ASU College of Health Solutions
Arizona State University

EMBEDDED MACHINE INTELLIGENCE LAB

True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning

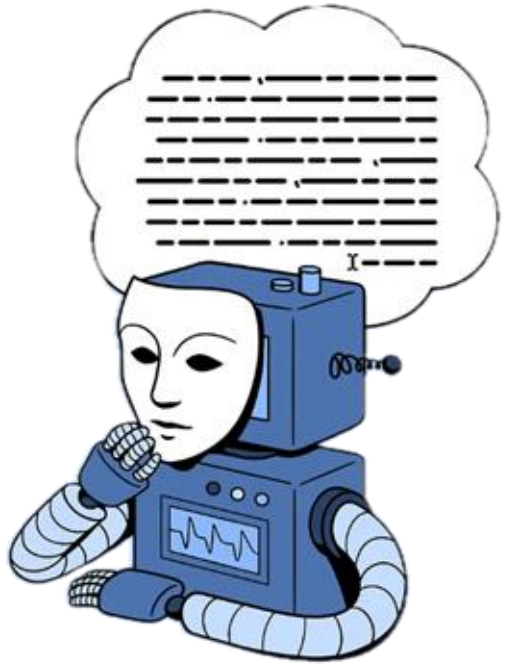
International Conference on Learning Representations (ICLR) – 2024
Citations: 90

Saman Khamesian

12/03/2025

Introduction

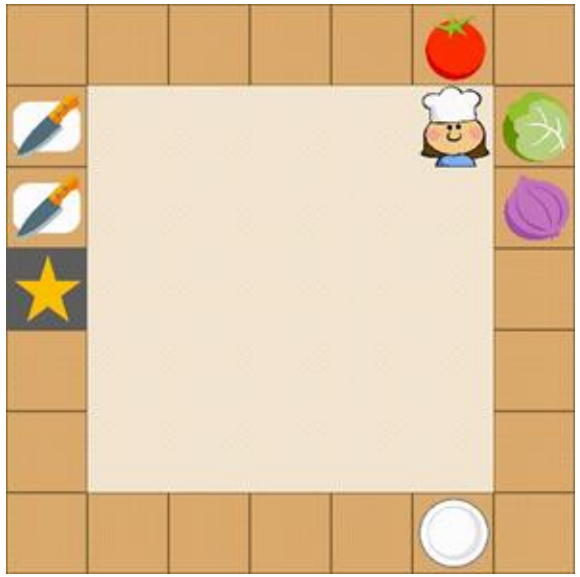
- **Large Language Models (LLMs)** can reason well, plan steps, and understand goals.
- But they're trained only on **text** — not on **how environments actually behave.**
- So when we ask them to act as agents, they often produce actions that "sound right" but **don't work in the real environment.**
- This gap between *knowing* and *doing* is the core problem we explore.

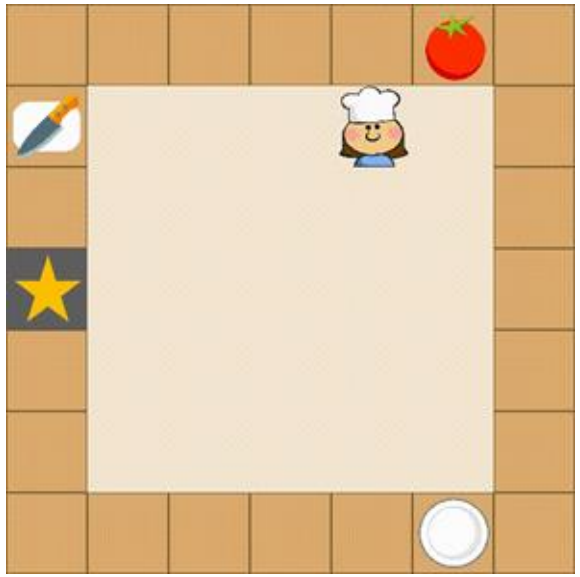


Large Language Model?
A deep learning model trained on vast text data to understand and generate human-like language. It can summarize, translate, answer questions, and create text.

