NUS National University of Singapore | Computing

LARGE LANGUAGE MODEL FOR TEXT-BASED DECISION MAKING

CS5446

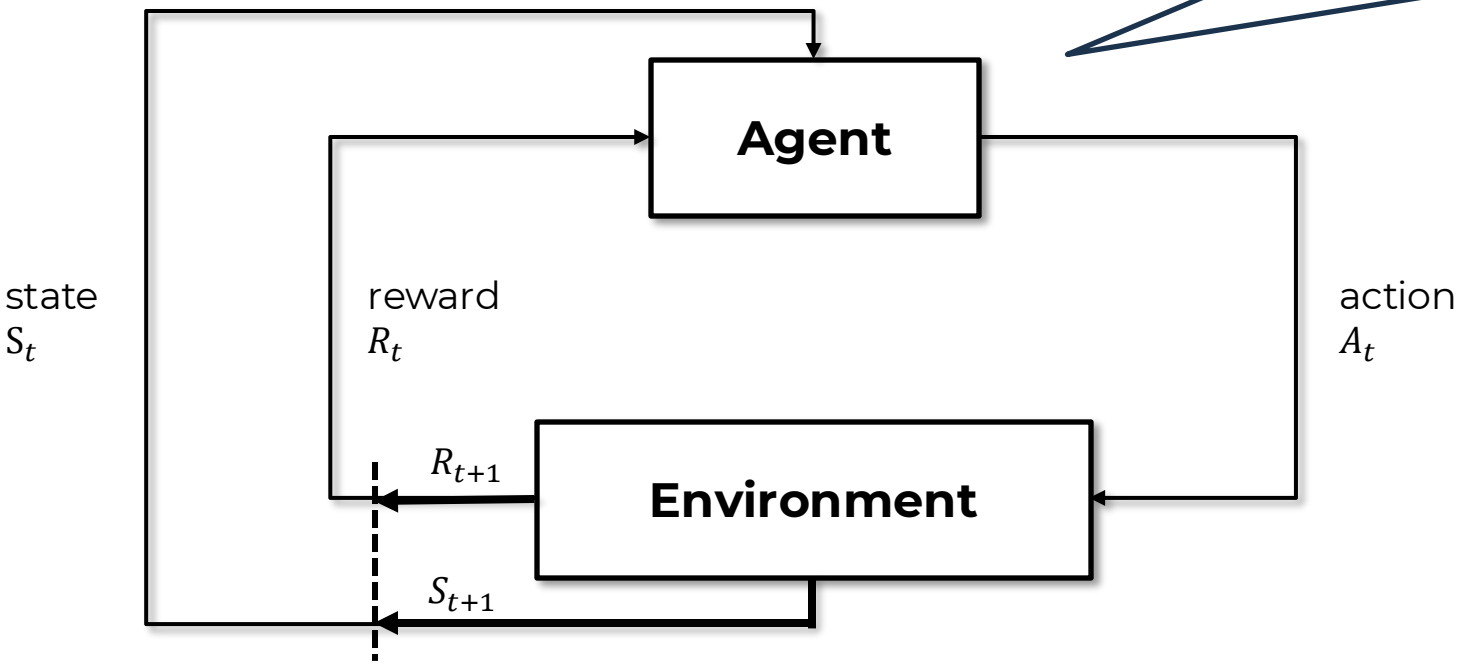
AI PLANNING AND DECISION MAKING

Group members:
LOW Shao Fu (A0305009R)
Nontaphat Charuvajana (A0309391B)
Liao Hung Chieh (A0304830N)
Chong Juang Ian (A0304474H)

Motivation

Reinforcement Learning workflow.

Learn to model environment
or
Directly learn policy to maximize reward



Motivation

Traditional reinforcement learning methods

- Dynamic Programming (DP)
- Monte Carlo Methods (MC)
- Temporal-Difference Learning (TD)
- Approximate Dynamic Programming (ADP)
- Actor-Critic Methods

Motivation

Problems with current methods of reinforcement learning

Sample Efficiency

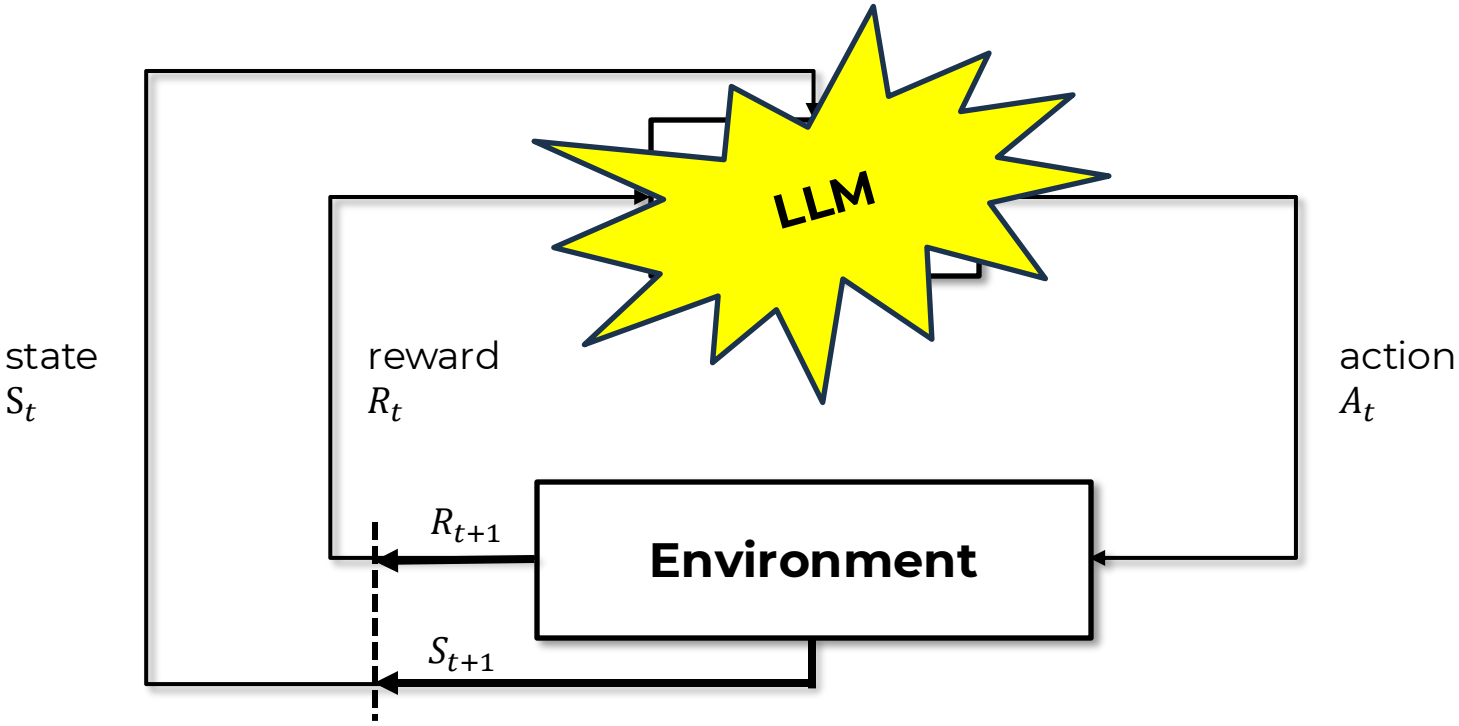
Generalizability

Training Difficulty

For every new environment, training an agent is no trivial task!

Motivation

Can large language models (LLM) act directly as an agent to perform decision making & planning?



