

## APPENDIX

### A. Failure Cases

a) *Threshold*: The uncertainty, threshold is 1.24 in the cooking robot case, where the uncertainty is 1.22 which is slightly lower than the threshold. For the case under, the system classifies ambiguous commands into certain commands.

Goal: Cook something delicious for me.  
 Robot Type: Cooking robot  
**GT label: Ambiguous**  
 robot action: robot.grab(bacon)  
**Uncertainty: 0.59**  
 robot action: robot.heat(bacon)  
**Uncertainty: 0.0**  
 robot action: robot.grab(bread)  
**Uncertainty: 0.82**  
 robot action: robot.heat(bread)  
**Uncertainty: 0.0**  
 robot action: robot.plate(bacon)  
**Uncertainty: 1.22**  
 ======  
**Pred label: Certain**

b) *Known keywords in the infeasible command*: If there is a known word in the goal in the infeasible commands, the large language models fail to predict it as an infeasible command. For example, in the case of a cooking robot, if the goal input contains words related to cooking, such as "after cooking a meal", the LLM predicts its goal as feasible.

Goal: Clean the kitchen after cooking a meal with bacon and banana  
 Robot Type: Cooking robot  
**GT label: infeasible**  
 ....  
 robot action: robot.plate(bacon)  
**Uncertainty: 1.49**  
 😊F: Yes, the robot can clean the kitchen after cooking a meal with bacon and banana. The robot can grab the banana and bacon, heat them, plate them, and then clean the kitchen by grabbing the water, desk, and table and wiping them down.  
 ======  
**Pred label: Ambiguous**

In addition, even though the possible action set is "clean" and "wipe" for the cleaning robot, in the "Organize the desk" commands, the LLM predicts that "organize" is in the action set. This sufficient understanding of the action set leads to failure, which can be relaxed via more prompts or fine-tuning approaches.

Goal: Organize the desk  
 Robot Type: Cleaning robot  
**GT label: infeasible**  
 robot action: robot.clean(desk)  
**Uncertainty: 0.93**  
 😊F: Yes, the robot can organize the desk. The robot action would be robot.organize(desk)  
 ======  
**Pred label: Ambiguous**

c) *Ambiguous commands with missing keywords*: Furthermore, the proposed methods sometimes fail on too vague commands with missing words. For example, "give a massage in the kitchen" can be interpreted as "give a massage to a person in the kitchen". However, the proposed method predicts it is infeasible because of the missing word "person" in this case.

Goal: Give a massage in the kitchen  
 Robot Type: Massage robot  
**GT label: Ambiguous**  
 robot action: robot.go to(kitchen)  
**Uncertainty: 1.17**  
 😊F: No, the robot cannot give a massage in the kitchen as it is not part of its action set. The robot can only "go to" and "give massage" to the person wearing the yellow shirt, white shirt, black shirt, or brown shirt.  
 ======  
**Pred label: infeasible**

d) *Lack of Information in Question*: In the scenario where the system has multiple missing information, the LLM fails to generate a question asking for both of the information.

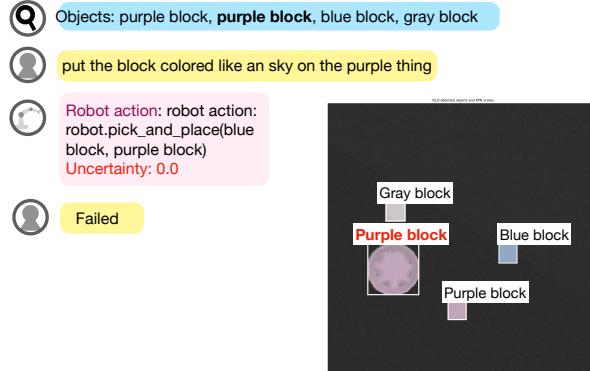
Goal: Give someone something to drink  
 Objects = [coca cola can, fanta can, lemon, apple]  
 People = [person wearing black shirts, person wearing white shirts]  
 robot.pick\_and\_give(coca cola can, person wearing blue shirt)  
**Uncertainty: 1.75**  
 😊F: Yes, I can give someone something to drink.  
 🤔R: It is not specified which person should receive the drink.  
 😊Q: Specify which person should receive the drink

e) *Malfunction in other modules*: Malfunctions on the vision module or robotic policy module can lead to wrong uncertainty quantification or overall failure. The examples are shown in Figure 1.

### B. Toy Example

Toy's example of a simple block-picking scenario is in Section III-B. In this example, the context sets  $C = \{C_1, C_2, C_3, C_4\}$  are prompted as follows:

### Vision Failure



### Policy Failure

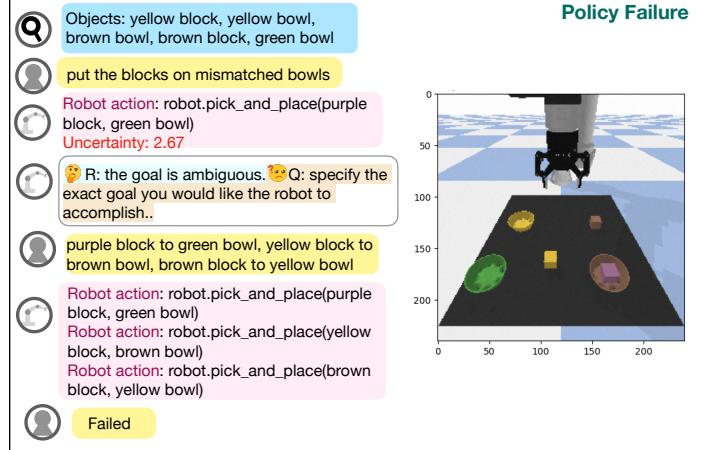


Fig. 1: Failure Cases due to other modules

- (C<sub>1</sub>) task: pick a red block - scene: [red block, blue block, yellow block] - robot action: robot.pick(red block)
- (C<sub>2</sub>) task: pick a block colored like a banana - scene: [green block, blue block, yellow block] - robot action: robot.pick(yellow block)
- (C<sub>3</sub>) task: pick something that resembles a tree - scene: [green block, blue block, yellow block, red block] - robot action: robot.pick(green block)
- (C<sub>4</sub>) task: pick something that is not blue - scene: [blue block, purple block] - robot action: robot.pick(purple block)

where the goal is "pick a block" with observing red block, blue block, yellow block in the scene. We first sample the contexts from  $C$  (e.g.,  $\{C_1, C_3, C_4\}$ ) and shuffle the scene description list. Then,  $c_1$  will be as follows:

- (C<sub>1</sub>) task: pick a red block - scene: [red block, blue block, yellow block] - robot action: robot.pick(red block)
- (C<sub>3</sub>) task: pick something that resembles a tree - scene: [green block, blue block, yellow block, red block] - robot action: robot.pick(green block)
- (C<sub>4</sub>) task: pick something that is not blue - scene: [blue block, purple block] - robot action: robot.pick(purple block)
- task: considering the ambiguity of the goal, **pick a block**  
**scene: [red block, yellow block, blue block]**

Similarly, if we sample  $\{C_3, C_2\}$  and shuffle the scene description list,  $c_3$  will be as follows:

- (C<sub>3</sub>) task: pick something that resembles a tree - scene: [green block, blue block, yellow block, red block] - robot action: robot.pick(green block)
- (C<sub>2</sub>) task: pick block colored like a banana - scene: [green block, blue block, yellow block] - robot action: robot.pick(yellow block)
- task: considering the ambiguity of the goal, **pick a block**  
**scene: [blue block, yellow block, red block]**

We then sample three generations with different prompts set and shuffled the order of observation.

