Harnessing Large Language Models for Planning

Vishal Pallagani
Class 23, CSCE 580

UNIVERSITY OF South Carolina

IBM Research

CSCE 580

Contents

01.

Introduction

Why Explore LLMs for Planning?

Large Language Models (LLMs) have demonstrated remarkable proficiency across diverse domains, excelling in natural language understanding, text generation, and even code generation tasks.

Planning problems involve the structured representation of tasks and goals, which are commonly expressed using Planning Domain Definition Language (PDDL). PDDL shares a logical and symbolic nature similar to Lisp programming.

Given LLMs' proven proficiency in programming code generation and understanding, it becomes imperative to explore their capabilities in comprehending PDDL and generating plans, tasks that demand even greater levels of reasoning.

02.

Brief Overview of Symbolic Planners and Plan Validators

Complex Decisions

- Making a sequence of decisions
  - Single actor or others in environment
- Making a single decision but with
  - Environment changing, not being observable
  - Actions not being deterministic, having durations
  - Perception not being perfect
  - All goals to be satisfied

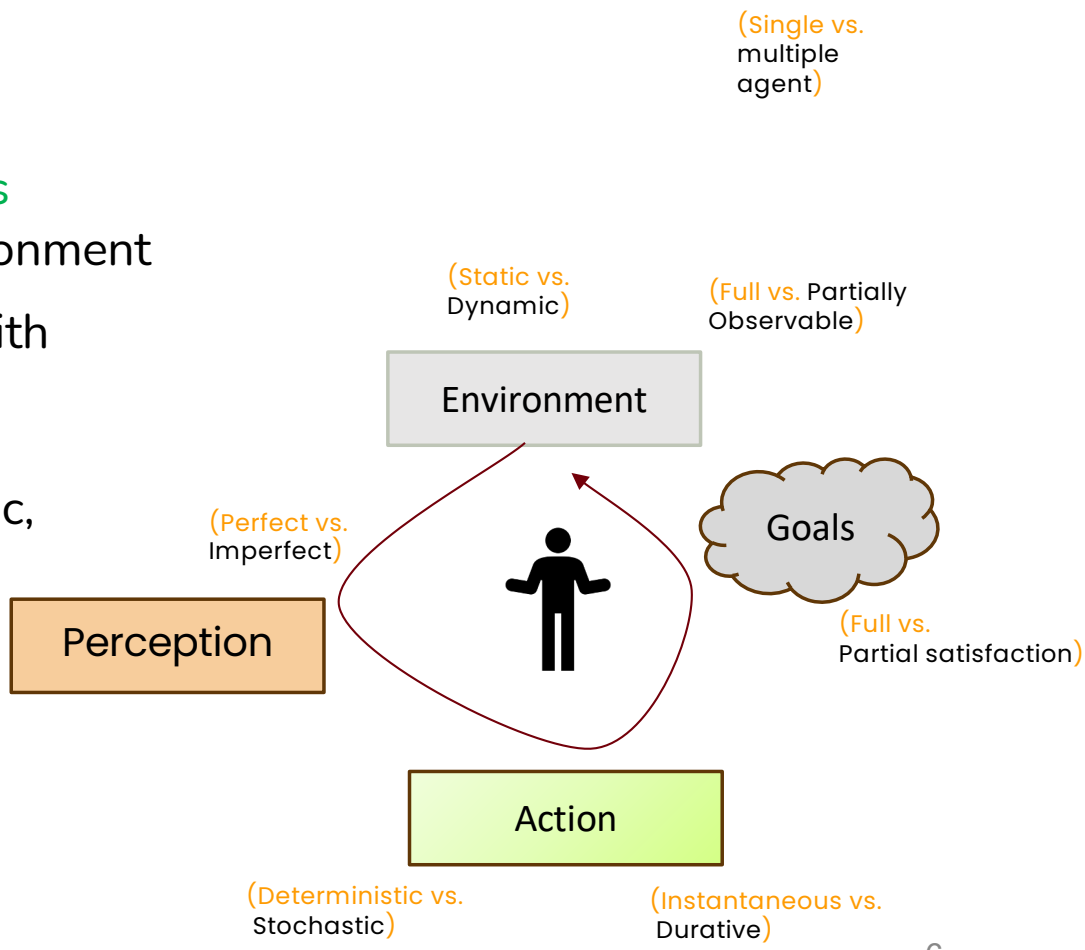
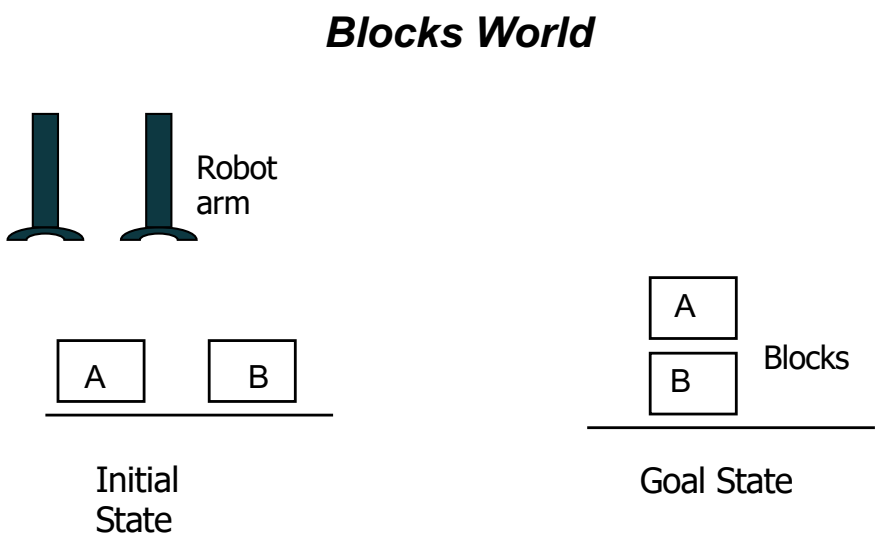


Illustration of a Planning Scenario

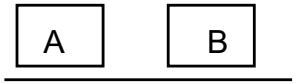


All robots are equivalent

Illustration of Problem Representation

States: ((On-Table A) (On-Table B) …)

Actions: ((Name: (Pickup ?block ?robot)
Precondition: ((Clear ?block)
(Arm-Empty ?robot)
(On-Table ?block))
Add: ((Holding ?block ?robot))
Delete: ((Clear ?block)
(Arm-Empty ?robot)))…)



