AdaptBot: Supplementary Material

Contents

1 Turtle Data Format Overview

We used the Turtle (Terse RDF Triple Language) format to represent the ontology of the environment we are operating in. Our ontology describes different objects, locations, tools, and the relationships between them. We used SPARQL to retrieve or update the state of the environment in a structured manner.

1.0.1 Turtle Syntax

Turtle syntax represents data as triples consisting of a subject, predicate, and object, which form a statement. The subject is the entity, the predicate is the relationship, and the object is the value.

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:robot rdf:type rdfs:agent;
    ex:agent_near ex:cabinet ;
    ex:agent_location ex:kitchen.
```

In this example, the data states that the robot is near the cabinet, which is located in the kitchen.

1.1 Querying the Turtle Data with SPARQL

We used SPARQL (SPARQL Protocol and RDF Query Language) to extract or modify the data from the RDF graph representing the environment. The robot interacts with the ontology using SPARQL queries to determine the state of the items in the environment and to perform actions on them. This structure allows for dynamic queries and updates, enabling the robot to navigate and manipulate the environment effectively.

1.1.1 Queries for Object Locations

The robot queries the ontology to determine the various states of objects, tools, and receptacles using SPARQL queries. For instance, to find the location of an *apple*, the following query is used:

```
PREFIX ex: <http://example.org/>
SELECT ?obj_location
WHERE {
    ex:apple ex:obj_location ?obj_location .
}
```

This query retrieves the location of apple.

```
PREFIX ex: <http://example.org/>
SELECT ?status
WHERE {
    ex:apple ex:IsCleaned ?status.
}
```

This query tells if the apple is cleaned or not.

```
PREFIX ex: <http://example.org/>
SELECT ?obj_name
WHERE {
    ?obj ex:obj_name ?obj_name.
    ?obj ex:obj_location ex:fridge.
    ?obj ex:IsCleaned false.
}
```

This query lists all the items present in the fridge that are not clean.

1.1.2 State-Based Actions

The agent (robot) uses the ontology to perform actions like moving to a location, picking up objects, and interacting with tools or receptacles. SPARQL queries are used to verify states (e.g., whether an object is in a specific location or if a task like cleaning or slicing has been completed).

For example, the robot queries whether an object is "cleaned" before proceeding with a task:

