DialFRED: Dialogue-Enabled Agents for **Embodied Instruction Following**

Xiaofeng Gao[®], *Member, IEEE*, Qiaozi Gao[®], Ran Gong, Kaixiang Lin, Govind Thattai, and Gaurav S. Sukhatme^(D), *Fellow*, *IEEE*

Abstract—Language-guided Embodied AI benchmarks requiring an agent to navigate an environment and manipulate objects typically allow one-way communication: the human user gives a natural language command to the agent, and the agent can only follow the command passively. We present DialFRED, a dialogueenabled embodied instruction following benchmark based on the ALFRED benchmark. DialFRED allows an agent to actively ask questions to the human user; the additional information in the user's response is used by the agent to better complete its task. We release a human-annotated dataset with 53 K task-relevant questions and answers and an oracle to answer questions. To tackle DialFRED, we propose a questioner-performer framework wherein the questioner is pre-trained with the human-annotated data and fine-tuned with reinforcement learning. Experimental results show that asking the right questions leads to significantly improved task performance. We make DialFRED publicly available and encourage researchers to propose and evaluate their solutions to building dialog-enabled embodied agents: https://github.com/ xfgao/DialFRED.

Index *Terms*—Human-robot collaboration, multi-modal perception for HRI, natural dialog for HRI.

I. INTRODUCTION

R OBOT assistants need to understand natural language and interact with the environment. interact with the environment. To help build languagedriven embodied agents, various benchmarks based on predefined tasks, datasets and evaluation metrics have been proposed [1], [2], allowing fair comparisons between different approaches. In these benchmarks, the agent is often given an instruction, following which it is supposed to execute the corresponding sequence of actions. Even with natural language instructions, such tasks are often overwhelming for the agent on its own due to two major challenges: 1) resolving ambiguities in natural language and grounding instructions to actions in a rich

Manuscript received 24 February 2022; accepted 8 July 2022. Date of publication 22 July 2022; date of current version 2 August 2022. This letter was recommended for publication by Associate Editor N. Yamanobe and Editor G. Venture upon evaluation of the reviewers' comments. This work was supported by Amazon Alexa AI. (Corresponding author: Xiaofeng Gao.)

Xiaofeng Gao and Ran Gong are with the Center for Vision, Cognition, Learning, and Autonomy, University of California, Los Angeles, CA 90034 USA (e-mail: xfgao@ucla.edu; nikepupu@ucla.edu).

Qiaozi Gao, Kaixiang Lin, and Govind Thattai are with the Amazon Alexa AI, Sunnyvale, CA 94086 USA (e-mail: qzgao@amazon.com; kaixianl@ amazon.com; thattg@amazon.com).

Gaurav S. Sukhatme is with the Amazon Alexa AI, Sunnyvale, CA 94086 USA, and also with the Department of Computer Science, USC Viterbi School of Engineering, Los Angeles, CA 90089 USA (e-mail: sukhatme@amazon.com).

Digital Object Identifier 10.1109/LRA.2022.3193254

Dialog Vision **Robot Action** Robot Human <turn left> Where is the The kitchen table <forward> is to your left. kitchen table? <turn left> Ok, what does the knife look The knife is yellow. <pick up [mask]> like? Got it!

Human Instruction: Move to the kitchen table and pick up the knife.

Fig. 1. Example dialogue between a robot and a human user during task completion. The robot raises questions to obtain additional information (e.g., when the target location is not clear) and to resolve ambiguities (e.g., when facing two knives on the table).

environment, and 2) planning for long-horizon action sequences and recovering from possible failures.

Humans, faced with inadequate information for a task, seek assistance from others. Similarly, embodied agents should be able to actively ask questions to humans, and utilize the verbal response to overcome challenges in understanding intent and task execution. For example, to deal with ambiguity in human instruction, clarifications are often necessary. As shown in Fig. 1, the instruction "pick up the knife," is ambiguous when there are two knives in front of the robot - knowing the color of the intended knife helps the agent ground the instruction to its environment.

We present **DialFRED**, an embodied instruction following benchmark allowing an agent to 1) actively ask questions to the human user, and 2) use the information in the response to better complete the task. DialFRED is built by augmenting ALFRED [2], an existing benchmark that pairs demonstrations of common household tasks with instructions. ALFRED language instructions are given as high level goals, e.g., Move a knife to the sink, and a sequence of step-by-step instructions (sub-goals), e.g., Move forward to the center table, Pick up the knife, Walk to the sink, Put the knife in the sink. ALFRED only contains 7 types of high level goals and 8 types of sub-goals. Existing work [3], [4] has exploited patterns in ALFRED task structures, and shown that models can acheive state-of-the-art performance by classifying the task type from high-level task instructions alone,

2377-3766 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

even without using step-by-step instructions. To mitigate this issue and ensure the necessity of instruction following, we build DialFRED by augmenting ALFRED for an increased number of task types. In addition, DialFRED facilitates agent-human dialogue by providing human-annotated task-relevant questions and answers.