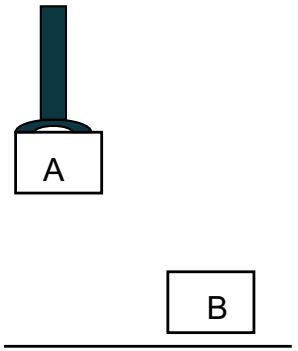


Illustration of Reasoning for Planning

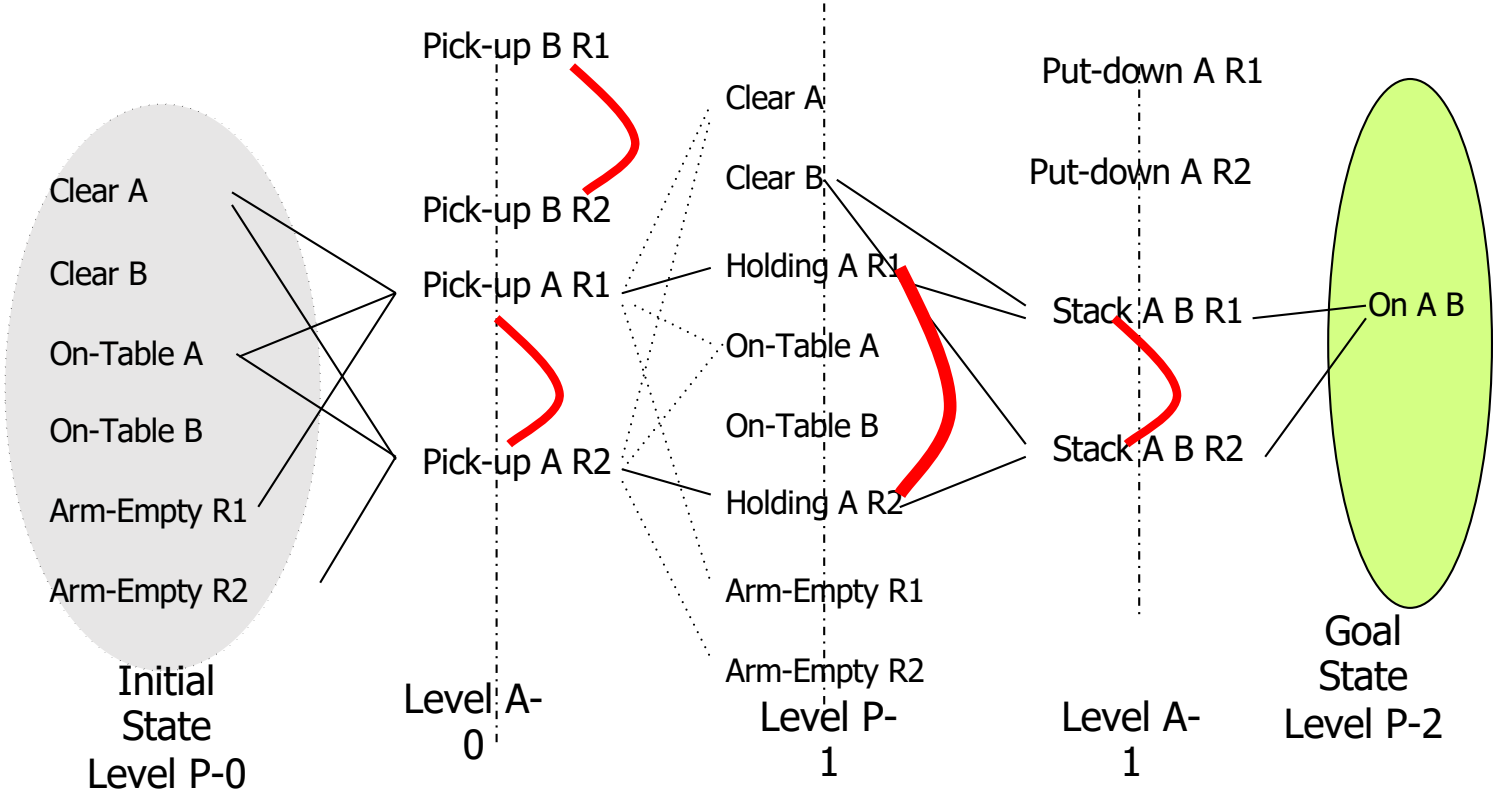


Illustration of a Larger Planning Scenario

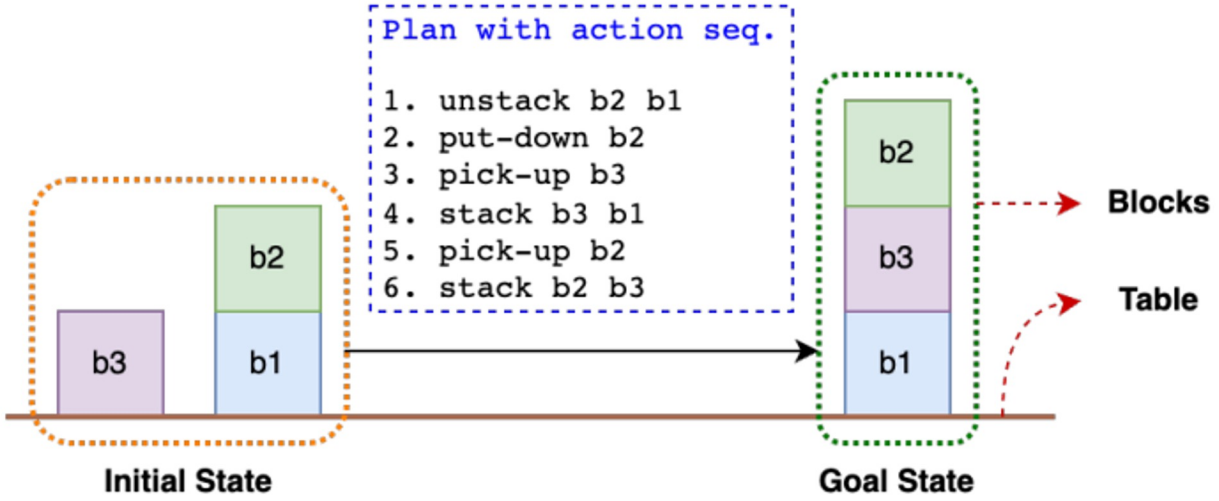


Figure. Demonstration of automated planning problem with blocksworld domain example

Active Areas of Research

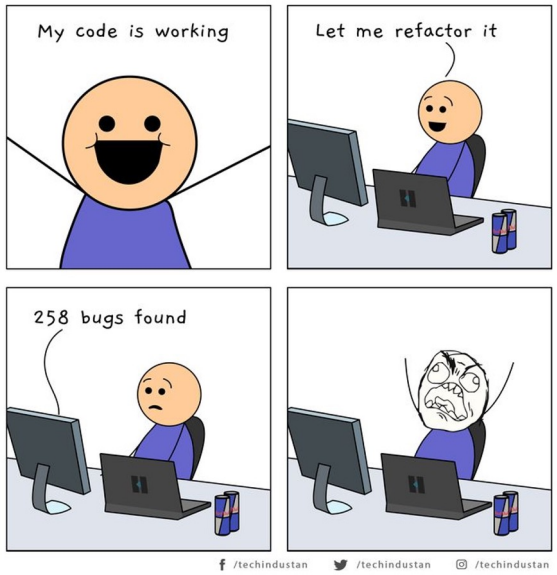
Considerations

- What plan to find?
  - **Any workable plan**
  - Optimal plan – but then what is the criteria
  - All plans
  - Diverse plans
- When to find plan?
  - **Plan at the end**
  - Plan anytime
- How to represent problem ?
  - **Planning Domain Description Language**
- How to explain solution ?

Plan Validation

- **Definition:** Plan validation refers to the process of assessing whether a generated plan satisfies the specified planning problem and domain constraints.
- **Objective:** Ensure that the plan is feasible, correct, and efficient in achieving the desired goals.
- **Key Components:**
  - Syntax Check: Verify that the plan adheres to the syntax rules defined in PDDL.
  - Semantics Check: Ensure that the plan semantics are consistent with the domain and problem definitions.
  - Goal Achievement: Confirm that the plan accomplishes all specified goals within the given constraints.
- **Tools for Classical Planning:**
  - Plan Validator - KCL-Planning/VAL: The plan validation system. (github.com)
  - INVAL - patrikhaslum/INVAL: The INVAL plan validator, and other PDDL tools. (github.com)

>> It is time for some coding



Source: r/ProgrammerHumor

03.

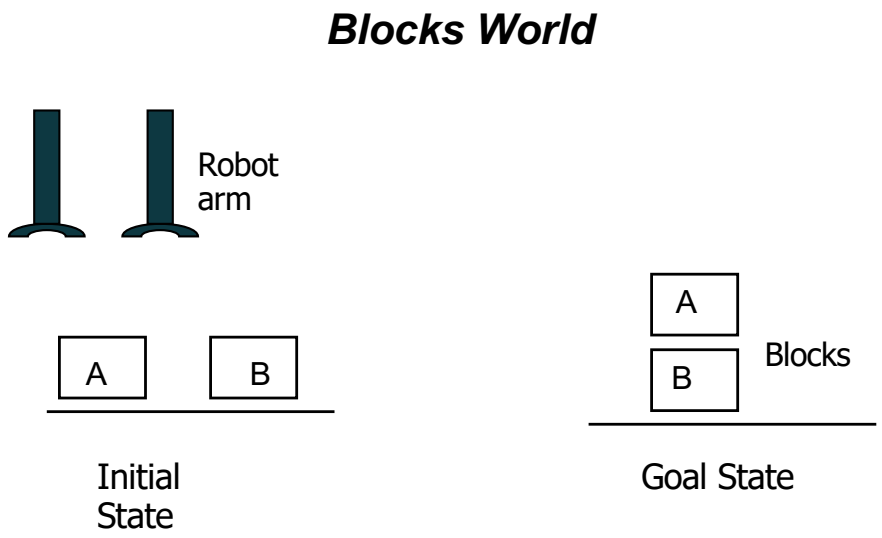
Deep-dive into LLMs

ChatGPT Time

Ask ChatGPT to Generate a Plan

- What should be the prompt ?
- What can one verify the output ?

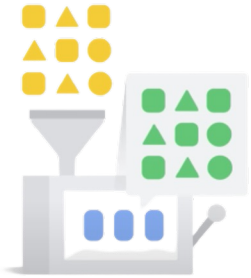
- Expected plan ?
  - Pickup-A-R1 // or R2
  - Stack-A-B-R1 // or R2



Now, from the Basics

Deep-dive into LLMs

- Large
  - Large training dataset
  - Large number of parameters
- General purpose
  - Commonality of human languages
  - Resource restriction
- Pre-trained and fine-tuned



Credits: Google Cloud Skills Boost

In this lab, we will be focussing our discussion on three different language modeling techniques, namely, masked, seq2seq, and causal.

