# WORLDAPIS: The World Is Worth How Many APIs? A Thought Experiment

**Jiefu Ou, Arda Uzunoğlu, Benjamin Van Durme, Daniel Khashabi**

Johns Hopkins University
{jou6, auzunog1, vandurme, danielk}@jhu.edu

## Abstract

AI systems make decisions in physical environments through primitive actions or affordances that are accessed via API calls. While deploying AI agents in the real world involves *numerous* high-level actions, existing embodied simulators offer *a limited set* of domain-salient APIs. This naturally brings up the questions: *how many primitive actions (APIs) are needed for a versatile embodied agent, and how should they look like?*

We explore this via a thought experiment: assuming that wikiHow tutorials cover a wide variety of human-written tasks, what is the space of APIs needed to cover these instructions? We propose a framework to iteratively induce new APIs by grounding wikiHow instruction to situated agent policies. Inspired by recent successes in large language models (LLMs) for embodied planning, we propose a few-shot prompting to steer GPT-4 to generate Pythonic programs as agent policies and bootstrap a universe of APIs by 1) reusing a seed set of APIs; and then 2) fabricate new API calls when necessary. The focus of this thought experiment is on defining these APIs rather than their excitability.

We apply the proposed pipeline on instructions from wikiHow tutorials. On a small fraction (0.5%) of tutorials, we induce an action space of 300+ APIs necessary for capturing the rich variety of tasks in the physical world. A detailed automatic and human analysis of the induction output reveals that the proposed pipeline enables effective reuse and creation of APIs. Moreover, a manual review revealed that existing simulators support only a small subset of the induced APIs (9 of the top 50 frequent APIs), motivating the development of action-rich embodied environments.

## 1 Introduction

Developing versatile and capable AI agents that follow natural language instructions for task execution in the real world has been an important research objective. A series of efforts (Ahn et al. 2022; Singh et al. 2022; Song et al. 2023) have been made towards this goal, with a notable upsurge of work on harnessing the power of large language models (LLMs) to generate task plans and policies for embodied agents.

The backbones of such success are benchmarks and simulations of embodied environments. (Anderson et al. 2018; Thomason, Gordon, and Bisk 2019; Puig et al. 2018; Shridhar et al. 2020; Ku et al. 2020). However, most of these existing embodied environments suffer from the limited handcrafted action spaces. Such restraint diminishes action diversity, leading to the oversaturation of a small group of basic actions. For example, as shown in Figure 1, existing embodied simulators/environments often define a small and closed set of actions an agent can perform. As a result, the possible instructions it can carry out are highly constrained by the limited action space. However, carrying out tasks in real-world scenarios across different domains requires a much larger and more diverse action space. To bridge this gap between embodied agent learning and real-world deployment, it is natural to ask the following questions: *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?*

In this work, we make the first step towards answering these questions by proposing WORLDAPIS(§3), a thought experiment that aims to approximate a realistic pool of agent-executable primitive actions (APIs)[1], the composition of which enables agents to accomplish diverse and realistic goals specified by high-level procedural natural language instructions. The thought experiment assumes an embodied agent to be deployed in a hypothetical world consisting of objects and primitive action space with predefined properties. The agent is tasked to perform a wide variety of daily tasks specified through natural language instructions, in the form of wikiHow tutorials. To complete these tasks, the agent needs to evoke and compose primitive actions (APIs) from the action space to interact with objects. As shown in Figure 1, in contrast to prior work, we propose to induce an open action space from diverse and realistic wikiHow procedural instructions.

To create the action space, we propose a pipeline (§4) for bootstrapping the API pool and corresponding agent policies as Pythonic programs that 1) iteratively generate semi-executable agent policies as Pythonic programs via few-shot prompting LLMs with procedural natural language tutori-



[1] In this work we refer to APIs as interfaces to affordances of agents that could be deployed in the real world. Upon calling an API, the agent can carry out a corresponding physical primitive action. We use APIs and primitive actions and APIs interchangeably henceforth