```
PREFIX ex: <http://example.org/>
SELECT ?status
WHERE {
    ex:object ex:IsCleaned ?status .
}
```

This query returns the cleaning status of the object.

1.1.3 Updating the RDF Graph

SPARQL is also used to update the environment. After the robot performs an action (e.g., cleaning an object), the RDF graph is updated using SPARQL `INSERT` and `DELETE` operations.
For example, after performing the cleaning action on an *item*, its `IsCleaned` status is updated using the query:

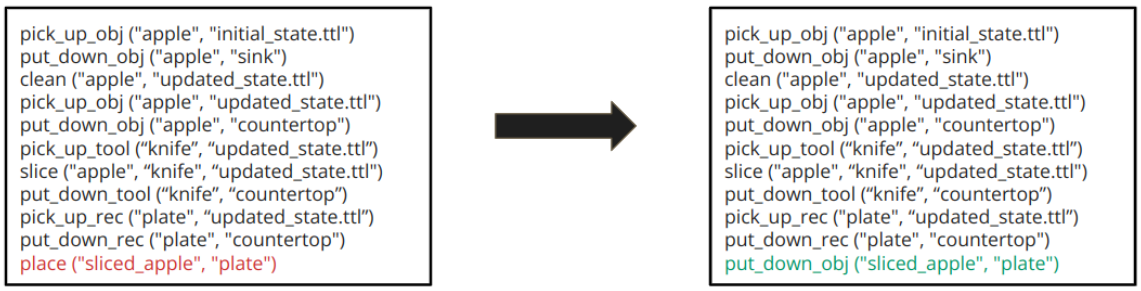
```
DELETE WHERE {{
        ex:{item} ex:IsCleaned false.
}}
INSERT DATA {{
    ex:{item} ex:IsCleaned true.
}}
```

2 Question asked during Knowledge Expansion

2.1 Question asked for unknown action

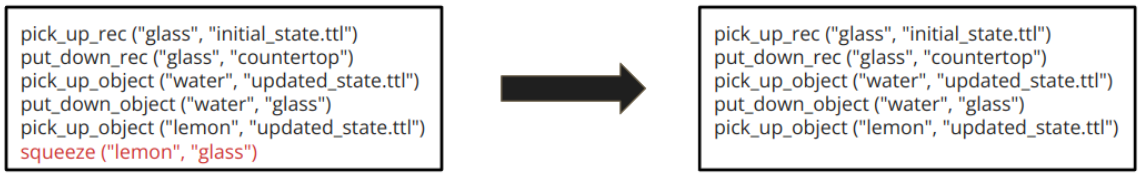
Revising action name.

- **Robot:** I do not know how to execute action *'place'*, am I mistaking it for another action? Please answer in 'Yes' or 'No'.
- **Human:** Yes
- **Robot:** Please revise the action name
- **Human:** put_down_obj



Removing unexecutable action.

- **Robot:** I do not know how to execute action *'squeeze'*, am I mistaking it for another action? Please answer in 'Yes' or 'No'.
- **Human:** No



Agent skips the action when it is not executable.

2.2 Question asked for unknown item

Adding an item to G_s and G_k.

Example 1 - Adding *mango* to KG

- **Robot:** Item *'mango'* is not present in my knowledge base. Is it present in the environment? Please answer in 'Yes' or 'No'. If I am mistaking it for another item, please enter 'correct_item'.
- **Human:** Yes.
- **Robot:** What kind of item is *'mango'*? Choose from: 'edible_object', 'receptacle', 'tool'.

- **Human: edible_object**
- **Robot:** What is the location of the object?
- **Human:** Fridge
- **Robot:** Is it sliceable? Please answer in 'True' or 'False.
- **Human:** True
- **Robot:** Is it boilable? Please answer in 'True' or 'False'.
- **Human:** False
- **Robot:** Is it fryable? Please answer in 'True' or 'False'.
- **Human:** False
- **Robot:** Does this object needs cleaning? Please answer in 'True' or 'False'.
- **Human:** True

Example 2 - Adding *bowl* to KG

- **Robot:** Item *'bowl'* is not present in my knowledge base. Is it present in the environment? Please answer in 'Yes' or 'No'. If I am mistaking it for another item, please enter 'correct_item'.
- **Human:** Yes.
- **Robot:** What kind of item is *'bowl'*? Choose from: 'edible_object', 'receptacle', 'tool'.
- **Human: receptacle**
- **Robot:** What is the location of the object?
- **Human:** Cabinet
- **Robot:** Is receptacle sensitive? Please answer in 'True' or 'False.
- **Human:** True
- **Robot:** Can this receptacle be used for slicing on? Please answer in 'True' or 'False'.
- **Human:** False
- **Robot:** Does this receptacle needs cleaning? Please answer in 'True' or 'False'.
- **Human:** True

Example 3 - Adding *fork* to KG

- **Robot:** Item *'fork'* is not present in my knowledge base. Is it present in the environment? Please answer in 'Yes' or 'No'. If I am mistaking it for another item, please enter 'correct_item'.
- **Human:** Yes.
- **Robot:** What kind of item is *'fork'*? Choose from: 'edible_object', 'receptacle', 'tool'.
- **Human: tool**

- **Robot:** What is the location of the object?
- **Human:** rack
- **Robot:** Is it a slicing tool? Please answer in 'True' or 'False.
- **Human:** False
- **Robot:** Does this tool needs cleaning? Please answer in 'True' or 'False'.
- **Human:** True

Revising the item name.

- **Robot:** Item *'sauce'* is not present in my knowledge base. Is it present in the environment? Please answer in 'Yes' or 'No'. If I am mistaking it for another item, please enter 'correct_item'.
- **Human:** correct_item.
- **Robot:** Please revise the object name.
- **Human:** ketchup

Item isn't present in the environment.

- **Robot:** Item *'grapes'* is not present in my knowledge base. Is it present in the environment? Please answer in 'Yes' or 'No'. If I am mistaking it for another item, please enter 'correct_item'.
- **Human:** No.

Agent skips the action when item is not available.

3 Classes of Tasks

In order to evaluate the ability of our framework to adapt to different classes of tasks, we considered cooking and cleaning tasks. We considered 30 different cooking tasks in a kitchen; These tasks were created by randomly sampling from the Recipe1M+ dataset[1]. In addition, we considered 12 variants of cleaning/clearing tasks that involved the agent cleaning specific objects or surfaces or arranging objects in desired configurations in particular rooms.

[1] `https://paperswithcode.com/dataset/recipe1m-1`

Recipes sampled from Recipe1M+ dataset

- chocolate cookies
- peanut butter cookies
- italian burger
- bbq crunch burger
- mac and cheese burger
- onion sandwich
- mushroom sandwich
- chicken salad sandwich
- french toast sandwich
- tomato soup
- cucumber salad
- greek salad
- peppermint iced tea
- chai tea latte
- ginger iced tea
- chargrilled bbq burger
- chili burger
- tea scones
- stop and go tea sandwich
- green tea lava cake
- egg and spinach sandwich
- chocolate and banana toasted sandwich
- veggie sandwich panini
- split pea soup
- potato soup
- weight watchers golden cream soup
- basil tomato mozzarella salad
- ham pea and cheese salad
- anja's egg salad
- mozzarella beef burger

Cleaning/Clearing tasks

- clean the bedroom
- dust the TV
- wash the clothes
- water the plant
- take out trash
- wash the dishes
- clean the window
- mop the countertop
- clean the table
- pick up and put all the toys to the toy box
- charge the phone
- play the music

4 LLM pormpts

Given below is the function that generated the main prompt for the GPT-3.5 and GPT-4o models.

Main Prompt

