A Framework for Learning to Request Rich and Contextually Useful Information from Humans

Khanh Nguyen [1] Yonatan Bisk [2] Hal Daumé III [1 3]

Abstract

When deployed, AI agents will encounter problems that are beyond their autonomous problem-solving capabilities. Leveraging human assistance can help agents overcome their inherent limitations and robustly cope with unfamiliar situations. We present a general interactive framework that enables an agent to request and interpret rich, contextually useful information from an assistant that has knowledge about the task and the environment. We demonstrate the practicality of our framework on a simulated human-assisted navigation problem. Aided with an assistance-requesting policy learned by our method, a navigation agent achieves up to a 7× improvement in success rate on tasks that take place in previously unseen environments, compared to fully autonomous behavior. We show that the agent can take advantage of different types of information depending on the context, and analyze the benefits and challenges of learning the assistance-requesting policy when the assistant can recursively decompose tasks into subtasks.

1. Introduction

Machine learning has largely focused on creating agents that can solve problems on their own. Despite much progress, these autonomous agents struggle to operate in unfamiliar environments and fulfill new requirements from users (Goodfellow et al., 2015; Jia & Liang, 2017; Eykholt et al., 2018; Qi et al., 2020; Shridhar et al., 2020). As autonomous agents are built to perform tasks by themselves, they are typically endowed with limited capabilities of communicating with humans during task execution. This design significantly constrains the performance of these agents, as human bystanders or supervisors can provide useful information that improves the agents' decisions. Uncommunicative autonomous agents are also highly unsafe. They do not consult humans on difficult decisions and can potentially commit catastrophic mistakes without warnings.

The reliability of autonomous agents can be dramatically enhanced if they are equipped with communication skills for leveraging knowledge of humans in the same environment (Rosenthal et al., 2010; Tellex et al., 2014; Nguyen et al., 2019; Nguyen & Daumé III, 2019; Thomason et al., 2020). In many cases, a task that is initially out of reach of an agent can be elaborated or re-formulated into a task that the agent is familiar with. Through effective communication, the agent can help a human revise the task specification into a form that it can easily interpret and accomplish. As a motivating example, consider a home-assistant robot assigned a task of finding a newly bought *rice cooker* in a house, which it has never heard of and therefore does not know how to proceed. The robot, however, knows how to navigate to familiar landmarks in the house. In this case, the robot can ask a human it encounters a question like "*where is the rice cooker?*", expecting an instruction like "*find it in the kitchen, near the sink*", which it may know how to execute. By communicating with the human, the robot can convert the initially impossible task into a more feasible one. On the other hand, by giving the right information to the agent, a human can lift its task performance without needing to collect extra data and acquire expertise to upgrade its model.

We present Hari: **H**uman-**A**ssisted **R**einforced **I**nteraction, a general, POMDP-based framework that allows agents to effectively request and interpret information from humans. We identify and provide solutions to two fundamental problems: the *speaker* problem and the *listener* problem. The speaker problem concerns teaching an agent to elicit information from humans that can improve its decision making. Inspired the intention-based theory of human communication (Sperber & Wilson, 1986; Tomasello et al., 2005; Scott-Phillips, 2014), we equip a learning agent with a set of general information-seeking intentions that are helpful in solving any POMDP problem. At every time step, the agent can express to a human an intention of requesting additional information about (i) its current state, (ii) the goal state

[1]University of Maryland, College Park [2]Carnegie Mellon University [3]Microsoft Research. Correspondence to: Khanh Nguyen <kxnguyen@umd.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

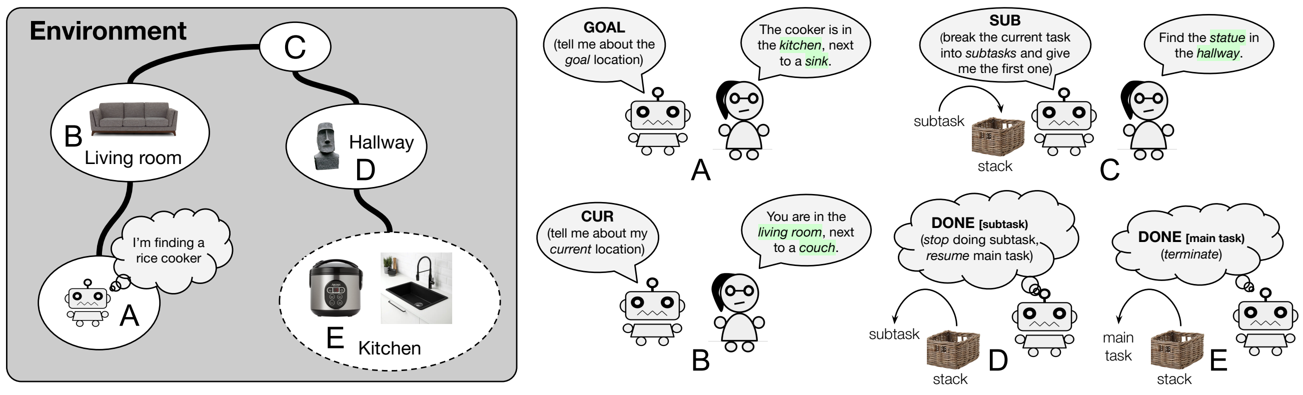

Environment
C
B
Living room
Hallway
D
I'm finding a rice cooker
A
E
Kitchen
GOAL (tell me about the goal location)
The cooker is in the kitchen, next to a sink.
A
SUB (break the current task into subtasks and give me the first one)
subtask
stack
Find the statue in the hallway.
C
CUR (tell me about my current location)
You are in the living room, next to a couch.
B
DONE [subtask] (stop doing subtask, resume main task)
subtask
stack
D
DONE [main task] (terminate)
main task
stack
E

Figure 1: An illustration of the HARI framework in a navigation task. The agent can only observe part of an environment and is asked to find a rice cooker. An assistant communicates with the agent and can provide information about the environment and the task. Initially (A) the agent may request information about the goal, but may not know where it currently is. For example, at location B, due to limited perception, it does not recognize that it is next to a couch in a living room. It can obtain such information from the assistant. If the current task becomes too difficult (like at location C), the agent can ask the assistant for a simpler subtask that helps it make progress toward the goal. When the agent receives a subtask, it pushes the subtask to the top of a goal stack, and when it decides to stop executing a subtask, it pops the subtask from the stack (e.g., at location D). At location E, the agent empties the stack and terminates its execution.

which it needs to reach, or (iii) a new subgoal state which, if reached, helps it make progress on the main goal. The agent learns to decide what information to request through reinforcement learning, by learning how much each type of information enhances its performance in a given situation. The agent's request decisions are thus grounded in its own perception of its (un)certainties about the current situation and are optimized to be maximally contextually useful to it.

Our approach contrasts with methods to train agents to ask questions by mimicking those asked by humans (Labutov et al., 2015; Mostafazadeh et al., 2016; De Vries et al., 2017; Rao & Daumé III, 2018; Thomason et al., 2020; Liu et al., 2020). While effective in some situations, this approach has a potential significant drawback: questions asked by humans may not always be contextually useful for agents. For example, humans are experts in recognizing objects, thus rarely ask questions like "*what objects are next to me*?" Meanwhile, such questions may be helpful for robot localization, as robots may have imperfect visual perception in unfamiliar environments. Rather than focusing on naturalness, we aim to endow an agent with the cognitive capability of determining which type of information would be contextually useful to itself. Such capability requires an understanding of the casual effects of the agent's decisions on its future performance, which is exactly the knowledge acquired through reinforcement learning.