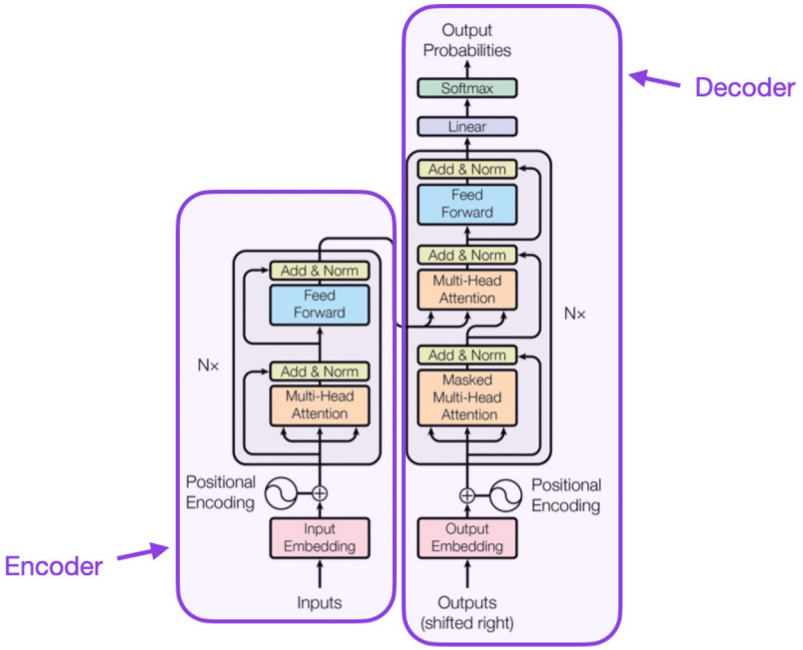
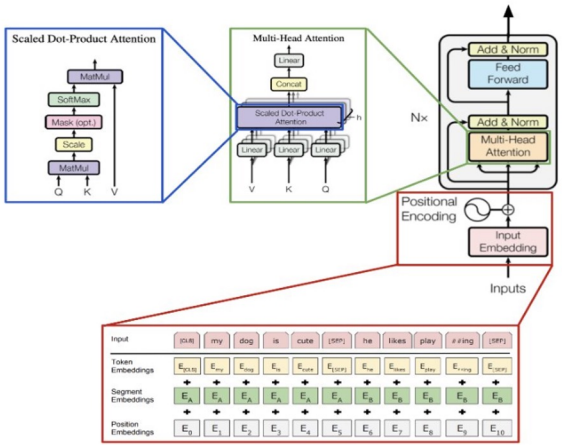


Figure. The Transformer - model architecture.

Masked Language Modeling (MLM)

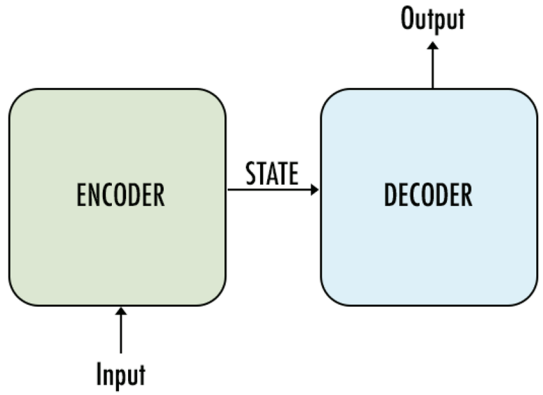
- MLMs like BERT are trained to understand the bidirectional context by predicting words randomly masked in a sentence.
- This approach allows the model to learn both forward and backward dependencies in language structure.
- MLMs have proven effective in NLP tasks such as sentiment analysis or question answering.
- MLMs are encoder-only.



Credits: Cameron R. Wolfe
Source: Language Model Training and Inference: From Concept to Code (substack.com)

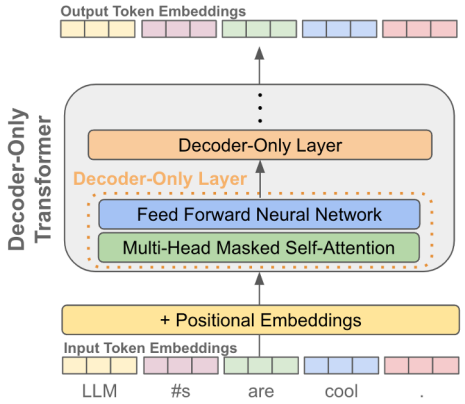
Seq2Seq Language Modeling (Seq2Seq)

- Seq2Seq models, like T5, are designed to transform an input sequence into a related output sequence.
- They are often employed in tasks that require a mapping between different types of sequences, such as language translation or summarization.
- They consist of an encoder-decoder architecture.



Causal Language Modeling (CLM)

- CLMs, such as GPT4, are designed for tasks where text generation is sequential and dependent on the preceding context.
- They predict each subsequent word based on the preceding words, modeling the probability of a word sequence in a forward direction.
- This characteristic makes CLMs particularly suitable for applications like content generation, where the flow and coherence of the text in the forward direction are crucial.
- CLMs consist of a decoder-only architecture.

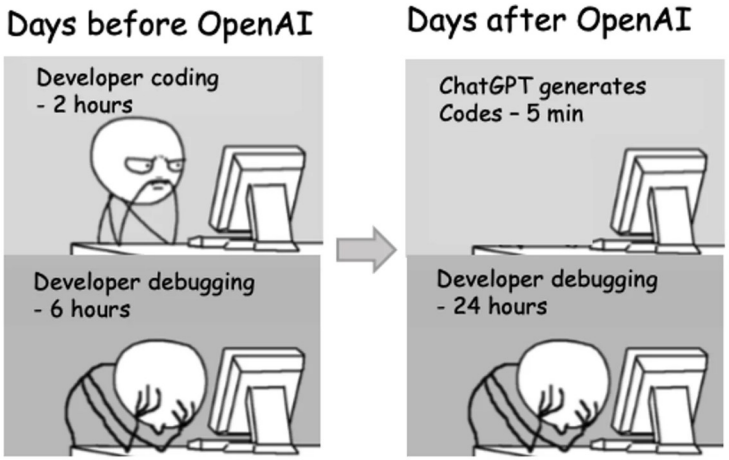


Credits: Cameron R. Wolfe
Source: Language Model Training and Inference: From Concept to Code (substack.com)

Given this basic description of three language modeling techniques used to build LLMs, what among them do you think is a good fit for **plan generation?**

Think in terms of the input and output.

>> It is time for some coding



Source: r/ProgrammerHumor

04.

Plansformer

[Hands-on session]

Plansformer

What 

Plansformer [1,2], is an LLM based planner that is capable of generating valid and optimal plans for classical planning problems.

Why 

Traditional planners, although sound and complete, often cannot solve problems with vast space in a stipulated time or generalize. A learning based planner, such as Plansformer, is extremely good at harnessing learnt representations to generalize to unseen problems, and solve any problem in constant time.

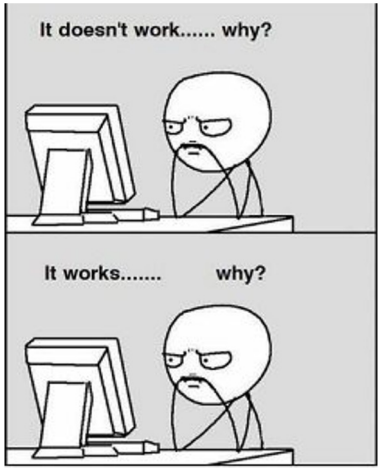
How 

Plansformer is obtained by fine-tuning CodeT5 on planning problems and their corresponding optimal plans. CodeT5 is an LLM that is pre-trained on programming language and associated natural language.

[1] Pallagani, V., Muppasani, B., Murugesan, K., Rossi, F., Horesh, L., Srivastava, B., Fabiano, F. and Loreggia, A., 2023. Plansformer: Generating symbolic plans using transformers. Generalized Planning (GenPlan) Workshop at NeurIPS.

[2] Pallagani, V., Muppasani, B., Srivastava, B., Rossi, F., Horesh, L., Murugesan, K., Loreggia, A., Fabiano, F., Joseph, R. and Kethepalli, Y., 2023, August. Plansformer tool: demonstrating generation of symbolic plans using transformers. In IJCAI (Vol. 2023, pp. 7158-7162). International Joint Conferences on Artificial Intelligence..

