Controlling Robots via Large Language Models

Sanjiban Choudhury
The Problem
The way we program robots today is ... rigid!

Engineers hand-craft behaviors

Ship robot

Choose option
1. Start
2. Clean
3. Stop

Frustrate users!

Cannot be flexibly re-programmed by everyday users
Instead of explicitly engineering behaviors, can we implicitly program robots via natural interactions?
Programming via natural interactions

Demonstrations, Language

Feedback, Interactive QA
Question: How do we translate between humans and robots?
Large Language Models to the rescue!
Prompt: Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.
LLMs parse language and plan tasks

**Prompt:** Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.

**GPT-4:**

One possible way to stack the objects onto each other in a stable manner is:

- Place the book flat on a level surface, such as a table or a floor. The book will serve as the base of the stack and provide a large and sturdy support.
- **Arrange the 9 eggs in a 3 by 3 square on top of the book, leaving some space between them.** The eggs will form a second layer and distribute the weight evenly. Make sure the eggs are not cracked or broken, and handle them gently to avoid dropping them.
- Place the laptop on top of the eggs, with the screen facing down and the keyboard facing up. The laptop will fit snugly within the boundaries of the book and the eggs, and its flat and rigid surface will provide a stable platform for the next layer. Make sure the laptop is closed and turned off, and do not press any keys or buttons.
- Place the bottle on top of the laptop, with the cap facing up and the bottom facing down. The bottle will add some height and balance to the stack, and its cylindrical shape will reduce the contact area with the laptop. Make sure the bottle is empty and closed, and do not shake or tilt it.
- Place the nail on top of the bottle cap, with the pointy end facing up and the flat end facing down. The nail will be the final and smallest object in the stack, and its sharp tip will prevent it from rolling or sliding off the bottle. Make sure the nail is clean and straight, and do not touch or move it.
An Example
HAL
Helping Out In the Kitchen

(Home Apprentice Learner)
Activity!
Think-Pair-Share!

Think (30 sec): Think of all the steps to go from what the human said to the code the robot has to execute.

Pair: Find a partner

Share (45 sec): Partners exchange ideas

Human: “Help me make vegetable soup”

Robot:

- go_to(SALT)
- pick_up_item(SALT)
- go_to(TABLE)
- place_item_at(TABLE)
How things worked pre-LLM
Two Fundamental Challenges
Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state

"Pick up the farthest red block on the left."
Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state

Challenge 2:
Planning actions to solve a task

"Pick up the farthest red block on the left."

Find “salt” Find “pepper”
Two Fundamental Challenges

Challenge 1: Ground natural language in robot state

Challenge 2: Planning actions to solve a task

"Pick up the farthest red block on the left."

Find “salt”

Find “pepper”
What is grounding? Why is it hard?

"Pick up the farthest red block on the left."
Grounding: Mapping language to robot’s internal state

Natural Language

“Pick up the farthest red block”

MDP

< S, A, R, T >
Grounding: Mapping language to robot’s internal state

Natural Language

“Pick up the farthest red block”

MDP

\[ \langle S, A, R, T \rangle \]

on('obj1','table')
on('obj2','table')
on('obj3','table')
on('obj4','table')
left('obj2','obj1')
left('obj3','obj2')
left('obj4','obj3')
...

obj1 obj2 obj3 obj4
Grounding: Mapping language to robot’s internal state

Natural Language → MDP

“Pick up the farthest red block”

< S, A, (R), T >

R(in(obj4, hand)) = 1
How did we solve grounding?

“Pick up the farthest red block”

Complex graphical models!

Train this on small, custom robot datasets
Why did this not scale?

1. Failure to generalize to different human utterances
2. Failure to capture common sense
3. Failure to capture complex instructions (while loops)
Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state

Challenge 2:
Planning actions to solve a task

Find “salt”  Find “pepper”
What is task planning? Why is it hard?

Take the apple from the shelf and put it on the table

shelf

apple

robot

R

table
What is task planning? Why is it hard?

