The World Is Worth How Many APIs? A Thought Experiment

Jiefu Ou, Arda Uzunoğlu, Benjamin Van Durme, Daniel Khashabi
{jou6, auzunog1, vandurme, danielk}@jhu.edu
Center for Language and Speech Processing
Johns Hopkins University

Introduction

Research Question: To build a versatile embodied agent that can carry out daily tasks in the physical world, how many primitive actions (APIs) should such an agent be equipped with, and what do they look like?

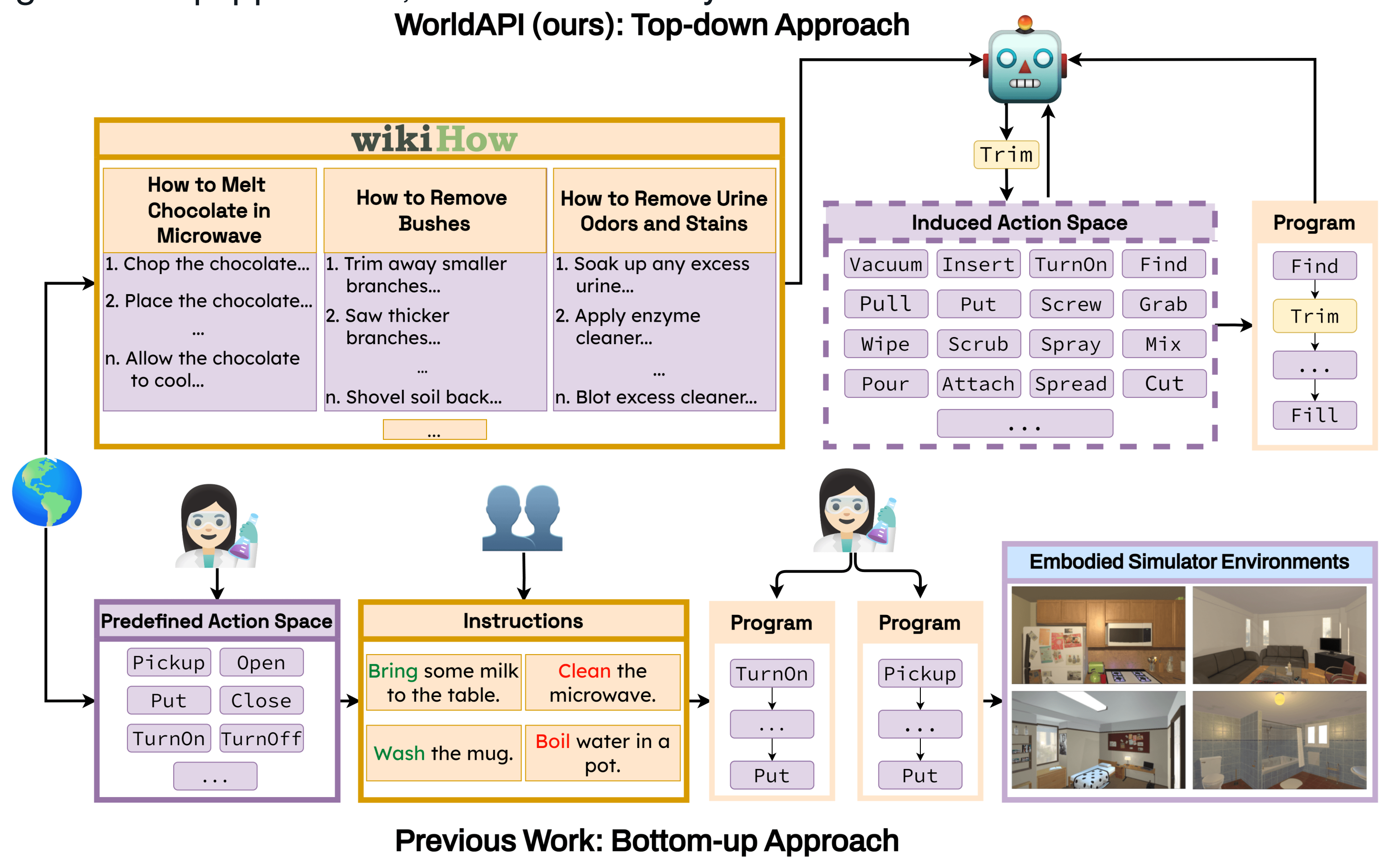

WorldAPI (ours): Top-down Approach
wikiHow
How to Melt Chocolate in Microwave
How to Remove Bushes
How to Remove Urine Odors and Stains
Trim
Induced Action Space
Vacuum Insert TurnOn Find Pull Put Screw Grab Wipe Scrub Spray Mix Pour Attach Spread Cut
Program
Find Trim Fill
Predefined Action Space
Pickup Open Put Close TurnOn TurnOff
Instructions
Bring some milk to the table.
Clean the microwave.
Wash the mug.
Boil water in a pot.
Embodied Simulator Environments
Previous Work: Bottom-up Approach

