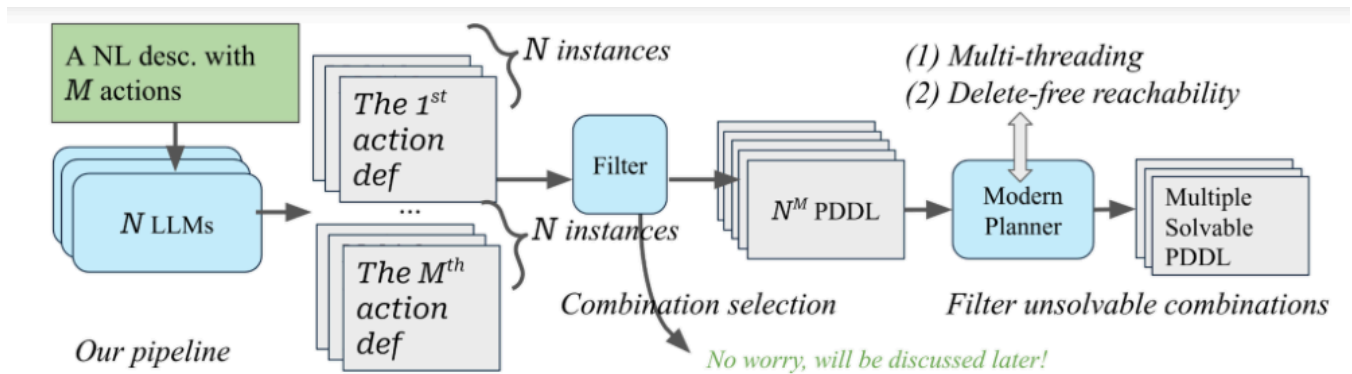


Introduction of the LLM for PDDL modeling pipeline



Introduction of the LLM for PDDL modeling pipeline

Background

Multi-step procedure to increase the chance for LLMs to correctly model the transition system.

Step 1: In context learning for LLMs

Step 2: Multiple LLMs and Combinatorial Selection

Step 3: Verify whether the generated Domain PDDL can solve a set of problem instances and filter out unsolvable ones

Step 4 (Optional, implement it if you have time): Ensure semantic alignment -- leveraging sentence encoder

Second Phase

Background

[COMP90054-AI-Planning-for-Autonomy assignment 2](#)

Your task is to read the description of the domain below and encode the three planning problem instances specified filling the template files provided in this repository in the folder [pddl template](#).

But this time, you ask LLMs to do this assignment

The Domain - Real Player Game

Help the hero to get out of a maze known as the UpsideDown!

A hero woke up in the UpsideDown (perhaps the party last night went wrong...), and wants to return home to Melbourne. The UpsideDown is a maze full of Demogorgons (monsters) and gates. The Hero needs your help to close all the gates in order to make sure monsters will not go back to Melbourne, and then get back home. I.e. The hero has to get home and close all gates. Here are basic facts for the UpsideDown domain:

- The UpsideDown contains rooms that are **connected** by corridors (UpsideDown can thus be represented by an undirected graph),
- each room can be **empty**, or can have a **Demogorgon**, a **key** or/and a **match** in it,
- one of the empty rooms is **home**: where the hero really wants to be.

The hero is lucky since she/he has full knowledge about the UpsideDown. But not that lucky – just after the hero leaves each room she visits, the room is **invigilated** by **Vecna** (the bad guy) and cannot be visited again.

The hero can perform the following actions – but only if s/he is alive!

- The hero can **move** to an adjacent room (connected by a corridor) that has not been invigilated (i.e., the hero has not already visited the room). If the hero moves into a room with a monster, she needs to be holding a match.
- **Pickup** a match or a key if present in the room the hero is currently in and the hero is empty handed
- **Drop** a match or a key, it will become present in the room, and the hero will have the hands free to pickup a key or a match.
- **Strike a match.** The hero can move into a room with a monster **but**, before leaving the room, it needs to strike a match, as the monster will eat the hero otherwise. For some reason, monsters are afraid of fire. The fire of a match lasts only until the hero moves. I.e. if a match is on fire, once the character moves, the match will go off. Once the match has been used, it doesn't disappear, you have to drop it if you want to pick up another object.
- **Close a Gate** – if there is a gate in an adjacent room to the current room where the hero is in, the hero is holding the correct color key, and the key has still some uses available, then the hero can close the gate. The hero can move through rooms with closed gates as long as she hasn't moved through that room yet. Once a gate is closed, the counter of how many times the key can be used will be decreased. Once a key has been used to completion, it doesn't disappear, the character needs to drop it if she wants to use another key or pickup another match. Keys can be:
 - Single-use keys, can be used only once.
 - Double-use Keys, can be used twice.
 - Master Keys, can be used infinitely often.