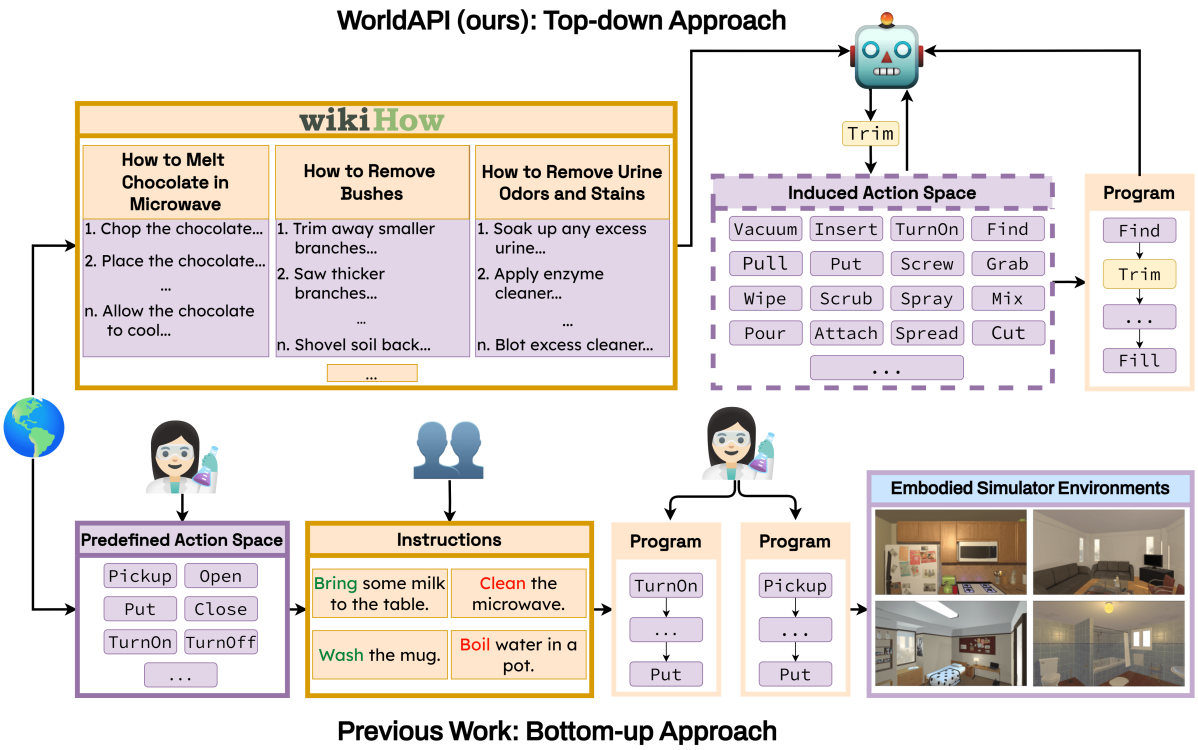


Figure 1: **Top**: WORLDAPIS, the proposed thought experiment that takes a top-down approach. Starting from daily tasks with sequences of instruction steps in wikiHow, and a seed action space (API pool), we iteratively prompt LLMs to generate agent programs and add the induced (hallucinated) APIs in 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.

als. 2) parse generated programs, filter and add full/snippets of programs to the pool of demonstrations. The proposed pipeline steers the LLMs to *1) reuse existing APIs when possible* and *2) hallucinate (induce) new APIs when the instruction cannot be covered with existing APIs* when converting a natural language instruction into a composition of APIs. While hallucination is often regarded as a pitfall of LLMs, in this work we harness it as a constructive capability to synthesize novel APIs that are semantically and stylistically consistent with our pre-defined APIs.

We conduct preliminary experiments (§5) to induce agent policies and action spaces on wikiHow [2], a rich semi-structured online resource of human-written procedural instructions covering daily tasks from a wide spectrum of domains. The set of APIs saturated after running the iterative induction pipeline on less than 0.5% of wikiHow tutorials, yielding a self-contained pool of over 300 frequently-evoked APIs (§6.2). This provides an approximation of the lower bound of the primitive action space. Human and automatic evaluations (§6.1) demonstrate the proposed pipeline's effectiveness in inducing APIs necessary to carry out diverse wikiHow instructions. Further comparison with existing embodied environments (Shridhar et al. 2020; Puig et al. 2018) highlights the limited coverage of our induced APIs in existing environments (§6.2), thus encouraging further efforts to enrich the action space of embodied simulations.

[2]https://www.wikihow.com

## 2 Related Work

**Embodied planning and simulation.** This literature focuses on enabling agents to navigate and manipulate their environments. While LLMs are shown to be effective at planning (Song et al. 2023; Huang et al. 2022, 2023), embodied planning usually requires more than textual data due to its interactive nature. Thus, creating a reflection of the real world (e.g., simulation or benchmarks) has been an important aspect of the progress. These simulators enable interactive learning and evaluation on a proxy of the world.

Various simulators have been developed for simulating human activities (Kolve et al. 2017; Xia et al. 2018; Nasiriany et al. 2024), each with distinct advantages. Yet, they are intrinsically confined to a set of interactions based on factors such as the feasibility of rendering actions, computational limitations, constraints of the physics engine, etc. These benchmarks for embodied reasoning (summarized in Table 1) can be split into two groups: (i) grounded in *physical* world and (ii) grounded in *software* world.

The first group of simulators is concerned with various real-world tasks such as navigation. In most of these works, plan execution resources are mostly limited to household

| Resource | Size of Action Space | Domain |
|---|---|---|
| PHYSICAL WORLD SIMULATORS | | |
| ALFRED (Shridhar et al. 2020) | 13 | Household Tasks |
| VirtualHome (Puig et al. 2018) | 12 | Household Tasks |
| TEACh (Padmakumar et al. 2021) | 16 | Household Tasks |
| LACMA (Yang et al. 2023) | 10 | Household Tasks |
| BEHAVIOR (Srivastava et al. 2021) | 6 | Household Tasks |
| BEHAVIOR-1K (Li et al. 2024) | 6 | Household Tasks |
| Mini-BEHAVIOR (Jin et al. 2023) | 15 | Household Tasks |
| House3D (Wu et al. 2018) | 12 | Navigation |
| R2R (Anderson et al. 2018) | 6 | Navigation |
| CVDN (Thomason et al. 2019) | 6 | Navigation |
| HM3D (Ramakrishnan et al. 2021) | 4 | Navigation |
| ThreeDWorld (Gan et al. 2021) | 6 | Navigation,Manipulation |
| CHAI (Misra et al. 2019) | 5 | Navigation,Manipulation |
| CALVIN (Mees et al. 2021) | 12 | Manipulation |
| Overcooked (Carroll et al. 2019) | 6 | Cooking |
| VRKitchen (Gao et al. 2019) | 6 | Cooking |
| KITchen (Younes and Asfour 2024) | 12 | Household & Cooking |
| SmartPlay (Wu et al. 2023) | 37 | Games |
| DIGITAL SIMULATORS | | |
| ToolAlpaca (Tang et al. 2023) | 400 | Function Call |
| API-Bank (Li et al. 2023) | 53 | Function Call |
| API-Bench (Patil et al. 2023) | 1.6K | Function Call |
| API-Pack (Guo et al. 2024) | 11.2K | Function Call |
| ToolBench (Xu et al. 2023) | 232 | Function Call |
| ToolLLM (Qin et al. 2023) | 16.4K | Function Call |
| **WORLDAPIS (ours)** | **300+** | **Household Tasks & Cooking & Crafts, etc** |