Environment

- To understand this gap more clearly, let's look at the environment!
- Overcooked is a simple **grid-based kitchen environment** where an agent follows a sequence of steps to prepare dishes.





- ❖ The agent moves, picks up ingredients, chops, cooks, and serves.
- ❖ Actions only work if the environment allows them.
- ❖ Clear, strict rules make mistakes easy to detect.

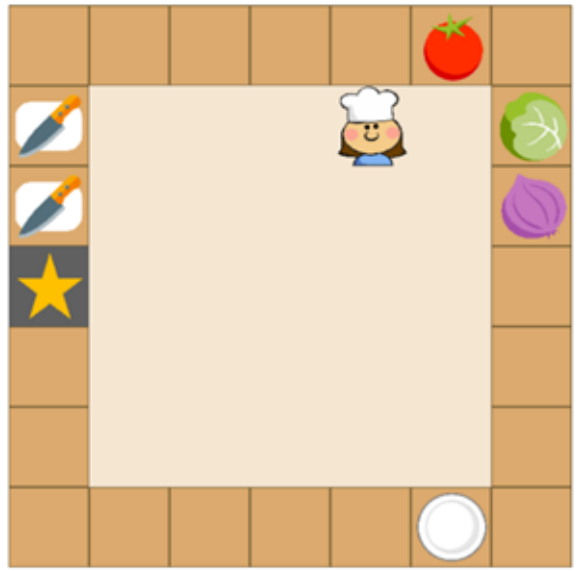
This makes Overcooked a perfect setting to show how LLMs break rules and what we mean by misalignment.

Action Misalignment

- Action Misalignment occurs when the LLM proposes actions that **sound correct** but **cannot be executed in the environment.**

- Examples from Overcooked:
  - Suggesting “pick up cucumber” even though cucumber doesn’t exist
  - Trying to place lettuce on a cutting board already holding tomato
  - Actions break environment rules despite being linguistically reasonable

- These mistakes show that language-based reasoning does not automatically translate into **valid environment actions.**



Only tomato, lettuce and onion are provided in the game.

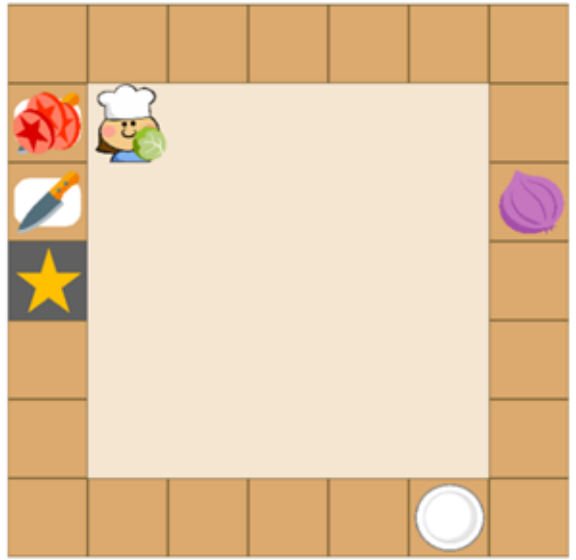
LLMs may choose to pick up additional ingredients, such as cucumber and pepper to cook the dish.

Learning Misalignment

- Learning Misalignment arises because LLMs are trained for next-token prediction, not reward-based improvement!

- **Key issues:**
  - They maximize *likelihood of text*, not *expected reward*
  - They lack *trial-and-error* adaptation
  - They do not improve their strategies across episodes
  - RL settings require feedback-driven learning, which LLMs are not naturally trained for.

- As a result, LLMs can produce **fluent actions** but fail to learn **effective long-horizon behaviors.**



LLMs guide the agent to put the lettuce on the cutting board, which already contains tomato, without knowing that each cutting board can only contain one item at a time.