PDDL Domain Template

```
8
9
10 (:types
11   matches keys - items
12   cells
13   colour
14 )
15
16 (:predicates
17   ;Indicates the number of uses left in a key
18   (key-infinite-uses ?k - keys)
19
20   (key-two-use ?k - keys)
21
22   (key-one-use ?k - keys)
23
24   (key-used-up ?k - keys)
25
26   ;Add other predicates needed to model this domain
27
28 )
29
30 ;Hero can move if the
31 ; - hero is at current location
32 ; - cells are connected,
33 ; - there is no monster in current loc and destination, and
34 ; - destination is not invigilated
35 ;Effects move the hero, and the original cell becomes invigilated.
36 (:action move
37   :parameters (?from ?to - cells)
38   :precondition (and
39     ;Fill this empty precondition
40   )
41   :effect (and
42     ;Fill this empty effect
43   )
44 )
45
46 --
```

Multi-step procedure to increase the chance for LLMs to correctly model the transition system.

Problem setting: We assume that *Problem PDDL model* is provided, *type*, *predicate*, *action name* and *action parameters* are also provided. You only ask LLMs to generate the transition model (i.e., precondition and effect of each actions)

Note

You can try to relax those assumptions if you have extra time. E.g., Ask LLMs to generate Problem PDDL model from natural language descriptions.

Step 1: In context learning for LLMs

Important

We will use Huggingface transformers library to **deploy** your LLMs <https://huggingface.co/docs/transformers/index>

We provide few-shot examples to help LLMs **learn** how to map natural language (NL) descriptions to structured preconditions and effects snippets

To achieve this, we collect (a) NL descriptions of actions and their PDDL domain skeleton template along with (b) their corresponding ground-truth action schemas. This data is then used as few-shot prompting

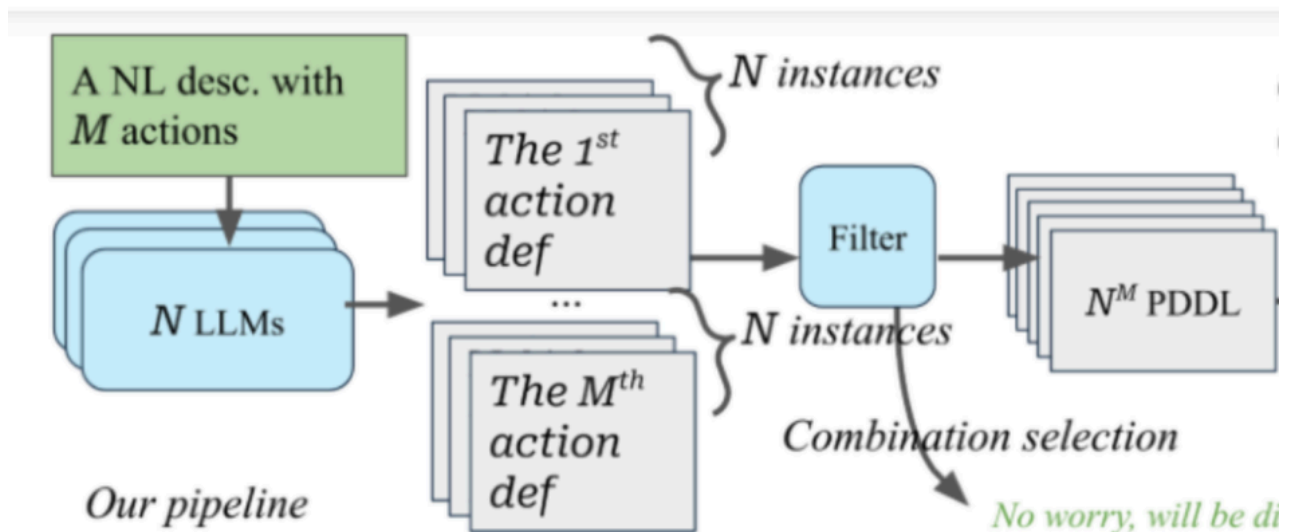
Tip

Check <https://huggingface.co/docs/transformers/main/en/tasks/prompting#few-shot> to see how to do In context learning for LLMs

For training data collection, you can collect it by yourself, or use mine. https://github.com/Sino-Huang/Official-LLM-Symbolic-Planning-without-Experts/tree/master/data/01_raw/pddl_domain

Step 2: Multiple LLMs and Combinatorial Selection

Unity makes strength

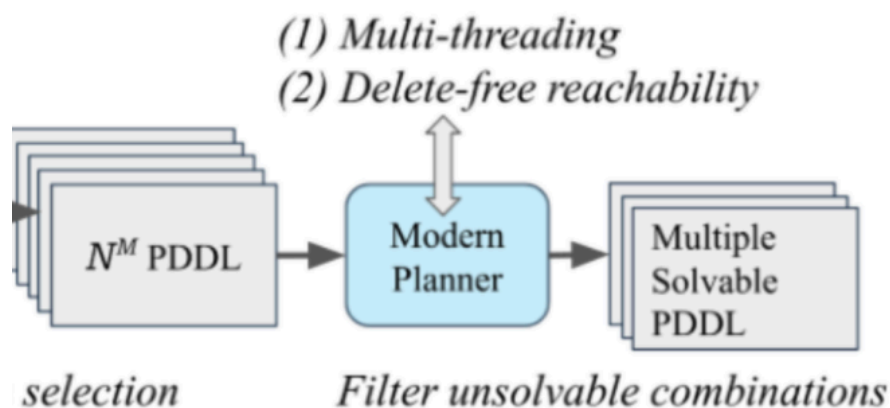


The likelihood of obtaining **at least one solvable** Domain PDDL is significantly increased.