Table 1: Existing resources, their sizes and domains.

and adjacent domains and rely on a hand-crafted small action space. Such limitations of the action space diminish the diversity of the tasks that can be executed with the given actions. Thus, they fail to capture the authenticity of real-world tasks due to their dependency on an extremely restrictive set of actions and incomplete modeling of the real world in virtual environments. In contrast, here, we ignore the concerns regarding the action (API) executability to allow us to broadly explore the space of various actions needed in the world.

The second category of benchmarks deals with executing APIs in digital spaces (e.g., interacting with web pages) without correspondence in the real world. We have highlighted notable works in this category, even though the focus of our thought experiment is on APIs needed to interact with the physical world.

**Procedural language understanding.** There is a broad range of works that focus on *procedures*, including reasoning on procedures (Zhang, Lyu, and Callison-Burch 2020; Uzunoglu, Safa, and Şahin 2024), generating procedures (Sakaguchi et al. 2021; Lyu, Zhang, and Callison-Burch 2021), and so forth. Most of this literature is confined to unimodal tasks, and the multimodal approaches are limited (Zhou et al. 2023; Zellers et al. 2021). Similarly, the focus has been on individual tasks, even though collective benchmarks that cover different tasks exist (Uzunoglu and Şahin 2023; Onoe et al. 2021) Therefore, most works are confined to shallow textual and visual representations. This leads to the lack of procedural data grounded in real-world tasks and scenarios. Similar to what we aspire to achieve, Puig et al. (2018) utilizes wikiHow to convert high-level procedures to executable actions in virtual environments. However, its breadth is highly narrow due to its limited domain and predefined action space of 12 actions.

# 3 Defining a Hypothetical World

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

Unlike prior work that takes a *bottom-up* approach (i.e., first define the action space and then build the simulator), we adopt a *top-down* formalism: we first collect diverse and realistic instructions from online resources (§3.1), then define *hypothetical* environment and agent that are capable of carrying out these instructions (§3.2), and finally induce agent programs and action spaces jointly (§4).

## 3.1 Collecting instructions from wikiHow

We leverage wikiHow, a prominent web platform with 200K+ professionally curated "how-to" tutorials across a diverse set of domains. Each wikiHow tutorial presents a sequence of instructional paragraphs in natural language that aim to accomplish a certain goal (see Figure 1 for an example). We follow prior work (Zhang, Lyu, and Callison-Burch 2020; Zhou et al. 2022) to use the tutorial title as the goal, the paragraph headline as instruction steps, and the paragraph body as additional descriptions.

Crucially, wikiHow is curated to be used *by humans*. It contains a significant diversity of tasks that require seemingly similar actions (cutting vs. pruning, tying vs. fastening, etc.) with subtly different physical control specifications, making it a rich source of tasks for simulating embodied agents of the real world.

While wikiHow covers a wide range of domains, we choose *Home & Garden* as our domain due to its feasibility to be a test bed for embodied agents and its ample action space reflective of the real world. Thus, we exclude categories that are abstract and social (i.e. *Relationships* and *Youth*), contain actions that would not be expected to be performed by embodied agents (i.e. *Work World*), or have repetitive actions that fail to illustrate the scale of realistic action space (i.e. *Computers & Electronics*).

## 3.2 A Text-based Hypothetical Environment based on wikiHow Instructions

In our formulation, a text-based hypothetical environment is defined by each wikiHow tutorial. The instructions in the tutorials provide all the objects explicitly or implicitly required to complete the task inferred by commonsense knowledge.

Since we are not building a real simulator, we make the following simplifications to the environment: 1) Scenery and spatial details are omitted and simplified to object receptacle relations. For example, liquid/powder is within the container; 2) As a result of spatial simplification, navigation, and object detection are simplified to a `find` API, which directs the agent to locate and navigate to the vicinity (within intractable distance) of a target object.

## 3.3 Principles for Translating Language Instructions to Pythonic Policies and Actions

Inspired by recent work (Liang et al. 2022; Singh et al. 2022), we jointly induce primitive actions (APIs) and poli-

Listing 1: Our in-context demonstrations for decomposing wikiHow tasks into API calls.

```
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.
    ...
# execute the program
robot_program()
```

cies (Pythonic programs) via prompting LLMs with few-shot demonstrations (details in §4). 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.

**Overview.** The example in Listing 1 shows a demonstration that translates the first wikiHow tutorial in the top-left of Figure 1, i.e. "How to Melt Chocolate in Microwave". After specifying the tutorial (`TASK`) and numbered instruction steps (`INSTRUCTIONS`), the following `PROGRAM` lists the objects and primitive actions through `import`, specifies object relations with `receptacles`, and incorporates instruction steps and sub-steps in comments for policy specification.