Sample 1.  $\{C_1, C_3, C_4\}$

task: considering the ambiguity of the goal, **pick a block**  
**scene: [red block, yellow block, blue block]**  
**robot.pick(yellow block)**

Sample 2.  $\{C_2, C_4\}$

task: considering the ambiguity of the goal, **pick a block**  
**scene: [red block, yellow block, blue block]**  
**robot.pick(red block)**

Sample 3.  $\{C_3, C_2\}$

task: considering the ambiguity of the goal, **pick a block**  
**scene: [blue block, yellow block, red block]**  
**robot.pick(blue block)**

In such case,  $y_1, y_2, y_3$  becomes "yellow block", "red block", and "blue block" respectively. The distance matrix in word embedding space will be 02.51.7 2.502.3

1.72.30, which results in total uncertainty as 2.1.

In the classification section, the prompts and the generated outcome of the sample example are as follows:

(Continue from previous prompts)

robot thought: I am a robot that can pick an object.

Considering the action set, pick, can I **pick a block** ? answer in yes or no

Yes, I can pick a block given more information.

With the generated answer, we used a heuristic parser to distinguish commands from infeasible and ambiguous based on the keyword (i.e., yes). If the generated sentence contains the keyword "yes", we denote the corresponding command as ambiguous. In the toy example, the command is classified as ambiguous.

For the disambiguation progress, the prompts and generated outcome are as follows:

(Continue from previous prompts)

robot thought: this code is uncertain because the task does not specify any specific criteria for selecting the block  
(robot thought: what can I ask to the user? question: Please provide more information about the criteria for selecting a block

The system provides the reason for the uncertainty and the following question for disambiguation. After obtaining the user's answer, the system returns to the uncertainty estimation step with extended prompts with the disambiguation process.

Sample 1. ( $C_4$ ), ( $C_1$ ) task: considering the ambiguity of the goal, pick a block scene: [red block, yellow block, blue block]

(Append Generated Explanation and Question)

user: red block

robot.pick(red block)

### C. Demonstration in Mobile Manipulation Task

To analyze the proposed method in the larger environment under partial observability, we have demonstrated the proposed method in Alfworld [1], following the prompt baseline from [2]. The threshold for uncertainty is set as 1.5 in this environment.

In certain cases, the uncertainty remains lower than the threshold.

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: put a saltshaker in drawer 1.

Act 1: go to countertop 1

Uncertainty 1: 1.21

Obs 1: On the countertop 1, you see a creditcard 2, a dishsponge 2, and a mug 2.

Act 2: go to countertop 2

Uncertainty 2: 0.0

Obs 2: On the countertop 2, you see a creditcard 1, a cup 2, a knife 2, a knife 1, a lettuce 1, a papertowelroll 1, a soapbottle 2, a statue 1, a tomato 2, and a tomato 1.

Act 3: go to countertop 3

Uncertainty 3: 0.0

Obs 3: On the countertop 3, you see a bread 1, a butterknife 1, a houseplant 1, a potato 1, a spoon 1, and a vase 2.

Act 4: go to shelf 1

Uncertainty 4: 0.0

Obs 4: On the shelf 1, you see a mug 3, and a statue 2.

Act 5: go to shelf 2

Uncertainty 5: 0.0

Obs 5: On the shelf 2, you see a creditcard 3, and a dishsponge 3.

Act 6: go to shelf 3

Uncertainty 6: 0.0

Obs 6: On the shelf 3, you see a vase 1.

Act 7: go to cabinet 1

Uncertainty 7: 0.713

Obs 7: On the cabinet 1, you see a glassbottle 1. Act 8: go to cabinet 2

Uncertainty 8: 0.0

Obs 8: The cabinet 2 is closed.

Act 9: open cabinet 2

Uncertainty 9: 0.0

Obs 9: You open the cabinet 2. The cabinet 2 is open. In it, you see a plate 1, a saltshaker 1, and a soapbottle 1.

Act 10: take saltshaker 1 from cabinet 2

Uncertainty 10: 0.0

Obs 10: You pick up the saltshaker 1 from the cabinet 2.

Act 11: go to drawer 1

Uncertainty 11: 0.0

Obs 11: The drawer 1 is closed.

Act 12: open drawer 1

Uncertainty 12: 0.0

Obs 12: You open the drawer 1. The drawer 1 is open. In it, you see a fork 1, and a peppershaker 1.

Act 13: put saltshaker 1 in/on drawer 1

Uncertainty 13: 0.0

Obs 13: You put the saltshaker 1 in/on the drawer 1.

=====

Pred label: certain

In the ambiguous scenario, where the goal is to put something in the drawer 1, the uncertainty increases in the beginning, asking the user to clarify what something is. During proceeding with the task, the robot feels that it is stuck, and asks the user where the saltshaker is.

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: put something in the drawer 1

Act 1: go to cabinet 1

Uncertainty 1: 1.55

😊F: Yes. 😊R: First I need to find something I can put in the drawer 1.

😊Q: Tell me what item you would like to put in the drawer 1.

User: saltshaker

Act 2: go to countertop 1

Uncertainty 2: 0.647

Obs 2: On the countertop 1, you see a creditcard 2, a dishponge 2, and a mug 2.

Act 3: go to countertop 2

Uncertainty 3: 2.14

🤔R: The saltshaker is not here.

🤔Q: Tell me where the saltshaker is located.

User: cabinet 2

Act 4: go to cabinet 2

Uncertainty 4: 0.0

Obs 4: The cabinet 2 is closed.

Act 5: open cabinet 2

Uncertainty 5: 0.0

Obs 5: You open the cabinet 2. The cabinet 2 is open. In it, you see a plate 1, a saltshaker 1, and a soapbottle 1.

Act 6: take saltshaker 1 from cabinet 2

Uncertainty 6: 0.0

Obs 6: You pick up the saltshaker 1 from the cabinet 2.

Act 7: go to drawer 1

Uncertainty 7: 0.0

Obs 7: The drawer 1 is closed.

Act 8: open drawer 1

Uncertainty 8: 0.0

Obs 8: You open the drawer 1. The drawer 1 is open. In it, you see a fork 1, and a peppershaker 1.

Act 9: put saltshaker 1 in/on drawer 1

Uncertainty 9: 0.0

Obs 9: You put the saltshaker 1 in/on the drawer 1.

=====

Pred label: ambiguous

In the infeasible scenario, the uncertainty becomes high in the beginning, and tells the user that the command is infeasible.

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: do laundry

Act 1: go to cabinet 1

Uncertiany 1: 1.538

🤔F: No.

=====

Pred label: infeasible

#### D. SaGC Data collections

The template to generate the dataset is as follows.

I am a [x] robot. Your possible action set is [y]. Here are the scene and the examples of the goal that [x] robot can do.

objects = [objects here]

floorplan = [floorplan here]

people = [people here]

Examples here

(Clear goal) create the high-level clear goal similar to the examples

(Ambiguous goal) create the creative goal that lacks the information to do the task but still can do.

(Infeasible goal) create the simple goal that only [x] robot cannot do

An example prompt for cooking the robot in scene nine is as follows.

I am a cooking robot. Your possible action is grab, heat and plate. Here are the scene and the examples of the goal that robot can do.

objects = [coffee, table, bread, desk, buns, water, bacon, pan, banana, apple]

floorplan = [kitchen, office, meeting room, gym]

people = [person wearing gray shirt", person wearing white shirt, person wearing black shirt, person wearing brown shirt]  
make breakfast consisted of bacon, bread, and coffee

I just worked out help me

cook a meal with apple and bread

(Clear goal) create the high-level clear goal similar to the examples

(Ambiguous goal) create the creative goal that lacks the information to do the task but still can do.

