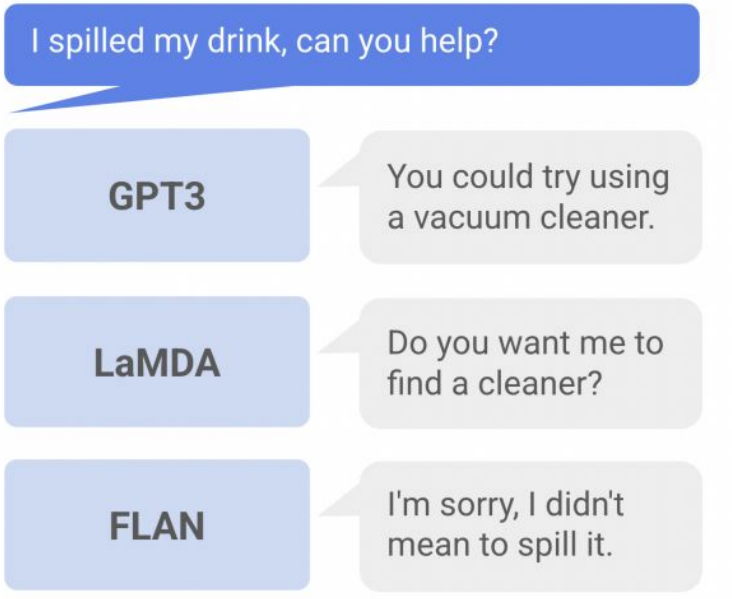
Do As I Can, Not As I Say

Along the same line as o1-preview but for robotics?

Have all this intelligence, none is practically usable



The end goal the paper talks about

Core Idea

- LLMs yap, don’t think about what is and isn’t feasible
- The main idea here is to ground language models in reality, show them the likelihood of a task succeeding
- Two separate components that can be hot swapped , The LLM, and the Value function (or Cost approximator)

The LLM

- Robots ships with predefined set of skills
  - Come with a small text description (“pick up a banana”)
  - Come with a policy (could be imitation learning based, or anything else)
  - Come with a reward function (we’ll talk about this later)
- Robots gets user task, and the prompt “I would 1) _”, the probability of each skill name being predicted is used as a score
- This process can be repeated over multiple steps until the task is done
- LLM’s job is to plan, predict next task options
- Receives sensor data, + past history

Cost Approximator (the main value here)

- This is a not so simple Markov Decision process using Temporal Difference (TD) based RL.
- The task is simple, given information on its current state and a certain skill, output the probability of that skill being a success.
- Sparse reward system received a 1 or 0 at the end of a chain of tasks if the task was successful or not.
- These turns are a cost approximator into an affordance function. Prioritizes end goal, not intermediately correct

Some Math Coming up

How do we know what action is affordable?

M = (S, A, P, R, γ)

- S – Set of states representing the environment
- A – Set of actions the agent can perform
- P – Probability function for state transitions. If we execute a skill, will we change our environment state?
- R – Reward function. Did we do good?
- γ - Discount function. They dont talk about it much, but values close to 1 help the robot plan longer paths, values close to 0 plan shorter ones