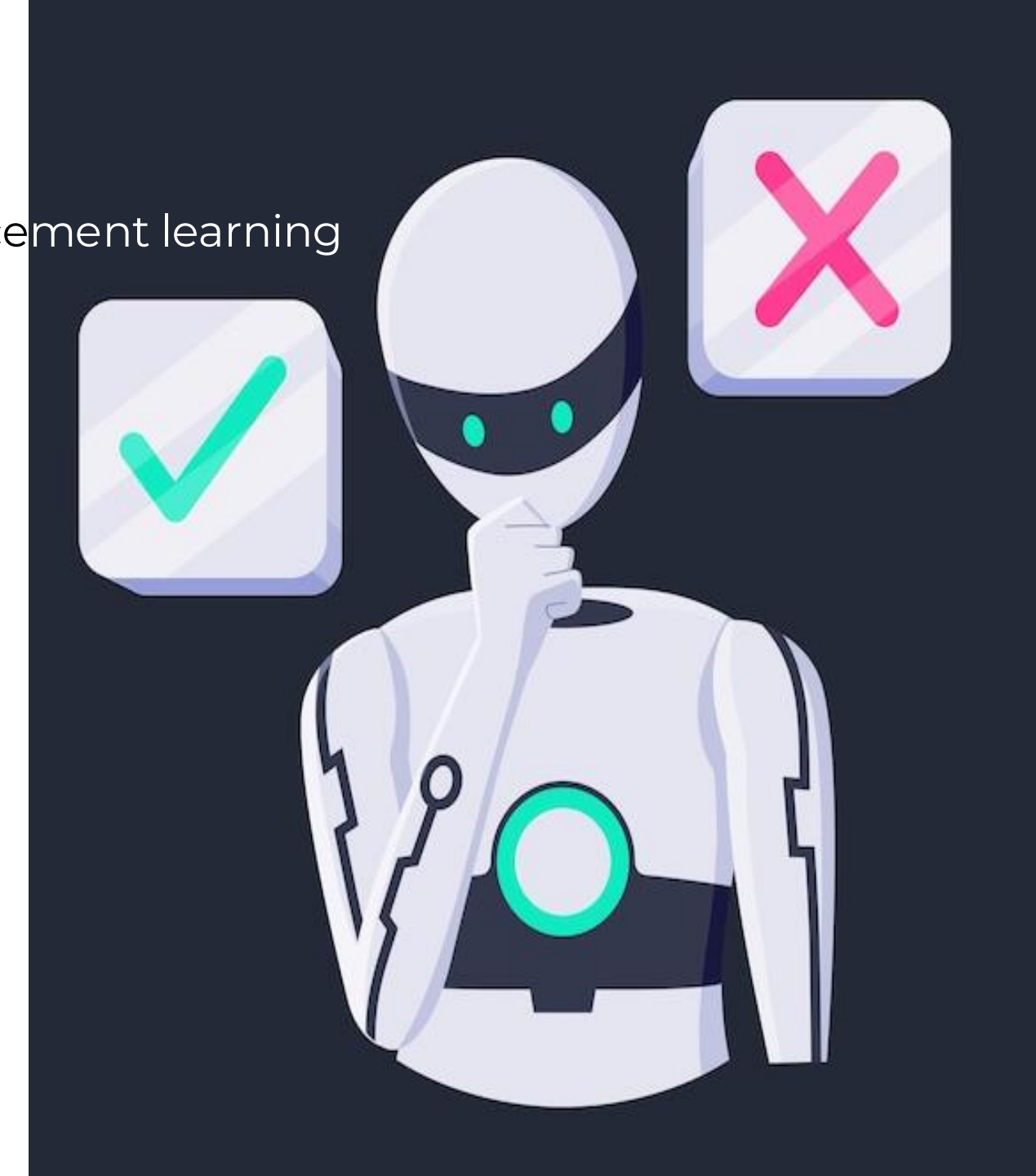
Benefits of LLM

Compared to traditional method in reinforcement learning

Commonsense Knowledge

Generalizable Reasoning Capability

Can be instructed with natural language



Problem statements

Find the optimal policy given a Markov Decision Process, where:

- The state/observation is in the form of **natural language**.
- The transition function and probabilities are unknown.

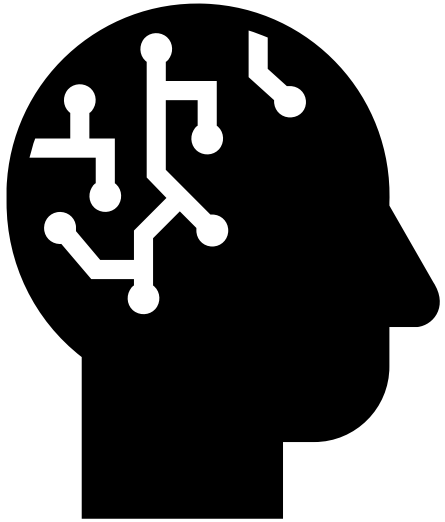
Objectives

- Implement and evaluate traditional reinforcement learning method for text-based environment.

- Implement and experiment several LLM-based methods for text-based environment.