Methodology

- With the misalignment problems clear, our goal is to create a method where:
  - LLM actions are grounded in **valid environment transitions**.
  - The model can learn from **rewards and feedback.**
  - Policies become **more consistent** over episodes.
  - The LLM keeps its language abilities while improving its decision-making
  - The whole system becomes more “agent-like” instead of just “text-like”

- To achieve these goals, the **T**rue kno**W**ledge c**O**me**S** fr**OM** practic**E** (TWOSOME) framework uses an RL-based approach to align LLMs with embodied environments.

LLM Prompt Design Principles

- Cohesion: Ensure observation and action prompts use linking phrases such as “you should”.
- Articles Sensitivity: Include proper articles (e.g., “pick up *the* tomato”).
- Context Reinforcement: Repeat key objects/goals to bias action probabilities.
- Context Adaptation: Modify phrasing based on state (e.g., “put the tomato on the plate” instead of “pick up the tomato” when the object is already held).

Valid Policy Generation

Problem: LLMs generate invalid actions that violate environment constraints.

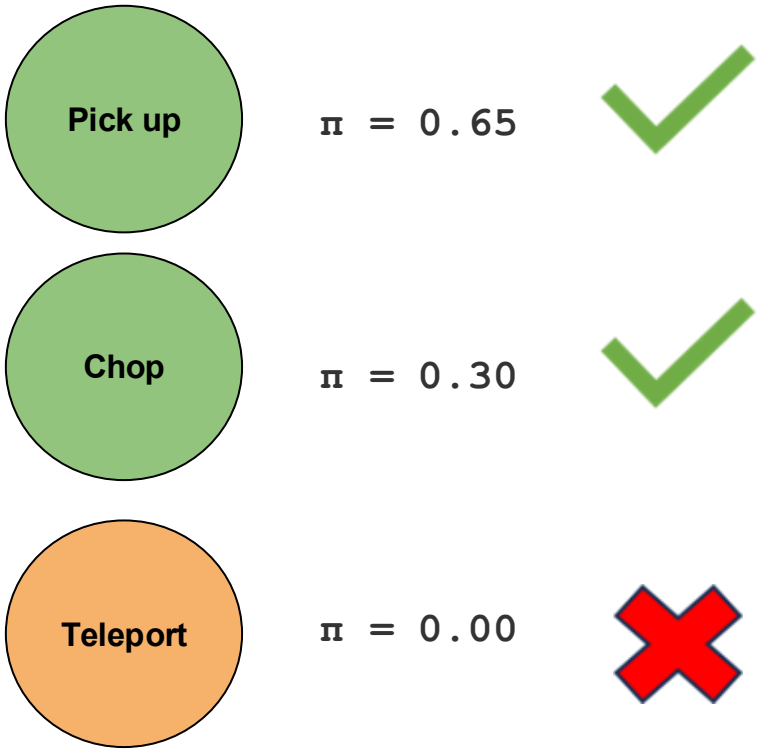
Solution : Instead of letting the LLM freely generate an action, TWOSOME evaluates the log-likelihood of each valid action prompt and normalizes these scores to form the policy.

$$P_{\text{token}}(a_k|s) = \prod_{i=1}^{N_k} P(w_i^k|s, w_1^k, \ldots, w_{i-1}^k)$$

