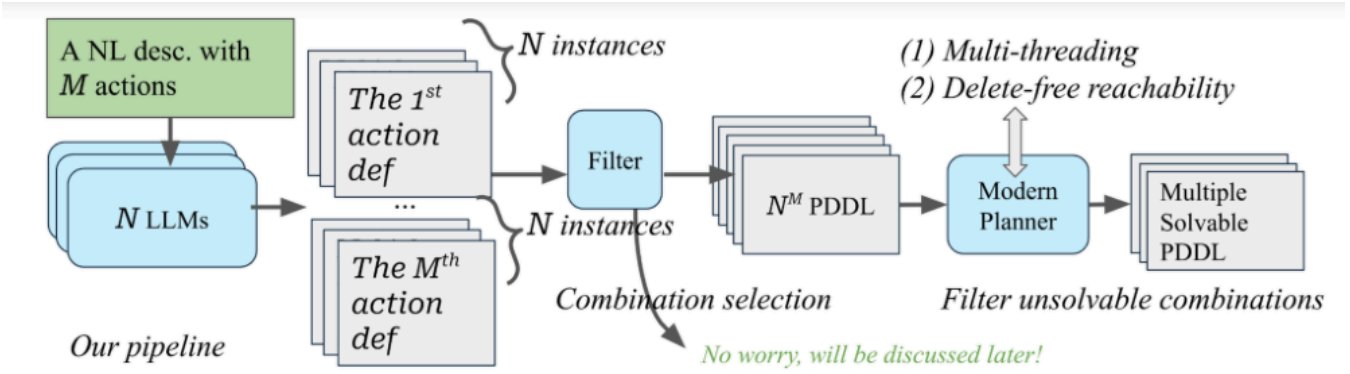
Introduction of the LLM for PDDL modeling pipeline



Background

COMP90054-AI-Planning-for-Autonomy assignment 2

> **Your task** is to read the description of the 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

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., precon 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

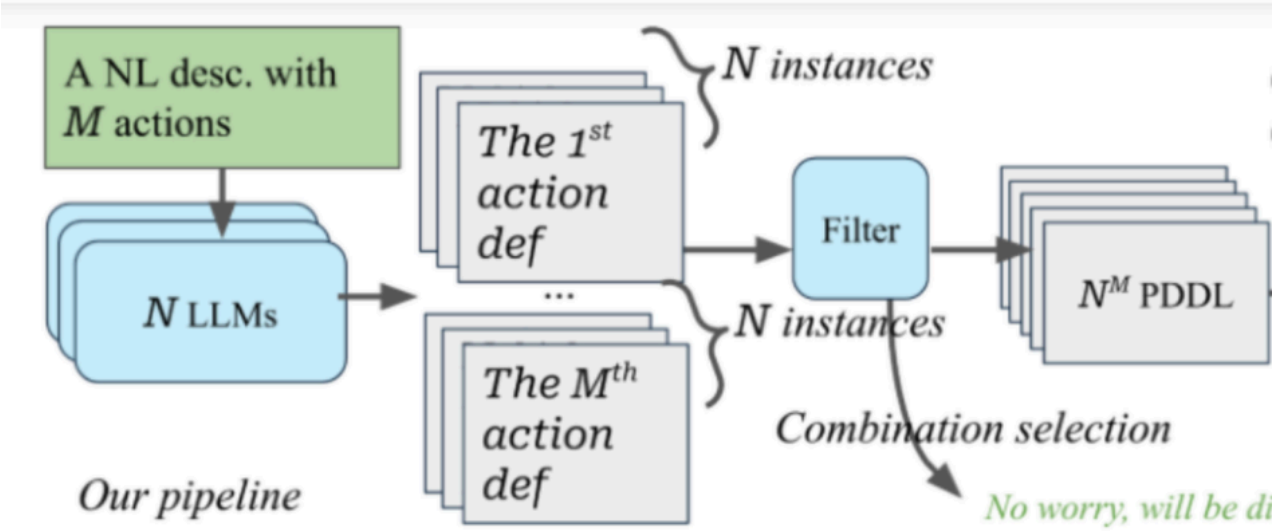
We provide few-shot examples to help LLMs **learn** how to map natural language (NL) descriptions to structured preconditions and effects snippets