Our methods

Our approaches in reinforcement learning

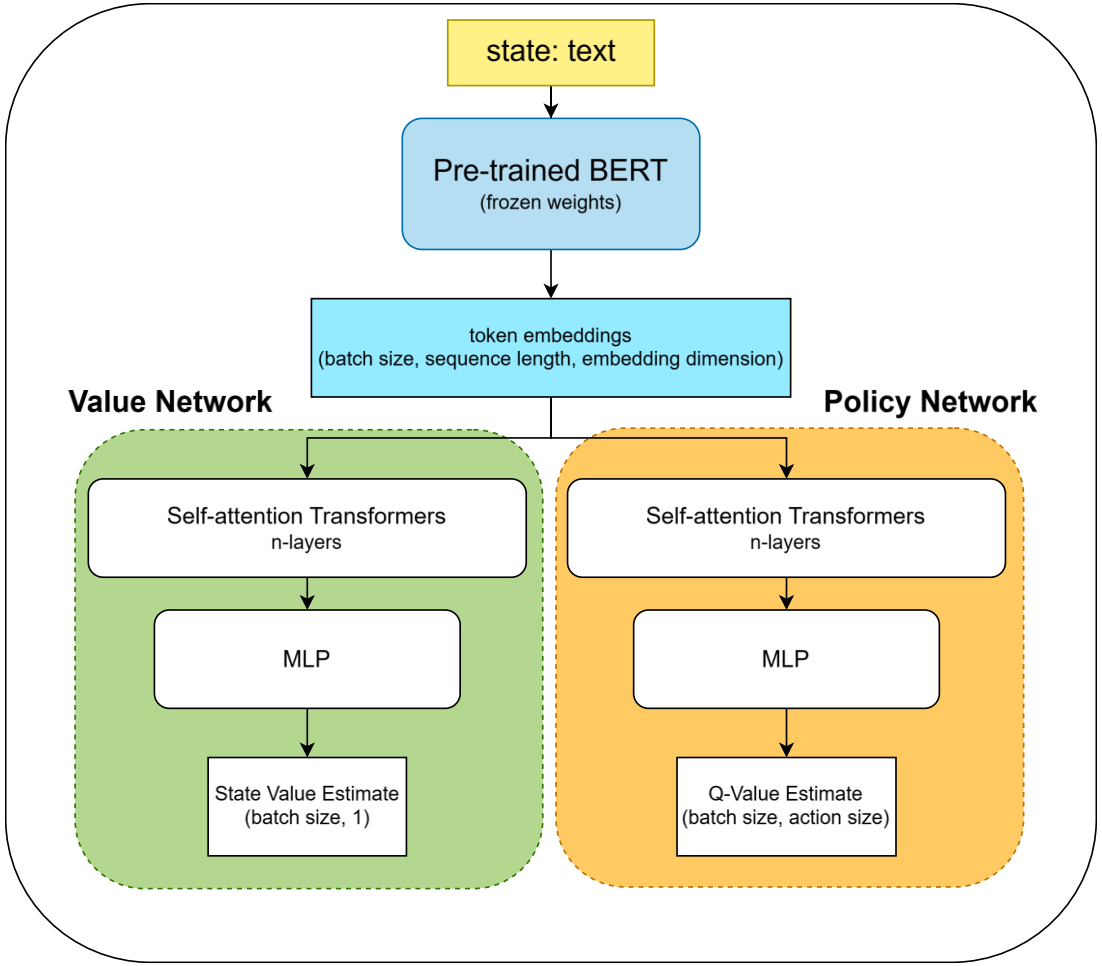


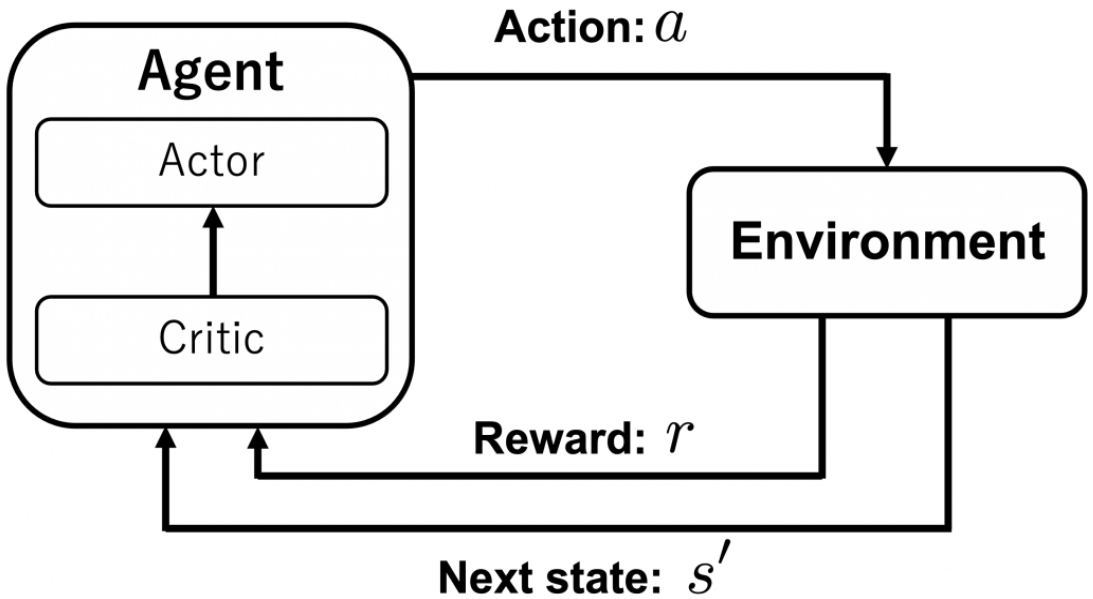
Deep Learning

Large Language Model

Deep Learning

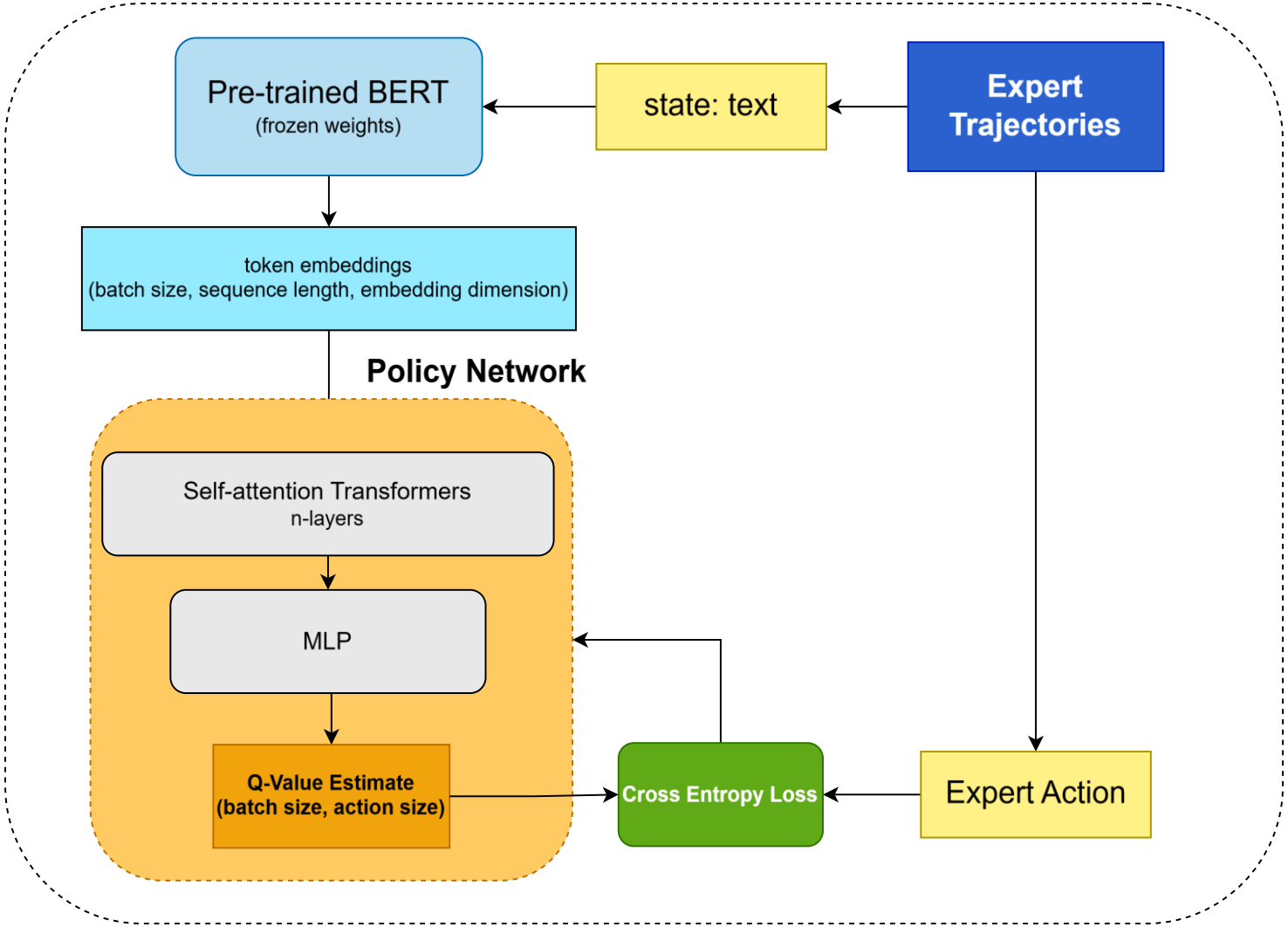
Proximal Policy Optimization with Transformer Networks





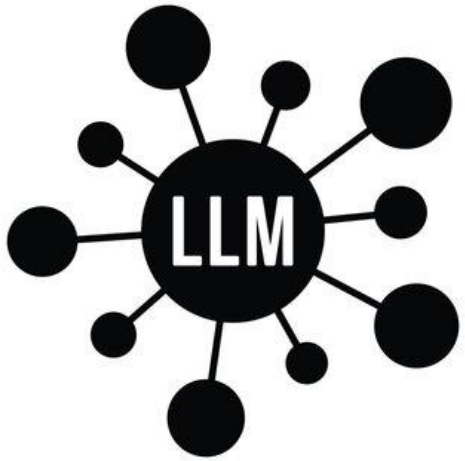
Deep Learning

We also train a policy network **DAgger** style to evaluate the effectiveness of our network.

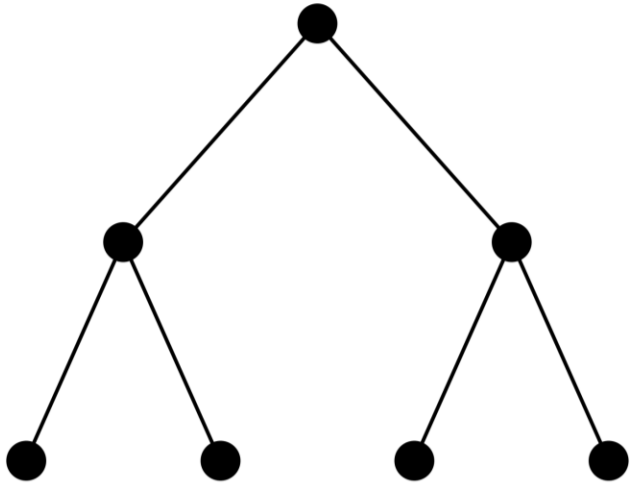


Large Language Model

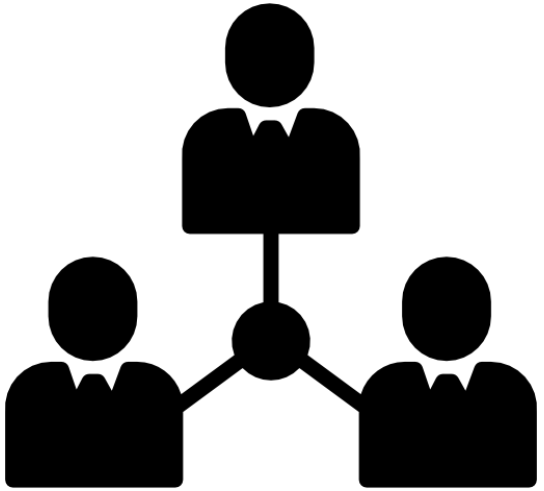
Large Language model approach for reinforcement learning.