💡 Tip

check https://huggingface.co/docs/transformers/generation_strategies#customize-text-generation to see how to let LLMs to generate multiple and diverse outputs

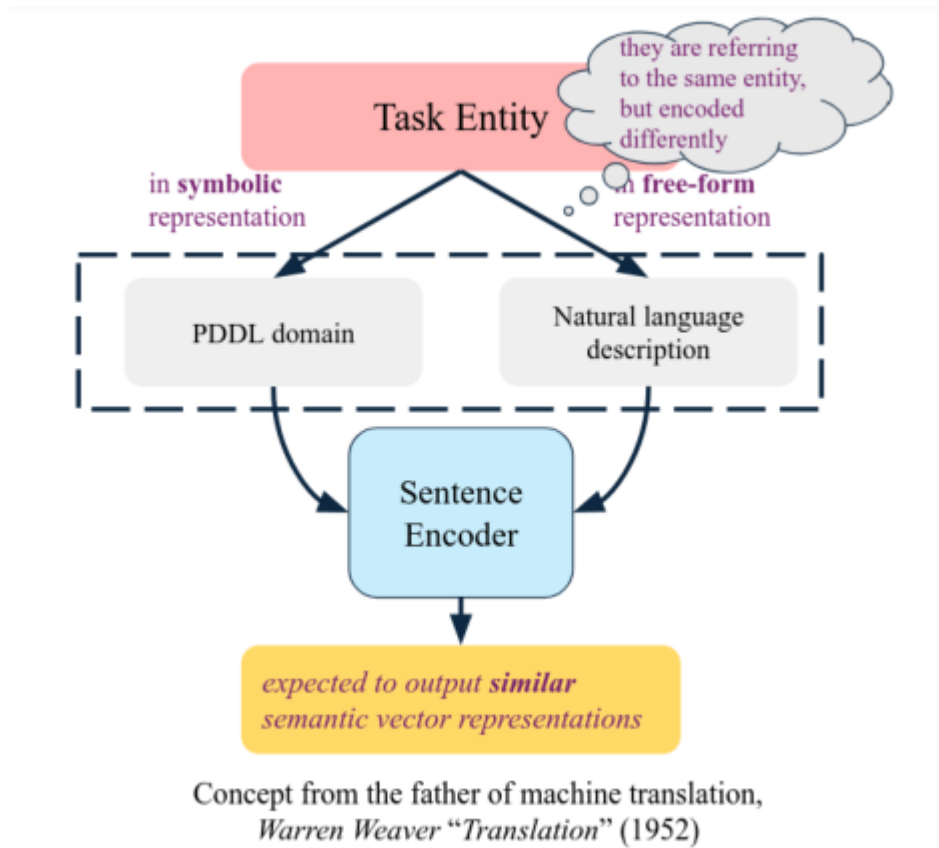
Step 3: Verify whether the generated Domain PDDL can solve a set of problem instances and filter out unsolvable ones



Step 4 (Optional, implement it if you have time): Ensure semantic alignment -- leveraging sentence encoder

📌 Important

A sentence encoder converts textual data into latent vectors that capture the semantic meaning of the textual content



LLMs may not always guarantee that the generated action definitions are semantically aligned with the natural language descriptions. Therefore, we propose using a sentence encoder to filter out misaligned action schema snippets.

Finetuning semantic encoder based on contrastive learning

1. we have ground truth action schema snippets as positive examples
2. we can create negative examples through manipulating ground-truth action schema (I have the [codebase](#) to do this manipulation to generate negative examples, no worry)

Manipulation Type	Description	Example
Swap	Exchanges a predicate between preconditions and effects	Precondition: (at ?x ?y) Effect: (not (at ?x ?z)) → Precondition: (not (at ?x ?z)) Effect: (at ?x ?y)
Negation	Negates a predicate in either preconditions or effects	Precondition: (clear ?x) → Precondition: (not (clear ?x))
Removal	Removes a predicate from either preconditions or effects	Precondition: (and (on ?x ?y) (clear ?x)) → Precondition: (on ?x ?y)
Addition	Adds mutually exclusive (mutex) predicates to preconditions or effects (Helmert 2009)	Effect: (on-table ?x) → Effect: (and (on-table ?x) (holding ?x))

💡 Tip

Library for finetuning sentence encoders: [sentence-transformers](#)

Second Phase

We may use the similar procedure for Planimation Animation Profile Construction

Tip

Documents for Planimation Animation Profile Construction https://planimation.github.io/documentation/ap_guide/