environments of the kind shown in Figure 7. To recover the treasure, the agent must navigate (while avoiding water) to its goal location, then perform a DIG action. At this point the episode ends; if the treasure is located in the agent's current position, it receives a reward, otherwise it does not. In every task, the treasure has consistently been buried at a fixed position relative to some landmark (like the heart in Figure 7). Both the offset and the identity of the target landmark are unknown to the agent, and the location landmark itself varies across maps. Indeed, there is nothing about the agent's observations or action space to suggest that landmarks and offsets are even the relevant axis of variation across tasks, but this structure is made clear in the natural language annotations. The high-level structure of these tasks is similar to one used by Hermer-Vazquez et al. (2001) to study concept learning in humans.

The interaction between language and learning in these tasks is rather different than in the supervised settings. In the supervised case, language served mostly as a guard against overfitting, and could...
### Structured few-shot learning

<table>
<thead>
<tr>
<th>examples</th>
<th>true description</th>
<th>true output</th>
</tr>
</thead>
<tbody>
<tr>
<td>emboldens</td>
<td>replace all (n) s with (c)</td>
<td>loocies</td>
</tr>
<tr>
<td>kisses</td>
<td>kisses</td>
<td>loonies</td>
</tr>
<tr>
<td>loneliness</td>
<td>locelicious</td>
<td></td>
</tr>
<tr>
<td>vein</td>
<td>veic</td>
<td>loocies</td>
</tr>
<tr>
<td>dogtrot</td>
<td>dogtrot</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6:** Example predictions for string editing.

A few interesting facts stand out. Under the ordinary evaluation condition (with no ground-truth annotations provided), language-learning with natural language data is actually better than language-learning with regular expressions. This might be because the extra diversity helps the model figure out the relevant axes of variation and avoid overfitting to individual strings. Allowing the model to do its own inference is also better than providing ground-truth natural language descriptions, suggesting that it is actually better at generalizing from the relevant concepts than our human annotators (who occasionally write things like *I have no idea* for the inferred rule).

Unsurprisingly, with ground truth REs (which unlike the human data are always correct) we can do better than any of the models that have to do inference. Coupling our inference procedure with an oracle RE evaluator, we essentially recover the synthesis-based approach of Devlin et al. (2017). Our findings are consistent with theirs: when a complete and accurate execution engine is available, there is no reason not to use it. But we can get almost 90% of the way there with an execution model learned from scratch.

Some examples of model behavior are shown in Figure 6; more may be found in Appendix D.

### 6 Policy Search

The previous two sections examined supervised settings where the learning signal comes from few examples but is readily accessible. In this section, we move to a set of reinforcement learning problems, where the learning signal is instead sparse and time-consuming to obtain. We evaluate on a collection of 2-D treasure hunting tasks. These tasks require the agent to discover a rule that determines the location of buried treasure in a large collection of environments of the kind shown in Figure 7. To recover the treasure, the agent must navigate (while avoiding water) to its goal location, then perform a DIG action. At this point the episode ends; if the treasure is located in the agent's current position, it receives a reward, otherwise it does not. In every task, the treasure has consistently been buried at a fixed position relative to some landmark (like the heart in Figure 7). Both the offset and the identity of the target landmark are unknown to the agent, and the location landmark itself varies across maps. Indeed, there is nothing about the agent's observations or action space to suggest that landmarks and offsets are even the relevant axis of variation across tasks, but this structure is made clear in the natural language annotations. The high-level structure of these tasks is similar to one used by Hermer-Vazquez et al. (2001) to study concept learning in humans.

The interaction between language and learning in these tasks is rather different than in the supervised settings. In the supervised case, language served mostly as a guard against overfitting, and could
Current order to execute on:

Send 2 peasants to mine upper ore
Future challenges
Fake data

Touch cyan block.

Move closer to magenta block.

Move a lot up.
Fake data

"Instructions" are synthesized from a grammar because of sample inefficiency!

Touch cyan block. → Move closer to magenta block. → Move a lot up.
Fake data

Train on synthetic language and test on synthetic.

Test on real.

[Blukis et al., CoRL '18]
Fake data

Start by eliciting real user utterances

Use synthetic data to augment, not replace, natural language

Sim-to-real transfer?
Neural planning

Figure 9: (a-c) Visualizations of tasks from the ISI Language Grounding dataset (Bisk et al., 2016) and our model's value map predictions. The agentive block and goal location are outlined in red for visibility. (d) The MSE of the value map prediction as a function of a subgoal's ordering in an overall task. The model performs better on subgoals later in a task despite the subgoals being treated completely independently during both training and testing.

Performance on the modified ISI dataset is reported in Table 4 and representative visualizations are shown in Figure 9. Our model outperforms both baselines by a greater margin in policy quality than on our own dataset.

Misra et al. (2017) also use this dataset and report results in part by determining the minimum distance between an agent and a goal during an evaluation lasting $N$ steps. This evaluation metric is therefore dependent on this timeout parameter $N$. Because we discretized the state space so as to be able to represent it as a grid of embeddings, the notion of a single step has been changed and direct comparison limited to $N$ steps is ill-defined. Hence, due to modifications in the task setup, we cannot compare directly to the results in Misra et al. (2017).

Understanding grounding evaluation

An interesting finding in our analysis was that the difficulty of the language interpretation task is a function of the stage in task execution (Figure 9(d)). In the ISI Language Grounding set (Bisk et al., 2016), each individual instruction (describing where to move a particular block) is a subgoal in a larger task (such as constructing a circle with all of the blocks). The value maps predicted for subgoals occurring later in a task are more accurate than those occurring early in the task. It is likely that the language plays a less crucial role in specifying the subgoal position in

6 When a model is available and the states are not overwhelmingly high-dimensional, policy quality is a useful metric that is independent of this type of parameter. As such, it is our default metric here. However, estimating policy quality for environments substantially larger than those investigated here is a challenge in itself.

Clear the columns, then the row

Take block 13 and place it directly above block 14 so they are almost touching.
Natural language subgoals

Solve the puzzle.

Clear out the right half of the puzzle.

Remove all the long columns.

Clear the remaining blocks.

Clear a row.

Clear a column.
Conclusions

Instruction following $\Leftrightarrow$ policy learning
But need to think carefully about state tracking, planning, compositionality

Instruction following $\Rightarrow$ other tasks
Language generation, machine teaching, structured exploration

Challenges
Better data efficiency, smarter inference
References

Branavan et al. Reinforcement learning for mapping instructions to actions. ACL 2009.


Tellex et al. Understanding natural language commands for robotic navigation and mobile manipulation. NCAI 2011.

Andreas & Klein. Alignment-based compositional semantics for instruction following. EMNLP 2016.

Andreas et al. Modular multitask reinforcement learning with policy sketches. ICML 2017.

Fried et al. Unified pragmatic models for generating and following instructions. NAACL 2018.


Andreas et al. Learning with latent language. NAACL 2018.