In addition to being able to *ask* for useful information, the agent must be able to incorporate the information it receives into its decision-making process: the *listener* problem. Humans can offer various types of assistance and may express them using diverse media (e.g., language, gesture, image). HARI implements a flexible communication protocol that is defined by the agent's task-solving policy: information coming from the human is given as state descriptions that the policy can take as input and map into actions. Hence, the space of information that the human can convey is as rich as the input space of the policy. Our design takes advantage of the flexibility of constructing an agent policy: (i) the policy can be further trained to interpret new information from humans, and (ii) it can leverage modern neural network architectures to be able to encode diverse forms of information. This contrasts with existing approaches based on reinforcement learning (RL) or imitation learning (IL) (Knox & Stone, 2009; Judah et al., 2010; Torrey & Taylor, 2013; Judah et al., 2014), in which feedback to the agent is limited to scalar feedback of rewards (in RL) or actions (in IL).

To evaluate our framework, we simulate a human-assisted navigation problem where an agent can request information about the environment from a human. Our agent learns an *intention policy* to decide at each step whether it wants to request additional information, and, if so, which type of information it wants to obtain. On tasks in previously unseen environments, the ability to ask for help improves the agent's success rate by 7× compared to performing tasks on its own. This human-assisted agent even outperforms an agent that has perfect perception and goal descriptions in unseen environments, thanks to the ability to simplify difficult tasks via requesting subgoals. We release code and data for the experiments at `https://github.com/khanhptnk/hari`.

2. Preliminaries

We consider an environment defined by a partially observed Markov decision process (POMDP) $E = (\mathcal{S}, \mathcal{A}, T, c, \mathcal{D}, \rho)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, cost function $c : (\mathcal{S} \times \mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, description space $\mathcal{D}$, and description function $\rho : \mathcal{S} \to \Delta(\mathcal{D})$, where $\Delta(\mathcal{Y})$ denotes the set of probability distributions over $\mathcal{Y}$.[1] We refer to this as the *execution environment*, where the agent performs tasks—e.g., a navigation environment.

A (goal-reaching) *task* is a tuple (s_1, g_1, d^g_1) where s_1 is the start state, g_1 is the goal state, and d^g_1 is a limited description of g_1.[2] Initially, a task (s_1, g_1, d^g_1) is sampled from a distribution $\mathfrak{T}$. An agent starts in s_1 and is given the goal description d^g_1. It must reach the goal state g_1 within H steps. Let g_t and d^g_t be the goal state and goal description executed at time step t, respectively. In a standard POMDP, $g_t = g_1$ and $d^g_t = d^g_1$ for $1 \le t \le H$. Later, we will enable the agent to set new goals via communication with humans.

At time t, the agent does not know its state s_t but only receives a state description $d^s_t \sim \rho(s_t)$. Given d^s_t and d^g_t, the agent makes a decision $a_t \in \mathcal{A}$, transitions to the next state $s_{t+1} \sim T(s_t, a_t)$, and incurs an *operation cost* $c_t \triangleq c(s_t, g_t, a_t)$. When the agent takes the a_{done} action to terminate its execution, it receives a *task error* $c(s_t, g_t, a_{\text{done}})$ that indicates how far it is from actually completing the current task. The agent's goal is to reach g_1 with minimum total cost $C(\tau) = \sum_{t=1}^{H} c_t$, where $\tau = (s_1, d^s_1, a_1, \ldots, s_H, d^s_H)$ is an execution of the task.

We denote by b^s_t a belief state—a representation of the history $(d^s_1, a_1, \ldots, d^s_t)$—and by $\mathcal{B}$ the set of all belief states. The agent maintains an *execution policy* $\hat{\pi} : \mathcal{B} \times \mathcal{D} \to \Delta(\mathcal{A})$. The learning objective is to estimate an execution policy that minimizes the expected total cost of performing tasks:

$$\min_{\pi} \mathbb{E}_{(s_1, g_1, d^g_1) \sim \mathfrak{T}, \tau \sim P_\pi(\cdot \mid s_1, d^g_1)} \left[C(\tau) \right] \tag{1}$$

where $P_\pi(\cdot \mid s_1, d^g_1)$ is the distribution over executions generated by a policy π given start state s_1 and goal description d^g_1. In a standard POMDP, an agent performs tasks on its own, without asking for any external assistance.

3. Leveraging Assistance via Communication

For an agent to accomplish tasks beyond its autonomous capabilities, we introduce an *assistant*, who can provide rich information about the environment and the task. We then describe how the agent requests and incorporates information from the assistant. Figure 1 illustrates an example communication between the agent and the assistant on an example object-finding task. Our formulation is inspired by the intention-based theory of human communication (Sperber & Wilson, 1986; Tomasello et al., 2005; Scott-Phillips, 2014), which characterizes communication as the expression and recognition of intentions in context. In this section, we draw connections between the theory and elements of POMDP learning, which allows us to derive reinforcement learning algorithms to teach the agent to make intentional requests.

[1] We say "description" in lieu of "observation" to emphasize (i) the description can come in varied modalities (e.g., image or text), and (ii) it can be obtained via perception as well as communication.

[2] Our notion of "goal state" refers not only to physical goals (e.g. a location in a house), but also to abstract states (e.g. a level of house cleanliness).

Assistant. We assume an ever-present assistant who knows the agent's current state s_t, the goal state g_t, and the optimal policy $\pi^\star$. Their first capability is to provide a description of a state, defined by a function $\rho_A : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{D})$ where $\rho_A(d' \mid s, d)$ specifies the probability of giving d' to describe state s given a current description d, which can be empty. The second capability is to propose a subgoal of a current goal, specified by a function $\omega_A : \mathcal{S} \times \mathcal{S} \to \Delta(\mathcal{S})$, where $\omega_A(g' \mid s, g)$ indicates the probability of proposing g' as a subgoal given a current state s and a goal state g.

Common ground. Common ground represents mutual knowledge between the interlocutors and is a prerequisite for communication (Clark & Brennan, 1991; Stalnaker, 2002). In our context, knowledge of the agent is contained in its execution policy $\hat{\pi}$, while knowledge of the assistant is given by $\pi^\star$. When $\hat{\pi}$ and $\pi^\star$ maps to the same action distribution given an input (b^s, d^g), we say that the input belongs to the common ground of the agent and the assistant. Substantial common ground is needed for effective communication to take place. Thus, we assume that execution policy has been pre-trained to a certain level of performance so that there exists a non-empty subset of inputs on which the outputs of $\hat{\pi}$ and $\pi^\star$ closely match.

3.1. The *Listener* Problem: Incorporating Rich Information Provided by the Assistant

Upon receiving a request from the agent, the assistant replies with a state description $d \in \mathcal{D}$. The generality of our notion of state description (see § 2) means that the assistant can provide information in any medium and format, so long as it is compatible with the input interface of $\hat{\pi}$. This reflects that only information interpretable by the agent is useful.

Concretely, the assistant can provide a new current-state description d^s_{t+1}, which the agent appends to its history to compute a new belief state b^s_{t+1} (e.g., using a recurrent neural network). The assistant can also provide a new goal description d^g_{t+1}, in which case the agent simply replaces the current goal description d^g_t with the new one. This formulation allows the human-agent communication protocol to be augmented in two ways: (i) enlarging the common ground by training $\hat{\pi}$ to interpret new state descriptions or (ii) designing the agent model architecture to support new types of input information. In comparison, frameworks that implement a reinforcement learning- or imitation learning-based communication protocol (e.g., Knox & Stone, 2009; Torrey & Taylor, 2013) allow humans to only give advice using low-bandwidth media like rewards or primitive actions.