**Defining objects.** Following Kolve et al. (2017), we unify all the objects under the `WorldObject` class that is defined with two types of properties as described below:

Actionable properties specify the APIs an agent can invoke to interact with an object. For example, the `grab` API can only target objects with the actionable property `grabbable`.

Material properties refer to the inherent attributes of objects that are not directly tied to interactions but can be altered through API calls. For instance, the chocolate object in Listing1 with the material property `{'form': 'bar'}` may change to `{'form': 'small pieces'}` after a `chop` API call.

Object-object relationships can be specified via `receptacles` and `receptacles_of`, when necessary. We refer readers to apendix on our arXiv draft for the complete definition of `WorldObject`.

**Defining primitive actions (APIs).** While there are numerous potential ways to define the action spaces, we seek to strike a balance in our definition among 1) stylistically and semantically similar to existing embodied environments (Shridhar et al. 2020); 2) simple and informative such that LLMs can effectively leverage its parametric knowledge of code generation to predict agent policies as programs for unseen instructions; 3) easily extendable where LLMs are capable of hallucinating new APIs that are consistent to the available APIs specified in the demonstrations. Specifically, we focus on the following aspects:

Base APIs: To ensure comparability with prior work, we provide a set of commonly used APIs for executing wikiHow instructions, such as `find`, `put`, `open_obj`, and `turn_on`.

Granularity: Defining the right abstraction level for actions is challenging. Nevertheless, in our definitions, we aim to strike a balance between the following two principles:

- Avoiding overly-abstract actions: We prefer to have tasks broken into actions that directly interact with the objects. For example, given the task of "dry your clothes with a drier," one straightforward approach is distilling it to a single `dry` action. However, we argue that it is not a desired approach to create the action space since it creates APIs at a problematic level of granularity from an action perspective - drying something with a dryer requires a totally different set of actions to drying something with iron. Instead, break up this task into sub-steps (e.g., "the dryer can be opened and turned on to perform drying"), each appropriate action.
- Avoiding too low-level actions: We prefer to avoid actions that involve low-level physics. As this degrades the action space into just a handful of control APIs with complicated spatial-temporal argument realization, which is infeasible in our hypothetical environment without explicit spatial relation specification, and harms the reusability of learned APIs in practice. For instance in Listing 1, given the instruction substep "chop chocolate until it 's into small pieces", we will define `chop(obj=chocolate_0, tool=knife_0, on=cutting_board_0)`, instead of `move_held_object(moveMagnitude=0.1)`.

State checking and feedback loop: Similar to prior work (Singh et al. 2022), in our demonstrations we implement steps where the agent collects feedback from the environments by checking the object state (material properties) and acting accordingly (details in appendix on our arXiv draft.)

Actionability: At times we might encounter instructions that are either too broad, subjective, or under-specified to be programmed by any existing/new APIs. For example, actions that involve personal feelings, or subjective preferences. Whenever we encounter such instructions, we skip them with a comment `# skip this instruction`.

## 4 Inducing the Action/Policy Space in the Hypothetical World

Following our definition of the hypothetical world (§3), we develop a pipeline for inducing action/policy of wikiHow tutorials via iterative few-shot code generation with LLMs. As depicted in Figure 2, 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.

**Prompt construction.** The prompts provided to LLM start with a system instruction that specifies the task requirements and the `WorldObject` definition. The system prompt is followed by a sequence of $k$ (we use $k = 10$) in-context examples. These examples are pairs of instructions and their corresponding programs, retrieved based on the semantic similarity to a particular instruction in the given tutorial.

Listing 2: Example API use cases

```
# Use Case of squeeze
# bring the sponge to the sink and squeeze out the
    water
find(obj=sink_0)
while sponge_0.material_properties['filled'] != '
    empty':
    squeeze(obj=sponge_0, target=sink_0)

# Use Case of insert
# attach hose attachment to vacuum
insert(obj=hose_attachment_0, target=vacuum_0)
```

**API use cases.** To provide more complete information about all existing APIs, we prepend use cases of all APIs discovered thus far to the input prompt. The use cases are the most similar use cases of each API retrieved from the pool. The sequence of use cases is inserted before the full programs as extra demonstrations. The use case of API $f$ is a program snippet starting from a sub-step comment and ending at the line that calls $f$.

**Expanding the pool of demonstrations and APIs.** We perform rejection sampling on program generation. Specifically, a program that either contains syntactic errors (i.e. failed in Abstract Syntax Tree parsing) or does not fully follow the instructions (i.e. does not contain comments of every numbered instruction step) will be rejected and the LLM will regenerate the program. For all the verified programs, we extract the full program and all the code snippets as API use cases and add them to the corresponding pools.

**Leveraging action descriptions in wikiHow.** While most of the prior work only uses the headline of each instructional paragraph in wikiHow, we find that their descriptions contain beneficial details to decompose and ground the instructions into primitive actions. For example, the header "*Combine the sugar, cocoa powder, and salt in a saucepan*" omits the action of mixing the ingredients that are specified in the description body "*Pour the sugar into a small saucepan. Add the unsweetened cocoa powder and a dash of salt. Stir everything together with a whisk.*" We, therefore, include these descriptions as part of the input instructions.

## 5 Experiment & Evaluation Metrics

We apply the proposed induction pipeline to the Home and Garden category of wikiHow tutorials.