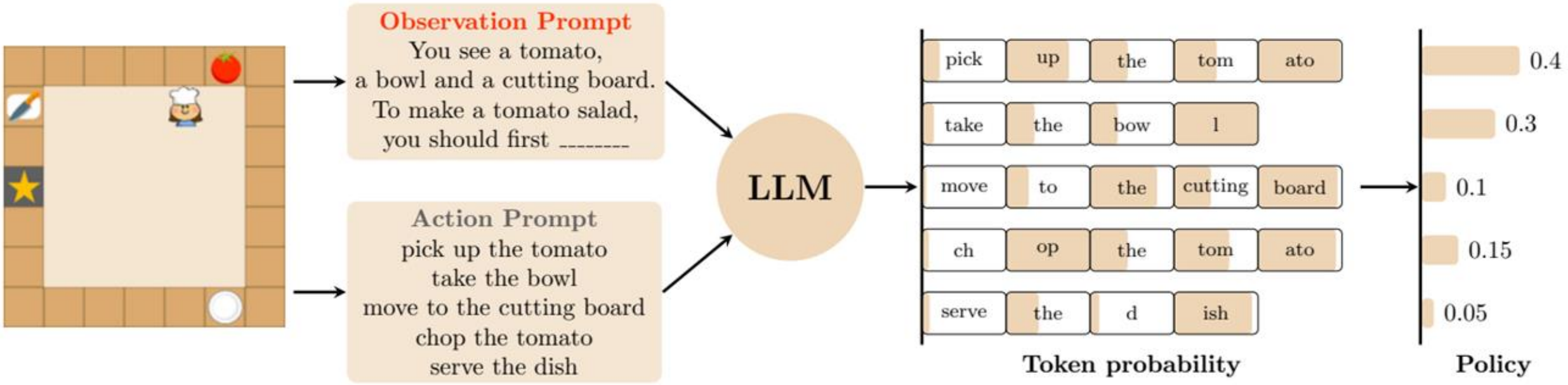
$$P(a_k|s) = \frac{\exp(\log P_{token}(a_k|s))}{\sum_{a\in\mathcal{A}} \exp(\log P_{token}(a|s))}$$