3.2. The *Speaker* Problem: Requesting Contextually Useful Information

Asking Questions as a Cognitive Capability. Asking questions is a cognitive process motivated by a person's self-recognition of knowledge deficits or common ground mismatches (Graesser et al., 1992). The intention-based theory of human communication suggest that a question, like other communicative acts, should convey an information-seeking intention that is grounded in the speaking context. In the case of question generation, the speaker's decision-making policy should be included in the context because, ultimately, a speaker asks a question to make better decisions.

Approaches that teach an agent to mirror questions asked by humans (e.g., De Vries et al., 2017), do not consider the agent's policy as part of the context for generating the question. The questions selected by humans may not be useful for the learning agent. To address this issue, we endow the agent with the cognitive capability of anticipating how various types of information would affect its future performance. The agent learns this capability using reinforcement learning, via *interacting* with the assistant and the environment, rather than imitating human behaviors.

Information-Seeking Intentions. While humans possess capabilities that help them come up with questions, it is unclear how to model those capabilities. Some approaches (e.g., Mostafazadeh et al., 2016; Rao & Daumé III, 2018) rely on pre-composed questions, which are domain-specific. We instead endow the agent with a set of intentions that correspond to speech acts in embodied dialogue (Thomason et al., 2020). They are relevant to solving general POMDPs, while being agnostic to the problem domain and the implementation of the execution policy:

(a) CUR: requests a new description of the current state s_t and receives $d^s_{t+1} \sim \rho_A\left(\cdot \mid s_t, d^s_t\right)$;

(b) GOAL: requests a new description of the current goal g_t and receives $d^g_{t+1} \sim \rho_A\left(\cdot \mid g_t, d^g_t\right)$;

(c) SUB: requests a description of a subgoal and receives $d^g_{t+1} \sim \rho_A\left(\cdot \mid g_{t+1}, \emptyset\right)$ where the subgoal $g_{t+1} \sim \omega_A\left(\cdot \mid s_t, g_t\right)$ and $\emptyset$ is an empty description.

We leave constructing more specific intentions (e.g., asking about a specific input feature) to future work.

Intention Selection. To decide which intention to invoke, the agent learns an intention policy ψ_θ, which is itself a function of the execution policy $\hat{\pi}$. The intention policy's action space consists of five intentions: $\bar{\mathcal{A}} = \{\text{CUR}, \text{GOAL}, \text{SUB}, \text{DO}, \text{DONE}\}$. The first three actions convey the three intentions defined previously. The remaining two actions are used for traversing in the environment:

(d) DO: executes the most-probable action $a^{\text{do}}_t \triangleq \arg\max_{a \in \mathcal{A}} \hat{\pi}\left(a \mid b^s_t, d^g_t\right)$. The agent transitions to a new state $s_{t+1} \sim T(s_t, a^{\text{do}}_t)$;

(e) DONE: decides that the current goal g_t has been reached. If g_t is the main goal ($g_t = g_1$), the episode ends. If g_t is a subgoal ($g_t \neq g_1$), the agent may choose a new goal to follow.

In our implementation, we feed the hidden features and the output distribution of $\hat{\pi}$ as inputs to ψ_θ so the agent can use its execution policy in choosing its intentions. Later we will specify the input space of the interaction policy and how the next subgoal is chosen (upon DONE), as these details depend on how the agent implements its goal memory. The next section introduces an instantiation where the agent uses a stack data structure to manage the goals it receives.

4. Learning When and What to Ask

In this section, we formalize the HARI framework for learning to select information-seeking intentions. We construct the POMDP environment that the intention policy acts in, referred to as the *intention environment* (§4.1) to distinguish with the environment that the execution policy acts in. Our construction employs a *goal stack* to manage multiple levels of (sub)goals (§4.2). A goal stack stores all the (sub)goals the agent has been assigned but has not yet decided to terminate. It is updated in every step depending on the selected intention. In §4.4, we design a cost function that trades off between taking few actions and completing tasks.

4.1. Intention Environment

Given an execution environment $E = (\mathcal{S}, \mathcal{A}, T, c, \mathcal{D}, \rho)$, the intention environment is a POMDP $\bar{E} = (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{\mathcal{T}}, \bar{c}, \bar{\mathcal{D}}, \bar{\rho})$:

- **State space** $\bar{\mathcal{S}} = \mathcal{S} \times \mathcal{D} \times \mathcal{G}_L$, where $\mathcal{G}_L$ is the set of all goal stacks of at most L elements. Each state $\bar{s} = (s, d^s, G) \in \bar{\mathcal{S}}$ is a tuple of an execution state s, description d^s, and goal stack G. Each element in the goal stack G is a tuple of a goal state g and description d^g;
- **Action space** $\bar{\mathcal{A}} = \{\text{CUR}, \text{GOAL}, \text{SUB}, \text{DO}, \text{DONE}\}$;
- **State-transition** function $\bar{T} = T_s \cdot T_G$ where $T_s : \mathcal{S} \times \mathcal{D} \times \bar{\mathcal{A}} \to \Delta(\mathcal{S} \times \mathcal{D})$ and $T_G : \mathcal{G}_L \times \bar{\mathcal{A}} \to \Delta(\mathcal{G}_L)$;
- **Cost function** $\bar{c} : (\mathcal{S} \times \mathcal{G}_L) \times \bar{\mathcal{A}} \to \mathbb{R}$, defined in §4.4 to trade off operation cost and task error;

- **Description space** $\bar{\mathcal{D}} = \mathcal{D} \times \mathcal{G}^d_L$ where $\mathcal{G}^d_L$ is the set of all goal-description stacks of size L. The agent cannot access the environment's goal stack G, which contains true goal states, but only observe descriptions in G. We call this partial stack a *goal-description stack*, denoted by G^d;
- **Description function** $\bar{\rho} : \bar{\mathcal{S}} \to \bar{\mathcal{D}}$, where $\bar{\rho}(\bar{s}) = \bar{\rho}(s, d^s, G) = (d^s, G^d)$. Unlike in the standard POMDP formulation, this description function is deterministic.

A belief states $\bar{b}_t$ of the intention environment summarizes a history $(\bar{s}_1, \bar{a}_1, \cdots, \bar{s}_t)$. We define the intention policy as $\psi_\theta : \bar{\mathcal{B}} \to \Delta(\mathcal{A})$, where $\bar{\mathcal{B}}$ is the set of all belief states.

4.2. Goal Stack

The goal stack is a list of tasks that the agent has not declared complete. The initial stack $G_1 = \{(g_1, d^g_1)\}$ contains the main goal and its description. At time t, the agent executes the goal at the top of the current stack, i.e. $g_t = G_t.\text{top}()$. Only the GOAL, SUB, and DONE actions alter the stack. GOAL replaces the top goal description of G_t with the new description d^g_{t+1} from the assistant. SUB, which is only available when the stack is not full, pushes a new subtask (g_{t+1}, d^g_{t+1}) to G_t. DONE pops the top element from the stack. The goal-stack transition function is $T_G(G_{t+1} \mid G_t, \bar{a}_t) = \mathbb{1}\{G_{t+1} = G_t.\texttt{update}(\bar{a}_t)\}$ where $\mathbb{1}\{.\}$ is an indicator and $G_t.\text{update}(a)$ is the updated stack after action a is taken.