Direct LLM



LLM-MCTS



LLMZero

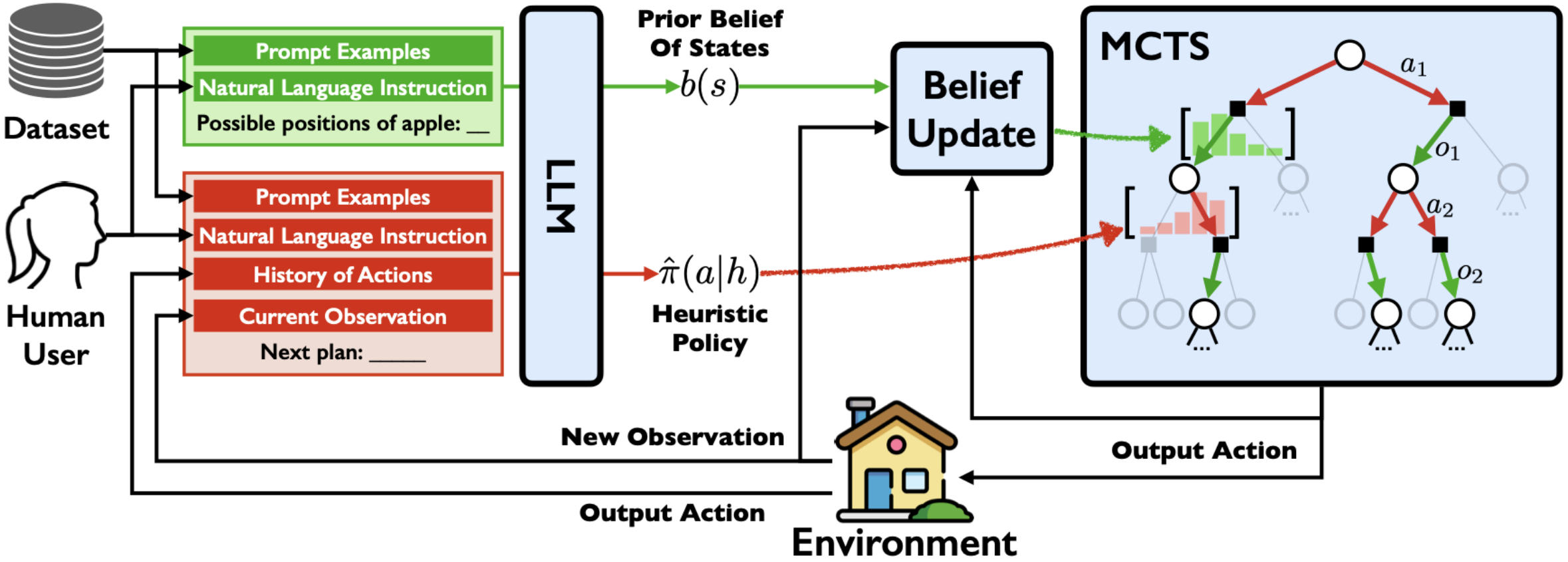
Large Language Model

Direct Large Language Model Policy

- Using prompt engineer to make the large language model understand the task.
- Giving them states, goals and available actions that they could take expecting large language model to output the optimal action for that states.
- Agent taking action directly from the large language model and then prompting the large language model again with new states and available set of actions.

Large Language Model

LLM-MCTS structure from paper

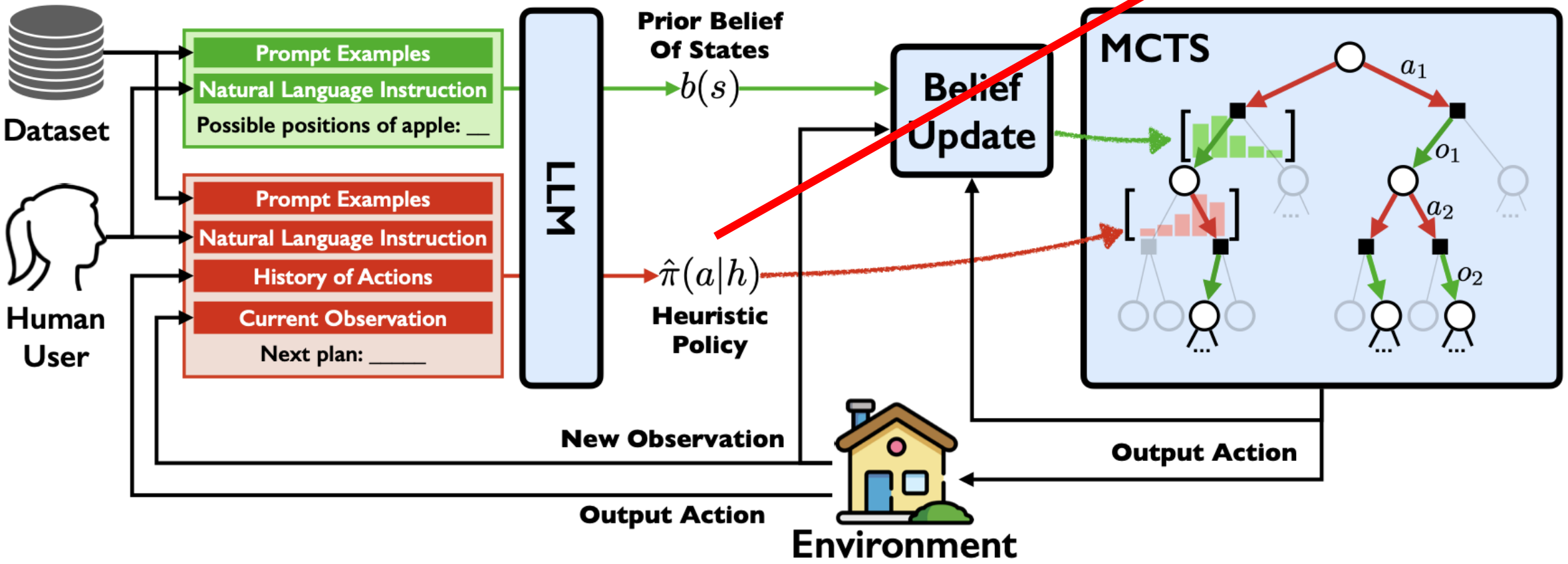


Reference: Zirui Zhao, Wee Sun, Lee David Hsu. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. 2023

Large Language Model

LLM-MCTS structure from paper

$$PUCT = Q(s,a) + c_{puct} \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

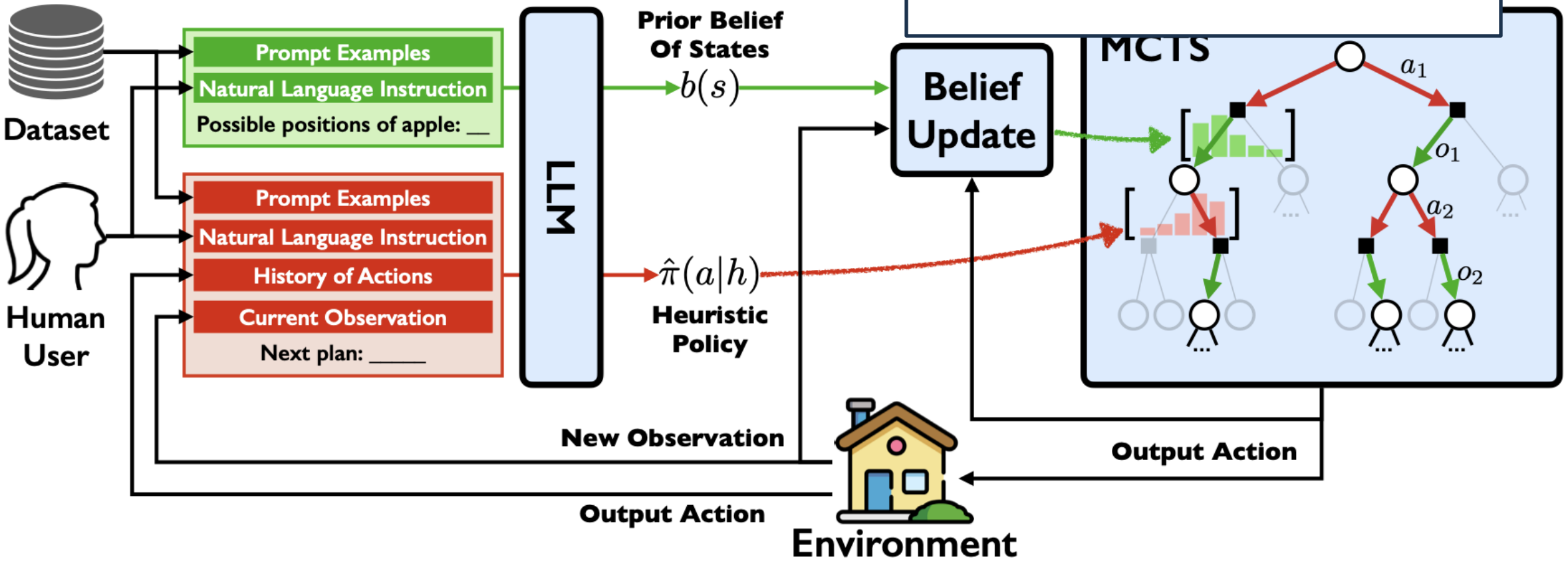


Reference: Zirui Zhao, Wee Sun, Lee David Hsu. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. 2023

Large Language Model

LLM-MCTS structure from paper

MCTS requires the transition function to perform node expansion and rollout.