>> It is time for some coding



Source: r/ProgrammerHumor

Plansformer's Architecture

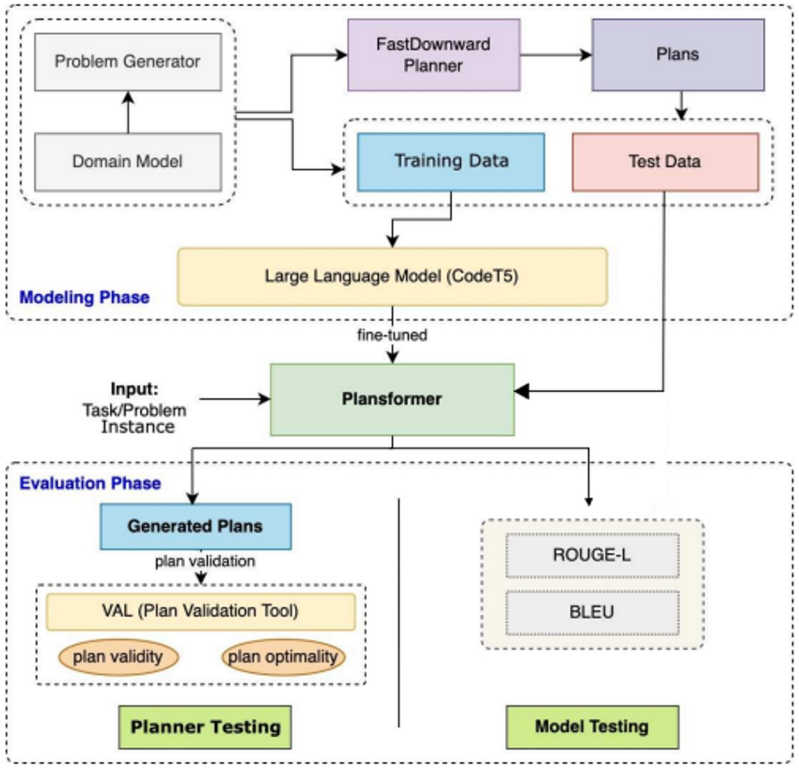


Figure. Plansformer Model Architecture showing modeling and evaluation phases. Modeling phase involves fine-tuning CodeT5 with data from planning domain. Evaluation phase shows both the planner and model testing.

Is Plansformer a Good Model?

Models	ROUGE-L_{recall}	ROUGE-$L_{precision}$	ROUGE-$L_{fmeasure}$	BLEU
Codex	0.72	0.52	0.60	0.36
GPT-2	0.04	0.14	0.06	0.07
T5-base	0.16	0.70	0.26	0.02
CodeT5-base	0.41	0.28	0.33	0.02
Plansformer	**0.93**	**0.93**	**0.93**	**0.89**
Plansformer-bw	0.97	0.99	0.98	0.90
Plansformer-hn	0.99	0.96	0.97	0.95
Plansformer-gr	0.94	0.94	0.94	0.92
Plansformer-dl	0.82	0.83	0.82	0.79

Table. Results of model testing (best performance in bold). Plansformer-[x] denotes Plansformer for a specific domain.

Is Plansformer a Good Planner?

Models	Valid Plans (%)	Invalid Plans		Optimal Plans (%)	Avg. Time (sec)
		Incomplete/Wrong (%)	Failed (%)		
FastDownward (Ground Truth)	100%	-	-	100%	10.28s
Codex	0.15%	0%	99.85%	0.15%	1s
GPT-2	0%	100%	0%	0%	0.05s
T5-base	0.25%	82.7%	17.3%	0.25%	0.47s
CodeT5-base	0.6%	99.4%	0%	0.6%	0.68s
Plansformer	**83.64%**	16.18%	0.19%	**73.27%**	**0.06s**
Plansformer-bw	90.04%	9.94%	0.02%	88.44%	0.05s
Plansformer-hn	84.97%	14.72%	0.31%	82.58%	0.05s
Plansformer-gr	82.97%	16.61%	0.42%	69.47%	0.06s
Plansformer-dl	76.56%	23.44%	0%	52.61%	0.09s

Table. Results of plan validation.

Can Plansformer Adapt to Another Domain?

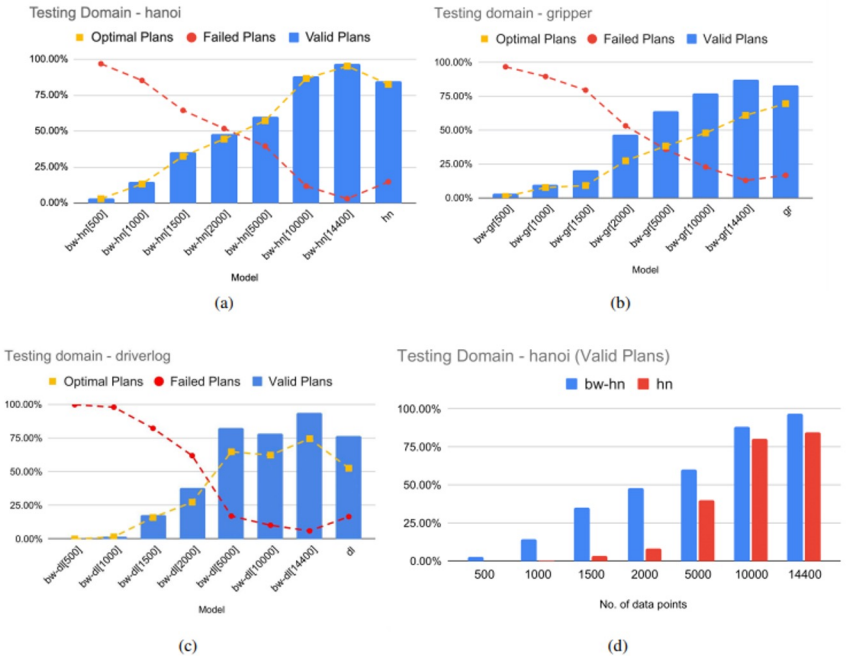


Table. Plansformer-**bw** as the base model fine-tuned with and tested on (a) **hanoi** (b) **grippers** (c) **driverlog**, and (d) shows the comparison of valid plans generated by Plansformer-**bw-hn** derived models with Plansformer-**hn** trained using similar data points.

05.

Overview of LLMs in Planning

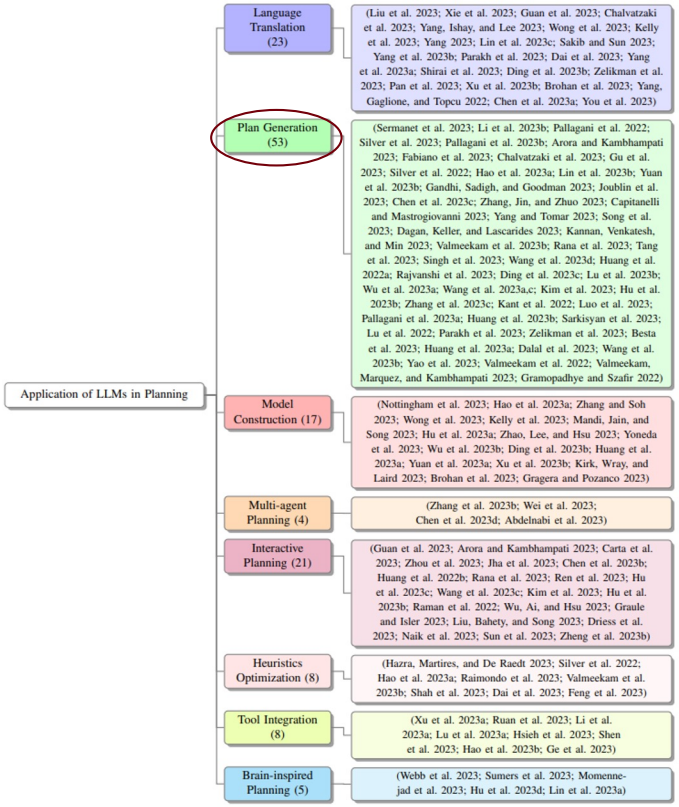
Applications of LLMs in Planning

Category	Description
Language Translation	Involves converting natural language into structured planning languages or formats like PDDL and vice-versa, enhancing the interface between human linguistic input and machine-understandable planning directives.
Plan Generation	Entails the creation of plans or strategies directly by LLMs, focusing on generating actionable sequences or decision-making processes.
Model Construction	Utilizes LLMs to construct or refine world and domain models essential for accurate and effective planning.
Multi-agent Planning	Focuses on scenarios involving multiple agents, where LLMs contribute to coordination and cooperative strategy development.
Interactive Planning	Centers on scenarios requiring iterative feedback or interactive planning with users, external verifiers, or environment, emphasizing the adaptability of LLMs to dynamic inputs.
Heuristics Optimization	Applies LLMs in optimizing planning processes through refining existing plans or providing heuristic assistance to symbolic planners.
Tool Integration	Encompasses studies where LLMs act as central orchestrators or coordinators in a tool ecosystem, interfacing with planners, theorem provers, and other systems.
Brain-Inspired Planning	Covers research focusing on LLM architectures inspired by neurological or cognitive processes, particularly to enhance planning capabilities.