4.3. State Transition

The transition function T_s is factored into two terms:

$$T_s(s_{t+1}, d^s_{t+1} \mid s_t, d^s_t, \bar{a}_t) = \underbrace{P(s_{t+1} \mid s_t, \bar{a}_t)}_{\triangleq p_{\text{state}}(s_{t+1})} \cdot \underbrace{P(d^s_{t+1} \mid s_{t+1}, d^s_t, \bar{a}_t)}_{\triangleq p_{\text{desp}}(d^s_{t+1})} \quad (2)$$

Only the DO action changes the execution state, with: $p_{\text{state}}(s_{t+1}) = T\left(s_{t+1} \mid s_t, a^{\text{do}}_t\right)$, where a^{do}_t is the action chosen by $\hat{\pi}$ and T is the execution environment's state transition function. The current-state description is altered when the agent requests a new current-state description (CUR), where $p_{\text{desp}}(d^s_{t+1}) = \rho_A(d^s_{t+1} \mid s_{t+1}, d^s_t)$, or moves (DO), where $p_{\text{desp}}(d^s_{t+1}) = \rho(d^s_{t+1} \mid s_{t+1})$ (here, ρ is the description function of the execution environment).

4.4. Cost Function

The intention policy needs to trade-off between two types of cost: the cost of operation (making information requests and traversing in the environment), and the cost of not completing a task (task error). The two types of cost are in conflict: the agent may lower its task error if it is willing to suffer a larger operation cost by making more requests to the assistant.

We employ a simplified model where all types of cost are non-negative real numbers of the same unit. Making a request of type a is assigned a constant cost γ_a. The cost of taking the DO action is $c(s_t, a^{\text{do}}_t)$, the cost of executing the a^{do}_t action in the environment. Calling DONE to terminate execution of the main goal g_1 incurs a task error $c(s_t, a_{\text{done}})$. We exclude the task errors of executing subgoals because the intention policy is only evaluated on reaching the main goal. The cost function is concretely defined as follows

$$\bar{c}(s_t, G_t, \bar{a}_t) = \begin{cases} c(s_t, g_t, a^{\text{do}}_t) & \text{if } \bar{a}_t = \text{DO}, \\ \gamma_{\bar{a}_t} & \text{if } \bar{a}_t \notin \{\text{DO}, \text{DONE}\}, \\ c(s_t, g_t, a_{\text{done}}) & \text{if } \bar{a}_t = \text{DONE}, |G_t| = 1, \\ 0 & \text{if } \bar{a}_t = \text{DONE}, |G_t| > 1 \end{cases}$$

The magnitudes of the costs naturally specify a trade-off between operation cost and task error. For example, setting the task errors much larger than the other costs indicates that completing tasks is prioritized over taking few actions.

To facilitate learning, we apply reward shaping (Ng et al., 1999), augmenting a cost $c(s, G, a)$ with $\Phi(s, g)$, the task error received if the agent terminated in s. The cost received at time step t is $\tilde{c}_t \triangleq \bar{c}_t + \Phi(s_{t+1}, g_{t+1}) - \Phi(s_t, g_t)$. We assume that after the agent terminates the main goal, it transitions to a special terminal state $s_{\text{term}} \in \mathcal{S}$ and remains there. We set $\Phi(s_{\text{term}}, \texttt{None}) = 0$, where $g_t = \texttt{None}$ signals that the episode has ended. The cumulative cost of an execution under the new cost function is

$$\sum_{t=1}^{H} \tilde{c}_t = \sum_{t=1}^{H} \left[\bar{c}_t + \Phi(s_{t+1}, g_{t+1}) - \Phi(s_t, g_t)\right] \quad (3)$$

$$= \sum_{t=1}^{H} \bar{c}_t - \Phi(s_1, g_1) \quad (4)$$

Since $\Phi(s_1, g_1)$ does not depend on the action taken in s_1, minimizing the new cumulative cost does not change the optimal policy for the task (s_1, g_1).

5. Modeling Human-Assisted Navigation

Problem. We apply HARI to modeling a human-assisted navigation (HAN) problem, in which a human requests an agent to find an object in an indoor environment. Each task request asks the agent to go to a room of type r and find an object of type o (e.g., find a mug in a kitchen). The agent shares its current view with the human (e.g. via an app). We assume that the human is familiar with the environment and can recognize the agent's location. Before issuing a task request, the human imagines a goal location (not revealed to the agent). We are interested in evaluating success in *goal-finding*, i.e. whether the agent can arrive at the human's intended goal location. Even though there could be multiple locations that match a request, the agent only succeeds if it arrives exactly at the chosen goal location.

Environment. We construct the execution environments using the house layout graphs provided by the Matterport3D simulator (Anderson et al., 2018). Each graph is generated from a 3D model of a house where each node is a location in the house and each edge connects two nearby unobstructed locations. At any time, the agent is at a node of a graph. Its action space $\mathcal{A}$ consists of traversing to any of the nodes that are adjacent to its current node.

Scenario. We simulate the scenario where the agent is pre-trained in simulated environments and then deployed in real-world environments. Due to mismatches between the pre-training and real-world environments, the capability of the agent degrades at deployment time. It may not reliably recognize objects, the room it is currently in, or the intent of a request. However, a simulated human assistant is available to provide additional information about the environment and the task, which helps the agent relate its current task to the ones it has learned to fulfill during pre-training.

Subgoals. If the agent requests a subgoal, the assistant describes a new goal roughly halfway to its current goal (but limited to a maximum distance $l_{\max}$). Specifically, let p be the shortest path from the agent's current state s_t to the current goal g_t, and p_i be the i-th node on the path ($0 \leq i < |p|$). The subgoal location is chosen as p_k where $k = \min(\lfloor |p|/2 \rfloor, l_{\max})$, where $l_{\max}$ is a pre-defined constant.

State Description. We employ a discrete bag-of-features representation for state descriptions.[3] A bag-of-feature description (d^s or d^g) emulates the information that the agent has extracted from a raw input that it perceives (e.g., an image, a language sentence). Concretely, we assume that when the agent sees a view, it detects the room and nearby objects. Similarly, when it receives a task request or a human response to its request, it identifies information about rooms, objects, and actions in the response. Working with this discrete input representation allows us to easily simulate various types and amounts of information given to the agent.

Specifically, we model three types of input features: the name of a room, information about an object (name, distance and direction relative to a location), and the description of a navigation action (travel distance and direction). The human can supply these types of information to assist the agent. We simulate two settings of descriptions: *dense* and *sparse*. A dense description is a variable-length lists containing the following features: the current room's name and features of at most 20 objects within five meters of a viewpoint. Sparse descriptions represent imperfect perception of the agent in real-world environments. A sparse description is derived by first constructing a dense description and then removing the features of objects that are not in the top 100 most frequent (out of $\sim$1K objects). The sparse description of the current location (d^s) does not contain the room name, but that of the goal location (d^g) does. As each description is a sequence of feature vectors, whose length varies depend on the location, we use a Transformer model (Vaswani et al., 2017) to be able to encode such inputs into continuous feature vectors. Details about the feature representation and the model architecture are provided in the Appendix.

[3]While our representation of state descriptions simplifies the object/room detection problem for the agent, it does not necessarily make the navigation problem easier than with image input, as images may contain information that is not captured by our representation (e.g., object shapes and colors, visualization of paths).

Experimental Procedure. We conduct our experiments in three phases. In the *pre-training* phase, the agent learns an execution policy $\hat{\pi}$ with access to only dense descriptions. This emulates training in a simulator that supplies rich information to the agent.

In the *training* phase, the agent is only given sparse descriptions of its current and goal locations. This mimics the degradation of the agent's perception about the environment and the task when deployed in real-world conditions. In this phase, the agent can request dense descriptions from the human. Upon a request for information about a (goal or current) location, the human gives a dense description with room and object features of that location. However, when the agent requests a subgoal (SUB) *and* is adjacent to the subgoal that the human wants to direct it to, the human simply gives a description of the next optimal navigation action to get to that location. We use advantage actor-critic (Mnih et al., 2016) to learn an intention policy ψ_θ that determines which type of information to request. The intention policy is now trained in environment graphs that are previously seen as well as unseen during the pre-training phase.