(Infeasible goal) create the simple goal that only cooking robot cannot do

The annotator is a large language model, and four validators were not blind to the research question. For annotation, we crafted 105 samples and then utilized gpt-3.5-turbo to generate the 5226 pairs of commands and the corresponding label. Utilizing LLMs to construct datasets is efficient yet powerful, which has been widely used in prior studies [3], [4]. The four validators were then asked to validate the corresponding generated pairs. The validators were given three options: discard the sample, change the label of the sample, or accept the sample. We provided guidelines to the validators, specifying that the term ambiguous pertains to scenarios where multiple answers could be applicable or when ambiguous keywords such as or, any are present. For the infeasible scenario, we instructed that it refers to situations either irrelevant to the type of robot or concerning the presence of an object in the scene. We have asked the validator to discard the sample if they can not decide. We observe that approximately 10% of the label changed after the validation. The dataset may have a bias as we have utilized only gpt-3.5-turbo model to generate all the pairs, while the effect of validators is not significant.

## E. SaGC Data Examples

After validation, the SaGC dataset is composed as follows. Label 0, 1, 2 is certain, ambiguous, and infeasible respectively.

```
"scene": {
    "floorplan": [ "kitchen", "living room", "bedroom" ],
    "objects": [ "water", "bacon", "bread", "pan", "coffee", "table",
        "orange juice", "sasauge", "banana", "apple" ],
    "people": [ "person wearing blue shirt", "person wearing white
        shirt", "person wearing red shirt" ]
},
"goal": "Prepare a meal consisting of bacon, toast and coffee
and serve it on the table.",
"label": 0,
"task": "cooking"
=====
"scene": {
    "floorplan": [ "kitchen", "living room", "bedroom", "bathroom" ],
    "objects": [ "bacon", "bread", "pan", "coffee", "table", "orange
        juice", "sasauge" ],
    "people": [ "person wearing yellow shirt", "person wearing white
        shirt", "person wearing black shirt" ]
},
"goal": "Clean the living room and wipe the table in the
kitchen.",
"label": 0,
"task": "cleaning"
=====
"task": "cleaning"
"scene": {
    "floorplan": [ "kitchen", "living room", "bedroom" ],
    "objects": [ "water", "bacon", "bread", "pan", "coffee", "table",
        "orange juice", "sasauge", "banana", "apple" ],
    "people": [ "person wearing blue shirt", "person wearing white
        shirt", "person wearing red shirt" ]
},
"goal": "Prepare and serve a meal or drink for the user.",
"label": 1,
"task": "cooking"
=====
"scene": {
    "floorplan": [ "kitchen", "living room", "bedroom" ],
    "objects": [ "water", "bacon", "bread", "pan", "coffee", "table",
        "orange juice", "sasauge", "banana", "apple" ],
    "people": [ "person wearing blue shirt", "person wearing white
        shirt", "person wearing red shirt" ]
},
"goal": "Person needs relaxation",
"label": 1,
"task": "massaging"
=====
```

```
"scene": {
    "floorplan": [ "kitchen", "office", "desk for students",
        "workspace for robots" ],
    "objects": [ "coffee", "table", "bread", "desk", "robot", "orange
        juice", "lemon", "salad" ],
    "people": [ "person wearing yellow shirt", "person wearing white
        shirt", "person wearing black shirt", "person wearing blue
        shirt" ]
},
"goal": "Teach a person wearing a yellow shirt how to cook a
meal.",
"label": 2,
"task": "cleaning"
=====
"scene": {
    "floorplan": [ "bedroom", "living room", "kitchen", "bathroom" ],
    "objects": [ "coffee", "table", "bread", "desk", "buns", "water",
        "bacon", "pan", "banana", "apple" ],
    "people": [ "person wearing gray shirt", "person wearing white
        shirt", "person wearing black shirt", "person wearing blue
        shirt" ]
},
"goal": "person wearing black shirt wants a glass of water",
"label": 2,
"task": "massaging"
```

## F. Supervised Learning Baselines

We have conducted additional experiments on the supervised classification model (i.e., BERT [5]), as shown in Table I. First, we trained the BERT with 1000, 500, and 300 train samples and evaluated the rest of the samples of the dataset (a total of 5222 pairs). We observe that with more than 500 samples, the supervised learning method can exceed the proposed method. However, using the supervised learning method can be vulnerable when the configuration of the robot changes. To see the generalizability, we trained the BERT on two different robot types and then tested on the pairs from the test robot type. In this robot-type transfer scenario, the average accuracy was 0.333, which shows the significant gap of 0.367 compared to the proposed method on `text-danvici-003`.

	Method		Acc.
SL	1000-shots		0.926
	500-shots		0.716
	300-shots		0.521
Transfer	Train	Test	Acc.
	Clean + Cook	MAS.	0.455
	Cook + MAS.	CLEAN	0.298
	Clean + MAS.	COOK	0.245
	Average		0.333
Zero-shot	Ours ( <code>text-danvici-003</code> )		0.710

TABLE I: Supervised Learning Baselines

## G. Vision Language Model

We selected vision-language models that excel in the environment. In the pick-and-place simulation setting, con-

structed similarly to SayCan [6], we discovered that using ViLD [7], as done in the original paper, outperforms the use of OWL-ViT [8]. However, the trend reverses in real-world scenarios. Figure 2 provides a comparison, illustrating the distinctions between the two vision-language models.

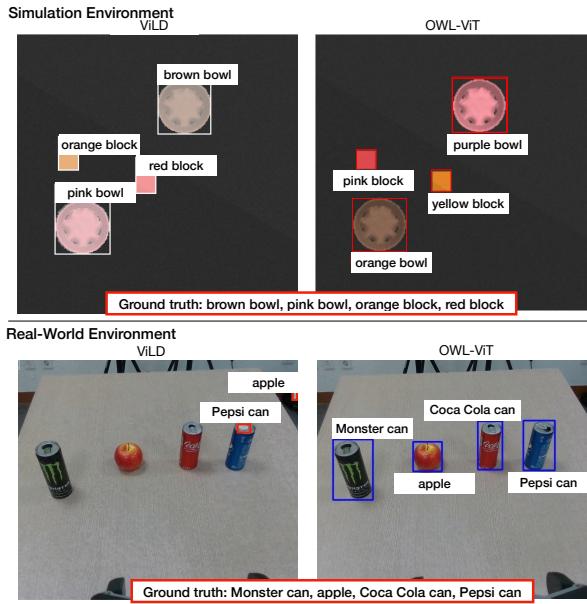


Fig. 2: Comparision in Vision-Language Models

#### H. Pick and Place Simulation

a) *Environment*: We conduct experiments on the tabletop pick and place environment with the Pybullet simulator. Following the previous approach on language model-based robots [9], we utilize ViLD [7] as a vision-language detection model to extract the scene information, a large language model to estimate low-level goals, and Cliprot [10] to transverse textual goals to robotic action. In the interaction phase, we assume that the system can request a question from the user only once. We define an ambiguous task as a goal, not specifying the pick or place object, as shown in Table II.

Categories	Tasks
Clear	pick [x] and put on [x] bowl place all blocks on [x] corner place all blocks on [x] bowl put all blocks on different corners place blocks on matching color place blocks on mismatching color stack all blocks on [x] corner
Ambiguous	pick block that user wants and place on [x] bowl pick [x] block and put on the bowl that the user wants pick the block and put in the bowl stack all blocks

TABLE II: Task explanation. We define an ambiguous task as a goal, not specifying the pick or place object.

b) *F1 Measure*: F1 measure shows the ability of the robot to generate questions only in ambiguous scenarios, as asking for additional information is not necessary for clear and infeasible scenarios. For calculating the F1 score, we

divided the commands into two categories: unambiguous (labeled negative) and ambiguous (labeled positive), as shown in Table III. We have added a statement about the importance of the F1 metric in the revised manuscript.