Reference: Zirui Zhao, Wee Sun, Lee David Hsu. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. 2023

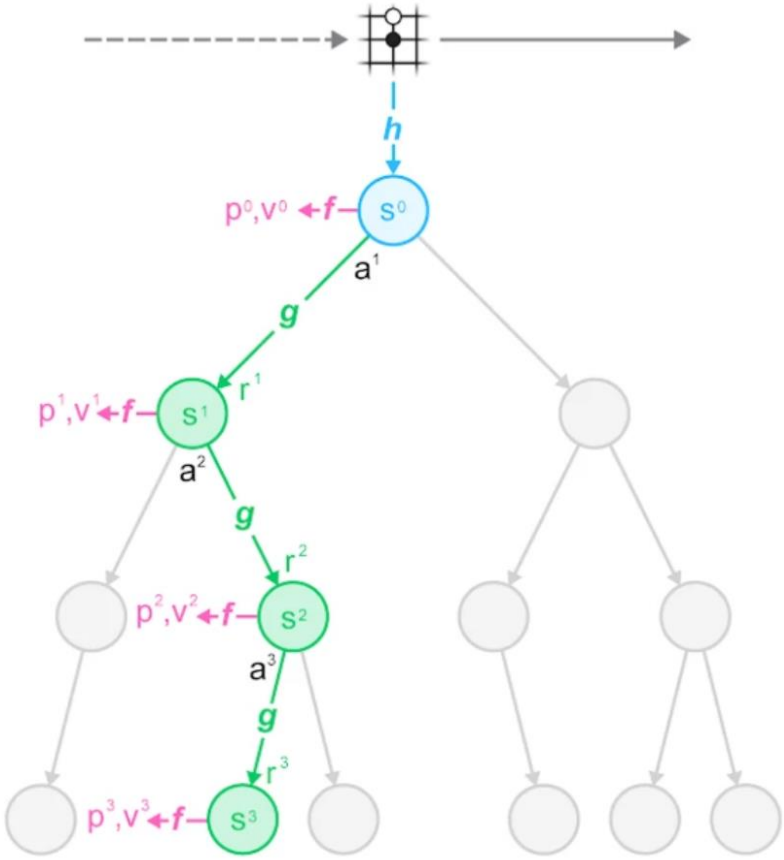
Large Language Model

MuZero architecture

representation $h_\theta(o_1, .., o_t) = s^0$

prediction $f_\theta(s^k) = \mathbf{p}^k, v^k$

dynamics $g_\theta(s^{k-1}, a^k) = r^k, s^k$



Reference: Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, et al. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. 2019

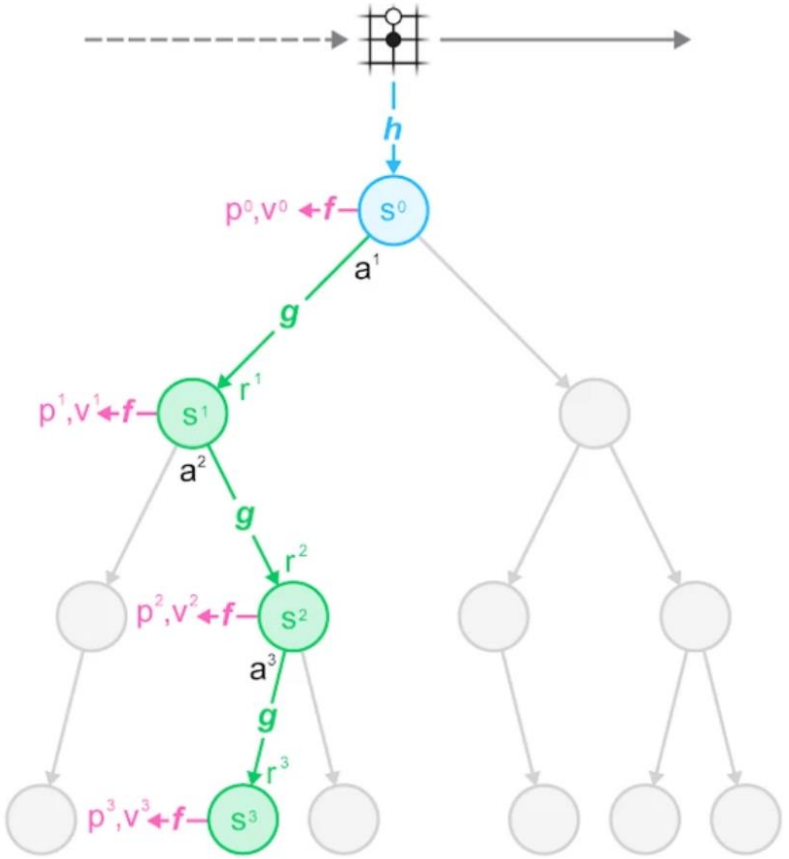
Large Language Model

LLM-Zero : replacing neural networks with pre-trained LLM

Model Representation: Natural Language

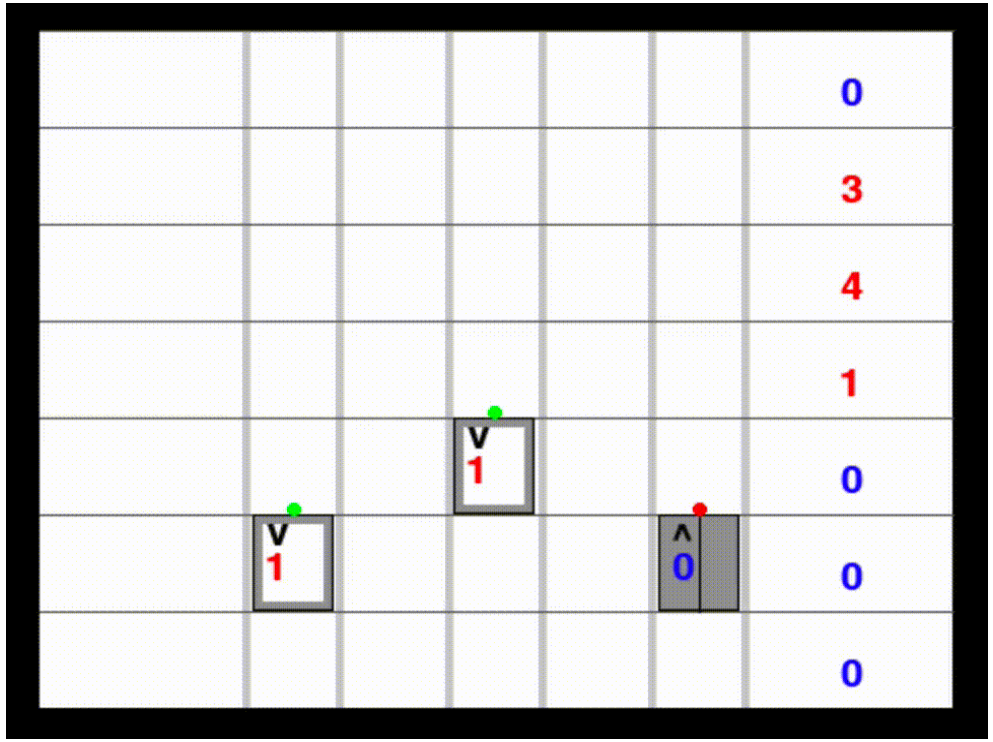
Action Prediction: Ask LLM
- "What is the next best move?"
- "What is the value of this state"

Transition: Ask LLM
- "What is the most probable next state from doing this action?"
- "What reward will I get from doing this action?"



Environments

Simulated environment for model interactions.