Finally, in the *evaluation* phase, the agent is tested on three conditions: seen environment and target object type but starting from a new room (UNSEENSTR), seen environment but new target object type (UNSEENOBJ), and unseen environment (UNSEENENV). The execution policy $\hat{\pi}$ is fixed during the training and evaluation phases. We created 82,104 examples for pre-training, 65,133 for training, and approximately 2,000 for each validation or test set. Additional details about the training procedure and dataset are included in the Appendix.

6. Results and Analyses

Settings. In our main experiments, we set: the cost of taking a CUR, GOAL, SUB, or DO action to be 0.01 (we will consider other settings subsequently), the cost of calling DONE to terminate the main goal (i.e. task error) equal the (unweighted) length of the shortest-path from the agent's location to the goal, and the goal stack's size (L) to be 2.

Table 1: Test goal success rates and the average number of different types of actions taken by the agent (on all task types).

Agent	Success Rate % ↑			Avg. number of actions ↓			
	Unseen Start	Unseen Object	Unseen Environment	CUR	GOAL	SUB	DO
Rule-based intention policy ψ_θ (when to call DONE is decided by the execution policy $\hat{\pi}$)							
(d^s: current-state description, d^g: goal description)							
No assistance (always DO until DONE)	43.4	16.4	3.0	-	-	-	13.1
Dense d^g (GOAL then always DO until DONE)	67.2	56.6	9.7	-	1.0	-	12.6
Dense d^s (always CUR then DO until DONE)	77.9	30.6	4.1	12.0	-	-	12.0
Dense d^g and d^s (GOAL then always CUR then DO until DONE)	97.8	81.7	9.4	11.0	1.0	-	11.0
Random + rules to match with # of actions of learned-RL ψ_θ	78.8	68.5	12.7	2.0	1.0	1.7	11.3
learned-RL intention policy ψ_θ							
With pre-trained navigation policy $\hat{\pi}$ (ours)	85.8	78.2	19.8	2.1	1.0	1.7	11.1
With uncooperative assistant (change a request randomly to CUR, GOAL or SUB)	81.1	71.2	16.1	2.7	2.7	1.5	9.6
With perfect navigation policy on sub-goals (skyline)	94.3	95.1	92.6	0.0	0.0	6.3	7.3

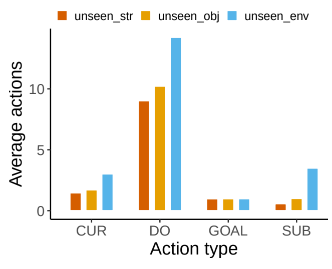
We compare our learned-RL intention policy with several rule-based *baselines*. Their descriptions are given in Table 1. We additionally construct a strong baseline that first takes the GOAL action (to clarify the human's intent) and then selects actions in such a way to match the distribution of actions taken by our RL-trained policy. In particular, this policy is constrained to take at most $\lfloor X_a \rfloor + y$ actions of type a, where $y \sim \text{Bernoulli}(X_a - \lfloor X_a \rfloor)$ and X_a is a constant tuned on the validation set so that the policy has the same average count of each action as the learned-RL policy. To prevent early termination, we enforce that the rule-based policy cannot take more DONE actions than SUB actions unless its SUB action's budget is exhausted. When there is no constraint on taking an action, the policy chooses a random action in the set of available actions. Our learned-RL policy can only outperform this policy by being able to determine contextually useful information to request, which is exactly the capability we wish to evaluate. We also construct a *skyline* where the intention policy is also learned by RL but with an execution policy that always fulfill subgoals perfectly.

In the problem we construct, different types of information are useful in different contexts (Table 1). Comparing the rule-based baselines reveals the benefits of making each type of request. Overall, we observe that information about the current state is helpful on only tasks in seen environments (UNSEENSTR and UNSEENOBJ). Information about the goal greatly improves performance of the agent in unseen environments (dense-d^g outperforms no-assistance by 6.4% in UNSEENENV). Dense-d^g outperforms dense-d^s in UNSEENOBJ but underperforms in UNSEENSTR, showing that goal information is more useful than current-state information in finding new objects but less useful in finding seen ones. The two types of information is complementary: dense-d^g-and-d^s improves success rate versus dense-d^g and dense-d^s in most evaluation conditions.

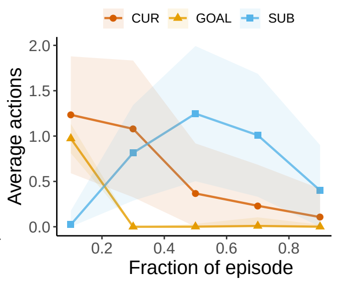
We expect subgoal information to be valuable mostly in unseen environments. This is confirmed by the superior performance of our learned-RL policy, which can request subgoals, over the rule-based baselines, which do not have this capability, in those environments.

Our learned-RL policy learns the advantages of each type of information (Table 1 & Figure 2a). Aided by our learned-RL policy, the agent observes a substantial $\sim$2× increase in success rate on UNSEENSTR, $\sim$5× on UNSEENOBJ, and $\sim$7× on UNSEENENV, compared to when performing tasks without assistance. This result indicates that the assistance-requesting skills are increasingly more helpful as the tasks become more unfamiliar. Figure 2a shows the request pattern of the policy in each evaluation condition. Our policy learns that it is not useful to make more than one GOAL request. It relies increasingly on CUR and SUB requests in more difficult conditions (UNSEENOBJ and UNSEENENV). The policy makes about 3.5× more SUB requests in UNSEENENV than in UNSEENOBJ. Specifically, with the capability of requesting subgoals, the agent impressively doubles the success rate of the dense-d^g-and-d^s policy in unseen environments, which *always* has access to dense descriptions in these environments. Moreover, our policy does not have to bother the assistant in every time step: only 1/4 of its actions are requests for information.

Our learned-RL policy selects contextually useful requests (Table 1, bottom half & Figure 2b). The policy is significantly more effective than the rule-based baseline that uses the same number of average actions per episode (+7.1% on UNSEENENV). Note that we enforce rules so that this baseline only differs from our learned-RL policy in where it places CUR and SUB requests. Thus, our policy has gained advantages by making these requests in more appropriate situations. To further showcase the importance of being able to obtain the right type of information, we use RL to train a policy with an *uncooperative* assistant, who disregards the

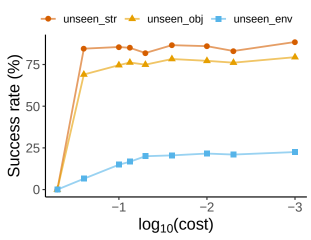


(a) Subgoals are requested much more in unseen environments.

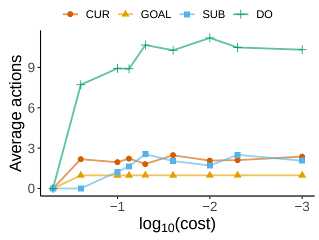


(b) Subgoals are requested in the middle, goal information at the beginning.

Figure 2: Analyzing the behavior of the learned-RL intention policy (on validation environments).



(a) Effect of cost on success.



(b) Effect of cost on actions.

Figure 3: Analyzing the effect of simultaneously varying the cost of the CUR, GOAL, SUB, DO actions (on validation environments), thus trading off success rate versus number of actions taken.

agent's request intention and instead replies to an intention randomly selected from $\{\text{CUR}, \text{GOAL}, \text{SUB}\}$. As expected, performance of the agent drops in all evaluation conditions. This shows that our agent has effectively leveraged the cooperative assistant by making useful requests.

We also observe that the agent adapts its request strategy through the course of an episode. As observed in Figure 2b. the GOAL action, if taken, is always taken only once and immediately in the first step. The number of CUR actions gradually decreases over time. The agent makes most SUB requests in the middle of an episode, after its has attempted but failed to accomplish the main goals. We observe similar patterns on the other two validation sets.