		Predicted	
		Negative	Positive
Actual	Negative	skip question in unambiguous command	give question in ambiguous command
	Positive	skip question unambiguous command	give question in ambiguous command

TABLE III: Confusion Matrix for calculating F1 score

c) *Detailed Results*: For the details of disambiguation evaluation in Table IV, we report the success rate for both before and after an interaction, with the portion of question generation.

#### I. GPT-Info Evaluation Examples

We assessed the informativeness of the generated questions and explanations with a fine-tuned GPT from TrustfulAI [13]. We measure how many questions and reasons generated are informative with respect to the total number of generations. The results are shown in Table V. We observe that the proposed method with GPT3.5-turbo generates the most informative questions, and the zero-shot based method (Ours, CLAM<sup>†</sup>[14]) had better quality in average than the few-shot based method (IM [9]). We believe that leveraging zero-shot performance in question generation contributes to better quality and variety in the resulting questions, as opposed to a one-shot approach.

```

goal: hide purple block
question: provide more information about the goal of the task.
info: false
=====
goal: put the blocks in their matching bowls
question: specify which bowl the blue block should go in.
info: true
=====
goal: put the blocks on mismatched bowls
question: specify the exact goal you would like the robot to accomplish.
info: false
=====
goal: put the blocks on mismatched bowls
question: provide more information about the task.
info: false
=====
goal: put all the blocks in different corners
question: specify which corner you would like the red block to be placed in.
info: true
=====
```

Model	Method	Clear				Ambiguous			
		Before	After	Gap	Por.	Before	After	gap	Por.
LLAMA [11]	IM [9]	32.4	34.3	2.1	0.05	33.3	38.9	5.6	0.07
	CLAM <sup>†</sup> [12]	25.9	40.7	14.8	0.98	27.8	50.0	22.2	0.93
	Ours	28.7	31.4	2.7	0.39	22.2	29.4	7.2	0.35
GPT3.5	IM [9]	36.1	40.7	4.6	0.46	12.5	61.1	<b>48.9</b>	0.81
	CLAM <sup>†</sup> [12]	30.5	39.8	9.3	0.81	22.2	46.2	24.0	0.81
	Ours	32.4	38.4	6.0	0.32	20.3	44.4	23.9	0.50
InstructGPT	IM [9]	30.6	34.3	3.7	0.88	22.2	48.1	25.9	0.88
	CLAM <sup>†</sup> [12]	30.5	25.5	-5.0	0.37	42.6	13.0	29.6	0.44
	Ours	38.3	39.8	1.5	0.27	16.6	55.6	<u>39.0</u>	0.72

TABLE IV: Success Rate with Portion of question generation

	LLAMA [11]	GPT3.5	InstructGPT
IM [9]	0.70	0.72	0.79
CLAM <sup>†</sup>	0.37	0.83	0.82
Ours	0.39	0.93	0.89

TABLE V: Informativeness of generated questions and reasons

goal: put a blue block in a bowl that user wants.  
 question: specify which bowl you would like the blue block to be placed in.  
 info: true  
 =====

Examples of the evaluation by GPT-info from Truthful QA [13] in are shown Box A.

### J. Prompts

a) SaGC Dataset: Prompts for cooking robots are as follows.

I am a cooking robot. Your possible action is grab, heat and plate.  
 objects: [bread, pan, table, water]  
 goal: cook me toast  
 robot action: robot.grab(bread)  
 robot action: robot.heat(bread)  
 robot action: robot.plate(bread)  
 robot action: robot.done()  
 objects: [table, watter, coffemachine]  
 goal: give a person wearing black shirt a coffee  
 robot action: robot.grab(coffee)  
 robot action: robot.pour(coffee)  
 robot action: robot.done()  
 objects: [bread, pan, bacon, watter, table]  
 goal: give me a sandwich  
 robot action: robot.grab(bread)  
 robot action: robot.heat(bread)  
 robot action: robot.grab(bacon)  
 robot action: robot.heat(bacon)  
 robot action: robot.plate(bread)  
 robot action: robot.plate(bacon)  
 robot action: robot.done()

objects: [water, table, pan, bread]  
 goal: I am thirsty  
 robot action: robot.grab(water)  
 robot action: robot.pour(water)  
 robot action: robot.done()  
 From this, predict the next action by considering the role of the robot and the ambiguity of the goal  
 objects: [Objects here]  
 goals: Goal Here

Prompts for cleaning robots are as follows.

I am a cleaning robot. Your possible action is clean and wipe.  
 scene: [kitchen, living room, bedroom, bathroom]  
 objects: [table, pan, bread]  
 goal: clean the kitchen  
 robot action: robot.clean(kitchen)  
 robot action: robot.done()

scene: [kitchen, living room, bedroom]  
 objects: [table, pan, bread]  
 goal: clean bedroom  
 robot action: robot.clean(bedroom)  
 robot action: robot.done()

scene: [kitchen, living room, bedroom]  
 objects: [table, pan, bread]  
 goal: clean table  
 robot action: robot.wipe(table)  
 robot action: robot.done()

scene: [kitchen, living room, bedroom]  
 objects: [table, pan, desk, bed]  
 goal: clean desk  
 robot action: robot.wipe(desk)  
 robot action: robot.done()

From this, predict the next action by considering the role of the robot and the ambiguity of the goal

scene: [Scene here]  
 objects: [Objects here]  
 goals: Goal Here

Prompts for massage robots are as follows.

I am a massage robot. Your possible action is go\_to and give\_massage.

people: [person wearing black shirt, person wearing yellow shirt, person wearing red shirt]

goal: Give a massage to a person wearing black shirt

robot action: robot.go\_to(person wearing black shirt)

robot action: robot.give\_massage(person wearing black shirt)

robot action: robot.done()

people: [person wearing green shirt, person wearing green shirt, person wearing blue shirt]

goal: Give a massage to a person wearing green shirt

robot action: robot.go\_to(person wearing green shirt)

robot action: robot.give\_massage(person wearing green shirt)

robot action: robot.done()

people: [person wearing green shirt, person wearing green shirt, person wearing blue shirt]

goal: A person wearing green shirt needs a massage

robot action: robot.go\_to(person wearing green shirt)

robot action: robot.give\_massage(person wearing green shirt)

robot action: robot.done()

people: [person wearing black shirt, person wearing yellow shirt, person wearing red shirt]

goal: A person wearing yellow shirt needs a massage

robot action: robot.go\_to(person wearing yellow shirt)

robot action: robot.give\_massage(person wearing yellow shirt)

robot action: robot.done()

From this, predict the next action by considering the role of the robot and the ambiguity of the goal

people: [people here]

goals: Goal Here

Prompts for classification and disambiguation is as follows. This is appended to the previous prompts.

Question: I am a [robot type here] robot. Considering the action set, Can I [goal here]?

Answer: 😊Generated Feasibility

Robot thought: this code is uncertain because 😊Generated Reason

Robot thought: what can I ask to user?