### 5.1 Experimental Setup

We perform human annotation on 19 randomly sampled tutorials according to the annotation guideline, which serves as the seed demonstrations. We then sample 1000 random tutorials to represent the diverse procedural instructions from users in real-world scenarios. Then, we evaluate 3 variants of our pipeline on the sampled set:

- 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 (discussed in §4).

For all the pipeline variants, we use OpenAI `gpt-4-1106-preview` as the backbone LLM, with temperature = 0. In prompt construction, we use top $k = 10$ similar full-program and use the top-1 similar use case for every API provided/induced so far at each step as in-context demonstrations. When selecting demonstrations, we measure retrieval similarity based on the embedding similarity of the concatenation of the tutorial name and instruction steps of the full program demonstrations and the leading comment of the API use case demonstrations. We use OpenAI `text-embedding-ada-002-v2` to obtain text embeddings.

We run all the pipeline variants for the first 50 tutorials within samples for human evaluation and apply **Base + Use Case + Description** to induce the action space on the full set of 1000 tutorials to induce the action space.

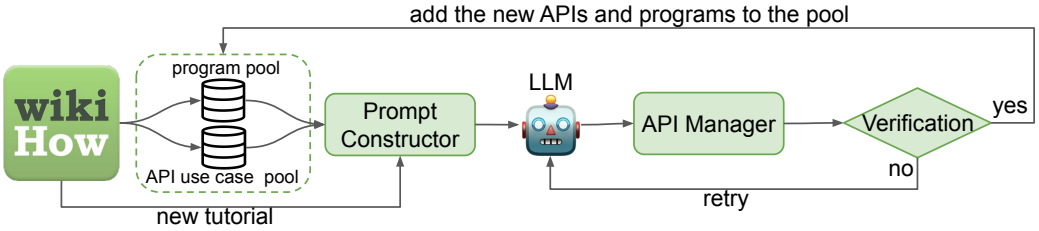


Figure 2: Proposed pipeline that jointly induces new APIs and programs.

| | 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 2: 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. **Adding API use case demonstrations helps decrease redundancy and significantly reduce the creation of new APIs, and further adding descriptions improves faithfulness.**

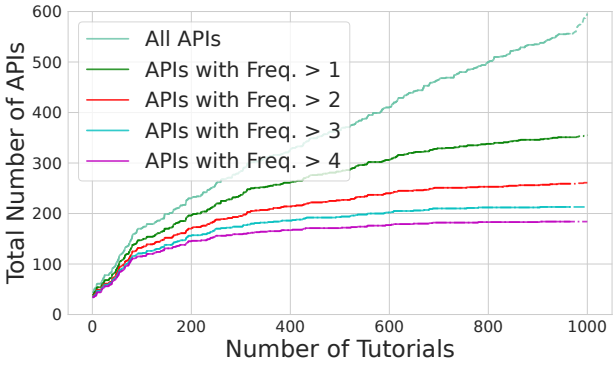


Figure 3: Size of API pool vs. # of tutorials. Lines represent different frequency thresholds used to filter the APIs. With increasing thresholds, the size of the pool of frequently-evoked APIs stabilized after inducing $\sim$ 600 tutorials.

## 5.2 Evaluation Metrics

Since our main contributions are based on the thought experiments, we focus on approximating the action space grounded in the hypothetical world (§3), instead of implementing a simulation of it. An exact execution-based evaluation is beyond the scope of our work. We thus approximate the execution-based evaluation with human evaluations and characterize the resulting action space through the lens of several automatic statistics.

**Human Evaluation.** Since we do not have a simulator to execute the induced APIs and policies, we manually evaluate the quality of programs and APIs for the first 50 tutorials produced by each pipeline variant. We define two metrics to measure the quality of APIs and programs:

Redundancy: In principle, we want the resulting pool of APIs to have low redundancy: each API should be atomic and unique in its functionality. For each new API, we quantify its redundancy with a $0-0.5-1$ scale measurement. An API is considered *fully redundant* with a score of 1 if there exist straightforward composition(s) of existing APIs that can replicate the functionality of that new API; *fully useful* with a score of 0 if there does not exist such compositions; and *partially redundant/useful* with a score of 0.5 for all the other cases - which includes complicated action where decomposition is complicated without expert knowledge (e.g. install electrical conduit), synonyms that are similar to existing APIs semantically but require different low-level physical specifications (e.g. wipe vs. scrub, cut vs. snip ).

Faithfulness: We approximate the simulator execution-based evaluation of generated programs with faithfulness measurement of $0-0.5-1$ scale. We define faithfulness on each individual instruction step: the objects are assumed to be in good initial conditions (e.g. the goal condition of successfully performing the previous step) at the beginning of each instruction step. Under this setup, a subprogram of its corresponding instruction step is *fully faithful* with a score of 1 if all the goal conditions will be met upon execution; 0.5 if only part of the goal conditions can be met, and 0 if non of the goal condition will be met. In addition to the absolute scores, we also evaluate the faithfulness across pipeline variants based on relative ranking. For each instruction, we ask the annotators to rank the code snippets of different configurations based on their relative faithfulness.