Valid Action Selection

State: Tomato Visible



TWOSOME Policy Generator



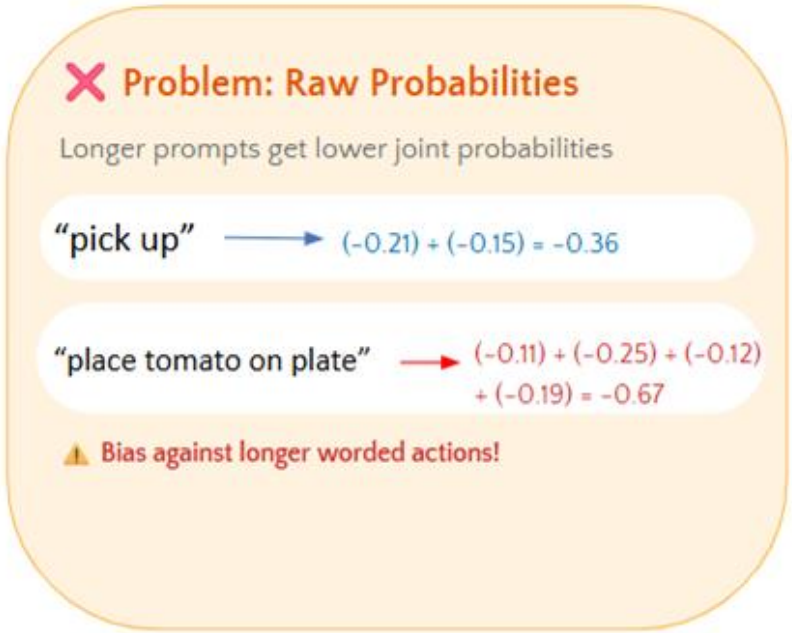
Action Prompt Normalization

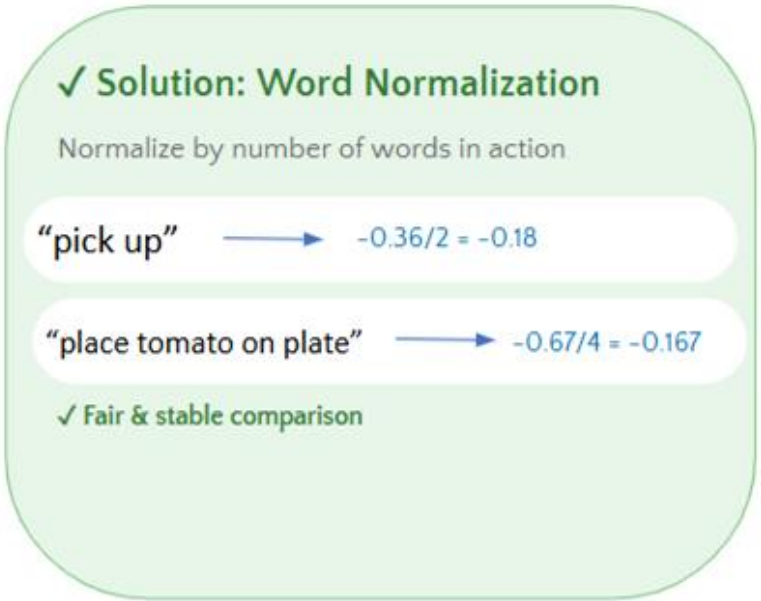
Problem: Longer prompts receive smaller joint probabilities, creating bias toward shorter actions.

Solution: Token and word normalization solve this issue by dividing the log-probability to create a balance. Word normalization is preferred because it treats multi-token words as single units, preventing unfair penalization of longer prompts due to tokenization quirks.

$$\log P^{\text{TN}}_{\text{token}}(a_k|s) = \frac{\log P_{\text{token}}(a_k|s)}{N_k}$$

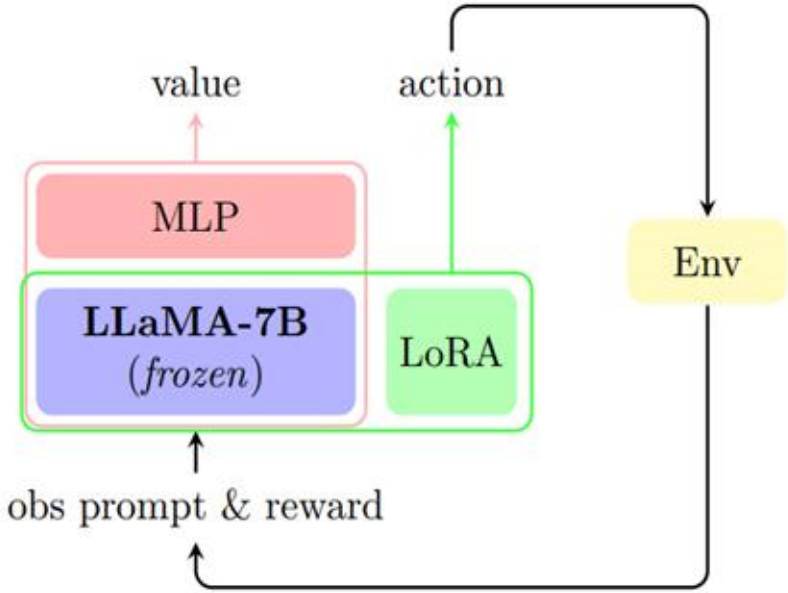
$$\log P^{\text{WN}}_{\text{token}}(a_k|s) = \frac{\log P_{\text{token}}(a_k|s)}{W_k}$$





Parameter-Efficient Architecture

Aspect	Actor (Policy)	Critic (Value)
Base Architecture	LLaMA-7B + LoRA	Multilayer Perceptron (MLP)
Rank/Layers	Rank r = 8-16 (low-rank decomposition on Query/Value projection matrix)	3-layer MLP Input(1024) → Hidden(512) → Output(1)
Trainable Parts	LoRA adapters (~4.2M)	All MLP weights
Pre-trained	Yes (LLaMA frozen)	No (learns from scratch)
Learning Rate	Low (~3e-5)	High (~1e-3)
Output	Policy π(a|s)	Value V(s)
Role in RL	Selects actions	Estimates advantage $A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