Question: Please 😊Generated Question

Answer: User's feedback

robot thought: continue the previous task based on the question and answer

b) *Pick and Place*: The prompts for the pick and place environment are as follows.

task: move all the blocks to the top left corner.

scene: objects = [red block, yellow block, blue block, green bowl]

robot action: robot.pick\_and\_place(blue block, top left corner)

robot action: robot.pick\_and\_place(red block, top left corner)

robot action: robot.pick\_and\_place(yellow block, top left corner)

robot action: done()

task: put the yellow one the green thing.

scene: objects = [red block, yellow block, blue block, green bowl]

robot action: robot.pick\_and\_place(yellow block, green bowl)

robot action: done()

task: move the light colored block to the middle.

scene: objects = [yellow block, blue block, red block]

robot action: robot.pick\_and\_place(yellow block, middle)

robot action: done()

task: stack all the blocks.

scene: objects = [blue block, green bowl, red block, yellow bowl, green block]

robot action: robot.pick\_and\_place(green block, blue block)

robot action: robot.pick\_and\_place(red block, green block)

robot action: done()

task: group the blue objects together.

scene: objects = [red block, blue block, green bowl, blue bowl, yellow block, green block]

robot action: robot.pick\_and\_place(blue block, blue bowl)

robot action: done()

task: put all blocks in the green bowl.

scene: objects = [red block, blue block, green bowl, blue bowl, yellow block]

robot action: robot.pick\_and\_place(red block, green bowl)

robot action: robot.pick\_and\_place(blue block, green bowl)

robot action: robot.pick\_and\_place(yellow block, green bowl)

robot action: done()

task: considering the ambiguity of the goal, **Goal Here** where the place object is not always dependent on the selected pick object

scene: objects = [Detection Results]

Prompts for disambiguation are as follows. This is appended to the previous prompts.

Robot thought: this code is uncertain because 😊Generated Reason

Robot thought: what can I ask to user?

Question: Please 😊Generated Question

Answer: User's feedback

robot thought: continue the previous task based on the question and answer

c) *Real-world Environment*: The few-shot contexts are as follows.

I am a robot that can pick and give objects to people.

task: give a person wearing a blue shirt a pepsi can.

scene: objects = [pepsi can, coca cola can, apple, banana]

scene: people = [person wearing blue shirt, person wearing black shirt]

robot action: robot.pick\_and\_give(pepsi can, person wearing blue shirt)

robot action: done()

task: give a person wearing light colored shirt a monster can.  
 scene: objects = [pepsi can, coca cola can, monster can, lime]  
 scene: people = [person wearing yellow shirt, person wearing black shirt]  
 robot action: robot.pick\_and\_give(monster can, person wearing yellow shirt)  
 robot action: done()

task: give a person wearing a white shirt a coca cola can.  
 scene: objects = [starbucks can, coca cola can, monster can, lime]  
 scene: people = [person wearing red shirt, person wearing white shirt]  
 robot action: robot.pick\_and\_give(coca cola can, person wearing white shirt)  
 robot action: done()

task: give a person wearing black shirt a monster can.  
 scene: objects = [pepsi can, coca cola can, starbucks can, lime, apple]  
 scene: people = [person wearing green shirt, person wearing black shirt]  
 robot action: robot.pick\_and\_give(monster can, person wearing black shirt)  
 robot action: done()

task: considering the ambiguity of the goal, **goal here**  
 scene: objects = [**objects here**]  
 scene: people = [**people here**]

The prompts used for classification and disambiguation are as follows. This is appended to the previous prompts.

robot thought: I am a robot that can only give the object to humans. Can I **goal here**?  
 robot thought: **Generated Feasibility**  
 Robot thought: this code is uncertain because **Generated Reason**  
 Robot thought: what can I ask to user?  
 Question: Please **Generated Question**  
 Answer: **User's feedback**  
 robot thought: continue the previous task based on the question and answer

## K. Fewshot Prompts

a) *Pick and place Environment*: Fewshot prompts with Inner Monologue style [9].

task: move all the blocks to the top left corner.  
 scene: objects = [red block, yellow block, blue block, green bowl]  
 robot action: robot.pick\_and\_place(blue block, top left corner)  
 robot action: robot.pick\_and\_place(red block, top left corner)  
 robot action: robot.pick\_and\_place(yellow block, top left corner)  
 robot action: done()

task: put the yellow one the green thing.  
 scene: objects = [red block, yellow block, blue block, green bowl]  
 robot action: robot.pick\_and\_place(yellow block, green bowl)  
 robot action: done()

task: move the light colored block to the middle.  
 scene: objects = [yellow block, blue block, red block]  
 robot action: robot.pick\_and\_place(yellow block, middle)  
 robot action: done()

task: stack all the blocks.  
 scene: objects = [blue block, green bowl, red block, yellow bowl, green block]  
 Robot action: robot.pick\_and\_place(green block, blue block)  
 Robot thought: This code is uncertain because the robot don't know which block to pick up first  
 Robot thought: What can I ask to the user?  
 Question: which block should I pick up first?  
 Answer: green block  
 robot action: robot.pick\_and\_place(green block, blue block)  
 robot action: robot.pick\_and\_place(red block, green block)  
 done()

task: group the blue objects together.  
 scene: objects = [red block, blue block, green bowl, blue bowl, yellow block, green block]  
 robot action: robot.pick\_and\_place(blue block, blue bowl)  
 robot action: done()

task: put all blocks in the green bowl.  
 scene: objects = [red block, blue block, green bowl, blue bowl, yellow block]  
 robot action: robot.pick\_and\_place(red block, green bowl)  
 robot action: robot.pick\_and\_place(blue block, green bowl)  
 robot action: robot.pick\_and\_place(yellow block, green bowl)  
 robot action: done()

b) *SaGC Dataset*: Prompts for cooking robots are as follows.

I am a cooking robot. Your possible action is grab, heat and plate.  
 objects: [bread, pan, table, water]  
 goal: cook me toast  
 robot action: robot.grab(bread)  
 robot action: robot.heat(bread)  
 robot action: robot.plate(bread)  
 robot action: robot.done()

objects: [table, watter, coffemachine]  
 goal: give a person wearing black shirt a coffee  
 robot action: robot.grab(coffee)  
 robot action: robot.pour(coffee)  
 robot action: robot.done()

objects: [bread, pan, bacon, watter, table]  
 goal: give me a sandwich  
 robot action: robot.grab(bread)  
 robot action: robot.heat(bread)  
 robot action: robot.grab(bacon)  
 robot action: robot.heat(bacon)  
 robot action: robot.plate(bread)

robot action: robot.plate(bacon)  
robot action: robot.done()

objects: [water, table, pan, bread, 'coffee]

goal: I am thirsty

robot thought: I am not sure what to do grab first

question: What should I do first?

answer: grab water

robot action: robot.grab(water)

robot action: robot.pour(water)

robot action: robot.done()

objects: [water, table, pan, bread, coffee]

goal: wipe here

robot thought: I can not do this task

question: provide more information about what to do

answer: I can not do this task

robot action: robot.done()

From this, predict the next action by considering the role of the robot and the ambiguity of the goal

objects: [Objects here]

goals: Goal Here

From this, predict the next action by considering the role of the robot and the ambiguity of the goal

scene: [Scene here]

objects: [Objects here]

goals: Goal Here

Prompts for massage robots are as follows.

I am a massage robot. Your possible action is go\_to and give\_massage.

people: [person wearing black shirt, person wearing yellow shirt, person wearing red shirt]

goal: Give a massage to a person wearing black shirt

robot action: robot.go\_to(person wearing black shirt)

robot action: robot.give\_massage(person wearing black shirt)

robot action: robot.done()

people: [person wearing green shirt, person wearing green shirt, person wearing blue shirt]

goal: Give a massage to a person wearing green shirt

robot action: robot.go\_to(person wearing green shirt)

robot action: robot.give\_massage(person wearing green shirt)

robot action: robot.done()

people: [person wearing green shirt, person wearing green shirt, person wearing blue shirt]

goal: A person wearing green shirt needs a massage

robot action: robot.go\_to(person wearing green shirt)

robot action: robot.give\_massage(person wearing green shirt)

robot action: robot.done()

people: [person wearing black shirt, person wearing yellow shirt, person wearing red shirt]

goal: Someone needs a massage

robot thought: this is uncertain because I am not sure who needs a massage

question: who needs a massage

answer: person wearing yellow shirt

robot action: robot.go\_to(person wearing yellow shirt)

robot action: robot.give\_massage(person wearing yellow shirt)

robot action: robot.done()

people: [person wearing black shirt, person wearing yellow shirt, person wearing red shirt]

goal: heat bread

robot thought: I can not do this task

question: provide more information about what to do

answer: I can not do this task

robot action: robot.done()

From this, predict the next action by considering the role of the robot and the ambiguity of the goal

people: [people here]

goals: Goal Here

scene: [kitchen, living room, bedroom]

objects: [table, pan, bread]

goal: clean bedroom

robot action: robot.clean(bedroom)

robot action: robot.done()

scene: [kitchen, living room, bedroom]

objects: [table, pan, bread]

goal: clean table

robot action: robot.wipe(table)

robot action: robot.done()

scene: [kitchen, living room, bedroom]

objects: [table, pan, desk, bed]

goal: clean here

robot thought: this is uncertain because I am not sure what is here

question: what is here

answer: desk

robot action: robot.wipe(desk)

robot action: robot.done()

objects: [water, table, pan, bread, coffee]

goal: heat bread

robot thought: I can not do this task

question: provide more information about what to do

answer: I can not do this task

robot action: robot.done()

c) Real-world Environment: The prompts used are as follows.