Table. Comprehensive description of the eight categories utilizing LLMs in Planning [1]

[1] Pallagani, V., Roy, K., Muppasani, B., Fabiano, F., Loreggia, A., Murugesan, K., Srivastava, B., Rossi, F., Horesh, L. and Sheth, A., 2024. On the prospects of incorporating large language models (llms) in automated planning and scheduling (aps). International Conference on Automated Planning and Scheduling (ICAPS).

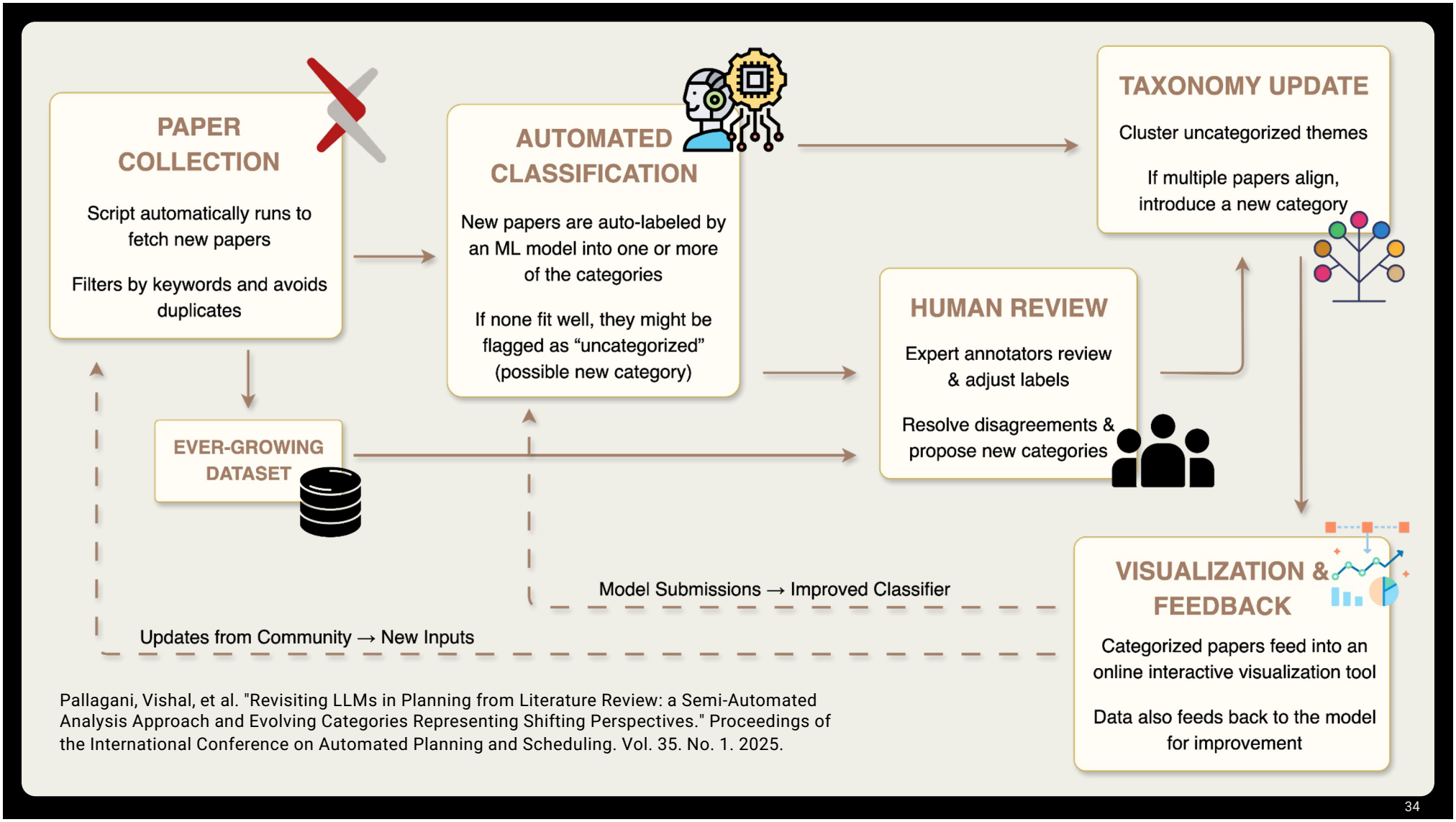
Applications of LLMs in Planning



In this tutorial, we focus on using LLMs for plan generation

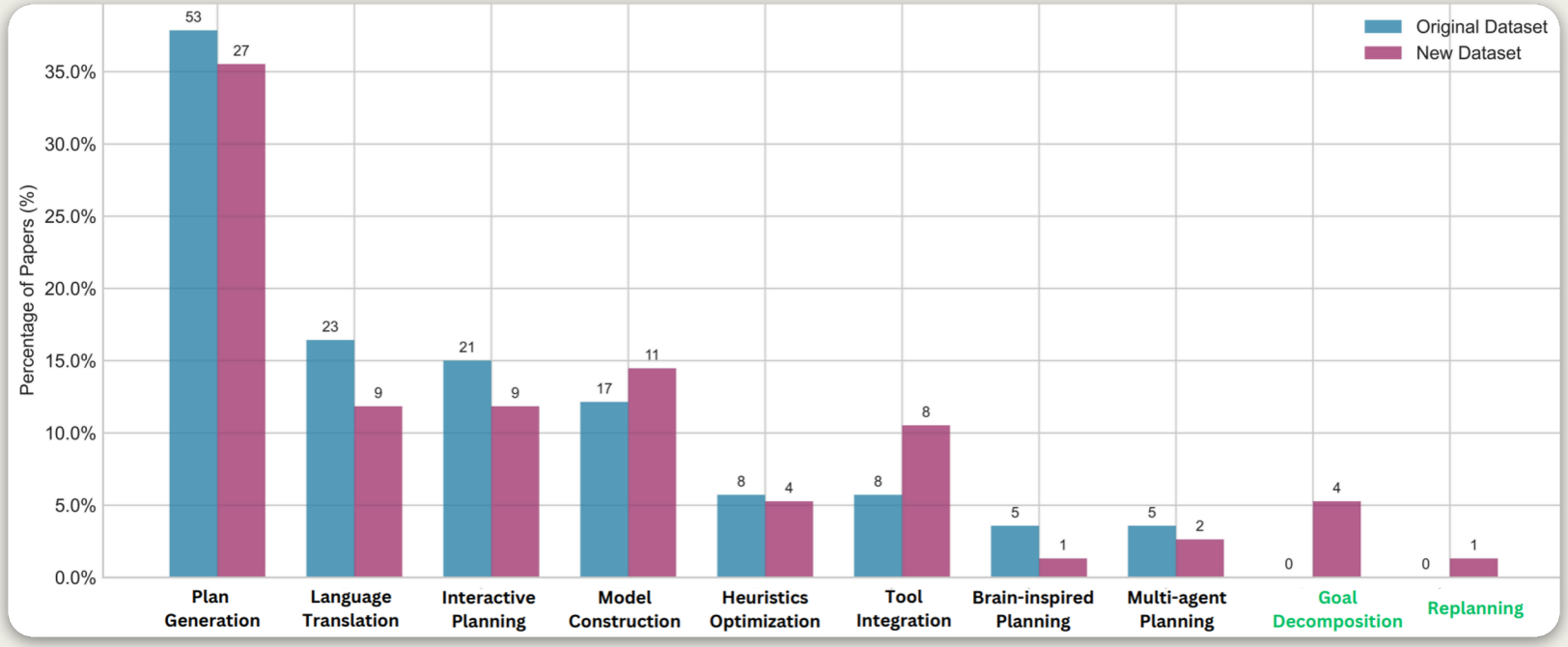
[1] Pallagani, V., Roy, K., Muppasani, B., Fabiano, F., Loreggia, A., Murugesan, K., Srivastava, B., Rossi, F., Horesh, L. and Sheth, A., 2024. On the prospects of incorporating large language models (llms) in automated planning and scheduling (aps). International Conference on Automated Planning and Scheduling (ICAPS).

Figure. Taxonomy of recent research in the intersection of LLMs and Planning with (#) mentioning the number of scholarly papers in each category [1].