Finally, results obtained by the skyline shows that further improving performance of the execution policy on short-distance goals would effectively enhance the agent's performance on long-distance goals.

Effects of Varying Action Cost (Figure 3). We assign the same cost to each CUR, GOAL, SUB, or DO action. Figure 3a demonstrates the effects of changing this cost on the success rate of the agent. An equal cost of 0.5 makes it too costly to take any action, inducing a policy that always calls DONE in the first step and thus fails on all tasks. Overall, the success rate of the agent rises as we reduce the action cost. The increase in success rate is most visible in UNSEENENV and least visible in UNSEENSTR. Similar patterns are observed with various time limits ($H = 10, 20, 30$). Figure 3b provides more insights. As the action cost decreases, we observe a growth in the number of SUB and DO actions taken by the intention policy. Meanwhile, the numbers of CUR and GOAL actions are mostly static. Since requesting subgoals is more helpful in unseen environments, the increase in the number of SUB actions leads the more visible boost in success rate on UNSEENENV tasks.

Table 2: Success rates and numbers of actions taken with different stack sizes (on validation). Larger stack sizes significantly aid success rates in unseen environments, but not in seen environments.

Stack size	Success rate (%) ↑ Unseen Start	Unseen Obj.	Unseen Env.	Average # of actions ↓ CUR	GOAL	SUB	DO
1 (no subgoals)	92.2	78.4	12.5	5.1	1.9	0.0	10.7
2	86.9	77.6	21.6	2.1	1.0	1.7	11.2
3	83.2	78.6	33.5	1.3	1.0	5.0	8.2

Recursively Requesting Subgoals of Subgoals (Table 2). In Table 2, we test the functionality of our framework with a stack size 3, allowing the agent to request subgoals of subgoals. As expected, success rate on UNSEENENV is boosted significantly (+11.9% compared to using a stack of size 2). Success rate on UNSEENOBJ is largely unchanged; we find that the agent makes more SUB requests on those tasks (averagely 4.5 requests per episode compared to 1.0 request made when the stack size is 2), but doing so does not further enhance performance. The agent makes less CUR requests, possibly in order to offset the cost of making more SUB requests, as we keep the action cost the same in these experiments. Due to this behavior, success rate on UNSEENSTR declines with larger stack sizes, as information about the current state is more valuable for these tasks than subgoals. These results show that the critic model overestimates the V values in states where SUB actions are taken, leading to the agent learning to request more subgoals than needed. This suggests that the our critic model is not expressive enough to encode different stack configurations.

7. Related Work

Knowledge Transfer in Reinforcement Learning. Various frameworks have been proposed to model information transfer between humans and agents (Da Silva & Costa, 2019). A basic human-agent communication protocol is instruction-following (Chen & Mooney, 2011; Bisk et al., 2016; Alomari et al., 2017; Anderson et al., 2018; Misra et al., 2018; Yu et al., 2018), where agents execute a natural language request. For multiple-round communication, Torrey & Taylor (2013) introduce the action-advising framework where a learner decides situations where it needs to request reference actions from a teacher. This can be viewed as active learning (Angluin, 1988; Cohn et al., 1994) in an RL setting. The request policy can be constructed based on uncertainty measurements (Da Silva et al., 2020; Culotta & McCallum, 2005) or learned via RL (Fang et al., 2017). An agent-to-agent variant poses the problem of learning a teaching policy in addition to the request policy (Da Silva et al., 2017; Zimmer et al., 2014; Omidshafiei et al., 2019). Overall, this literature uses IL-based communication protocols that assume the teacher shares a common action space with the learner and provides actions in this space in response to the agent's requests. Others employ standard RL communication protocols, where the human conveys knowledge through numerical scores or categorical feedback (Knox & Stone, 2009; Judah et al., 2010; Griffith et al., 2013; Peng et al., 2016; Christiano et al., 2017; Lee et al., 2021). In Maclin & Shavlik (1996), humans advise the agent with domain-specific language, specifying rules to incorporate into the agent's model.

Recent frameworks (Jiang et al., 2019; Kim et al., 2020; Nguyen et al., 2019; Nguyen & Daumé III, 2019) allow the teacher to specify high-level goals instead of low-level actions or rewards. HARI generalizes these, allowing specification of subgoal information and descriptions of the current and goal states. Importantly, the framework enables the agent to convey specific intentions rather than passively receiving instructions or calling for generic help.

Information expressed in human language has been incorporated into RL frameworks to guide task execution (Hermann et al., 2017) or assist with learning (Ammanabrolu et al., 2020)—see Luketina et al. (2019) for a thorough survey. Language-based inverse RL frameworks infer the reward function from language utterances (Fu et al., 2019; Sumers et al., 2021; Akakzia et al., 2021; Zhou & Small, 2021). In contrast, we investigate human-agent communication *during* task execution—the agent does *not* update its task-executing policy. Similar to language-conditioned RL, we directly incorporate feedback as input to the agent's policy. This approach also bears a resemblance to work on learning from language description (Jiang et al., 2019; Chan et al., 2019; Colas et al., 2020; Cideron et al., 2020; Nguyen et al., 2021), except that the agent does not learn from the response and can incorporate more diverse feedback. Our setting is closely related to He et al. (2013), where an agent selects a subset of useful input features, but we remove the assumption that features arrive in a fixed order.

Our work requires solving a hierarchical reinforcement learning problem (Sutton et al., 1999; Kulkarni et al., 2016; Le et al., 2018; Chane-Sane et al., 2021). While standard formulations mostly consider two levels of goal, our stack-based implementation offers a convenient way to construct deeper goal hierarchies.

Task-Oriented Dialog and Natural Language Questions. HARI models a task-oriented dialog problem. Most problem settings require the agent to compose questions to gather information from humans (De Vries et al., 2017; Das et al., 2017; De Vries et al., 2018; Thomason et al., 2020; Padmakumar et al., 2022; Narayan-Chen et al., 2019; Shi et al., 2022). A related line of work is generating language explanations of model decisions (Camburu et al., 2018; Hendricks et al., 2016; Rajani et al., 2019). The dominant approach in these spaces is to mimic pre-collected human utterances. However, naively mirroring human behavior cannot enable agents to understand the limits of their knowledge. We take a different approach, teaching the agent to understand its intrinsic needs through interaction. Teaching agents to act and communicate with intentions is understudied (Karimpanal, 2021). Notable work in this space proposes a computational theory-of-mind for agents to enable them to reason about their own mental states and those of others (Andreas & Klein, 2016; Fried et al., 2018; Kang et al., 2020; Roman Roman et al., 2020; Zhu et al., 2021; Bara et al., 2021). Our work is not concerned about designing theory-of-mind models, instead concentrating on highlighting the importance of interaction in the learning of intentional behavior.

8. Conclusion

Our framework can theoretically capture rich human-agent communication, and even communication with other knowledge sources like search engines or language models (Liu et al., 2022). An important empirical question is how well it generalizes to more realistic interactions and environments (e.g., (Shridhar et al., 2020)). Especially when deploying with real humans, using human simulations to pre-train the agent can help reducing human effort. Large language models (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022) can be leveraged for constructing such simulators. Towards generating more specific, natural questions, methods for generating faithful explanations (Kumar & Talukdar, 2020; Madsen et al., 2021), measuring feature importance (Zeiler & Fergus, 2014; Koh & Liang, 2017; Das & Rad, 2020), or learning the casual structure of black-box policies (Geiger et al., 2021) are relevant to investigate. Another exciting direction is to enable the agent to learn from the obtained information. We expect that, similar to curriculum-based learning (Wang et al., 2019), the communication protocol of our framework would guide the agent to request and learn increasingly more difficult tasks over time.