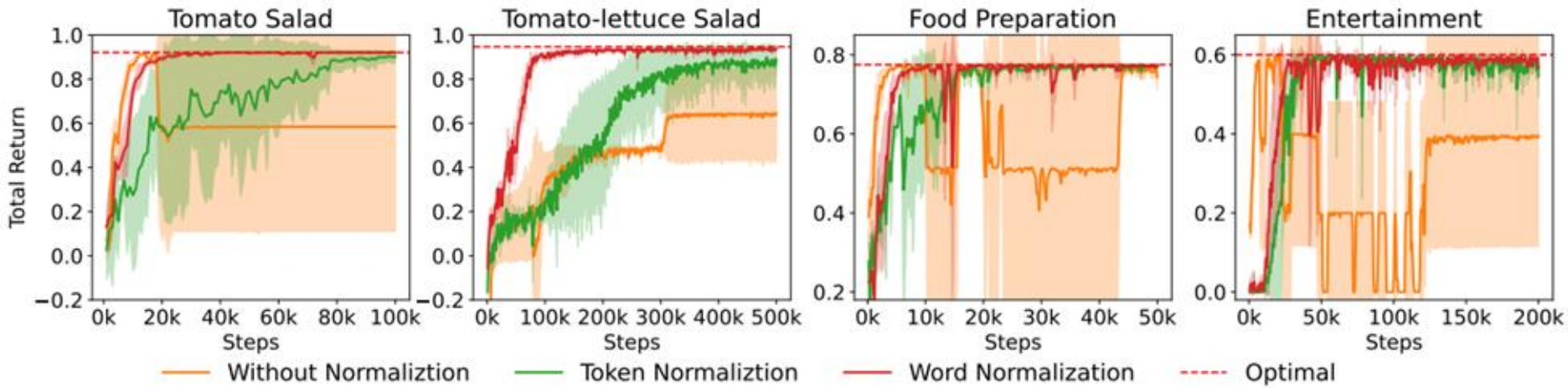


Key Benefit: The frozen LLaMA-7B backbone (7B parameters) preserves language knowledge while LoRA adapters (~4.2M parameters, 94% reduction) enable efficient task-specific fine-tuning.

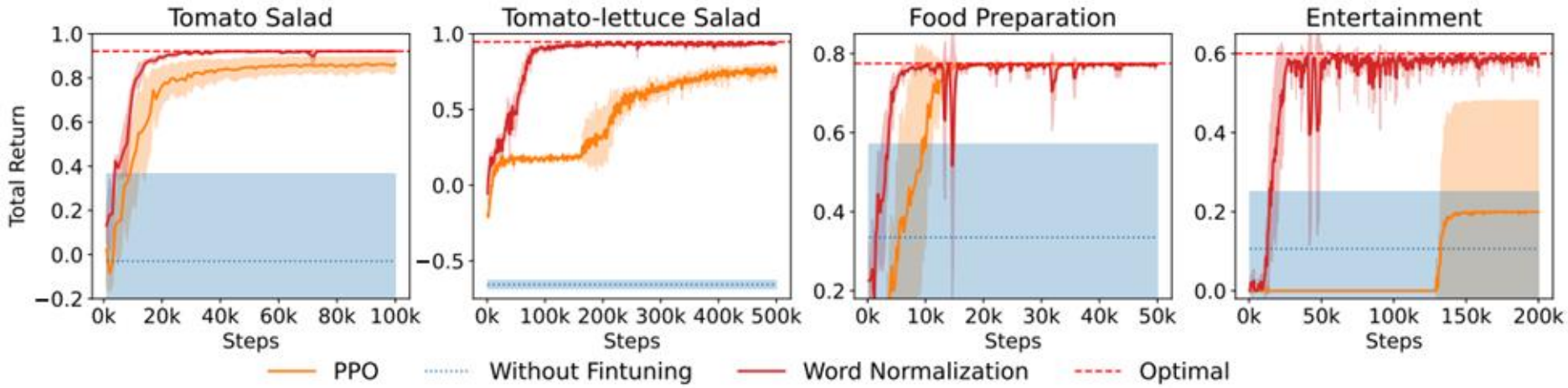
Complete Pipeline

1) Environment observation is converted into an **observation prompt**
2) All valid actions mapped to **action prompts (e.g. “Action: move_up”**)
3) LLM + LoRA computes joint token probabilities for each action
4) Select and execute the highest-probability valid action
5) Environment returns next state + reward
6) Update **LoRA (actor) + MLP critic (via an RL Algorithm like PPO)**
7) Loop continues until policy aligns with environment dynamics