Figure 1: **Top**: WorldAPIs, the proposed thought experiment that takes a top-down approach. Starting from daily tasks with sequences of instruction steps in wikiHow, and a seed action space (API pool), we iteratively prompt LLMs to generate agent programs and add the induced (hallucinated) APIs in generated programs to the API pool. **Bottom**: in contrast, most of the prior work in building embodied environments often adopts a bottom-up approach. The simulation and collection of instructions and programs are all based on a close set of predefined actions.

WorldAPI: Defining a Hypothetical World

Our goal is to formulate simulations that allow us to approximate the action space of versatile robots physical world:

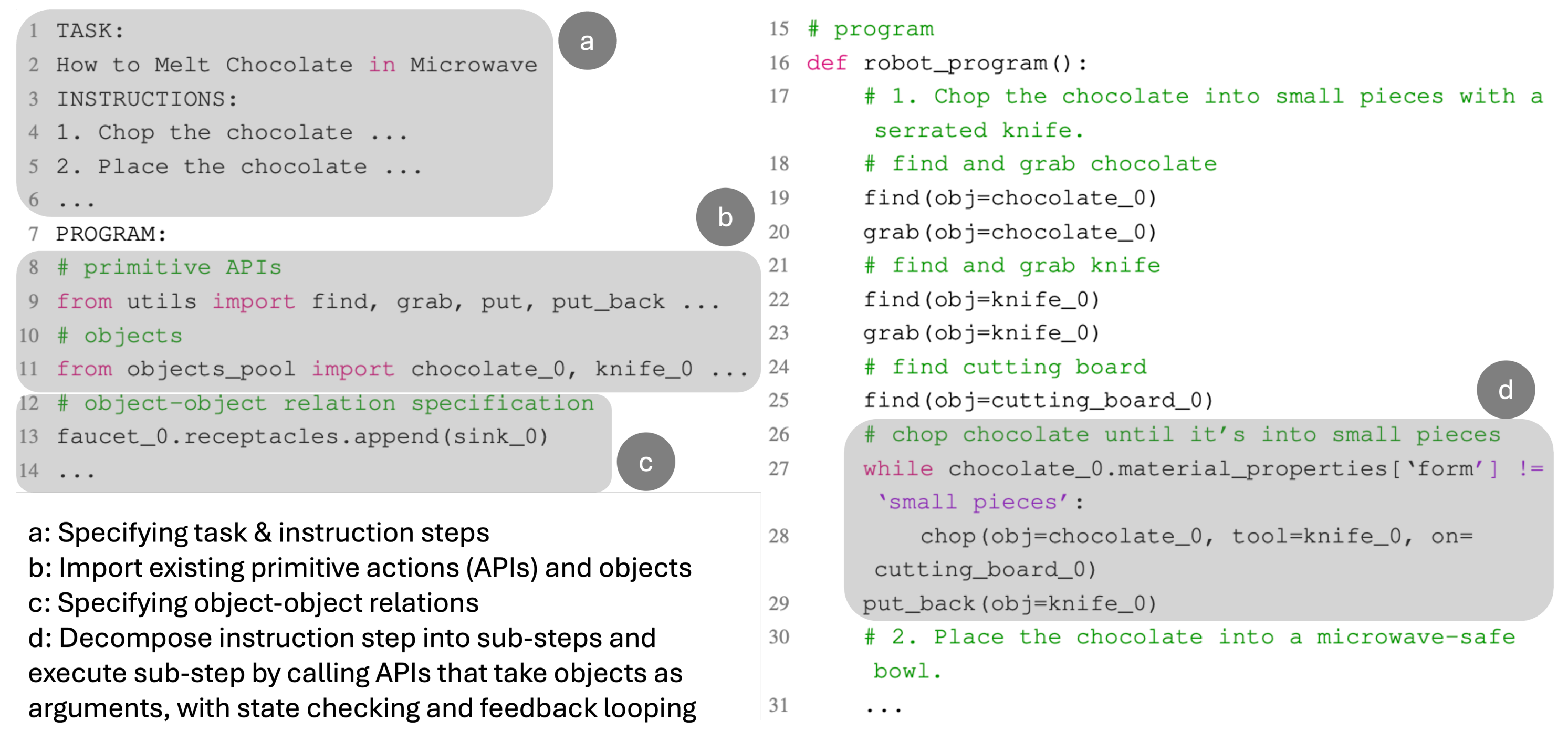
1. Collect diverse and realistic instructions from online resources
2. Define hypothetical environment and agent that are capable of carrying out these instructions
3. Induce agent programs and action spaces jointly