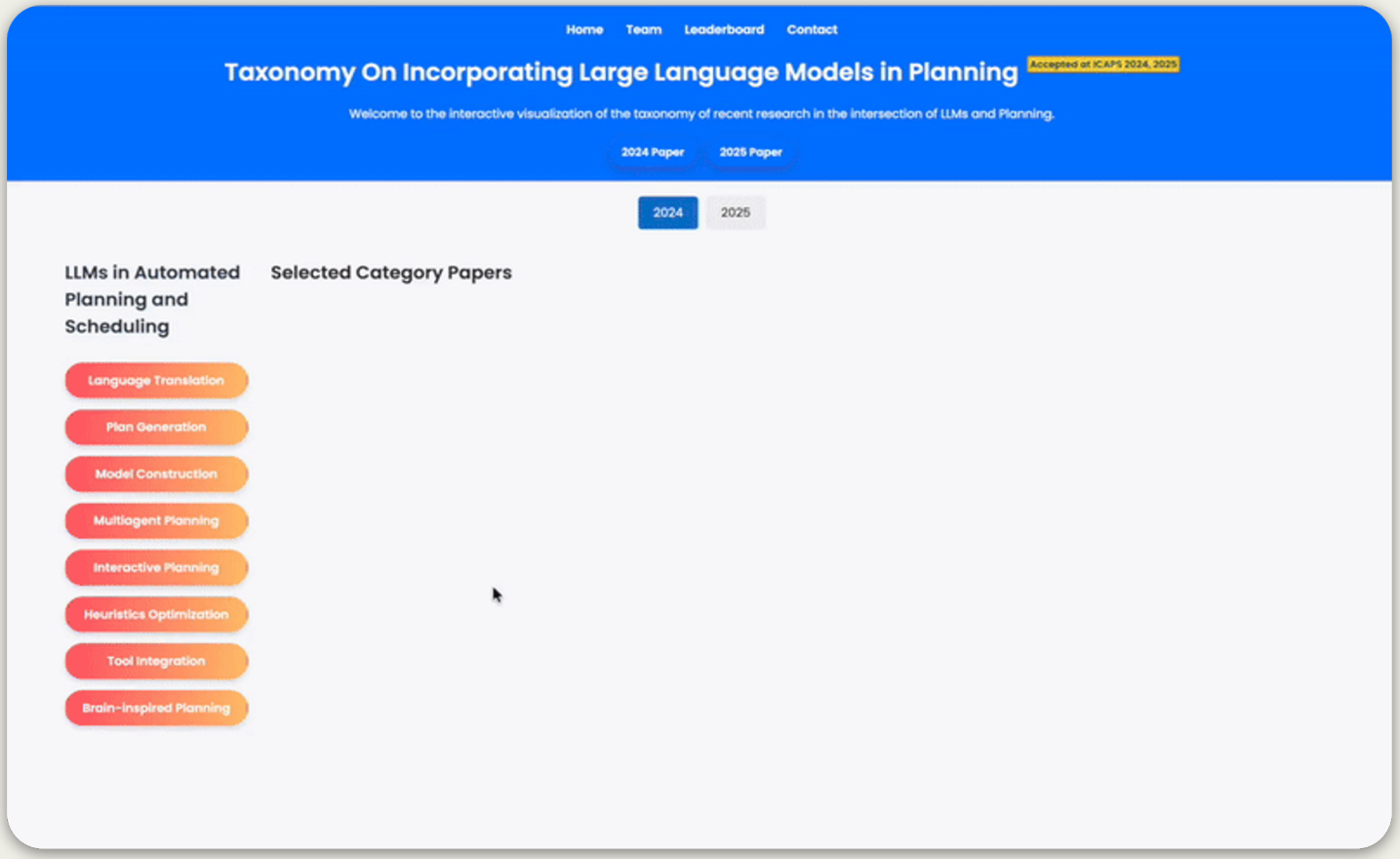
Pallagani, Vishal, et al. "Revisiting LLMs in Planning from Literature Review: a Semi-Automated Analysis Approach and Evolving Categories Representing Shifting Perspectives." Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 35. No. 1. 2025.

Results – Category Distribution & Drift



6 of the original 8 categories saw their paper share decrease, **2 increased**, and importantly **2 new categories emerged**. This reflects a shift in community interest: moving toward practical, integrative uses of LLMs (model building, tool use) and away from some earlier explorations that proved less fruitful.

Interactive Visualization Platform



https://ai4society.github.io/LLM-Planning-Viz/

Capabilities of LLMs for Plan Generation

- A majority of the works make use of prompting generative (causal) language models to generate plans.
- However, none of them have explored fine-tuning or looked beyond generative (causal) models for planning.
- In this lab, we will provide an understanding on the capabilities of LLMs in their ability to solve for plans given a PDDL problem as input using both fine-tuning and prompting approaches.

Fine-tuning is a method that updates the parameters of an LLM using a labeled dataset for the target task.

Prompting is a method that modifies the input of an LLM using a template or a cue to elicit the desired output.

Capabilities of LLMs for Plan Generation

We focus on answering four research questions [1]:

(a) To what extent can LLMs solve planning problems?

(b) What pre-training data is effective for plan generation?

(c) Does fine-tuning and prompting improve LLMs plan generation?

(d) Are LLMs capable of plan generalization?

[1] Pallagani, V., Muppasani, B., Murugesan, K., Rossi, F., Srivastava, B., Horesh, L., Fabiano, F. and Loreggia, A., 2023. Understanding the Capabilities of Large Language Models for Automated Planning. *arXiv preprint arXiv:2305.16151*.

Capabilities of LLMs for Plan Generation

For this study, we constructed a dataset comprising 18,000 planning problems along with their corresponding optimal plans across 6 domains. The dataset was divided into an 80%-20% train-test split, with the purpose of fine-tuning and evaluating the performance of the LLMs.

Planning Domain	Difficulty	State Space	Branching Factor
`Ferry`	Easy	$O(2^n * 2 * m * n!)$, n is no. of cars, m is no. of locations	$O(n+1)$
`Blocksworld`	Easy	$O(3^n)$, n is no. of blocks	$O(4n/2+1)$
`Miconic`	Medium	$O(n^{(m+1)} * 2^m * m!)$, n is no. of floors, m is no. of passengers	$O(m+1)$
`Tower of Hanoi`	Medium	$O(3^n)$, n is no. of disks	$O((k-1)k/2)$, k is no. of pegs
`Grippers`	Hard	$O(2^n * 3^{nr})$, n is no. of balls, r is no. of robots	$O(3nr+r)$
`Driverlog`	Hard	$O(L^{(D+T+P)} * K^P * D * T * 2^T)$, L is no. of locations, D is no. of drivers, T is no. of trucks, P is no. of packages	$O(L*(D+T+P+DT+TD))$

Table. Difficulty of planning domains

Capabilities of LLMs for Plan Generation

The LLMs along with their architectures considered for this study are as follows

Model-Parameters	Pre-training	Layers	Heads	Embed. Size	Context Length
T5-base-220M	NL	12	12	768	512
CodeT5-base-220M	NL+code	12	12	768	512
Codegen-multi-350M	NL+code	24	16	1024	1024
text-davinci-03	NL	NA	NA	NA	8000
code-davinci-02	NL+code	NA	NA	NA	8000

Table. Architecture of benchmark LLMs

Research Question 1: To what extent can LLMs solve planning

Models	Type	Inf. Time	Sat. Plans (%)			Opt. Plans (%)			Deg. Corr.		
			E	M	H	E	M	H	E	M	H
T5	Pre-trained	4.03s	0	0	0	0	0	0	0	0	0
	Fine-tuned	1.59s	0.11	0	1.16	0.11	0	0.36	0.02	0	0.03
CodeT5	Pre-trained	4.22s	2.70	0.6	0	1.73	0.6	0	0.07	0	0
	Fine-tuned	**1.50s**	**97.57**	**92.46**	**89.54**	**86.21**	**90.36**	**66.71**	**0.99**	**0.95**	**0.95**
Codegen	Pre-trained	4.52s	1.70	0.16	0	0.17	0	0	0.01	0.01	0
	Fine-tuned	1.93s	37.95	23.74	9.83	35.33	11.67	1.92	0.67	0.32	0.07
text-davinci	Zero-shot	3.89s	8.78	6.43	0	3.92	0	0	0.27	0.21	0
	Few-shot	3.88s	10.82	6.90	3.31	7.23	2.84	1.21	0.32	0.17	0.04
code-davinci	Zero-shot	3.58s	17.42	11.77	4.38	8.23	6.11	1.84	0.38	0.27	0.21
	Few-shot	3.51s	23.52	17.48	11.89	17.57	8.85	4.69	0.57	0.32	0.28

Table. Evaluation of plan generation capabilities of LLMs (both prompting pre-trained model and fine-tuned model). For each model, we report the inference time (Inf. Time), the percentage of satisficing plans (Sat. Plans), the percentage of optimal plans (Opt. Plans), and the degree of correctness (Deg. Corr.).