```
def prompt(onto_file):

    o,r,t,im = fetch_details(onto_file)
    actions, funcs = fetch_actions()

    main_prompt = f"""
- I am working on a problem where I use an LLM along with a knowledge graph to perform task planning in a given environment.
- Given is the ontology of the environment. The environment's ontology defines classes named object, receptacle, and tool, along with their
        properties. These classes are instantiated with the items present in the environment. We also define a set of actions that the robot can
        perform within this environment.

- Given this information, LLM should give me two things in the output:
1) Chain of thought: An explanation of why it selected a particular task sequence, describing the process by breaking it into subproblems. For
        example, for cooking, the problem should be broken down into the following sub-parts: 1. Fetching the ingredients and tools required for the
        cooking task, 2. Cooking process, 3. Plating and serving.
2) The action sequence for the task.


- Here is the information of my environment:

Environment Description:
- Rooms: ['kitchen']
- Objects: {o}
- Tools: {t}
- Receptacles: {r + im}
- Actions: {actions}
The first action would take initial ontology, initial_onto.ttl as the argument and then the next actions in the sequence would take updated_onto.ttl
        as the argument.

- I will give you an example. In order to make a coffee, the chain of thought and the sequence of actions would be the following:

COT & ACTION SEQUENCE:

1. Fetching the ingredients and tools required for the cooking task:

To prepare the coffee, I will first fetch the items I need. I will bring the pan, milk, coffee, and glass. If the glass is not cleaned, I will clean
        it first.

pick_up_obj("milk", "initial_onto.ttl") # picking up the milk
put_down_obj("milk", "countertop")   # putting down the milk on the countertop
pick_up_obj("coffee", "updated_onto.ttl") # picking up the coffee
put_down_obj("coffee", "countertop")   # putting down the coffee on the countertop
clean("glass", "updated_onto.ttl")   # cleaning the glass
pick_up_rec("glass", "updated_onto.ttl") # picking up the glass
put_down_rec("glass", "countertop")    # putting down the glass on the countertop

2. Cooking process:

To prepare coffee, milk should be boiled, so I will boil it.

boil("milk", "initial_onto.ttl")     # boiling the milk for the coffee

3. Plating and serving:

I will put the boiled milk into the glass, then I will put the coffee into the glass. I will stir the contents. The coffee is ready!

pick_up_obj("milk", "updated_onto.ttl") # picking up the milk
put_down_obj("milk", "glass")        # putting down the milk into the glass
pick_up_obj("coffee", "updated_onto.ttl") # picking up coffee
put_down_obj("coffee", "glass")        # putting the coffee into the glass filled with milk
stir_contents("glass", "updated_onto.ttl") # stirring the contents of the glass

STRICT GUIDELINES:

- READ THIS INFORMATION I JUST GAVE YOU CAREFULLY. STICK TO THE DESCRIPTION & STRUCTURE OF THE ACTIONS.
- STRICTLY USE THE TOOLS, RECEPTACLES AND THE ACTIONS GIVEN IN THE ENVIRONMENT, AND THE MINIMUM INGREDIENTS PROVIDED BY THE USER IN THE PROMPT.
- IN THE NEXT PROMPTS, I WILL GIVE YOU SOME NEW TASKS FOR WHICH YOU HAVE TO COME UP WITH COT & ACTION SEQUENCE (IN THE SAME FORMAT I WROTE ABOVE).
- MAKE SURE TO DIVIDE THE COT AND TASK SEQUENCE INTO SUB-PARTS:
  1. Fetching the ingredients and tools required for the cooking task,
  2. Cooking process,
  3. Plating and serving.
- SECOND ARGUMENT OF THE put_down FUNCTIONS SHOULD BE THE LOCATION ONLY, NOT AN ONTOLOGY FILE.
- SECOND ARGUMENT OF THE pour_contents FUNCTION SHOULD BE A LOCATION ONLY, NOT AN ONTOLOGY FILE.
"""

    return main_prompt
```

5 Results for other LLMs

LLM Models ↓	**Metrics →**	**LLM**	**LLM + KG**	**LLM + KG + Human**
Gemma2 9B	**Success Rate (in %) ↑**	21.04	37.75	86.93
	Avg. Tokens Used ↓	8122	8099	3999
Llama3 70B	**Success Rate (in %) ↑**	26.33	40.89	93.0
	Avg. Tokens Used ↓	7103	7451	4031