**Automatic evaluation.** We further quantify the characteristics of the induced API pool with the following statistics:

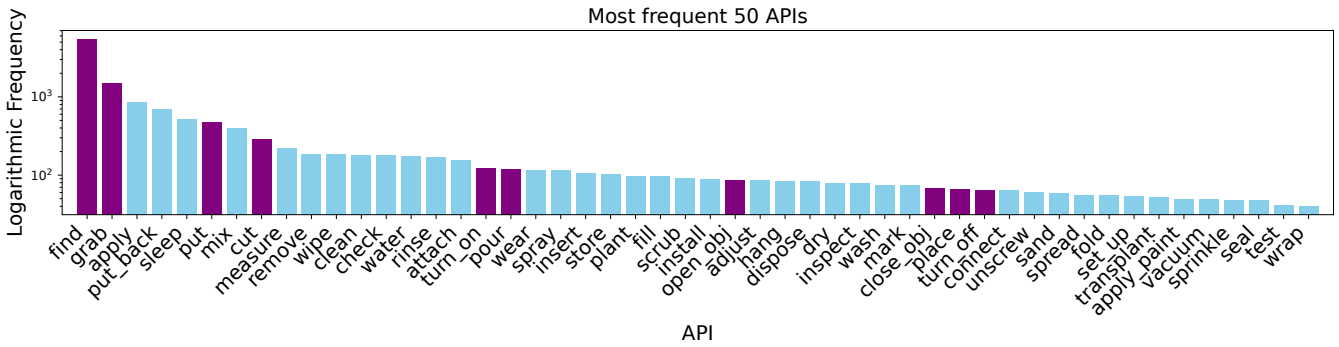


Figure 4: Top-50 most frequent APIs in the induced action space, with frequency in log scale. We use ■ to mark the APIs with exact/overlapping affordance to the primitive actions in existing embodied environments (Shridhar et al. 2020; Puig et al. 2018) and use ■ to mark APIs that are beyond the action space of exiting environments.

- **Size of API Pool** that records the total number of unique APIs defined and induced after each induction step.
- **New API Induction Ratio** that measures the fraction of induced APIs over all the unique APIs evoked at each induction step.
- **API frequency** that counts the number of calls on each API over the full induction

## 6 Results & Analysis

### 6.1 Human Evaluation

Table 2 shows the human evaluation results of running the induction pipeline variants on the first 50 wikiHow tutorials from the 1000 test samples. In addition to the absolute redundancy score, we also report the redundancy scores when converting all the `partially redundant` APIs that are either too complicated to be further decomposed without expert knowledge (-Complex), or synonyms to existing APIs (-Synonym) to `fully useful`. As we believe in practice the further optimization of these APIs should be done at the implementation level. As indicated by the results, adding API use cases as demonstrations significantly reduces the number of new APIs induced and also decreases the redundancy. Further incorporating detailed wikiHow descriptions improves the faithfulness of the generated program.

### 6.2 Automatic Evaluation

We illustrate in Figure 3 the expansion of the API pool while iteratively inducing on the sampled 1000 wikiHow tutorials, with filtering of infrequent APIs. At all levels of filtering, the pool of APIs demonstrates a diminishing increase in size and reaches a plateau after $\sim$ 600 steps of inductions. This indicates the induction pipeline approximates the **stabilized** space of frequently-evoked primitive actions to be $\sim 200 - 400$. Moreover, the saturation of the action space also indicates that as induction proceeds, LLMs shift from frequently inducing new APIs to largely reusing existing APIs. In appendix of our arXiv draft we provide the average ratio of induced APIs at each step of the induction.

Figure 4 displays the top 50 most frequently evoked APIs in the induced action space. It can be observed that the induced APIs cover a wide variety of physical actions that are necessary for carrying out the rich space of tasks in the real world. It is worth noting that the induction is only performed on a diverse yet very small fraction ($< 0.5\%$) of wikiHow tutorials. And with such limited scale yields a space of hundreds of APIs. The find suggests that the approximated $\sim 200 - 400$ frequent primitive actions can only serve as a proxy of the lower bound of a potentially much larger action space reflected in wikiHow.

Furthermore, we highlight with ■ the induced APIs whose affordances are covered/supported by primitive actions of existing embodied environments (Shridhar et al. 2020; Puig et al. 2018). The limited coverage (9 out of 50) provides evidence of the gap between action spaces of simulations and the real world, and motivates future work on simulations with richer primitive actions.

## 7 Conclusion and Limitations

Here, we attempt to approximate the size and properties of the action space of a versatile embodied agent that can carry out diverse tasks in the real world. We design a thought experiment with an induction pipeline to jointly induce primitive actions as APIs and agent policies as Pythonic programs, based on the hypothetical environments defined with wikiHow tutorials. Human and automatic evaluation verifies the usability of our induced APIs and provides an approximation of the lower bound of the realistic action space (300+). Moreover, our analysis highlights the deficiency in action diversity of existing embodied environments and motivates the development of action-rich simulations.

On the other hand, the induced action space is still noisy with a relatively high portion of redundant APIs, and it is challenging to scale up the evaluation without realizing APIs and executing policies induced by our pipeline. These limitations motivate future work in several interesting directions including reducing redundancy via self-correction (Madaan et al. 2023; Jiang et al. 2024) and evaluation in video-based embodied simulations (Du et al. 2023).

## Acknowledgements

This work is in-part supported by ONR grant N00014-241-2089, and generous gifts from Amazon and the Allen Institute for AI. We are also grateful to the broader JHU CLSP community and our anonymous reviewers for their support and constructive feedback

## References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Ho, D.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jang, E.; Ruano, R. J.; Jeffrey, K.; Jesmonth, S.; Joshi, N. J.; Julian, R. C.; Kalashnikov, D.; Kuang, Y.; Lee, K.-H.; Levine, S.; Lu, Y.; Luu, L.; Parada, C.; Pastor, P.; Quiambao, J.; Rao, K.; Rettinghouse, J.; Reyes, D. M.; Sermanet, P.; Sievers, N.; Tan, C.; Toshev, A.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; and Yan, M. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Conference on Robot Learning*.