Contributions: To enable the development and evaluation of dialogue-enabled agents in complex manipulation and navigation tasks, DialFRED consists of a) **25** types of sub-goal level tasks, compared to 8 sub-goals originally available in ALFRED; b) **53 K** human-annotated task-relevant questions and answers; and c) models for a questioner-performer framework showing that adding dialogue helps to significantly improve the instruction following performance. We make DialFRED publicly available and encourage researchers from related robotics disciplines to propose and evaluate their solutions to dialog-enabled embodied agents.

II. RELATED WORK

Embodied Question Answering and Instruction Following: Various virtual environments have been developed to accommodate robot agents completing common household tasks [5]–[7]. Building on these environments, tasks and benchmarks that require an interactive agent to extract information from the environment to answer specific questions have been proposed [8], [9]. The other line of work that uses these environments focuses on creating agents to interpret natural language instructions and perform tasks in the environment [1]. ALFRED [2], a recently proposed benchmark along this direction, requires the agent to complete complex household tasks by following natural language instructions. Dialogue-enabled agents in navigation or manipulation tasks have recently been proposed [10], [11] these focus on action prediction from dialogue history, and do not emphasize the agent's ability to ask clarification questions for better instruction grounding [12]. In this letter, we take a further step in this direction by presenting a benchmark for the agent to actively ask questions and learn from the answers to better finish tasks that require navigation and object interactions. We refer readers to this survey [13] for dataset comparison.

Task-Oriented Dialogue: In task-oriented dialogue, agents rely on skills beyond language modeling (e.g., processing multimodal sensory data, querying knowledge bases, reasoning based on observations and knowledge [14]–[17]). Towards building robust dialogue systems, both data-driven [18], [19] and reinforcement learning approaches [20]–[22] have been studied. Studies have shown that the ability to ask for help from humans is crucial for agent failure recovery [23]. For visual language navigation, multiple works study when and how to ask for help [24]–[27]. Household tasks however, pose greater challenges to agents compared to navigation-only tasks, due to longer action sequences, compositional task structures and irreversible object state changes. Our benchmark focuses on these complex household tasks requiring both navigation and manipulation.

finish a household task in a virtual environment. We further enable the agent to ask questions, and use the extra information in the responses to better complete the task.

Each task instance in DialFRED is a tuple of the initial environment state, the target environment state and an instruction. The agent's goal is to perform a sequence of actions to change the environment states to the target. Given a natural language instruction, the agent can choose to ask questions to the human, or to execute physical actions in its environment based on the information in the original instructions together with the questions and answers. Physical actions include all 5 navigation actions (e.g., Turn Left) and all 7 manipulation actions (e.g., Pickup). These actions can change environment states, and some of the changes are irreversible (e.g. Slice). Instead of always emitting a physical action, a dialog-enabled agent may emit a question. To standardize this benchmark, we provide an oracle that can answer a set of predefined types of questions (a crude 'simulated' human user). The oracle has access to the ground-truth states of the virtual environment, allowing it to provide accurate information regarding objects and tasks.

A. Hybrid Data Collection

We collect human annotations on Amazon Mechanical Turk for questions and answers. Each instance in the dataset is a tuple of question type $q \in Q$ asking about a specific property of an object $o \in O$ (o could be empty for questions not related to objects) and a human answer $a \in A$ for the question at the beginning of the task. Fig. 2 shows the data collection interface. The hybrid data collection (HDC) process is as follows:

- 1) Each annotator watches a 10 s video clip, which displays the state of the environment right before the task. Annotators also see the original language instructions for the task.
- 2) The annotator then selects one pertinent question (from several predefined questions) that they think may help the agent complete the task. The annotator may also type in a question of their own choosing if none of the provided questions are a good fit.
- 3) The annotator watches a second video clip this time of an expert agent performing the task.
- The annotator answers their own question and provides feedback (yes/no) on whether asking the question was necessary in the given scenario.

To generate the predefined question choices for the annotator to choose from, we consider three types of questions Q, related to the location and appearance of the query object o that needs to be interacted with to finish the task, and the relative direction between the agent's current position and the target position to guide the navigation:

- 1) Location: where is *o*?
- 2) Appearance: what does *o* look like?
- 3) Direction: which direction should I turn to?

Given a natural language instruction (e.g. *Put the egg in the microwave*), we parse it and extract all the nouns (e.g. *egg* and *microwave*). We insert each noun into one of the question templates to generate questions (e.g. *Where is the egg*? and *What does the microwave look like*?).