Research Question 2: What pre-training data is effective for plan generation?

LLMs pre-trained on programming code outperform those solely trained on the textual corpus.

Research Question 3: Does fine-tuning and prompting improve LLMs plan generation?

Fine-tuning is a superior approach to solving planning problems with LLMs. Overall, it has been observed that fine-tuned LLMs are capable of generating outputs for planning problems at a rate four times faster than pre-trained LLMs.

Research Question 4: Are LLMs capable of plan generalization?

There are three tasks that help measure the capability of LLMs in plan generalization -

- **Task 1** - **Plan Length Generalization:** Evaluate the ability of LLMs to generalize to plan lengths that are out of distribution from the training set.
- **Task 2** - **Object Name Randomization:** Test LLMs ability to generate plans using randomized object names not present in the training set.
- **Task 3** - **Unseen Domain Generalization:** Evaluate LLMs ability to generalize to planning problems from new domains that are not included in the training set.

Research Question 4: Are LLMs capable of plan generalization?

Task 1: Plan length generalization

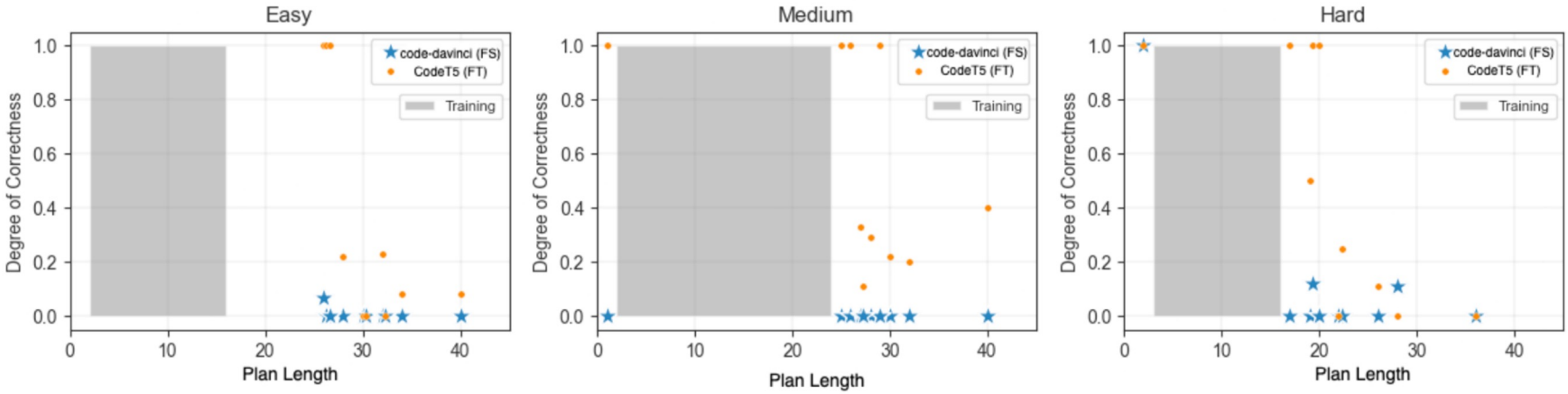


Figure. Fine-tuned CodeT5 and code-davinci with few-shot prompting show poor plan length generalization capabilities: plans from fine-tuned CodeT5 overall have a higher degree of correctness. The x-axis represents the plan length and the y-axis represents the degree of correctness. The training plan lengths are highlighted in grey.

Fine-tuned CodeT5 model can generalize to plan lengths to some extent, while the few-shot prompting of code-davinci generates only a single valid plan

Research Question 4: Are LLMs capable of plan generalization? **Task 2:** Object name randomization

Object Names	Model	ST	Sat. Plans (%) / Opt. Plan (%)			Deg. Corr.			E_{pg}		
			E	M	H	E	M	H	E	M	H
Version 1	CodeT5(FT)	✗	47.12% / 33.78%	42.74% / 38.68%	39.58% / 21.22%	0.57	0.51	0.51	0.82	0.84	0.84
		✓	97.52% / 86.21%	92.46% / 90.36%	89.12% / 66.71%	0.98	0.95	0.95	0.15	0.18	0.18
	code-davinci(FS)	✗	23.52% / 17.57%	17.48% / 8.85%	11.89% / 4.69%	0.57	0.32	0.28	0.77	0.83	0.96
		✓	23.52%/ 17.57%	17.48% / 8.85%	11.89% / 4.69%	0.57	0.32	0.28	0.77	0.83	0.96
Version 2	CodeT5(FT)	✗	66.01% / 64.72%	61.98% / 52.80%	55.17% / 37.5%	0.79	0.72	0.58	0.47	0.49	0.67
		✓	97.52%/86.21%	92.46%/90.36%	89.12%/66.71%	0.98	0.95	0.95	0.15	0.18	0.18
	code-davinci(FS)	✗	23.52%/17.57%	17.48%/8.85%	11.89%/4.69%	0.57	0.32	0.28	0.77	0.83	0.96
		✓	23.52%/17.57%	17.48%/8.85%	11.89%/4.69%	0.57	0.32	0.28	0.77	0.83	0.96
Version 3	CodeT5(FT)	✗	11.82% / 2.10%	4.92% / 1.47%	0.17% / 0%	0.24	0.04	0.01	0.87	0.95	1
		✓	97.52%/86.21%	92.46%/90.36%	89.12%/66.71%	0.98	0.95	0.95	0.15	0.18	0.18
	code-davinci(FS)	✗	23.52%/17.57%	17.48%/8.85%	11.89%/4.69%	0.57	0.32	0.28	0.77	0.83	0.96
		✓	23.52%/17.57%	17.48%/8.85%	11.89%/4.69%	0.57	0.32	0.28	0.77	0.83	0.96

Figure. Evaluating the capabilities of LLMs in handling randomized object names. In version 1, we used only single-digit numeric values as object names. In version 2, we used alphanumeric strings of length 2 (similar to the convention followed by IPC generators), where the combinations of alphabets and numerals used were unseen during training. Version 3 consisted of object names named after three alphabets.

Fine-tuned models can only generalize to object names belonging to the same vocabulary as that of training, while few-shot prompting of code-davinci handles the randomized object names but has poor plan generation capabilities.

Research Question 4: Are LLMs capable of plan generalization?

Task 3: Unseen domain generalization

Ground Truth Plan - π^* (*Domain:* `logistics`)

```
load-airplane p2 a0 l1-0, fly-airplane a0 l1-0 l0-0, unload-airplane p2 a0 l0-0,
load-airplane p0 a0 l0-0, fly-airplane a0 l0-0 l1-0, unload-airplane p0 a0 l1-0
```

Fine-tuned CodeT5 (*Inference Time:* 1.12s)

```
load-airplane driver0 l0, load-truck package3 truck0 s0, fly-airplane truck0
s0 s1 driver0, load-truck package2 truck0 s1, unload-airplane p0 a0 l1-0,
drive-truck truck0 s1 s0 driver0, unload-truck package2 truck0 s0 ✗
```

Few-shot prompting code-davinci (*Inference Time:* 3.51s)

```
load-truck p0 t0 l0-0, load-truck p1 t1 l1-0, drive-truck t0 l0-0 l1-0 c1,
unload-truck p0 t0 l1-0, drive-truck t1 l1-0 l0-0 c0, unload-truck p1 t1 l0-0,
load-airplane p2 a0 l1-0 ✗
```

Figure. Example of an incorrect generations from LLMs for a problem from an unseen domain.

Both fine-tuned and few-shot prompting approaches of CodeT5 and code-davinci respectively show no capabilities of plan generation for unseen domains.

Invalid Generations from Plansformer

Failed Plans
Actual Plan: unstack b2 b4, put-down b2, unstack b4 b1, put-down b4, unstack b1 b3, put-down b1, pick-up b2, stack b2 b1, pick-up b4, stack b4 b2
Generated Plan: unstack b2 b4, put-down b2, unstack b4 b1, put-down b4, unstack b1 b3, put-down b4, unstack b4 b1, put-down b4, unstack b1 b3, put-down b4, unstack b4 b2, put-down b4, unstack b2 b4, stack b2 b1, pick-up b4, stack b4 b2

Incomplete Generations
Actual Plan: unstack b1 b3, put-down b1, pick-up b4, stack b4 b1
Generated Plan: unstack b1 b3, put-down b1, pick-up

Figure. Different types of invalid plans generated by Plansformer on Blocksworld domain

We notice that even in cases of incorrect generations, LLMs often generate partially correct action sequences. In our upcoming research, we introduce a neuro-symbolic approach where these partially correct plans are employed to inform the heuristics of symbolic planners, enabling faster replanning as opposed to starting the planning process from scratch.