Take the apple from the shelf and put it on the table

1. Move to the shelf
2. Pick up the apple
3. Move back to the table
4. Place the apple
What is task planning? Why is it hard?

(At robot table)
(At apple shelf)
(HandEmpty robot)

Move(table, shelf)

(At robot shelf)
(At apple shelf)
(HandEmpty robot)
What is task planning? Why is it hard?

Diagram showing task planning process with objects and actions:
- Shelf
- Apple
- Robot
- Table

Sequence of actions:
1. \(\text{Move}(\text{table}, \text{shelf})\)
2. \(\text{Pick}(\text{apple}, \text{shelf})\)
3. \(\text{Move}(\text{shelf}, \text{table})\)
What is task planning? Why is it hard?
What is task planning? Why is it hard?
What is **task planning**? Why is it hard?
What is **task planning**? Why is it hard?
How did we solve it?

Good old fashioned search!

Lots of heuristics to make it real time

Why did it not scale?

Combinatorially large search tree

Had no notion of common sense
Two Fundamental Challenges

Challenge 1:
Ground natural language in robot state

Challenge 2:
Planning actions to solve a task

"Pick up the farthest red block on the left."

Find “salt”  Find “pepper”
LARGE LANGUAGE MODELS

Episode IV

A NEW HOPE
So ... we just ask an LLM to tell us what to do?
No! LLMs can say *anything* ..
Idea: Constrain LLM by what the robot can do (affordance)
Idea: Constrain LLM by what the robot can do (affordance)
The “SayCan” Approach

Instruction Relevance with LLMs

Combined

Task Affordances with Value Functions

Value Functions

How would you put an apple on the table?

I would: 1. _____

LLM

Find an apple 0.6
Find a coke 0.6
Find a sponge 0.6
Pick up the apple 0.2
Pick up the coke 0.2
...
...
...
Place the apple 0.1
Place the coke 0.1
Go to the table 0.8
Go to the counter 0.8

I would: 1. Find an apple, 2. _____
User input: Bring me a fruit flavoured drink without caffeine.

Robot: 1.
How does SayCan solve both challenges?

Challenge 1:
Ground natural language in robot state

Have the robot query the LLM with it’s own internal state

Challenge 2:
Planning actions to solve a task

LLMs are capable of planning tasks using chain of thought reasoning
But there are still problems!

Problem 1: What if actions fail?

Problem 2: How do we verify correctness?
But there are still problems!

Problem 1: What if actions fail?

Idea: Close the loop on LLMs

Problem 2: How do we verify correctness?
Inner Monologue:
Embodied Reasoning through Planning with Language Models

Wenlong Huang  Fei Xia  Ted Xiao  Harris Chan  Jacky Liang  Pete Florence
Andy Zeng  Jonathan Tompson  Igor Mordatch  Yevgen Chebotar  Pierre Sermanet
Noah Brown  Tomas Jackson  Linda Luu  Sergey Levine  Karol Hausman  Brian Ichter

Robotics at Google
But there are still problems!

Problem 1: What if actions fail?

Idea: Close the loop on LLMs

Problem 2: How do we verify correctness?

Idea: Get LLMs to generate code
Code as Policies:
Language Model Programs for Embodied Control

Jacky Liang  Wenlong Huang  Fei Xia  Peng Xu  Karol Hausman  Brian Ichter  Pete Florence  Andy Zeng

Robotics at Google
Stack the blocks on the empty bowl.

```python
block_names = detect_objects("blocks")
bowl_names = detect_objects("bowls")
for bowl_name in bowl_names:
    if is_empty(bowl_name):
        empty_bowl = bowl_name
        break
objs_to_stack = [empty_bowl] + block_names
stack_objects(objs_to_stack)

def is_empty(name):
    # Implementation

def stack_objects(obj_names):
    n_objs = len(obj_names)
    for i in range(n_objs - 1):
        obj0 = obj_names[i + 1]
        obj1 = obj_names[i]
        pick_place(obj0, obj1)
```
It’s an exciting time for robot learning!