III. TASK AND DATASET

DialFRED requires an agent to follow natural language instructions and perform navigation and object manipulation to

10050

	Step 1. Hit the blue "Begin" button and watch the video till it stops.			
	Step 2. Now you are shown the following command: " Move to the bed, Grab the pillow. "			
	Select a question that can help you execute the above command.			
	 Which direction should I turn to? What does the <i>bed</i> look like? Where is the <i>bed</i>? What does the <i>pillow</i> look like? Where is the <i>pillow</i>? Other (type in below) 			
TTTTT	If no questions apply, type in your own question Step 3. Hit the red "Show Demonstration" button to watch the video that executes the command. Answer the above question:			
▶ 0.00/0.47 · · · · · · · · · · · · · · · · · · ·				
Begin Show demonstration	Step 4. Do you think the question is necessary to execute the command? \circ Yes \circ No			

Fig. 2. The annotation interface for hybrid data collection. The worker first clicks the "begin" button to watch a video clip showing the initial states of the environment. Given the instruction, the worker selects a question to help perform the task. Next, the worker clicks the "show demonstration" button to watch the expert demonstration on how to complete the task. The worker then answers their own question based on what they have learned from the videos. Finally, workers choose whether they think the questions and answers are necessary to help the agent carry out the command.

Tasks	Move & Pick	Pick & Move	Pick, Move & Slice	Move & Open
Instructions	Grab the fork.	Take the tissuebox to the floorlamp.	Cut the tomato with a knife.	Open the microwave.
Expert demonstrations				
Questions	Where is the fork?	Where is the floorlamp?	What does the knife look like?	Which direction should I turn to?
Oracle answers	The fork is behind you on the countertop.	The floorlamp is to your front right.	The knife is silver and made of metal.	You should turn right.
Human answers	It is on the kitchen island behind you.	It is in the corner of the room by the window.	It is long and silver with a blade.	Turn right and you will see the microwave.

Fig. 3. Examples in our QA dataset. We show instructions and questions asked by humans, and answers provided by both the oracle and humans. Compared to step-by-step instructions, our augmented instructions are concise and general, thus requiring the agent to understand its current state to generate the correct action sequences.

We collected human questions and answers for 29,376 subgoals (e.g., Take the knife to the counter) on Amazon Mechanical Turk via crowd-sourcing. For each sub-goal, we ask two different annotators to provide questions and answers. And we see a modest level of agreement in question selection between different annotators (Fleiss' $\kappa = 0.13$). To ensure the quality of the dataset, we first remove invalid annotations when the annotation time is less than 15 seconds. A worker is compensated \$0.25 for each Human Intelligence Task (HIT); the dataset collection cost is \sim \$10 K. The dataset is gathered over 112 rooms and 80 types of objects. Each human answer contains 6.73 words on average. A lexical complexity analysis on the human answers [28] shows that the number of different words (NDW) in the answers are 7915. The lexical sophistication (the proportion of words not in the 2000 most frequent words in the American National Corpus) is 49%. Example human questions and answers are shown in Fig. 3.

B. Generating Answers

In addition to the human answers, we build an oracle that provides templated answers which can be easier for the agent to understand. In Fig. 3, we show answers generated by the oracle for some task examples. To create the oracle, we take advantage of the ground-truth states of objects and the agent in the simulated environment: (i) to answer object location questions, we compute the direction of the object relative to the agent, and the receptacle that contains the object; (ii) to answer object appearance questions, we focus on its color and material. Object material is extracted from scene metadata in the underlying simulation environment (AI2-Thor). For object color, we extract object pixel RGB values from images, and map them to color names; (iii) for direction questions, we check the agent's location at the end of the task in the ground-truth action sequences, and compare it to the agent's initial location. Based on these metadata, we use language templates to generate answers. Some example templates include:

- Location: The *o* is to your [*direction*] in/on the [*container*].
- Appearance: The *o* is [*color*] and made of [*material*].
- Direction: You should turn [direction] / You don't need to move.

C. Data Augmentation on ALFRED

Each task in ALFRED has a goal, split into multiple sub-goals. Each sub-goal requires the agent to manipulate some objects or move to a target location. Two types of language instructions are given. The high-level task instruction (e.g., *Move a knife to the sink*) describes the overall goal. The step-by-step instructions (e.g., *Pick up the knife*) guide the agent to complete each subgoal. ALFRED exhibits clear patterns in both high-level task structures and sub-goal action sequences: tasks of the same type require very similar sub-goal sequences to complete them, and sub-goals of the same type (especially manipulation sub-goals) have almost fixed action sequences. The limited variety in tasks and sub-goals precludes instructions understanding – allowing models that directly classify the task type from high-level task instructions alone to perform well, without even using the step-by-step instructions [3], [4].