Data

We leverage wikiHow, a prominent web platform with 200K+ professionally curated "how-to" tutorials across a diverse set of domains. We follow prior work to use the tutorial title as the goal, the paragraph headline as instruction steps, and the paragraph body as additional descriptions.

Translating Language Instructions to Pythonic Policies and Actions

We jointly induce primitive actions (APIs) and policies (Pythonic programs) via prompting LLMs with few-shot demonstrations. The demonstrations provide information about the hypothetical environment to the LLMs: available objects, primitive actions (APIs), as well as how to interact with objects through API calls and state checking. We create an annotation guideline that defines the semantic formalism of objects and APIs for the hypothetical environment. We annotate the programs for a small set of wikiHow tutorials as seed demonstrations.



```
TASK:
How to Melt Chocolate in Microwave
INSTRUCTIONS:
1. Chop the chocolate ...
2. Place the chocolate ...
...
PROGRAM:
# primitive APIs
from utils import find, grab, put, put_back ...
# objects
from objects_pool import chocolate_0, knife_0 ...
# object-object relation specification
faucet_0.receptacles.append(sink_0)
...
```

```
# program
def robot_program():
    # 1. Chop the chocolate into small pieces with a
     serrated knife.
    # find and grab chocolate
    find(obj=chocolate_0)
    grab(obj=chocolate_0)
    # find and grab knife
    find(obj=knife_0)
    grab(obj=knife_0)
    # find cutting board
    find(obj=cutting_board_0)
    # chop chocolate until it's into small pieces
    while chocolate_0.material_properties['form'] !=
     'small pieces':
        chop(obj=chocolate_0, tool=knife_0, on=
    cutting_board_0)
    put_back(obj=knife_0)
    # 2. Place the chocolate into a microwave-safe
     bowl.
    ...
```

a: Specifying task & instruction steps
b: Import existing primitive actions (APIs) and objects
c: Specifying object-object relations
d: Decompose instruction step into sub-steps and execute sub-step by calling APIs that take objects as arguments, with state checking and feedback looping

Figure 2: Our in-context demonstrations for decomposing wikiHow tasks into API calls.