I am a robot that can pick and give objects to people.

task: give a person wearing a blue shirt a pepsi can.  
 scene: objects = [pepsi can, coca cola can, apple, banana]  
 scene: people = [person wearing blue shirt, person wearing black shirt]  
 robot action: robot.pick\_and\_give(pepsi can, person wearing blue shirt)  
 robot action: done()

task: give a person wearing light colored shirt a monster can.  
 scene: objects = [pepsi can, coca cola can, monster can, lime]  
 scene: people = [person wearing yellow shirt, person wearing black shirt]  
 robot action: robot.pick\_and\_give(monster can, person wearing yellow shirt)  
 robot action: done()

task: give a person wearing a white shirt a coca cola can.  
 scene: objects = [starbucks can, coca cola can, monster can, lime]  
 scene: people = [person wearing red shirt, person wearing white shirt]  
 robot action: robot.pick\_and\_give(coca cola can, person wearing white shirt)  
 robot action: done()

task: give a person wearing light colored shirt a monster can.  
 scene: objects = [pepsi can, coca cola can, monser can, lime]  
 scene: people = [person wearing yellow shirt, person wearing white shirt]  
 robot thought: This is code is uncertain because I don't know whom to give.  
 robot thought: What can I ask to the user?  
 question: Which person do you want to give monster can?  
 answer: person wearing yellow shirt  
 robot action: robot.pick\_and\_give(monster can, person wearing yellow shirt)  
 robot action: done()

task: person yellow shirts wants a handshakes.  
 scene: objects = [pepsi can, coca cola can, monser can, lime]  
 scene: people = [person wearing yellow shirt, person wearing black shirt]  
 robot thought: I can not do this task.  
 robot action: done()

task: considering the ambiguity of the goal, [goal here](#)  
 scene: objects = [[objects here](#)] scene: people = [[people here](#)]

## L. Real-world Experiments Environment and Demonstrations

a) *Environments:* The setup consists of a UR5e robot arm equipped with a wrist-mounted OpenCV OAKD camera overlooking a workspace of the tabletop. We run our method via Robot Operating System (ROS) and the rate of publishing the joint position is set at 500HZ which is a default setting for a UR5e manipulator. In the table scene, we randomly place objects on the table and shuffle them throughout the experiments. The setup is shown in Figure 4.

We investigate three tasks considering goal information: (i) Certain cases involve the explicit provision of information regarding the location or name of the target object. (ii) Am-

biguous cases, in contrast, are characterized by the absence of detailed information about the target object. (iii) infeasible cases arise when objects or actions are encountered that have not been previously observed within the given context prompts. The goals are illustrated in Table VI.

Categories	Tasks
Clear	give [x] to [y]
Ambiguous	give [x] to someone give something to drink to [y] give something to drink to someone
Infeasible	Wipe the desk Smash the [x] put [x] on the ground

TABLE VI: Task explanation for the real-world experiment.

b) *User Guidelines:* A high-level goal was predefined during the experiment, as shown in Table V in the manuscript. There were four participants in each scene, and they were asked to answer the question from the robot during the disambiguation process. The instructions given to the participants are as follows:

Given the goal, the robot may ask you some questions if it is uncertain about what to do. If the robot asks for the information, please answer it via the interface and as detailed as possible, letting the robot be certain about the task.

An example of the response is as follows:

Scene: [coca cola can, pepsi can, apple, lemon]  
 Q. Please specify what kind of drink you would like.  
 A. Coca cola can

c) *Failure Cases:* The failure can arise due to both VLM and LLM. We have shown the failure cases in Figure 3. On the vision failure part, the VLM fails to detect the Red Bull can, leading to predict "give red bull can to a person wearing green shirt" as in feasible command. In the LLM Failure part, LLM failed to understand that coffee with sleeve is infeasible in the scene and picks the Starbucks can instead.

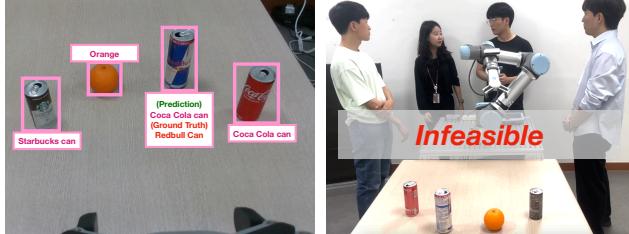
d) *Detailed Results:* The detailed results on disambiguation in the real world are shown in Table VII. We report the success rate of both before and after the interaction, along with a portion of question generation for each label.

The snapshots for real-world demonstrations are shown in Figure 6 and 7.

## M. Additional Results on SaGC Dataset

The dataset contains partially feasible goals, such as "bake buns and repair the desk" for a cooking robot, "cook breakfast and clean the kitchen for a cleaning room, and "make coffee and give a massage to the person wearing a gray shirt" for massage robot. For additional 216 data, we have validated the proposed method and observed that the proposed method on text-davinci-003 model has the 0.87 accuracy in those samples. Although the large-language model has hallucination issues, ensuring

## Vision Failure



- Q** Object: ['right coca cola can', 'left coca cola can', 'starbucks can', 'orange']
- Q** Person: ['person wearing blue shirt', 'right person wearing black shirt', 'left person wearing black shirt', 'person wearing green shirt']
- Q** give **redbull** can to a person wearing green shirt
- Q** Robot action: robot.pick\_and\_give(redbull can, person wearing green shirt) **Uncertainty:** 0.54
- Q** F: Given the current scene, there is no redbull can available. Therefore, I cannot give a redbull can to a person wearing a green shirt.
- Q** Failed!

## Clear Goal Predicted as Infeasible



## Large Language Model Failure

- Q** Object: ['apple', 'coca cola can', 'monster can', 'starbucks can']
- Q** Person: ['left person wearing black shirt', 'right person wearing black shirt', 'person wearing blue shirt', 'person wearing green shirt']
- Q** Give a **coffee with a sleeve** to a person wearing blue shirt
- Q** Robot action: robot.pick\_and\_give(starbucks can with sleeve, person wearing blue shirt) **Uncertainty:** 1.16
- F** F: Yes, if given more information about the specific object and person, I can determine if there is a starbucks can with a sleeve and a person wearing a blue shirt in the scene.. R: there might be multiple people wearing a blue shirt, and it's unclear which specific person the robot should give the can to. Q: Can you provide more specific details or distinguishing characteristics of the person wearing the blue shirt?
- Q** there is just one person who is wearing a blue shirt
- Q** Robot action: robot.pick\_and\_give(starbucks can with sleeve, person wearing blue shirt) **Uncertainty:** 0.0
- Q** Failed!