Results and Observations

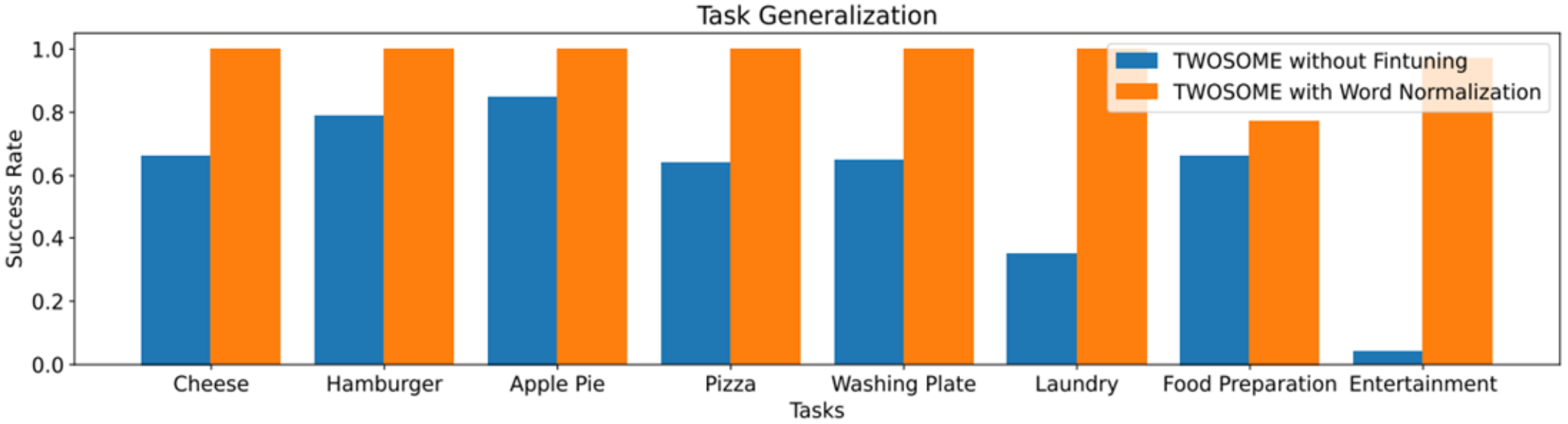


(a) Comparison of performance among finetuned TWOSOME.



(b) Comparison of performance among TWOSOME with word normalization and baselines.

Results and Observations



- VirtualHome environment
- TWOSOME agents that were originally trained on the Food Preparation and Entertainment tasks within VirtualHome are evaluated on eight new VirtualHome tasks.
- This setting checks whether the agent can transfer what it learned in one VirtualHome scenario to others without retraining.

Limitations of TWOSOME

Open-World Generalization

- Moving beyond scripted environments like Overcooked and VirtualHome to dynamic, real-world scenarios.

Computationally Expensive and Inefficient

- Every decision step requires the LLM to score all possible valid actions, leading to high latency and GPU demand.
- Real-time or large-scale deployment remains difficult due to slow inference speeds.

Limited Action Space

- The framework depends on a predefined list of macro-actions.
- This restricts adaptability in more open-ended or real-world environments.

Limitations of TWOSOME

Prompt Sensitivity

- Performance varies greatly with prompt phrasing, grammar, and structure.
- Lack of robustness to natural language variability limits reliability.

Data and Training Cost

- PPO fine-tuning with LoRA adapters is sample-efficient but computationally heavy.
- Requires significant resources and time for each environment.

THANK YOU !