Inducing the Action/Policy Space in the Hypothetical World

We develop a pipeline for inducing action/policy of wikiHow tutorials via iterative few-shot code generation with LLMs. As depicted in Figure 3, at each step of induction, a random tutorial is sampled from wikiHow. Given the input tutorial, a prompt is constructed with a system instruction, retrieved programs, and API use cases that are used as demonstrations to guide LLM generation. The LLMs process this prompt, after which we verify the syntactic well-formedness of the generated program. If it passes the verification, we add the full program and the extracted API into the pool of demonstrations. This is done iteratively, monotonically expanding the pool of APIs/programs. At each round, LLM leverages the program examples generated by itself in previous steps, essentially bootstrapping from its prior output.

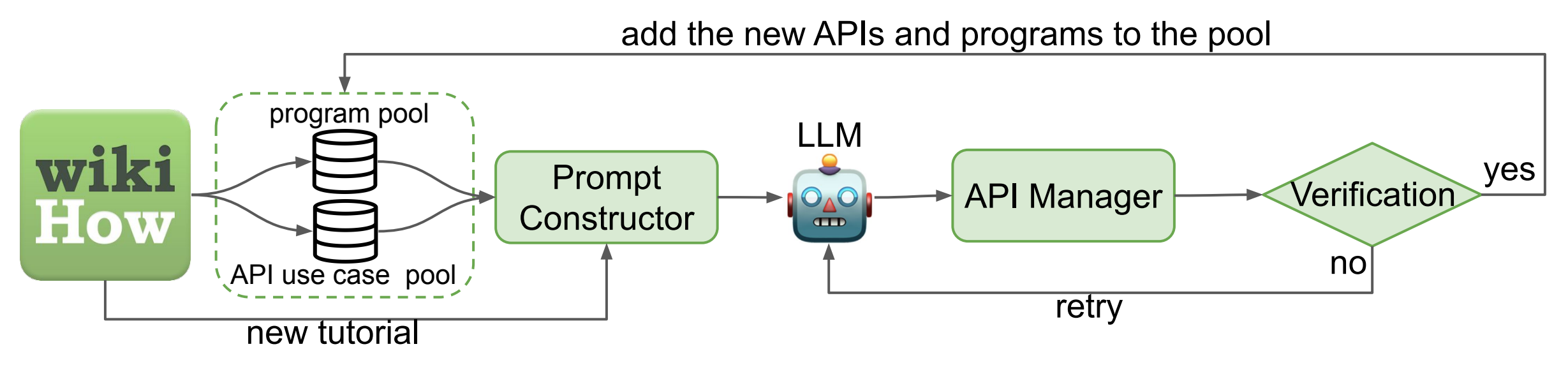


Figure 3: Proposed pipeline that jointly induces new APIs and programs.

Experiment & Evaluation Metrics

We experiment with 3 variants of our pipeline on 1000 sampled wikiHow tutorials:

- The **Base** variant takes in only pairs of (instruction steps, full program) as in-context demonstrations.
- The **Base + Use Case** version that additionally takes in code snippets of API use cases as demonstrations.
- The **Base + Use Case + Description** version that includes API use cases in demonstrations and adds descriptions to each instruction step.

We evaluate the quality of generated APIs with two metrics: For each new API, we quantify its redundancy with a $0-0.5-1$ scale measurement. We approximate the simulator execution-based evaluation of generated programs with faithfulness measurement of $0-0.5-1$ scale.