Elevator Environment

Alfworld Environment

ALF world

- **States**: List of items
  - Containers
  - Moveable objects
  - Surfaces
  - Rooms

- **Goal**: Move object(s) to the desired location

- **Actions**:
  - Go to (<location>): Walk to the location.
  - Open(<item>): Open a container.
  - Close(<item>): Close a container.
  - Take (<item>): Grab the object with the agent
  - Put (<item>) in/on the location: Put a movable object on a surface

Elevators

Long horizons text-based environment

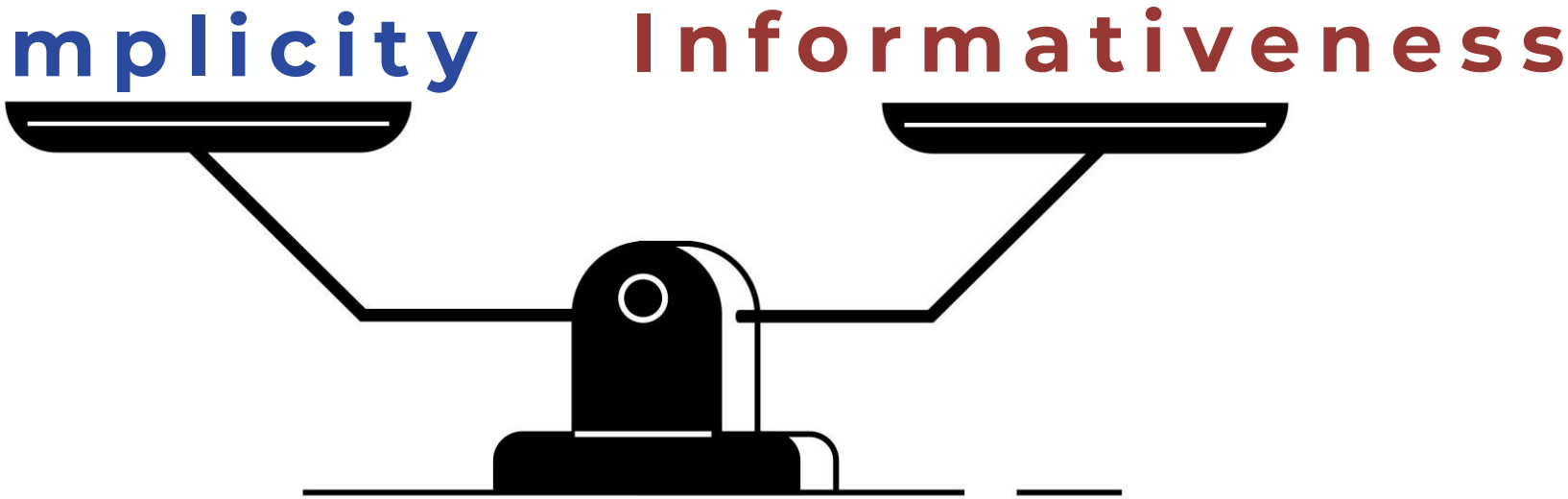
```
Elevator door is closed.

---------------------- Step 0 ----------------------
Enter action: move
Next state:
People waiting at floor 2: 0
People waiting at floor 3: 1
People waiting at floor 4: 1
People waiting at floor 5: 0
Elevator at floor 2.
There are 0 people in the elevator.
Elevator is moving up.
Elevator door is closed.

Reward: 0.0
---------------------- Step 1 ----------------------
Enter action: move
Next state:
People waiting at floor 2: 0
People waiting at floor 3: 1
People waiting at floor 4: 1
People waiting at floor 5: 0
Elevator at floor 3.
There are 0 people in the elevator.
Elevator is moving up.
Elevator door is closed.

Reward: -6.0
---------------------- Step 2 ----------------------
Enter action: open door
Next state:
People waiting at floor 2: 0
People waiting at floor 3: 1
People waiting at floor 4: 1
People waiting at floor 5: 0
Elevator at floor 3.
There are 0 people in the elevator.
Elevator is moving down.
Elevator door is open.

Reward: -6.0
---------------------- Step 3 ----------------------
Enter action:
```

ALFworld

Short horizons partially observable text-based environment

```
 -= Welcome to TextWorld, ALFRED! =-

You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 1, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a laundryha

Your task is to: put a alarmclock in desk.

valid actions: ['go to bed 1', 'go to desk 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to garbagecan 1', 'go to laundryhamper 1', 'go
---------------------- Step 0 ----------------------
Enter action: go to sidetable 1

Next state:
You arrive at loc 20. On the sidetable 1, you see a alarmclock 3, a alarmclock 2, a alarmclock 1, a creditcard 1, a desklamp 1, a keychain 1, a pen 3,

valid actions:
['examine sidetable 1', 'go to bed 1', 'go to desk 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to garbagecan 1', 'go to laundryhamper
 1', 'take alarmclock 2 from sidetable 1', 'take alarmclock 3 from sidetable 1', 'take creditcard 1 from sidetable 1', 'take keychain 1 from sidetable
le 1', 'use desklamp 1']

You arrive at loc 20. On the sidetable 1, you see a alarmclock 3, a alarmclock 2, a alarmclock 1, a creditcard 1, a desklamp 1, a keychain 1, a pen 3,
---------------------- Step 1 ----------------------
Enter action: take alarmclock 1 from sidetable 1

Next state:
You pick up the alarmclock 1 from the sidetable 1.

valid actions:
['examine alarmclock 1', 'examine sidetable 1', 'go to bed 1', 'go to desk 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to garbagecan
mclock 1 in/on sidetable 1', 'use desklamp 1']

You pick up the alarmclock 1 from the sidetable 1.
---------------------- Step 2 ----------------------
Enter action: go to desk 1
```

Prompt engineering



- Less domain knowledge
- Exploration
- More adaptive

- Require more domain knowledge
- Exploitation
- Case-specific

Prompt engineering

Elevators – Policy Network

Simplicity

- State expression
- Ultimate goal
- Action space
- Environment

Informativeness

- + Regulations / Domain knowledge
- + Reward details
- + Successful Examples
- + Instruction on how to make decision

Prompt Structure

Elevators

LLM Transition

- Transition Rules
- Examples and State format

LLM Reward

- Explain problem
- Reward formula
- Examples

LLM Value

- Explain problem
- Value estimation formula
- Examples

Prompt engineering

ALF world – Policy Network

Simplicity

- State expression
- Ultimate goal
- Action space
- Item features

Informativeness

- + Regulations / Domain knowledge
- + Reward details
- + Successful Examples

Prompt Structure

ALF world

LLM Transition

- Transition Rules
- Examples and State format

LLM Reward

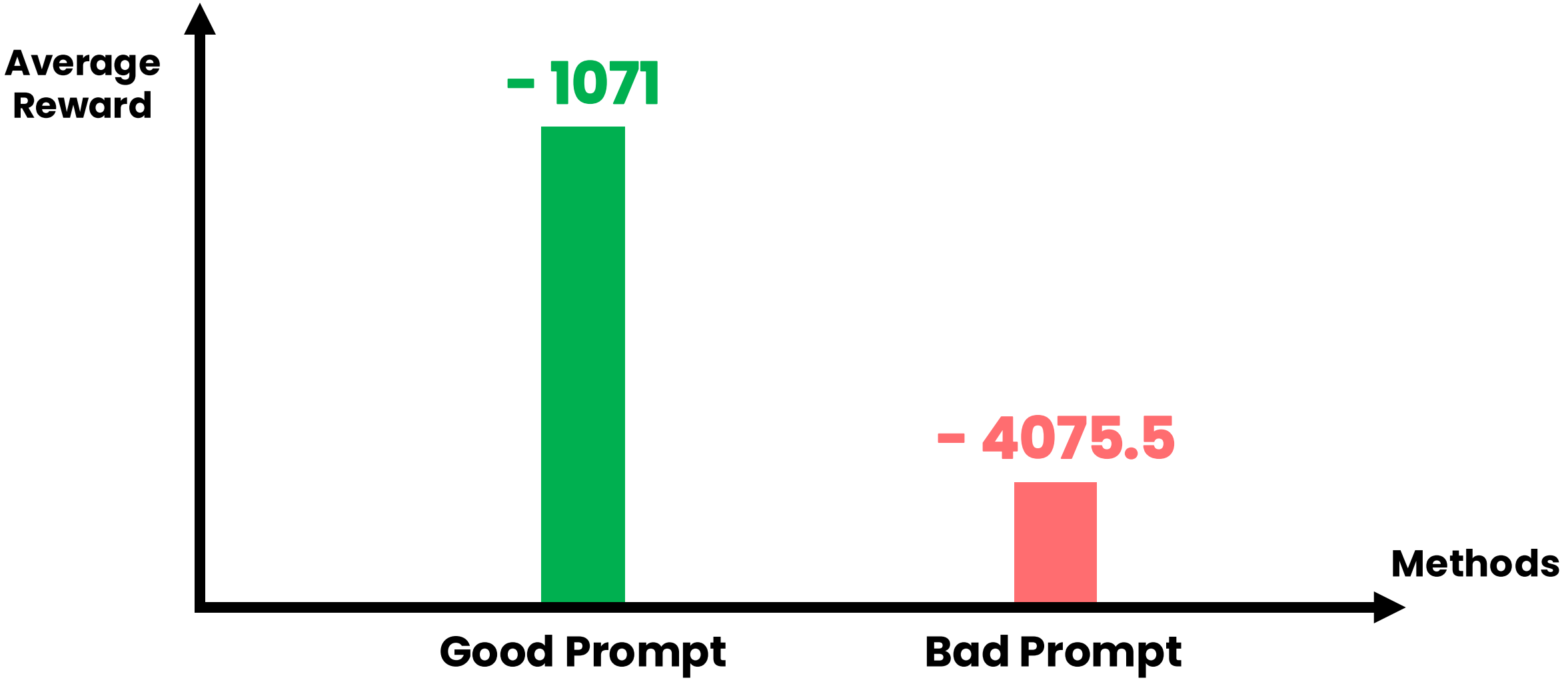
- Explain problem
- Reward formula
- Examples