Discussion Pointers

- Ongoing debate: is it Retrieval vs Reasoning in LLMs [1]
- What is the impact of licensing of LLMs for planning applications?

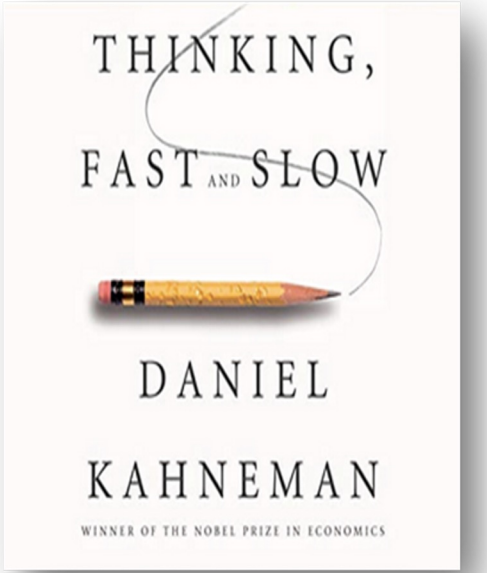
[1] Subbarao Kambhampati, Can LLMs Really Reason and Plan?. https://cacm.acm.org/blogs/blog-cacm/276268-can-llms-really-reason-and-plan/fulltext [Last accessed on Feb 17, 2024]

06.

Neuro-symbolic Approaches

Thinking Fast and Slow in Humans

System 1	System 2
Fast	Slow
Unconscious	Conscious
Automatic	Effortful
Everyday Decisions	Complex Decisions
Error prone	Reliable



System 1 and System 2 in AI

System 1 Solvers	System 2 Solvers
Rely on past experiences	Rely on procedures
React to new problems	Called by metacognition
Generate solutions with a certain confidence	Generate correct solutions
Complexity independent on input size	Complexity dependent on input size

Meta-cognition

The Role

To monitor and control:

- cognitive activities
- processes
- structures

Main Goal

To improve the System's decisions quality

Our Choice

- Centralized meta-cognitive agent
- Exploiting both internal and external information
- Arbitrates between Sys-1 and Sys-2 solvers

SOFAI = System 1 + System 2 + Metacognition

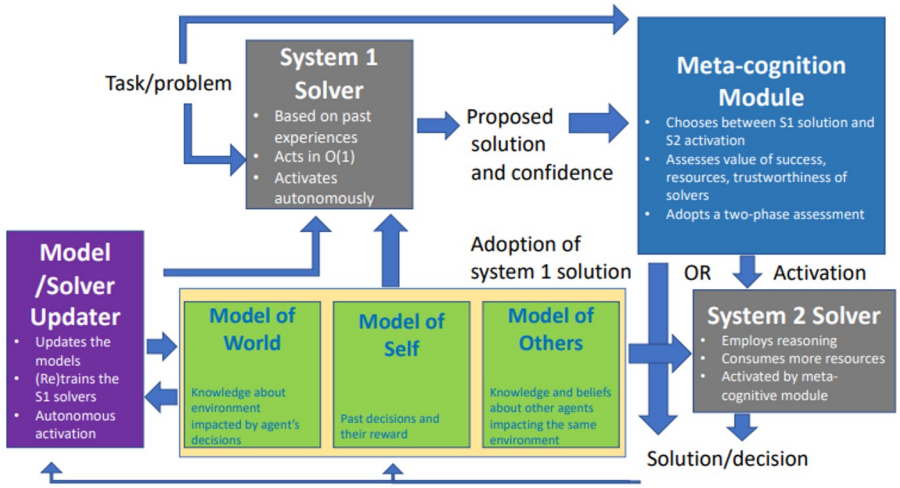


Table. The SOFAI architecture, supporting System 1/System 2 agents and meta-cognition.

Ganapini, M.B., Campbell, M., Fabiano, F., Horesh, L., Lenchner, J., Loreggia, A., Mattei, N., Rossi, F., Srivastava, B. and Venable, K.B., 2022, September. Thinking fast and slow in AI: The role of metacognition. In *International Conference on Machine Learning, Optimization, and Data Science* (pp. 502-509). Cham: Springer Nature Switzerland.

Plan-SOFAI: Instantiating SOFAI for Planning

(a) **System 1**

Plansformer

(b) **System 2**

A traditional planning system, such as FastDownward (for plan generation from scratch) or LPG (for partial plan completion).

(c) **Metacognition Module**

A rule-based function that chooses when to adopt S1 or S2 based on ***previous performance, expected cost,*** *and* ***expected accuracy on similar problems.*** It also includes a plan evaluator and has various hyperparameters to be more efficient.

Fabiano, F., Pallagani, V., Ganapini, M.B., Horesh, L., Loreggia, A., Murugesan, K., Rossi, F. and Srivastava, B., 2023, December. Plan-SOFAI: A Neuro-Symbolic Planning Architecture. In *Neuro-Symbolic Learning and Reasoning in the era of Large Language Models*.

Experimental Results

	FD	LPG	PF	SOFAI-PF-FD	SOFAI-PF-LPG	SOFAI-PF-FDxLPG	SOFAI-PF-LPGxLPG
Valid Plans	451 (90.20%)	**500 (100.00%)**	402 (80.40%)	483 (96.60%)	**500 (100.00%)**	490 (98.00%)	490 (98.00%)
Optimal Plans	451 (90.20%)	264 (52.80%)	386 (77.20%)	**464 (92.80%)**	434 (86.80%)	445 (89.00%)	448 (89.60%)
S1 Plans	–	–	402 (80.40%)	398 (79.60%)	401 (80.20%)	397 (79.40%)	394 (78.80%)
Optimal	–	–	386 (96.02%)	379 (95.23%)	383 (95.51%)	377 (94.96%)	378 (95.94%)
S2 Plans	451 (90.20%)	500 (100.00%)	–	85 (17.00%)	99 (19.80%)	93 (18.60%)	96 (19.20%)
Optimal	451 (100.00%)	264 (52.80%)	–	85 (100.00%)	51 (51.52%)	68 (73.12%)	70 (72.92%)
FD	451 (90.20%)	–	–	85 (17.00%)	–	5 (1.00%)	–
Optimal	451 (100.00%)	–	–	85 (100.00%)	–	5 (100.00%)	–
LPG	–	500 (100.00%)	–	–	99 (19.80%)	–	5 (1.00%)
Optimal	–	264 (52.80%)	–	–	51 (51.52%)	–	2 (40.00%)
PF + LPG	–	–	–	–	–	88 (17.60%)	91 (18.20%)
Optimal	–	–	–	–	–	63 (71.59%)	68 (74.73%)
Time (avg)	8.479s	**0.675s**	2.079s	4.915s	2.318s	2.199s	2.163s
Optimality (avg)	**+0.00%**	+23.68%	+0.34%	+0.02%	+9.78%	+1.13%	+4.86%

- **FD** has the **best optimality** (+0.00%) but **high** overall **resource** consumption (8.479 sec)
- **LPG** is the **most efficient** (0.675 sec) but has **less**-than-**ideal optimality** statistics (+23.68%)
- PF solves less problems (402) than FD and LPG but many (386) are optimal and time is constant (2.079 sec)
- Among the SOFAI instances:
  - SOFAI-PF-FDxLPG represents the most **balanced trade-off** among all the analyzed techniques
    - 490 solved problems, 2.199 sec, +1.13% optimality
  - SOFAI-PF-LPG best if we care less about optimality (all problems solved, 434 optimal)

Other Neuro-symbolic Approaches to Combine LLMs and Planning

- LLMs as planner generators [1]
  - LLMs are trained to generate planners (Python code) for a specific planning problem instance
  - The code is then run to generate a plan
- The pipeline approach [2]
  - LLMs to translate natural language to PDDL specification of a planning problem instance
  - Classical planners to solve the planning problem
  - LLM to translate the formal plan to natural language

[1] Silver, T., Dan, S., Srinivas, K., Tenenbaum, J.B., Kaelbling, L.P. and Katz, M., 2023. Generalized Planning in PDDL Domains with Pretrained Large Language Models. *arXiv preprint arXiv:2305.11014*.
[2] Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J. and Stone, P., 2023. Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477.

THANK YOU ALL

Contact Information

Vishal Pallagani –
vishalp@mailbox.sc.edu