Anderson, P.; Wu, Q.; Teney, D.; Bruce, J.; Johnson, M.; Sünderhauf, N.; Reid, I.; Gould, S.; and van den Hengel, A. 2018. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. arXiv:1711.07280.

Carroll, M.; Shah, R.; Ho, M. K.; Griffiths, T. L.; Seshia, S. A.; Abbeel, P.; and Dragan, A. D. 2019. On the Utility of Learning about Humans for Human-AI Coordination. *ArXiv*, abs/1910.05789.

Du, Y.; Yang, M.; Dai, B.; Dai, H.; Nachum, O.; Tenenbaum, J. B.; Schuurmans, D.; and Abbeel, P. 2023. Learning Universal Policies via Text-Guided Video Generation. *ArXiv*, abs/2302.00111.

Gan, C.; Zhou, S.; Schwartz, J.; Alter, S.; Bhandwaldar, A.; Gutfreund, D.; Yamins, D. L. K.; DiCarlo, J. J.; McDermott, J. H.; Torralba, A.; and Tenenbaum, J. B. 2021. The ThreeDWorld Transport Challenge: A Visually Guided Task-and-Motion Planning Benchmark Towards Physically Realistic Embodied AI. *2022 International Conference on Robotics and Automation (ICRA)*, 8847–8854.

Gao, X.; Gong, R.; Shu, T.; Xie, X.; Wang, S.; and Zhu, S.-C. 2019. VRKitchen: an Interactive 3D Virtual Environment for Task-oriented Learning. *ArXiv*, abs/1903.05757.

Guo, Z.; Soria, A. M.; Sun, W.; Shen, Y.; and Panda, R. 2024. API Pack: A Massive Multilingual Dataset for API Call Generation. *ArXiv*, abs/2402.09615.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *arXiv preprint arXiv:2201.07207*.

Huang, W.; Xia, F.; Shah, D.; Driess, D.; Zeng, A.; Lu, Y.; Florence, P.; Mordatch, I.; Levine, S.; Hausman, K.; and Ichter, B. 2023. Grounded Decoding: Guiding Text Generation with Grounded Models for Embodied Agents. In *Neural Information Processing Systems*.

Jiang, D.; Zhang, J.; Weller, O.; Weir, N.; Durme, B. V.; and Khashabi, D. 2024. SELF-[IN]CORRECT: LLMs Struggle with Refining Self-Generated Responses. *ArXiv*, abs/2404.04298.

Jin, E.; Hu, J.; Huang, Z.; Zhang, R.; Wu, J.; Li, F.-F.; and Mart'in-Mart'in, R. 2023. Mini-BEHAVIOR: A Procedurally Generated Benchmark for Long-horizon Decision-Making in Embodied AI. *ArXiv*, abs/2310.01824.

Kolve, E.; Mottaghi, R.; Han, W.; VanderBilt, E.; Weihs, L.; Herrasti, A.; Deitke, M.; Ehsani, K.; Gordon, D.; Zhu, Y.; Kembhavi, A.; Gupta, A. K.; and Farhadi, A. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *ArXiv*, abs/1712.05474.

Ku, A.; Anderson, P.; Patel, R.; Ie, E.; and Baldridge, J. 2020. Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding. arXiv:2010.07954.

Li, C.; Zhang, R.; Wong, J.; Gokmen, C.; Srivastava, S.; Martín-Martín, R.; Wang, C.; Levine, G.; Ai, W.; Martinez, B.; Yin, H.; Lingelbach, M.; Hwang, M.; Hiranaka, A.; Garlanka, S. S.; Aydin, A.; Lee, S.; Sun, J.; Anvari, M.; Sharma, M.; Bansal, D.; Hunter, S.; Kim, K.-Y.; Lou, A.; Matthews, C. R.; Villa-Renteria, I.; Tang, J. H.; Tang, C.; Xia, F.; Li, Y.; Savarese, S.; Gweon, H.; Liu, C. K.; Wu, J.; Li, F.-F.; and Research, S. 2024. BEHAVIOR-1K: A Human-Centered, Embodied AI Benchmark with 1,000 Everyday Activities and Realistic Simulation.

Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; and Li, Y. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. arXiv:2304.08244.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P. R.; and Zeng, A. 2022. Code as Policies: Language Model Programs for Embodied Control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500.

Lyu, Q.; Zhang, L.; and Callison-Burch, C. 2021. Goal-Oriented Script Construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, 184–200. Aberdeen, Scotland, UK: Association for Computational Linguistics.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Mees, O.; Hermann, L.; Rosete-Beas, E.; and Burgard, W. 2021. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters*, 7: 7327–7334.

Misra, D.; Bennett, A.; Blukis, V.; Niklasson, E.; Shatkhin, M.; and Artzi, Y. 2019. Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction. arXiv:1809.00786.

Nasiriany, S.; Maddukuri, A.; Zhang, L.; Parikh, A.; Lo, A.; Joshi, A.; Mandlekar, A.; and Zhu, Y. 2024. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots.

Onoe, Y.; Zhang, M. J. Q.; Choi, E.; and Durrett, G. 2021. CREAK: A Dataset for Commonsense Reasoning over Entity Knowledge. arXiv:2109.01653.