Results & Analysis

	Redundancy↓			Faithfulness↑		APIs
Induction Pipelines	Score	-Complex	-Complex -Synonym	Score	Ranking	Avg. #
Full (a)	46.50	38.11	35.32	82.0	1.756	2.88
+UseCase	**43.44**	**36.07**	34.43	81.0	1.732	1.24
+UseCase+Desc	47.46	36.59	**33.70**	**84.0**	**1.439**	1.74

Table 1: Human evaluation results on the output of 50 wikiHow tutorials. For redundancy, "Score" is the full score, and "-Complex"/"-Synonyms" refers to rescoring all the new APIs that are too complicated to be further decomposed/synonyms to existing APIs from 0.5 (partially redundant) to 1 (fully use full), respectively. For faithfulness, "Score" is the absolute score, and "Rank" is the preference-based ranking. "Avg. #" of APIs lists the average number of new APIs induced per tutorial.

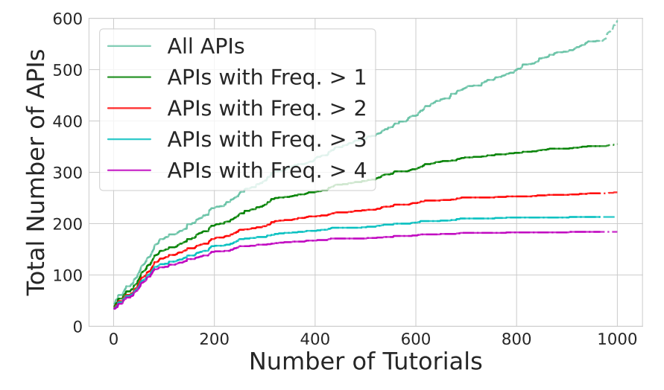


Figure 4: Size of API pool vs. # of tutorials. Lines represent different frequency thresholds used to filter the APIs.

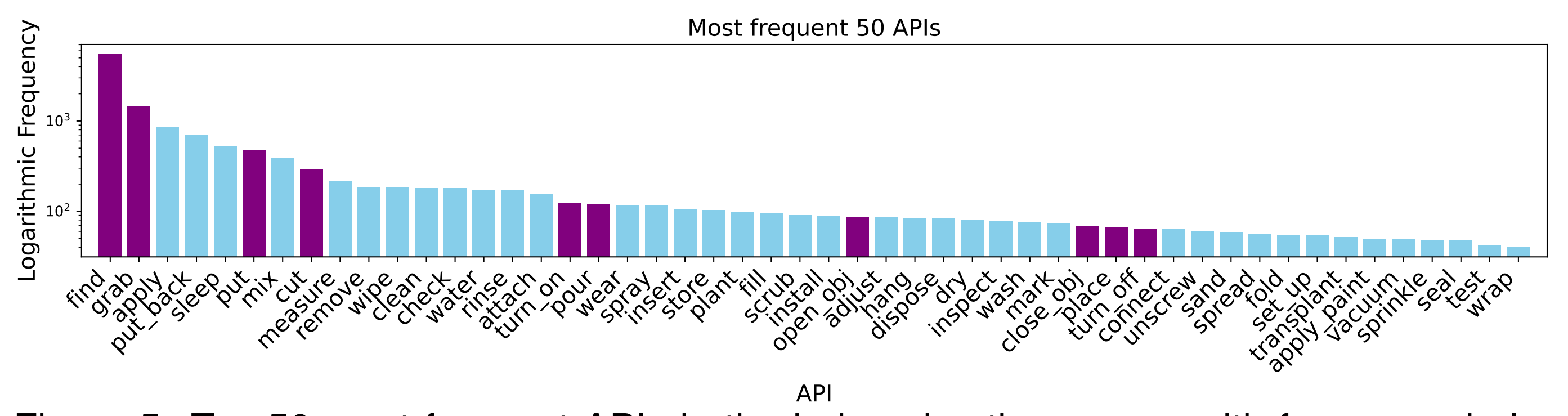


Figure 5: Top-50 most frequent APIs in the induced action space, with frequency in log scale. We use 🟪 to mark the APIs with exact/overlapping affordance to the primitive actions in existing embodied environments (ALFRED and VirtualHome) and use 🟦 to mark APIs that are beyond the action space of exiting environments.