References

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. Grounding language to autonomously-acquired skills via goal generation, 2021.

Alomari, M., Duckworth, P., Hogg, D., and Cohn, A. Natural language acquisition and grounding for embodied robotic systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Ammanabrolu, P., Tien, E., Hausknecht, M., and Riedl, M. O. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*, 2020.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

Andreas, J. and Klein, D. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1125.

Angluin, D. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

Bara, C.-P., CH-Wang, S., and Chai, J. MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1112–1125, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.85.

Bisk, Y., Yuret, D., and Marcu, D. 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*, pp. 751–761, 2016.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. e-snli: Natural language inference with natural language explanations. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.

Chan, H., Wu, Y., Kiros, J., Fidler, S., and Ba, J. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546*, 2019.

Chane-Sane, E., Schmid, C., and Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pp. 1430–1440. PMLR, 2021.

Chen, D. and Mooney, R. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 859–865, 2011.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Cideron, G., Seurin, M., Strub, F., and Pietquin, O. Higher: Improving instruction following with hindsight generation for experience replay. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 225–232. IEEE, 2020.

Clark, H. H. and Brennan, S. E. Grounding in communication. 1991.

Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine learning*, 15(2): 201–221, 1994.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P., and Oudeyer, P.-Y. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Advances in Neural Information Processing Systems*, 33:3761–3774, 2020.

Culotta, A. and McCallum, A. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pp. 746–751, 2005.

Da Silva, F. L. and Costa, A. H. R. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

Da Silva, F. L., Glatt, R., and Costa, A. H. R. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*, pp. 1100–1108, 2017.

Da Silva, F. L., Hernandez-Leal, P., Kartal, B., and Taylor, M. E. Uncertainty-aware action advising for deep reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5792–5799, 2020.

Das, A. and Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 326–335, 2017.

De Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., and Courville, A. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5503–5512, 2017.

De Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., and Kiela, D. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1625–1634, 2018.

Fang, M., Li, Y., and Cohn, T. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 595–605, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1063.

Fried, D., Andreas, J., and Klein, D. Unified pragmatic models for generating and following instructions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1951–1963, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1177.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *Proceedings of the International Conference on Learning Representations*, 2019.

Geiger, A., Wu, Z., Lu, H., Rozner, J., Kreiss, E., Icard, T., Goodman, N. D., and Potts, C. Inducing causal structure for interpretable neural networks. *arXiv preprint arXiv:2112.00826*, 2021.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

Graesser, A. C., Person, N., and Huber, J. Mechanisms that generate questions. *Questions and information systems*, 2:167–187, 1992.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

He, H., Daumé III, H., and Eisner, J. Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1455–1464, 2013.

Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. Generating visual explanations. In *European conference on computer vision*, pp. 3–19. Springer, 2016.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Hong, Y., Wu, Q., Qi, Y., Rodriguez-Opazo, C., and Gould, S. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Judah, K., Roy, S., Fern, A., and Dietterich, T. Reinforcement learning via practice and critique advice. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Judah, K., Fern, A. P., Dietterich, T. G., and Tadepalli, P. Active imitation learning: Formal and practical reductions to iid learning. *Journal of Machine Learning Research*, 15(120):4105–4143, 2014.

Kang, Y., Wang, T., and de Melo, G. Incorporating pragmatic reasoning communication into emergent language. *Advances in Neural Information Processing Systems*, 33: 10348–10359, 2020.

Karimpanal, T. Intentionality in reinforcement learning. 2021.

Kim, D.-K., Liu, M., Omidshafiei, S., Lopez-Cot, S., Riemer, M., Habibi, G., Tesauro, G., Mourad, S., Campbell, M., and How, J. P. Learning hierarchical teaching policies for cooperative agents. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

Knox, W. B. and Stone, P. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, 2009.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29: 3675–3683, 2016.

Kumar, S. and Talukdar, P. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8730–8742, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.771.

Labutov, I., Basu, S., and Vanderwende, L. Deep questions without deep understanding. 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)*, pp. 889–898, 2015.

Le, H., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and Daumé, H. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pp. 2917–2926. PMLR, 2018.

Lee, K., Smith, L., and Abbeel, P. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *Proceedings of the International Conference of Machine Learning*, 2021.

Liu, B., Wei, H., Niu, D., Chen, H., and He, Y. Asking questions the human way: Scalable question-answer generation from text corpus. In *Proceedings of The Web Conference 2020*, pp. 2032–2043, 2020.

Liu, I.-J., Yuan, X., Côté, M.-A., Oudeyer, P.-Y., and Schwing, A. G. Asking for knowledge: Training rl agents to query external knowledge using language. In *Proceedings of the International Conference of Machine Learning*, 2022.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. In *International Joint Conference on Artificial Intelligence*, 2019.

Maclin, R. and Shavlik, J. W. Creating advice-taking reinforcement learners. *Machine Learning*, 22(1):251–281, 1996.

Madsen, A., Meade, N., Adlakha, V., and Reddy, S. Evaluating the faithfulness of importance measures in nlp by recursively masking allegedly important tokens and retraining. *arXiv preprint arXiv:2110.08412*, 2021.

Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2667–2678, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1287.

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

Mostafazadeh, N., Misra, I., Devlin, J., Mitchell, M., He, X., and Vanderwende, L. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1802–1813, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1170.

Narayan-Chen, A., Jayannavar, P., and Hockenmaier, J. Collaborative dialogue in Minecraft. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5405–5415, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1537.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Nguyen, K. and Daumé III, H. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2019.

Nguyen, K., Dey, D., Brockett, C., and Dolan, B. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Nguyen, K., Misra, D., Schapire, R., Dudík, M., and Shafto, P. Interactive learning from activity description. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Omidshafiei, S., Kim, D.-K., Liu, M., Tesauro, G., Riemer, M., Amato, C., Campbell, M., and How, J. P. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6128–6136, 2019.

Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., Piramuthu, R., Tur, G., and Hakkani-Tur, D. Teach: Task-driven embodied agents that chat. In *Association for the Advancement of Artificial Intelligence*, 2022.

Peng, B., MacGlashan, J., Loftin, R., Littman, M. L., Roberts, D. L., and Taylor, M. E. A need for speed: Adapting agent action speed to improve task learning from non-expert humans. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2016.

Qi, Y., Wu, Q., Anderson, P., Wang, X., Wang, W. Y., Shen, C., and Hengel, A. v. d. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9982–9991, 2020.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Rajani, N. F., McCann, B., Xiong, C., and Socher, R. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4932–4942, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1487.

Rao, S. and Daumé III, H. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2737–2746, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1255.

Roman Roman, H., Bisk, Y., Thomason, J., Celikyilmaz, A., and Gao, J. RMM: A recursive mental model for dialogue navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1732–1745, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.157.

Rosenthal, S., Biswas, J., and Veloso, M. M. An effective personal mobile robot agent through symbiotic human-robot interaction. In *AAMAS*, volume 10, pp. 915–922, 2010.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Scott-Phillips, T. *Speaking our minds: Why human communication is different, and how language evolved to make it special*. Macmillan International Higher Education, 2014.

Shi, Z., Feng, Y., and Lipani, A. Learning to execute or ask clarification questions. *arXiv preprint arXiv:2204.08373*, 2022.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

Sperber, D. and Wilson, D. *Relevance: Communication and cognition*, volume 142. Citeseer, 1986.

Stalnaker, R. Common ground. *Linguistics and philosophy*, 25(5/6):701–721, 2002.