## Infeasible Goal Predicted as Ambiguous

Fig. 3: Failure Cases in Real-world Environment

Model	Method	Before	After	Gap	C. Por.	A. Por.	I. Por.
GPT3.5	IM [9]	0.38	0.55	0.17	0.83	0.83	0.17
	CLAM <sup>†</sup> [12]	0.33	0.61	0.28	0.66	1.0	0.33
	Ours	0.22	0.55	0.33	0.33	1.0	0.33
InstructGPT	IM [9]	0.27	0.44	0.17	0.33	0.66	0.0
	CLAM <sup>†</sup> [12]	0.33	0.39	0.06	0.33	0.33	0.16
	Ours	0.38	0.67	0.28	0.17	0.83	0.0

TABLE VII: Success Rates on real-world environment. C. Por., A. Por., I. Por. denotes a portion of question generation on clear, ambiguous, and infeasible commands respectively.

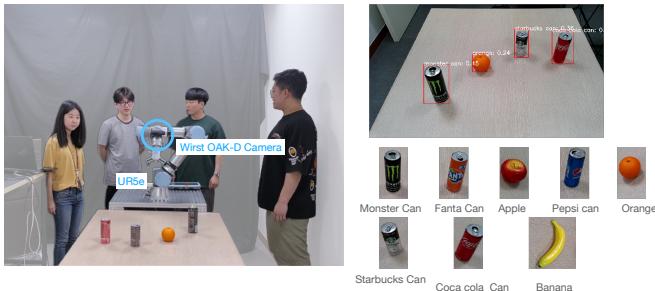


Fig. 4: Real-world Experiment Setting

the model to be aware of their capabilities and roles helps in reducing those issues. The generated outputs in such samples are illustrated in Figure 5, with the failure case.

## REFERENCES

- [1] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021.
- [2] Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.
- [3] Ehsan Kamalloo, Aref Jafari, Xinyu Zhang, Nandan Thakur, and Jimmy Lin. HAGRID: A human-llm collaborative dataset for generative information-seeking with attribution. *arXiv:2307.16883*, 2023.
- [4] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv preprint arXiv:2301.01820*, 2023.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [6] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding

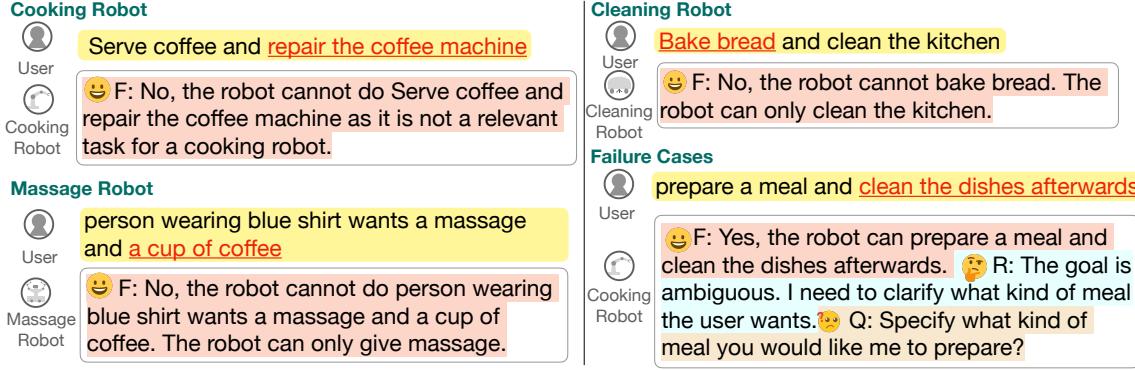
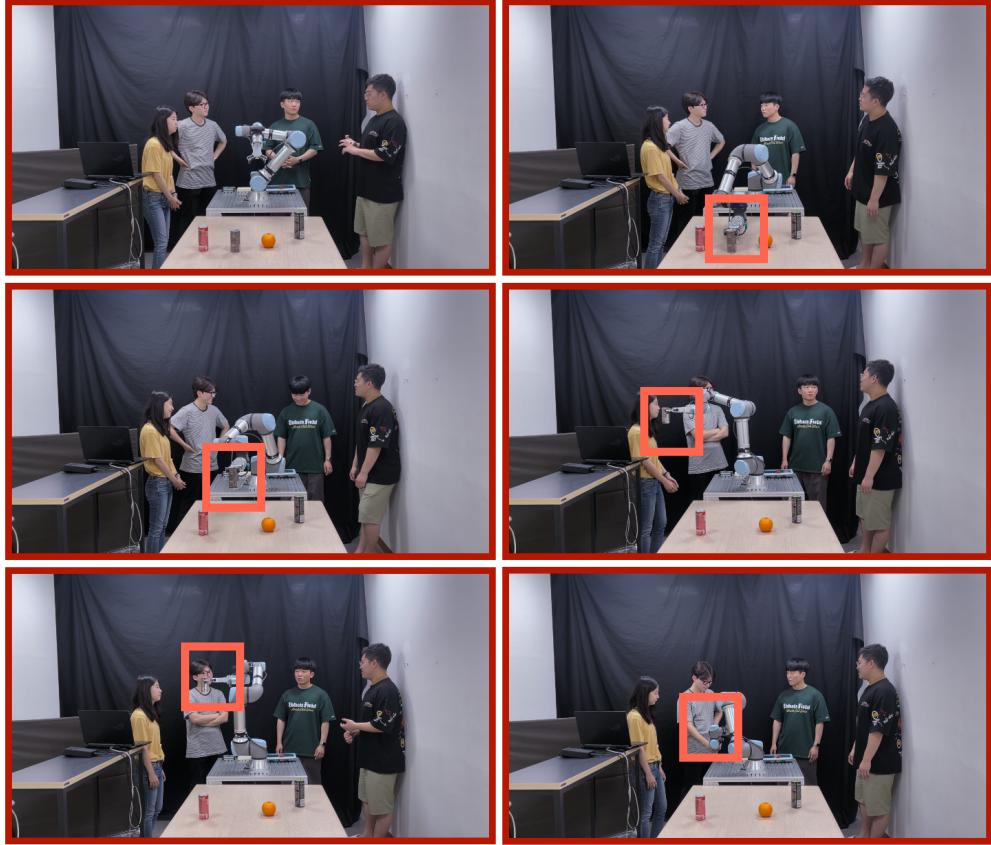


Fig. 5: Generated output on the additional samples

language in robotic affordances. In *Proc. of the 6th Annual Conference on Robot Learning (CoRL)*, 2022.

- [7] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2021.
- [8] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection with vision transformers. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2022.
- [9] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Proc. of the 6th Annual Conference on Robot Learning (CoRL)*, 2022.
- [10] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliporl: What and where pathways for robotic manipulation. In *Proc. of the Conference on Robot Learning (CoRL)*, pages 894–906. PMLR, 2022.
- [11] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. 2023.
- [12] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with large language models. *arXiv preprint arXiv:2212.07769*, 2022.
- [13] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2022.
- [14] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2023.