To get rid of the strong patterns in both high-level task structures and sub-goal actions in ALFRED, DialFRED uses data augmentation to increase the number of task types, and focuses on instruction following at the sub-goal level to encourage learning from instructions for each task. We also introduce augmentations on the original ALFRED sub-goal instructions to add ambiguities in language so that the sequence of actions cannot be fully determined by only focusing on the instruction – reasoning based on knowledge of the environment state is needed. We show examples for DialFRED tasks and instructions in Fig. 3.

Sub-goal augmentations: In ALFRED there are 8 only subgoal types (go to, pick up, put, cool, heat, clean, slice, toggle). The action sequences required to finish these are almost fixed. For example, *cool object* always corresponds to: *open fridge*, put object into fridge, close fridge, open fridge, take out object. To increase task variations, we augment the sub-goals in two ways. First, we split an original sub-goal into multiple low level actions, each action corresponding to a new sub-goal. For example, the sub-goal *clean object* is split into three sub-goals: put the object in the sink, turn on the faucet and turn off the faucet. Second, we merge multiple sub-goals into a new sub-goal. For example, sub-goals go to the fridge and open the fridge are merged into a new sub-goal which requires the agent to first go to the fridge and open it. Using these operations, we arrive at 25 sub-goal types in our augmented dataset. In our experiments, we evaluate agent performance on these sub-goals; henceforth referred to as tasks. To standardize the benchmark, we divide the sub-goal task instances into training and validation folds. We further divide the validation fold into seen and unseen splits depending on whether the environment presents in the training fold. This results in 34,253 tasks in the training fold, 1,296 tasks in the validation seen fold and 1,363 tasks in the validation unseen fold. The corresponding low level actions for all DialFRED tasks is displayed in Table IV.

Instruction augmentation: To generate instructions for Dial-FRED tasks, we create instruction templates for each low level action. For tasks created from split sub-goals, we directly use the template as instruction. For tasks created from combined sub-goals, we concatenate the instructions of low level actions within the sub-goal. In addition to the step-by-step instructions describing low level actions, we generate new instructions that only describe one major action in the task. For example, for the task go to the microwave and open the microwave (Fig. 3), a human may only describe the main action "open the microwave" in the instruction. The agent cannot determine the action sequence solely based on the instruction. It needs to have an understanding of its position in the room to decide which action to take next. We belive these instructions match human commands in real-world scenrios and are more challenging than the original step-by-step instructions. The example instructions for all DialFRED tasks is displayed in Table IV.

IV. METHOD

Our baseline for the **DialFRED** benchmark has two key components, a questioner and a performer (Fig. 4). The questioner asks questions based on the task instruction and agent observations. The performer predicts a sequence of actions to execute in the environment based on the original instruction and the questions and answers. A good questioner knows both when to ask a question and what question(s) to ask, so that the task can be better completed by the performer. We first train the questioner using human-annotated data, i.e., give the model a good starting point by mimicking human judgement. To improve the coordination between the questioner and the performer, we fine-tune the questioner with reinforcement learning. The questions and answers (QAs) together form a dialogue between two entities: the agent (represented by the performer and questioner model) and the human (represented by the instruction and the answers).

A. Architecture

Questioner: Our questioner model (Fig. 5) is based on a sequence-to-sequence architecture with attention [29]. It has two modules: an encoder and a decoder, both implemented using LSTM [30]. The instructions are first tokenized and go through an embedding layer to produce token embeddings $e_{1:N}$. At each time step, the LSTM encoder takes the embedding of the current token e_n and the previous encoder hidden state h_{n-1} to produce a new hidden state h_n . As a result, the encoder generates a sequence of hidden states $h_{1:N}$. The final encoder hidden state h_N is used to initialize the decoder's hidden state d_0 . The decoder uses the image ResNet feature I, the previous question token w_{i-1} and the previous decoder hidden state d_{i-1} to produce the new docoder hidden state d_i . The decoder hidden state d_i is used to attend over encoder hidden states $h_{1:N}$ and predict the next question token w_i via a dot-product attention mechanism. Each question is represented by two tokens, including one for the question type (i.e. loc for location, app for appearance and dir for direction), and the other for the target object. The questioner can also choose not to ask a question by generating a none token. We pre-train the questioner using questions selected by Turkers (Section III-A) on the training split.