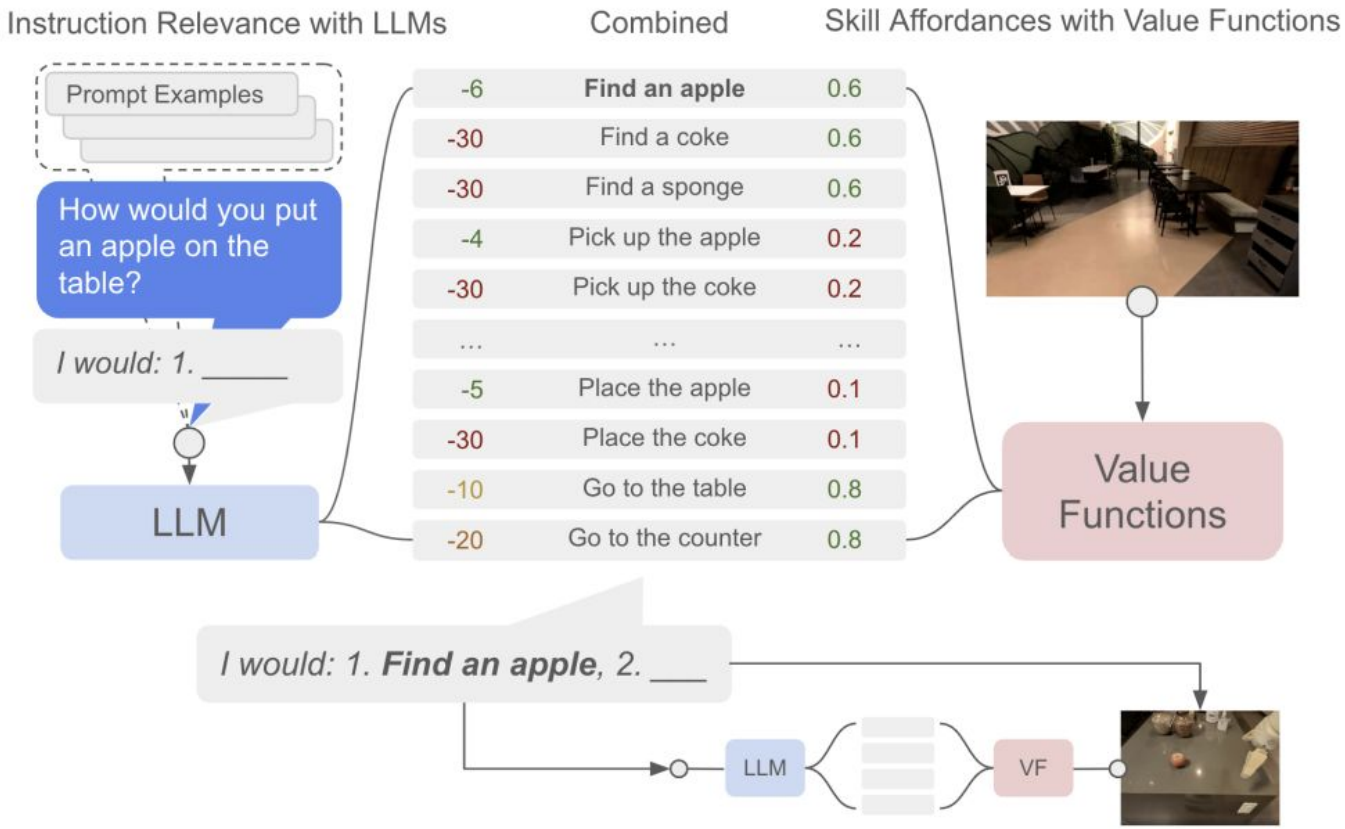
All of these variables do not explicitly appear in the math behind the Markov predictor but were used to brainstorm what directions the robot should be thinking in.

How do we know what action is affordable?-2

- $Q^{\pi}(s,a)$ – Cumulative reward, S is the state A is the action. π is the policy being used to generate actions. This is trying to estimate our future possible reward
- We want to give Q a dual purpose: we also wanted to judge how likely an action is to succeed. This is done with sparse rewards, good actions that do not result in the task being completed are still given a negative weight.
- Q is tuned by minimizing loss by the equation provided below. You will notice that not all of the variables in the previous slide are present there.

$$L_{TD}(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[R(s,a) + \gamma \mathbb{E}_{a^*\sim\pi} Q_\theta^\pi(s', a^*) - Q_\theta^\pi(s,a)\right]$$

Put it together



How did they test this?

- 15 objects, 5 semantically meaningful locations, 100ish skills, PaLM 504B model
- Mock kitchen – planning success rate of 84% and an execution rate of 74%
- Real kitchen – planning success rate of 81% and an execution rate of 60%
- Why the drop in execution?
- 65% LLM failure, 35% affordance miscalculation
- LLM struggled with negation (bring me a snack that isn't an apple)
- Long Horizon tasks were often terminated early (clean this spill, then bring me a new drink)

What is easy to add here

LLMs ship with many nice add on features, we can benefit from them here too!

- Multi language queries
- Chain of Thought (“asked for a drink with no caffeine, in front of me is…”)
- Closed loop queries? Currently all steps are generated at the beginning, resulting in no failure recovery
- Survey surrounding first/maintain more context? – (get me apple case study)
- Vision transformers???? 😳
- How do we add more skills here? Some sort of closed self teaching loop?