## Ambiguous 1



## Clear

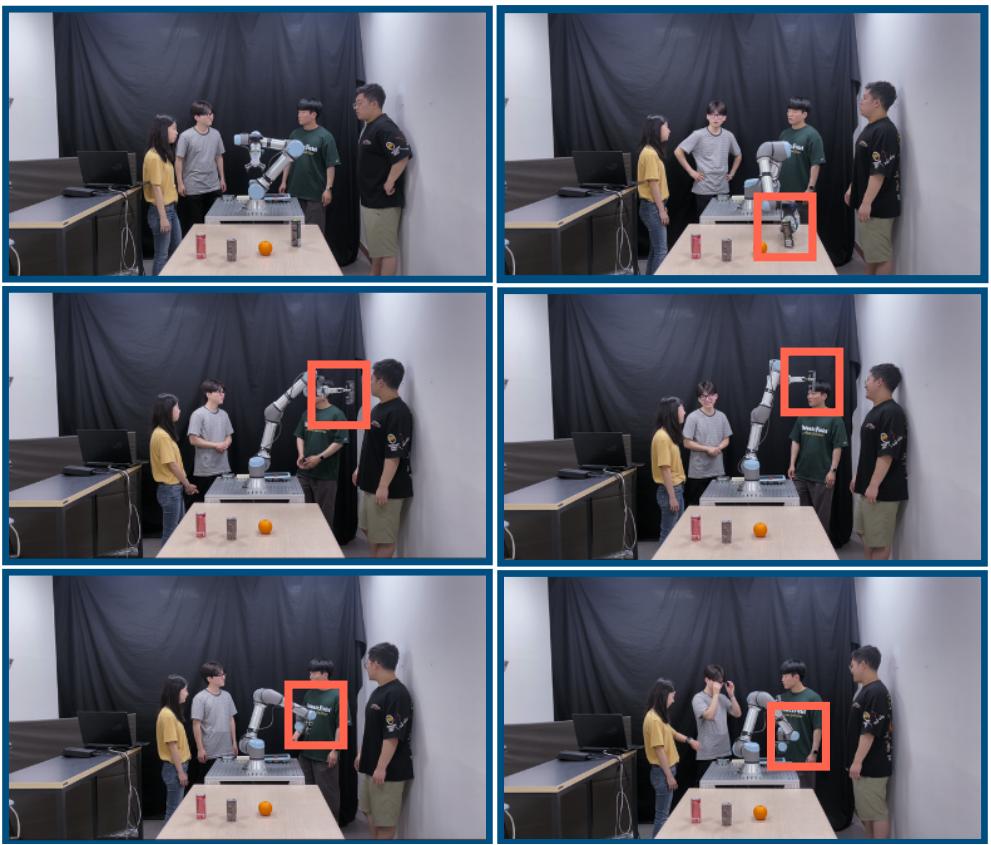
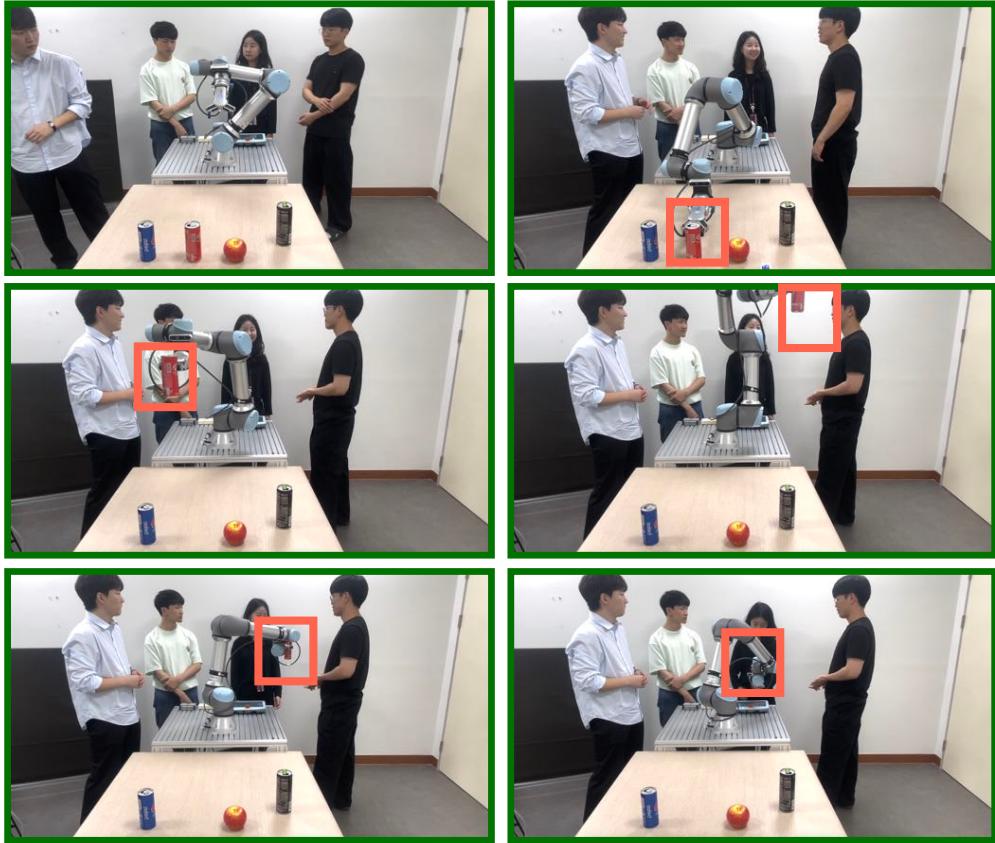


Fig. 6: Snapshot1

## Ambiguous 2



## Infeasible

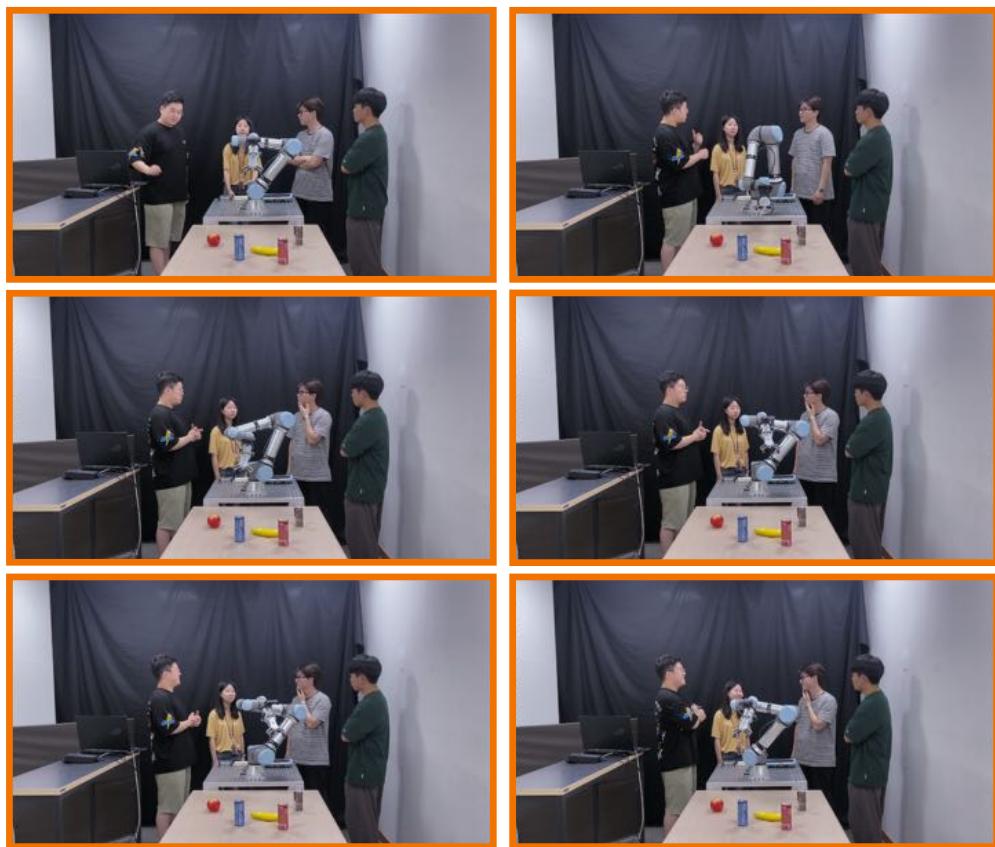


Fig. 7: Snapshot2