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

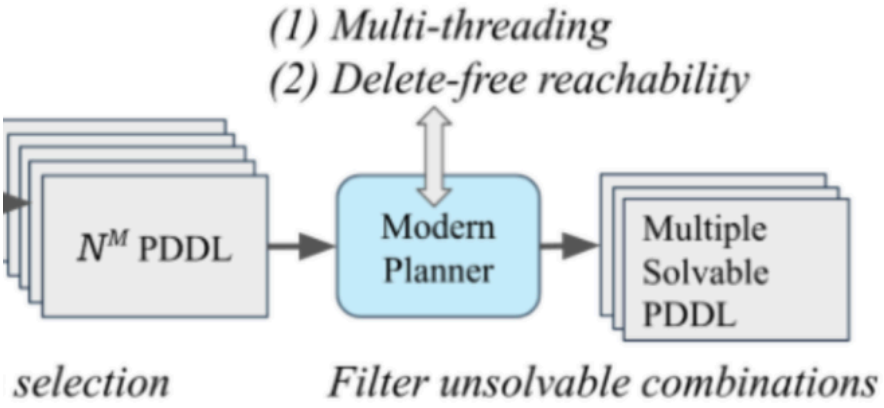
Unity makes strength



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

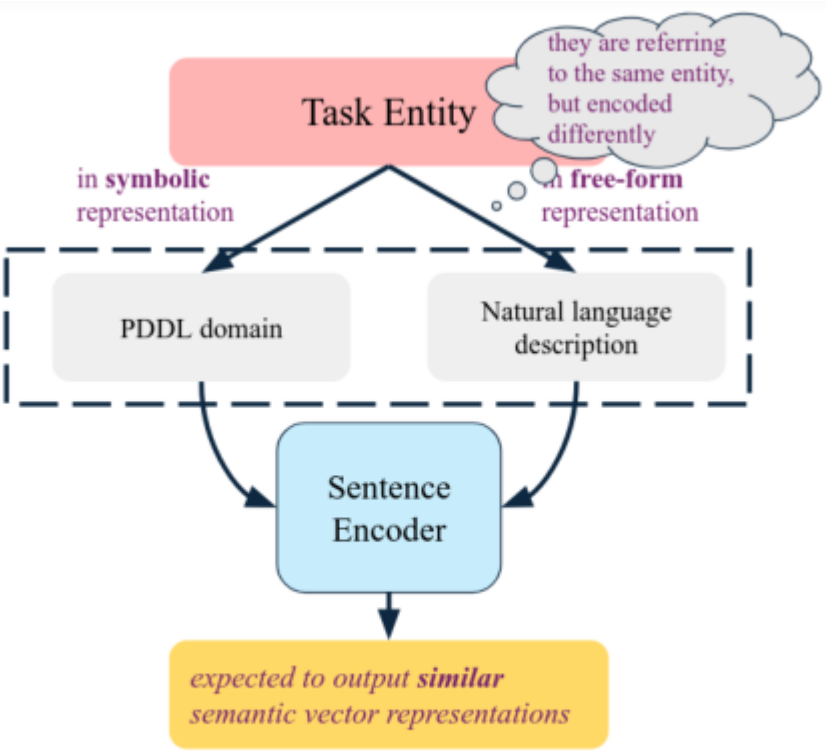
> **Tip**
>
> check https://huggingface.co/docs/transformers/generation_strategies#customize-text-generation to see how to let LLMs to generate multiple and diverse outputs

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



Concept from the father of machine translation,
Warren Weaver "Translation" (1952)

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/