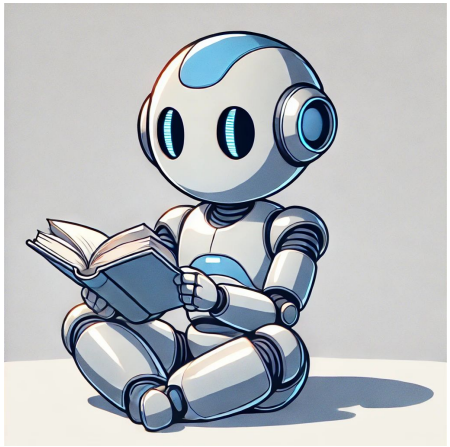
SoupOperator: Implementing Language-Driven Planning Strategies in Autonomous Robotics

Souradeep Nanda
Jonathan Nguyen
Gaurav Sharma

Large Language Models are World Models



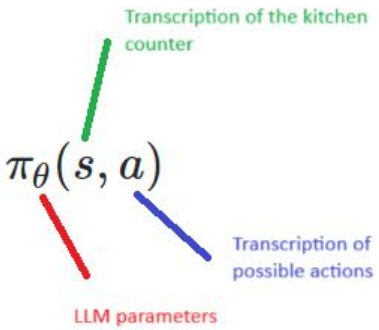
- Although, LLMs have never seen the world they have pretty good grasp of how the world works just by reading text
- This was an emergent phenomenon and accidental discovery by the scientists working in the field
- As such, this allows LLMs to guess how the world would react if it takes certain actions in it, although it has never really interacted with the world

Analogy to Reinforcement Learning (1 / 4)

- In reinforcement learning, we have a policy function pi, which takes a state and parameters as input and outputs the probability distribution of which ever actions it should take next.

$$\pi_\theta(s, a) = \frac{e^{\phi(s,a)^\intercal \theta}}{\sum_{k=1}^{N} e^{\phi(s,a_k)^\intercal \theta}}$$

Analogy to Reinforcement Learning (2 / 4)



- The state of the environment and the model is encoded into the prompt as state. This acts like a scene transcription handed to a blind person.

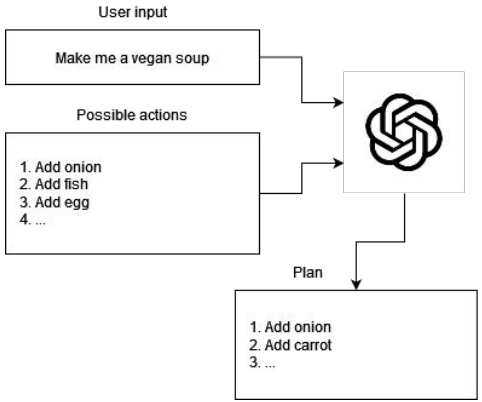
  Theta - LLM parameters

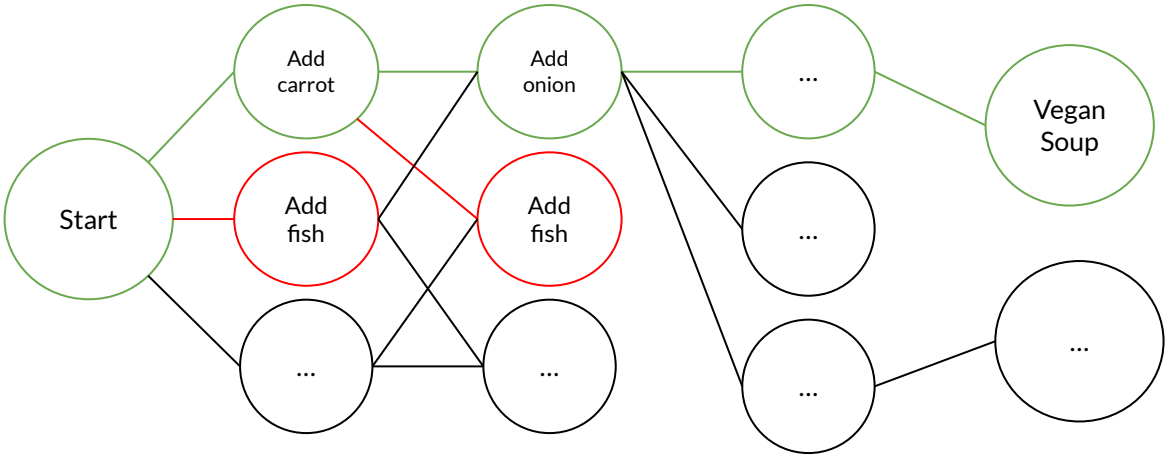
  State - e.g. You are tasked to cook an onion soup, you have one onion, butter, pot, …