LLM Value

- Explain problem
- Value estimation formula
- Examples

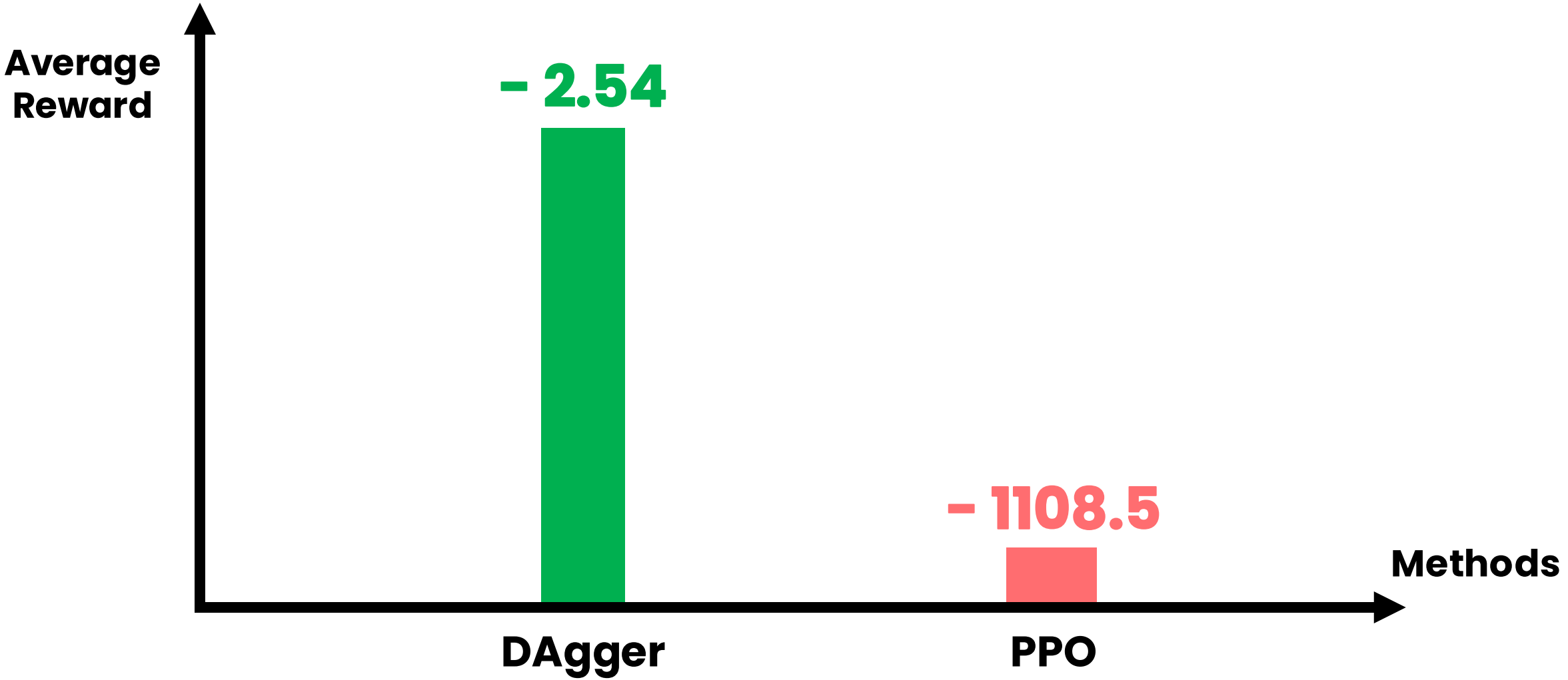
LLM Policy: Good Prompt vs Bad Prompt

Comparing the reward between good and bad prompt.