Padmakumar, A.; Thomason, J.; Shrivastava, A.; Lange, P.; Narayan-Chen, A.; Gella, S.; Piramuthu, R.; Tur, G.; and Hakkani-Tur, D. 2021. TEACh: Task-driven Embodied Agents that Chat. arXiv:2110.00534.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2023. Gorilla: Large Language Model Connected with Massive APIs. *ArXiv*, abs/2305.15334.

Puig, X.; Ra, K. K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. VirtualHome: Simulating Household Activities Via Programs. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8494–8502.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.-T.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; Zhao, S.; Tian, R.; Xie, R.; Zhou, J.; Gerstein, M. H.; Li, D.; Liu, Z.; and Sun, M. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *ArXiv*, abs/2307.16789.

Ramakrishnan, S. K.; Gokaslan, A.; Wijmans, E.; Maksymets, O.; Clegg, A.; Turner, J.; Undersander, E.; Galuba, W.; Westbury, A.; Chang, A. X.; Savva, M.; Zhao, Y.; and Batra, D. 2021. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI. *ArXiv*, abs/2109.08238.

Sakaguchi, K.; Bhagavatula, C.; Le Bras, R.; Tandon, N.; Clark, P.; and Choi, Y. 2021. proScript: Partially Ordered Scripts Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2138–2149. Punta Cana, Dominican Republic: Association for Computational Linguistics.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10737–10746.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2022. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530.

Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Srivastava, S.; Li, C.; Lingelbach, M.; Mart'in-Mart'in, R.; Xia, F.; Vainio, K.; Lian, Z.; Gokmen, C.; Buch, S.; Liu, C. K.; Savarese, S.; Gweon, H.; Wu, J.; and Fei-Fei, L. 2021. BEHAVIOR: Benchmark for Everyday Household Activities in Virtual, Interactive, and Ecological Environments. In *Conference on Robot Learning*.

Tang, Q.; Deng, Z.; Lin, H.; Han, X.; Liang, Q.; Cao, B.; and Sun, L. 2023. ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases. *ArXiv*, abs/2306.05301.

Thomason, J.; Gordon, D.; and Bisk, Y. 2019. Shifting the Baseline: Single Modality Performance on Visual Navigation & QA. arXiv:1811.00613.

Thomason, J.; Murray, M.; Cakmak, M.; and Zettlemoyer, L. 2019. Vision-and-Dialog Navigation. arXiv:1907.04957.

Uzunoglu, A.; Safa, A. R.; and Şahin, G. G. 2024. PARADISE: Evaluating Implicit Planning Skills of Language Models with Procedural Warnings and Tips Dataset. arXiv:2403.03167.

Uzunoglu, A.; and Şahin, G. 2023. Benchmarking Procedural Language Understanding for Low-Resource Languages: A Case Study on Turkish. In Park, J. C.; Arase, Y.; Hu, B.; Lu, W.; Wijaya, D.; Purwarianti, A.; and Krisnadhi, A. A., eds., *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 804–819. Nusa Dua, Bali: Association for Computational Linguistics.

Wu, Y.; Tang, X.; Mitchell, T. M.; and Li, Y. 2023. SmartPlay : A Benchmark for LLMs as Intelligent Agents. *ArXiv*, abs/2310.01557.

Wu, Y.; Wu, Y.; Gkioxari, G.; and Tian, Y. 2018. Building Generalizable Agents with a Realistic and Rich 3D Environment. *ArXiv*, abs/1801.02209.

Xia, F.; Zamir, A.; He, Z.-Y.; Sax, A.; Malik, J.; and Savarese, S. 2018. Gibson Env: Real-World Perception for Embodied Agents. arXiv:1808.10654.

Xu, Q.; Hong, F.; Li, B.; Hu, C.; Chen, Z.; and Zhang, J. 2023. On the Tool Manipulation Capability of Open-source Large Language Models. *ArXiv*, abs/2305.16504.

Yang, C.; Chen, Y.-C.; Yang, J.; Dai, X.; Yuan, L.; Wang, Y.-C. F.; and Chang, K.-W. 2023. LACMA: Language-Aligning Contrastive Learning with Meta-Actions for Embodied Instruction Following. *ArXiv*, abs/2310.12344.

Younes, A.; and Asfour, T. 2024. KITchen: A Real-World Benchmark and Dataset for 6D Object Pose Estimation in Kitchen Environments.

Zellers, R.; Lu, X.; Hessel, J.; Yu, Y.; Park, J. S.; Cao, J.; Farhadi, A.; and Choi, Y. 2021. MERLOT: Multimodal Neural Script Knowledge Models. In *Advances in Neural Information Processing Systems 34*.

Zhang, L.; Lyu, Q.; and Callison-Burch, C. 2020. Reasoning about Goals, Steps, and Temporal Ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4630–4639. Online: Association for Computational Linguistics.

Zhou, H.; Martín-Martín, R.; Kapadia, M.; Savarese, S.; and Niebles, J. C. 2023. Procedure-Aware Pretraining for Instructional Video Understanding. arXiv:2303.18230.

Zhou, S.; Zhang, L.; Yang, Y.; Lyu, Q.; Yin, P.; Callison-Burch, C.; and Neubig, G. 2022. Show Me More Details: Discovering Hierarchies of Procedures from Semi-structured Web Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2998–3012. Dublin, Ireland: Association for Computational Linguistics.