Sumers, T. R., Ho, M. K., Hawkins, R. D., Narasimhan, K., and Griffiths, T. L. Learning rewards from linguistic feedback. In *Association for the Advancement of Artificial Intelligence*, 2021.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Tellex, S., Knepper, R., Li, A., Rus, D., and Roy, N. Asking for help using inverse semantics. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014. doi: 10.15607/RSS.2014.X.024.

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. Vision-and-dialog navigation. In *Conference on Robot Learning*, pp. 394–406. PMLR, 2020.

Tomasello, M., Carpenter, M., Call, J., Behne, T., and Moll, H. Understanding and sharing intentions: The origins of cultural cognition. *Behavioral and brain sciences*, 28(5): 675–691, 2005.

Torrey, L. and Taylor, M. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1053–1060, 2013.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Yu, H., Lian, X., Zhang, H., and Xu, W. Guided feature transformation (gft): A neural language grounding module for embodied agents. In *Conference on Robot Learning*, pp. 81–98. PMLR, 2018.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhou, L. and Small, K. Inverse reinforcement learning with natural language goals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11116–11124, 2021.

Zhu, H., Neubig, G., and Bisk, Y. Few-shot language coordination by modeling theory of mind. In *International Conference on Machine Learning*, pp. 12901–12911. PMLR, 2021.

Zimmer, M., Viappiani, P., and Weng, P. Teacher-student framework: a reinforcement learning approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

A. Training Procedure

Training Algorithms. We pre-train the execution policy $\hat{\pi}$ with DAgger (Ross et al., 2011), minimizing the cross entropy between its action distribution with that of a shortest-path oracle (which is a one-hot distribution with all probability concentrated on the optimal action).

We use advantage actor-critic (Mnih et al., 2016) to train the intention policy ψ_θ. This method simultaneously estimates an actor policy $\psi_\theta : \bar{\mathcal{B}} \rightarrow \Delta(\bar{\mathcal{A}})$ and a critic function $V_\phi : \bar{\mathcal{B}} \rightarrow \mathbb{R}$. Given an execution $\bar{\tau} = (\bar{s}_1, \bar{a}_1, \bar{c}_1 \cdots, \bar{s}_H)$, the gradients with respect to the actor and critic are

$$\nabla_\theta \mathcal{L}_{\text{actor}} = \sum_{t=1}^{H} \left(V_\phi(\bar{b}^v_t) - C_t \right) \nabla_\theta \log \psi_\theta(\bar{a}_t \mid \bar{b}^a_t) \tag{5}$$

$$\nabla_\phi \mathcal{L}_{\text{critic}} = \sum_{t=1}^{H} \left(V_\phi(\bar{b}^v_t) - C_t \right) \nabla_\phi V_\phi(\bar{b}^v_t) \tag{6}$$

where $C_t = \sum_{j=t}^{H} c_j$, $\bar{b}^a_t$ is a belief state that summarizes the partial execution $\bar{\tau}_{1:t}$ for the actor, and $\bar{b}^v_t$ is a belief state for the critic.

Model Architecture. We adapt the V&L BERT architecture (Hong et al., 2020) for modeling the execution policy $\hat{\pi}$. Our model has two components: an encoder and a decoder; both are implemented as Transformer models (Vaswani et al., 2017). The encoder takes as input a description d^s_t or d^g_t and generates a sequence of hidden vectors. In every step, the decoder takes as input the previous hidden vector b^s_{t-1}, the sequence of vectors representing d^s_t, and the sequence of vectors representing d^g_t. It then performs self-attention on these vectors to compute the current hidden vector b^s_t and a probability distribution over navigation actions p_t.

The intention policy ψ_θ (the actor) is an LSTM-based recurrent neural network. The input of this model is the execution policy's model outputs, b^s_t and p_t, and the embedding of the previously taken action $\bar{a}_{t-1}$. The critic model also has a similar architecture but outputs a real number (the V value) rather than an action distribution. When training the intention policy, we always fix the parameters of the execution policy. We find it necessary to pre-train the critic before training it jointly with the actor.

Representation of State Descriptions. The representation of each object, room, or action is computed as follows. Let f^{name}, f^{horz}, f^{vert}, f^{dist}, and f^{type} are the features of an object f, consisting of its name, horizontal angle, vertical angle, distance, and type (a type is either `Object`, `Room`, or `Action`; in this case, the type is `Object`). For simplicity, we discretize real-valued features, resulting in 12 horizontal angles (corresponding to $\pi/6 \cdot k, 0 \le k < 12$), 3 vertical angles (corresponding to $\pi/6 \cdot k, -1 \le k \le 1$), and 5 distance values (we round down a real-valued distance to the nearest integer). We then lookup the embedding of each feature from an embedding table and sum all the embeddings into a single vector that represents the corresponding object. For a room, f^{horz}, f^{vert} f^{dist} are zeroes. For an action, f^{name} is either `ActionStop` for the stop action a_{done} or `ActionGo` otherwise.

During the pre-training phase, we randomly drop features in d^s_t and d^g_t so that the execution policy is familiar with making decisions under sparse information. Concretely, we refer to all features of an object, room or action as a *feature set*. For d^s_t, let M be the number of objects in a description. We uniformly randomly keep m feature sets among the $M + 1$ feature sets of d^s_t (the plus one is the room's feature set), where $m \sim \text{Uniform}(\min(5, M + 1), M + 1)$.

For d^s_t, we have two cases. If g_1 is not adjacent or equals to s_1, we uniformly randomly alternate between giving a dense and a sparse description. In this case, the sparse description contains the features of the target object and the goal room's name. Otherwise, with a probability of ⅓, we give either (a) a dense description (b) a (sparse) description that contains the target object's features and the goal room's name, or (c) a (sparse) description that describes the next ground-truth action.

We pre-train the execution policy on various path lengths (ranging from 1 to 10 graph nodes) so that it learns to accomplish both long-distance main goals and short-distance subgoals.

Data. Table 3 summarizes the data splits. From a total of 72 environments provided by the Matterport3D dataset, we use 36 environments for pre-training, 18 as unseen environments for training, 7 for validation UNSEENENV, and 11 for test

Table 3: Dataset statistics.

Split	Number of examples
Pre-training	82,104
Pre-training validation	3,000
Training	65,133
Validation UNSEENSTR	1,901
Validation UNSEENOBJ	1,912
Validation UNSEENENV	1,967
Test UNSEENSTR	1,653
Test UNSEENOBJ	1,913
Test UNSEENENV	1,777

Table 4: Hyperparameters.

Hyperparameter Name	Value
Environment	
Max. subgoal distance (l_{max})	3 nodes
Max. stack size (L)	2
Max. object distance for d_t^s	5 meters
Max. object distance for d_t^g	3 meters
Max. number of objects (M_{max})	20
Cost of taking each CUR, GOAL, SUB, DO action	0.01
Execution policy $\hat{\pi}$	
Hidden size	256
Number of hidden layers	2
Attention dropout probability	0.1
Hidden dropout probability	0.1
Number of attention heads	8
Optimizer	Adam
Learning rate	10^{-4}
Batch size	32
Number of training iterations	10^5
Max. number of time steps (H)	15
Intention policy ψ_θ	
Hidden size	512
Number of hidden layers	1
Entropy regularization weight	0.001
Optimizer	Adam
Learning rate	10^{-5}
Batch size	32
Number of critic pre-training iterations	5×10^3
Number of training iterations	5×10^4
Max. number of time steps (H)	30
Max. number of time steps for executing a subgoal	$3\times$ shortest distance to the subgoal

UNSEENENV. We use a vocabulary of size 1738, which includes object and room names, and special tokens representing the distance and direction values. The length of a navigation path ranges from 5 to 10 graph nodes.

Hyperparameters. See Table 4.