  Actions - e.g. PICKUP_ONION, ADD_CARROT_TO_POT etc

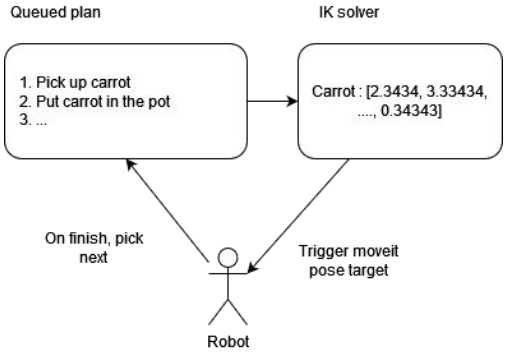
Analogy to Reinforcement Learning (3 / 4)

- We pass the objective we want to achieve in our environment, to condition the network to direct our goal. In essence, we are using the LLM as the heuristic function for path finding. E.g. Make me a vegan soup



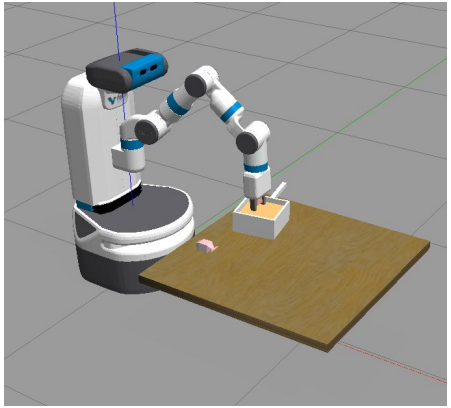


Analogy to Reinforcement Learning (4 / 4)



- We request the LLM to output in a specified format so that we can parse the response using python.
- We execute this sequence of actions using moveit in gazebo simulator.
- We use inverse kinematic solver to calculate joint angles required to grasp items.
- The drop off location is calculated and actuated with IK solver as well

Experimental Setup



- We use the fetch_gazebo robot for all our experiments
- We have a desk with the following items, onion, carrot, fish, butter
- We also have a pot in which the robot will use to cook
- The user request is typed in natural language through a CLI terminal

Demo

https://drive.google.com/file/d/1UzHWXH5BnMttcilg2avIqrt0FUQELXo5/view?usp=drive_link

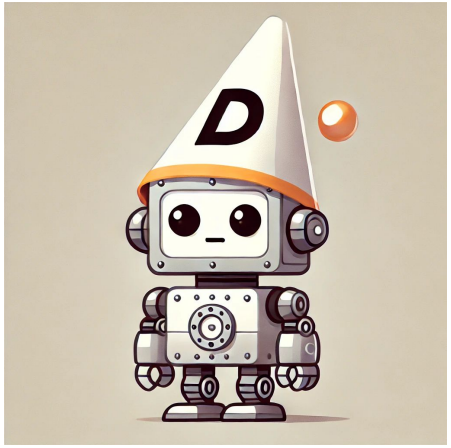
Error rates as a function of task complexity

TABLE I

ERROR RATES AS A FUNCTION OF TASK COMPLEXITY (GPT4)