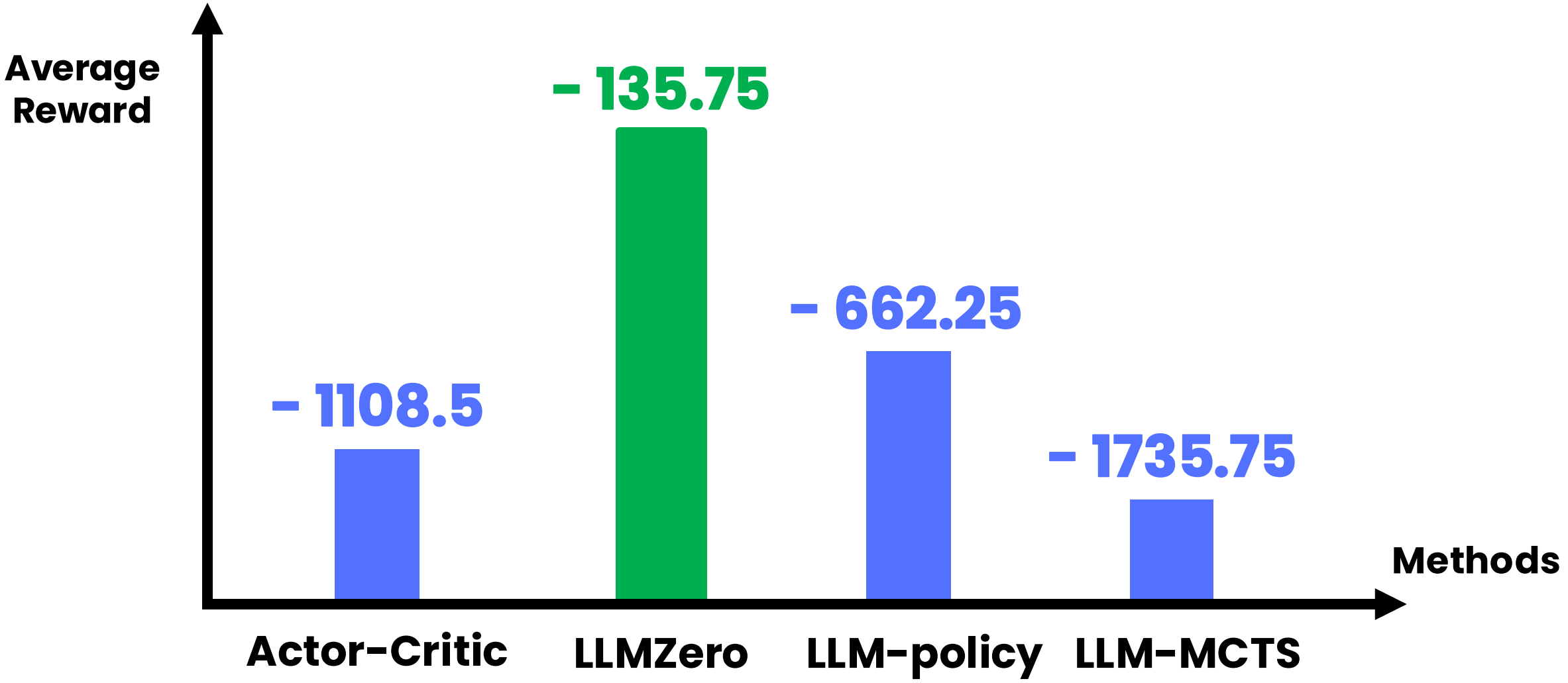
Evaluation: DAgger v.s. PPO

Evaluation on elevator environment



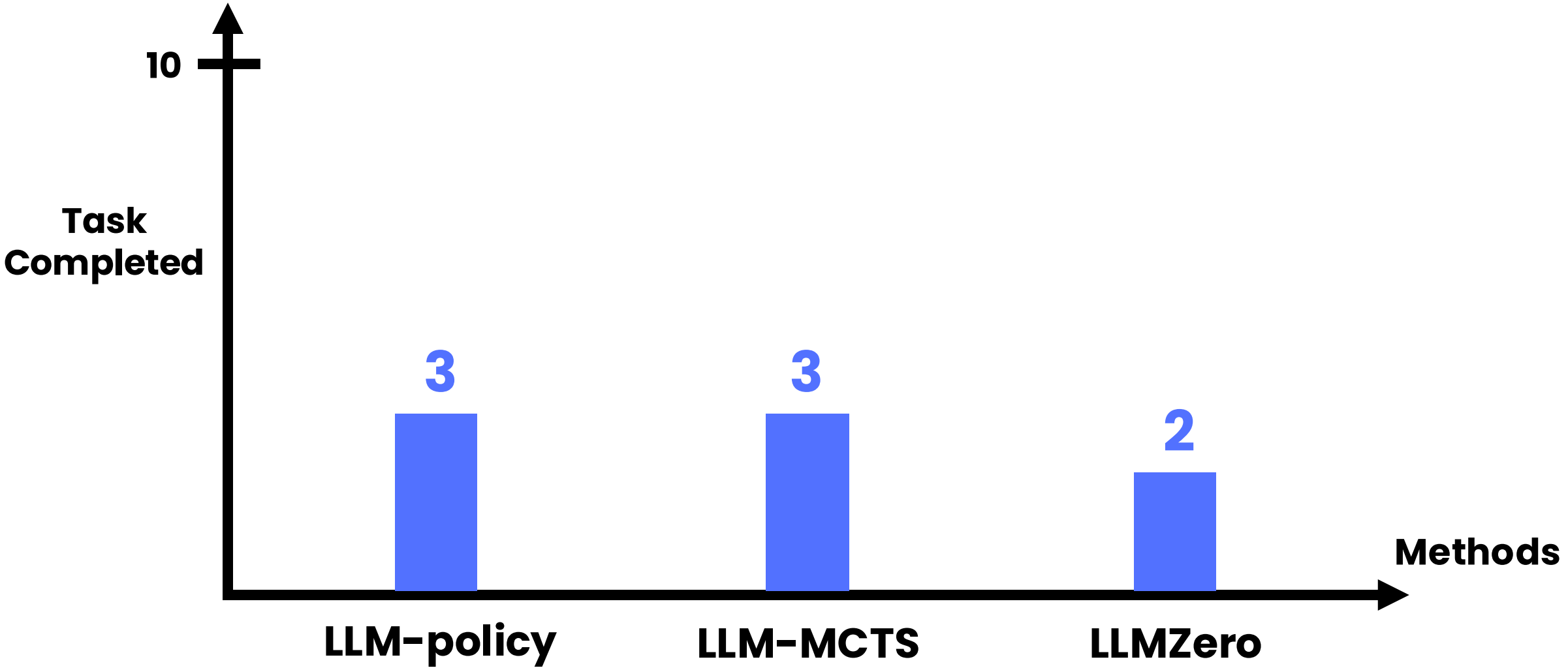
Evaluation

Evaluation on elevator environment



Evaluation

Evaluation on ALFworld environment.



Evaluation

Failure Analysis (AlfWorld): Effects of reward structure

Recall the formula for selecting nodes expansion:

$$PUCT = Q(s,a) + c_{puct} \cdot P(s,a) \cdot \frac{\sqrt{(\sum N(s,b))}}{1 + N(s,a)}$$

$Q(s,a)$ is the mean of cumulative rewards over simulations

$$Q(s,a) = \frac{\sum_{m=1}^{M} \gamma^m R(s,a) + \gamma^{\{M+1\}} \cdot V(s)}{N(s,a)}$$

If $R(s,a) \geq 0 \; \forall \; s, a$ and $V(s) \geq 0 \; \forall \; s$, and we initialize $Q(s,a) = 0$, the MCTS degrades to following the first action picked by $P(s,a)$ until the exploration penalty $\frac{\sqrt{(\sum N(s,b))}}{1+N(s,a)}$ becomes large enough.

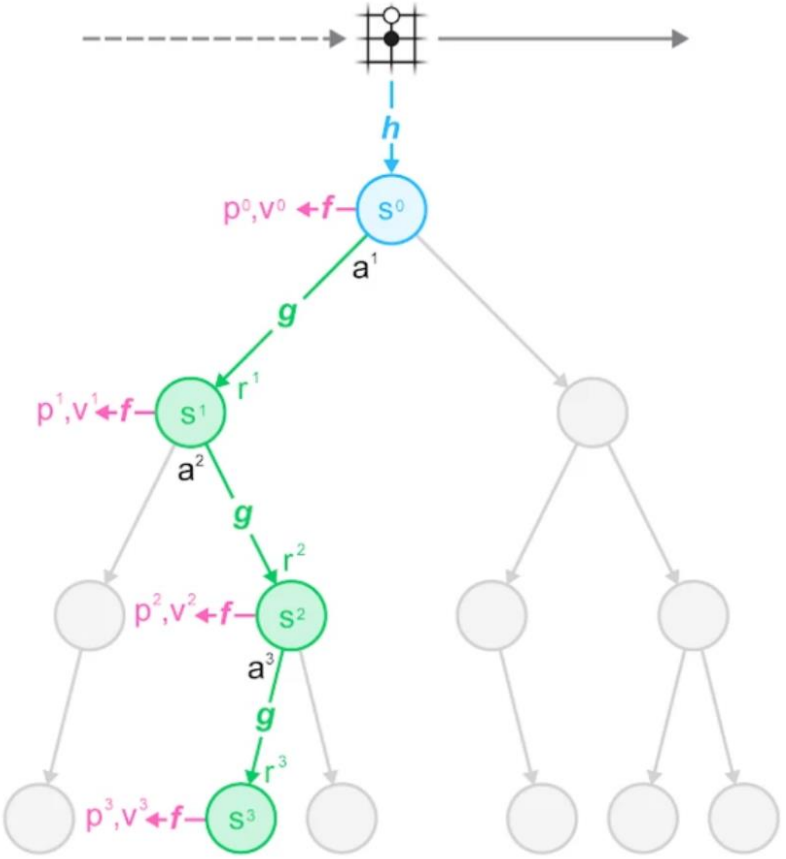
c_{puct} has no effect here because it also scales $P(s,a)$.

Evaluation

Failure Analysis (AlfWorld): Compounding error

Errors from transition and reward compounds as
Search tree gets deeper!

Accuracy of the transition model is crucial.



Evaluation

Pros and Cons of LLM comparing with Deep Learning.

- Pros:
    - No training and hyperparameter tuning needed.
    - No need to define a new network architecture to fit a new environment.
    - More generalizable in unseen situations.
- Cons:
    - LLM inferences are much more expensive and time consuming compared to small neural networks.
    - Sophisticated prompt engineering needed for complex tasks – expert knowledge needed.
    - Unlikely to produce super-human performance.

Conclusion

Pros and Cons of LLM comparing with Deep Learning.

- Implementing existing deep reinforcement learning method and large language based model methods across 2 env.

- Deep Reinforcement Learning
  - Actor-Critic with transformer network

- LLM-based
  - LLM-policy
  - LLM-MCTS

- Proposed our own method
  - LLMZero
    - Improvement upon LLM-MCTS needing the transition function from env.
    - Showed that it can perform as well as LLM-MCTS on at least one environment.

References

1. pyrddlgym-project/pyRDDLGym: A toolkit for auto-generation of OpenAI Gym environments from RDDL description files. [Internet]. [cited 2024 Nov 9]. Available from: https://github.com/pyrddlgym-project/pyRDDLGym/tree/main

2. Shridhar M, Yuan X, Côté MA, Bisk Y, Trischler A, Hausknecht M. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning [Internet]. arXiv; 2020 [cited 2024 Nov 9]. Available from: https://arxiv.org/abs/2010.03768

3. Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, et al. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. 2019 [cited 2024 Nov 9]; Available from: https://arxiv.org/abs/1911.08265

4. Zirui Zhao, Wee Sun, Lee David Hsu. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. 2023 [cited 2024 Nov 9]; Available from: https://arxiv.org/pdf/2305.14078