Performer: Our performer is based on the Episodic Transformer [31], an attention-based multi-layer transformer model that encodes the full history of the instruction and QAs, visual observations and action history to predict future actions. To enable the model to handle all possible QAs from the questioner and oracle, we pre-train it on the training split by providing it with instructions and all possible questions, including a combination of question types and answers. Model parameters are optimized by minimizing the cross-entropy between predicted and expert actions.



Fig. 4. The questioner-performer architecture. The questioner generates questions based on the first person image of the agent and the task instruction. The oracle answers the question based on the scene metadata. The performer takes the image, the instruction, and question and answers as input to predict actions.



Fig. 5. The Questioner model. Given the instruction and image feature I, our model generates question tokens $w_{1:i}$.

B. Questioner Fine-Tuning Using RL

The goal of the questioner is to ask necessary questions to help the performer finish the task. Thus we fine-tune the questioner using reinforcement learning to learn when to ask a question and *what* question(s) to ask on the validation seen split. The learning system for question asking is modeled as a Markov Decision Process, specified by a tuple $\langle S, A, T, R \rangle$, where S is the state space (including the language instructions and visual observations), $A = Q \times O$ is the action space of all possible questions (in our cases a combination of question types Q and target object O), T(s'|s, a) is the transition function encoding how the performer can advance the task given the question and its corresponding answers, and R(s, a) is the reward function encoding the reward for each (s, a) pair. Our questioner model can be viewed as a policy network $\pi_{\theta}(s, a) = p(a|s; \theta)$ mapping each state vector s to a stochastic questioning policy, which can be learned based on the reward function to strike a balance between the number of questions and performance gain.

We adopt the following reward function structure to address this trade-off: reward for task completion $r_{suc} = 1.0$, penalty for each step $r_{step} = -0.01$, penalty per question asked $r_q =$ -0.05, penalty per invalid question $r_{invalid} = -0.1$. Some questions generated by the questioner cannot be answered by the oracle, e.g. the appearance of task-irrelevant objects. Thus we add a penalty for invalid questions. The reward forms the actor critic loss function [32], the gradient of which is used to update the model so that it can learn to ask necessary questions at correct time. The performer model is not updated during questioner fine-tuning.

C. Heuristic-Based Questioner

Inspired by [27], we implemented a questioner based on model confusion. The idea is that if the performer is not confident, the output action distribution would have high entropy; model confusion could be a good heuristic to know when to ask a question. We sort action probabilities in decreasing order; an agent is confused if the minimum difference between the top two actions is less than a threshold ϵ throughout the action sequences:

$$\min_{t}(p_{sorted}^{t}[0] - p_{sorted}^{t}[1]) < \epsilon \tag{1}$$

where the threshold ϵ is used to control the degree of confusion for asking questions. In practice, we set the confusion threshold $\epsilon = 0.5$ in the experiment.

V. EXPERIMENTS

We evaluate the baseline models on our dataset. We terminate the episode when it exceeds 1000 steps or has more than 10 failed actions.

A. Evaluation Metrics

We evaluate model performance using success rate and path weighted success rate. In DialFRED, task success *s* is a binary indicator of whether the task goal conditions have been achieved. For example, the task "clean the fork" requires the states of the fork to be changed from "dirty" to "clean". Task success *s* is defined as 1 if the goal conditions have been fully achieved, and 0 otherwise. Success rate is calculated via averaging *s* across all tasks. Path weighted success rate further takes into consideration the number of actions the robot takes. The path weighted success score for a task is calculated by:

$$p_s = s * L^* / \max(L^*, \hat{L}) \tag{2}$$

TABLE I

PERFORMANCE OF THE BASELINES. SEEN SR AND UNSEEN SR REPRESENT THE SUCCESS RATE ON VALID SEEN AND VALID UNSEEN SPLITS. PWSR IS THE PATH WEIGHTED SUCCESS RATE. NQ IS THE NUMBER OF QUESTIONS ASKED BY THE QUESTIONER. THE BEST RESULTS ARE HIGHLIGHTED IN BOLDFACE

#	Expt setting	Seen SR	Unseen SR	Seen PWSR	Unseen PWSR	NQ
1	Instruction only	25.4	18.3	18.4	11.4	0
2	All QAs	43.4	32.0	31.2	19.9	3.24
3	Random QA	39.9	27.9	28.4	17.3	0.81
4	Random MC	46.6	29.5	35.5	18.7	0.52
5	RL begin	47.3	32.7	33.5	20.1	0.37
6	RL anytime	47.8	33.6	34.2	20.4	0.71