Description of Task	Error Rate (%)
Make vegan soup	8%
Make fish soup	8%
Make vegan soup but I am allergic to tomatoes	12%
Make fish soup but I am allergic to fish	16%

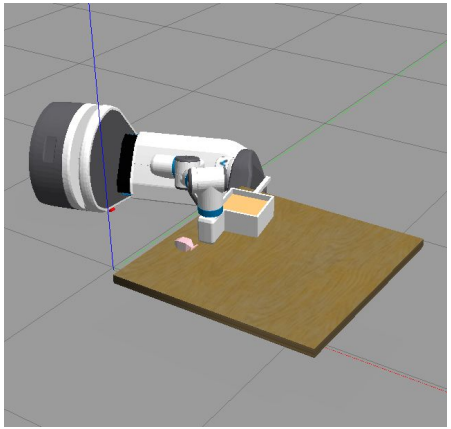
Error Mode Analysis (LLM side)



Common failure modes include

- Disregarding instructions e.g. adding fish to soup in spite the user explicitly asks not to add fish
- Losing count of inventory e.g. adding more ingredients than available
- Hallucinating actions that are not possible in our environment e.g. Stirring the pot, chopping the onions

Error Mode Analysis (Simulation side)



Common failure modes include

- Knocking ingredients off the desk
- Failing to grasp for no apparent reason
- Entire robot falling backwards for no apparent reason
- End effector getting stuck in the table/pot
- Gazebo crashing for no apparent reason

Error rates vs model size

TABLE II

ERROR RATES AS A FUNCTION OF LANGUAGE MODEL SIZE (FISH SOUP)

Model Name	Error Rate (%)
GPT-4o-mini	16%
GPT-4o	12%
GPT-4	8%

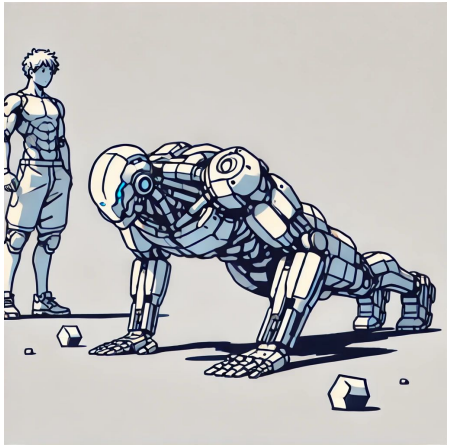
Error rate over time for continuous operation

TABLE III

ERROR RATES OVER CONTINUOUS OPERATION PERIODS (GPT4)

Operation Duration (mins)	Error Rate (%)
1	4%
2	4%
5	8%
10	12%
15	16%

Conclusion and Future Work



Reinforcement learning in simulated environments are expensive to train because the environment needs to be simulated. However if we can get a reasonably trained policy model off-the-shelf without even interacting with the environment, it can help bring down cost of training such agents.

It is still possible further finetune this LLM with simulated environment in the loop with RLHF-like methods for further increase in accuracy. This has been left as future work.

References

[1] X. Wang, Y. Chen, L. Yuan, Y. Zhang, Y. Li, H. Peng, and H. Ji, “Executable code actions elicit better llm agents,” ArXiv, vol.abs/2402.01030, 2024.

[2] C. Nandkumar and L. Peternel, “Enhancing supermarket robot interaction: A multi-level llm conversational interface for handling diverse customer intents,” ArXiv, vol. abs/2406.11047, 2024.

[3] P. Allgeuer, H. Ali, and S. Wermter, “When robots get chatty:Grounding multimodal human-robot conversation and collaboration,”ArXiv, vol. abs/2407.00518, 2024.

[4] Z. Yang, S. S. Raman, A. Shah, and S. Tellex, “Plug in the safety chip: Enforcing constraints for llm-driven robot agents,” 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 14 435–14 442, 2023.

[5] S. Hassan, H.-Y. Chung, X. Z. Tan, and M. Alikhani, “Coherence driven multimodal safety dialogue with active learning for embodied agents,” 2024