TABLE II

ABLATION STUDY BY PERTURBING THE ORACLE. WE START FROM TWO SETTINGS: A QUESTIONER THAT HAS BEEN FINE-TUNED FOR ASKING QUESTIONS AT ANY TIME AND A QUESTIONER THAT ASKS A RANDOM QUESTION AT THE BEGINNING. WE PERTURB THE ORACLE BY NOT PROVIDING ANSWERS FOR ONE QUESTION TYPE 50% OF THE TIME. LOC PERC, APP PERC AND DIR PERC REPRESENT THE PERCENTAGE QUESTIONS ABOUT OBJECT LOCATIONS, OBJECT APPEARANCE AND DIRECTIONS RESPECTIVELY

Expt setting	Unseen SR	Unseen PWSR	NQ	Loc Perc.	App Perc.	Dir Perc.
Human	-	-	0.66	0.72	0.22	0.06
RL anytime	33.6	20.4	0.71	0.65	0.14	0.21
RL Loc perturb	32.2	19.8	0.66	0.47	0.15	0.38
RL App perturb	32.4	19.9	0.40	0.92	0.04	0.04
RL Dir perturb	33.0	20.4	0.61	0.51	0.45	0.04
Random	28.4	17.3	0.81	0.36	0.31	0.33
Random Loc perturb	26.0	16.0	0.81	0.36	0.31	0.33
Random App perturb	26.1	16.1	0.81	0.36	0.31	0.33
Random Dir perturb	26.2	15.8	0.81	0.36	0.31	0.33

TABLE III

EFFECT OF QUESTION TIMING. WE MANIPULATE THE NUMBER OF STEPS THE PERFORMER ROLLS OUT BEFORE THE QUESTIONER CAN ASK THE NEXT QUESTION. FOR (*Fixed 1*), We Modify the Rewards ($r_{invalid} = -0.01$, $r_q = -0.002$) to Promote Question Asking. For (*MC*), the Questioner Asks Questions Based on the Performer Model Confusion

Experiment setting	Seen SR	Unseen SR	Seen PWSR	Unseen PWSR	NQ
RL anytime (Fixed 1)	51.9	34.7	36.3	21.2	21.39
RL anytime (Fixed 5)	47.8	33.6	34.2	20.4	0.71
RL anytime (Fixed 10)	46.3	32.4	32.7	19.9	0.36
RL anytime (MC)	47.1	33.0	33.3	20.4	0.31

where s is the original task success indicator (0 or 1), \hat{L} is the number of steps taken by the agent to complete the task, and L^* is the number of steps in the expert demonstration. Intuitively, the more steps the robot took, the lower the score. Again, the path weighted success rate is calculated via averaging p_s across all tasks. To understand the trade-off between the number of questions asked and task performance, we measure the number of questions throughout the task.

B. Baselines

We implemented 6 baselines and evaluated their performance on our augmented dataset (Table I). In all baselines we use the Episodic Transformer model as the performer. In baselines 2–6, the performer is trained using imitation learning on expert action sequences with language instructions and QAs. In baselines 5–6, the questioner is pre-trained on human dialogues in the training split, and fine-tuned using reinforcement learning on the valid seen split. Baseline details are itemized below:

- 1) In this baseline, no questions are allowed, the performer is trained to predict the action sequences based on the language instruction and visual observations.
- 2) In addition to the instruction, the performer gets all valid QAs at the beginning of the task, including a combination

of all three question types (i.e. location, appearance and direction) and objects mentioned in the instruction.

- 3) Questions are sampled randomly based on type: 25% for each of the 3 types and 25% for no question. Given a selected question type, questions and answers for all relevant objects are given to the performer in addition to the instructions.
- 4) When the action sequences generated by the performer satisfy the model confusion criterion (1), the performer is randomly provided with a valid QA as input.
- 5) The questioner is fine-tuned using reinforcement learning to learn whether to ask a question and what question to ask at the beginning of a task. The performer uses the QA to generate the action sequence to finish the task.
- 6) Similar to baseline 5, the questioner is fine-tuned using reinforcement learning, but now it can ask questions in the middle of the task. Given the instruction and previous QAs, we roll out the performer for 5 steps, following which the questioner is allowed to generate new questions and thus get new answers.

C. Results

We display (Table I) the task performance (success rate) on validation seen and unseen splits for all baseline models.

#	Tasks	Example Low Level Actions	Example Instructions
1	Clean	put sink, turnon faucet, turnoff faucet, pickup pot	Wash the pot.
2	Close	close drawer	Close the drawer.
3	Cool	open fridge, put fridge, close fridge, open fridge, take tomato, close fridge	Chill the tomato.
4	Heat	open microwave, put microwave, close microwave, turnon microwave, turnoff microwave, open microwave, pickup apple, close microwave	Cook the apple.
5	Move	goto cart	Go to the cart.
6	Open	open safe	Open the safe.
7	Pick	pickup egg	Take the egg.
8	Put	put plate	Put on the plate.
9	Slice	slice letttuce	Cut the lettuce.
10	Turn on	turnon desklamp	Power on the desklamp.
11	Turn off	turnoff microwave	Switch off the microwave.
12	Move & Clean	goto sinkbasin, put sink, turnon faucet, turnoff faucet, pickup ladle	Go to the sinkbasin, wash the ladle.
13	Move & Close	goto laptop, close laptop	Close the laptop.
14	Move & Cool	goto fridge, open fridge, put fridge, close fridge, open fridge, take potato, close fridge	Move to the fridge, cool the potato.
15	Move & Heat	goto microwave, open microwave, put microwave, close microwave, turnon microwave, turnoff microwave, open microwave, pickup potato, close microwave	Heat up the potato.
16	Move & Open	goto microwave, open microwave	Open the microwave.
17	Move & Pick	goto countertop, pickup fork	Grab the fork.
18	Move & Put	goto countertop, put countertop	Place on the countertop.
19	Move & Slice	goto bread, slice bread	Cut the bread.
20	Move & Turn on	goto desklamp, turnon desklamp	Switch on the desklamp.
21	Open & Pick & Close	open microwave, pickup cup, close microwave	Pick up the cup.
22	Open & Put & Close	open fridge, put fridge, close fridge	Put into the fridge.
23	Pick & Move	pickup mug, goto sidetable	Take the mug to the sidetable.
24	Pick & Move & Put	pickup keychain, goto armchair, put keychain	Put the keychain on the armchair.
25	Pick & Move & Slice	pickup knife, goto bread, slice bread	Cut the bread with a knife

TABLE IV List of All 25 Tasks in DialFRED and Their Example Low Level Actions and Instructions. Note That Here We Do Not Further Split "goto" Into Low Level Actions (E.g. Move Forward, Turn Left, Etc.)

Comparing the results of baselines 1–3, we see that adding QAs to the instructions improves task performance on both splits. Comparing the results of baselines 2–5, we see that the fine-tuned questioner achieves the best performance, with a smaller number of questions. Comparing the results of two reinforcement learning baselines (5, 6), we see that enabling the agent to ask questions in the middle of the task improves performance at the cost of more questions and answers. In addition, the *random MC* baseline achieves reasonable performance on the valid seen split, but not on the valid unseen split. Since the performer is pre-trained on the training split, which has the same scene as the valid seen split, and is given random combinations of different types of QAs as input, it is not surprising that the *random MC* baseline does not generalize well to unseen environments.

D. Ablation Study

1) Perturbed Oracle: We perform an ablation study (Table II) by perturbing the oracle. For each question type, we limit the oracle to provide answers for 50% of the asked questions. The original fine-tuned questioner (*RL anytime*) has a similar number of questions (and distribution) as the human data it is pre-trained on – most questions are about object locations. With the perturbation, the model asks slightly fewer questions. We observe that the fine-tuned questioner model adapts to the deficient oracle. For example, after perturbation on object location answers, the questioner asks 28% fewer location questions and 81% more direction questions instead. Looking at model performance after perturbation, we find that perturbing the location answers reduces the performance the most. This

result matches the large proportion of location questions asked by both humans and the learning-tuned model, indicating that location answers are probably most useful for task completion. We perform the same perturbations on the oracle for the *random* questioner. Comparing the performance of models before and after perturbation, we see that the drop in SR caused by the perturbations for the learning-tuned questioners are smaller than the drops for the *random* questioners. The difference can be explained by the learning-tuned questioner's ability to adjust the proportion of questions to adapt to the deficient oracle.

2) Question Timing: To understand how question timing affects performance, we change the number of action steps the performer executes before allowing the questioner to ask a question. We add a setting, *RL anytime (MC)*, which allows the questioner to ask questions based on model confusion (1) during fine-tuning. The results (Table III) show that reducing the number of steps between questions leads to slightly better performance, but it also requires the oracle to answer significantly more questions. Comparing the results of model confusion with fixed timing, we find that it achieves reasonable performance at relatively low cost.

VI. CONCLUSION

We presented **DialFRED**, a dialogue-enabled embodied instruction following benchmark that allows an agent to actively ask questions while interacting with the environment to finish a household task. DialFRED is generated by augmenting AL-FRED to increase task and language variations. It includes an oracle to answer questions and a human-annotated dataset with 53 K task-relevant questions and answers – a potential resource to model how humans ask and answer task-oriented questions. To tackle DialFRED, we propose a questioner-performer baseline (and variants) wherein the questioner is pre-trained with the human-annotated data and fine-tuned with reinforcement learning. Experimental results show that asking the right questions leads to significantly improved task performance. Extending existing embodied instruction following benchmarks with dialogue is a promising avenue of research towards truly interactive embodied agents. Along these lines, we posit that the general framework of oracle-guided reinforcement training and the hybrid data annotation method we employ may be useful to "dialogue-enable" other embodied instruction following tasks.

REFERENCES

- P. Anderson et al., "Vision-and-language navigation: Interpreting visuallygrounded navigation instructions in real environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3674–3683.
- [2] M. Shridhar et al., "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10740–10749.
- [3] V. Blukis, C. Paxton, D. Fox, A. Garg, and Y. Artzi, "A persistent spatial semantic representation for high-level natural language instruction execution," in *Proc. Conf. Robot Learn.*, 2021, pp. 706–717.
- [4] S. Y. Min, D. S. Chaplot, P. K. Ravikumar, Y. Bisk, and R. Salakhutdinov, "FILM: Following instructions in language with modular methods," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–17.
- [5] E. Kolve et al., "Ai2-thor: An interactive 3D environment for visual ai," 2017, arXiv:1712.05474.
- [6] X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S. Zhu, "Vrkitchen: An interactive 3D virtual environment for task-oriented learning," 2019, arXiv:1903.05757.
- [7] F. Xia et al., "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 713–720, Apr. 2020.
- [8] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1–10.
- [9] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual question answering in interactive environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4089–4098.
- [10] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer, "Vision-anddialog navigation," in *Proc. Conf. Robot Learn.*, 2020, pp. 394–406.
- [11] A. Padmakumar et al., "TEACh: Task-driven embodied agents that chat," in Proc. Conf. Artif. Intell., 2022, pp. 2017–2025.
- [12] K. Jokinen and M. McTear, "Spoken dialogue systems," Synth. Lectures Hum. Lang. Technol., vol. 2, no. 1, pp. 1–151, 2009.
- [13] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. Wang, "Vision-and-language navigation: A survey of tasks, methods, and future directions," in *Proc.* 60th Annu. Meeting Assoc. Comput. Linguistics, vol. 1, Dublin, Ireland, May 2022, pp. 7606–7623.

- [14] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, "The first question generation shared task evaluation challenge," in *Proc.* 6th Int. Natural Lang. Gener. Conf., 2010, pp. 1–7.
- [15] J. Gao, M. Galley, and L. Li, "Neural approaches to conversational AI," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 1371–1374.
- [16] J. Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu, "Language to action: Towards interactive task learning with physical agents," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2–9.
- [17] J. Thomason et al., "Jointly improving parsing and perception for natural language commands through human-robot dialog," J. Artif. Intell. Res., vol. 67, pp. 327–374, 2020.
- [18] N. Mrkšić, D. Ó. Séaghdha, T.-H. Wen, B. Thomson, and S. Young, "Neural belief tracker: Data-driven dialogue state tracking," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, vol. 1, pp. 1777–1788.
- [19] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," *Adv. Neural Inf. Process. Syst.*, vol. 1, pp. 20179–20191, 2020.
- [20] P.-H. Su et al., "On-line active reward learning for policy optimisation in spoken dialogue systems," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, vol. 1, pp. 2431–2441.
- [21] B. Peng et al., "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2231–2240.
- [22] X. Hu, Z. Wen, Y. Wang, X. Li, and G. de Melo, "Interactive question clarification in dialogue via reinforcement learning," in *Proc. 28th Int. Conf. Comput. Linguistics: Ind. Track*, 2020, pp. 78–89.
- [23] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy, "Asking for help using inverse semantics," in *Proc. Robot.: Sci. Syst.*, 2014, pp. 1–9.
- [24] K. Nguyen, D. Dey, C. Brockett, and B. Dolan, "Vision-based navigation with language-based assistance via imitation learning with indirect intervention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12527–12537.
- [25] K. Nguyen and H. Daumé III, "Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2019, pp. 684–695.
- [26] H. R. Roman, Y. Bisk, J. Thomason, A. Celikyilmaz, and J. Gao, "RMM: A recursive mental model for dialog navigation," in *Proc. Findings Conf. Empirical Methods Natural Lang, Process.*, 2020, pp. 1732–1745.
- [27] T.-C. Chi, M. Shen, M. Eric, S. Kim, and D. Hakkani-tur, "Just ask: An interactive learning framework for vision and language navigation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 2459–2466.
- [28] X. Lu, "The relationship of lexical richness to the quality of ESL learners' oral narratives," *Modern Lang. J.*, vol. 96, no. 2, pp. 190–208, 2012.
 [29] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to
- [29] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.
- [30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [31] A. Pashevich, C. Schmid, and C. Sun, "Episodic transformer for visionand-language navigation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 15942–15952.
- [32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.