Long Term LLM+RL Procedural Assistant

David Roskin*[1] and Omer Maayany Laor†[1]

[1]School of Electrical Engineering / Tel-Aviv University

October 22, 2025

Abstract

Large Language Models (LLMs) are increasingly used as procedural assistants but remain primarily *reactive* they respond to current inputs without anticipating future steps or reasoning about long-term outcomes. Moreover, their reliance on cloud inference limits use in offline or resource-constrained environments. We introduce a hybrid *LLM+RL procedural assistant* that integrates a reinforcement learning (RL) policy for *foresight* with a lightweight, locally run LLM for *reactive support*. Using a curated subset of EPIC-KITCHENS-100, we train a sequence policy to predict the next correct action from observed prefixes, constrained by a legality mask and a prefix–action bias table derived from dataset statistics. The policy learns to distinguish *correct*, *legal but incorrect*, and *illegal* continuations, achieving **97.7%** overall next-action accuracy and **97.5%** on multi-branch prefixes after 50,000 episodes. Predicted next actions are then provided to small, offline-capable LLMs (Mistral-7B, Phi-3-Mini, Llama-3.2-1B) to choose supportive helper actions via dynamic prompting. In a human evaluation of 1,200 outputs, RL-provided foresight substantially improves contextual relevance, with Mistral-7B achieving the highest quality (mean 8.28, median 10). Together, these components form a robust, offline assistant capable of proactive, context-aware guidance in multi-step tasks. The results show that coupling RL-based foresight with lightweight LLM reasoning bridges the gap between *reaction* and *planning* in autonomous assistance.

1 Introduction

1.1 Problem Description

Most LLM-based agents operate re-actively meaning they respond to immediate input and provide short-term solutions or the next step in a procedural task. However, they typically lack awareness or anticipation of future steps or the long term consequences of their actions.

*ID: 317491868, davidroskin@mail.tau.ac.il

†ID: 200477057, omermaayany@mail.tau.ac.il

This limitation can prevent the LLM from offering the most effective guidance or support, especially in multi-step processes. A second problem is the dependence of most LLMs on continuous internet connectivity. Due to their size and computational requirements, these models often run on remote servers or in the cloud. Yet in many real-world scenarios, such as medical procedures, industrial maintenance, or cooking, a constant online access may not be available or reliable. This restricts the deployment of helpful AI systems in offline or limited resource environments where real-time, local assistance is needed most.

1.2 Importance

The proposed solution addresses both key limitations of current LLM agents. First, by incorporating an RL component that predicts upcoming steps, the agent gains foresight into the process and can reason about the long-term effects of suggested actions and their sequence. This enables the agent to provide more strategic, context-aware guidance - a crucial capability for tasks that require careful planning, safety considerations, and procedural precision. Second, deploying a lightweight, domain-specific LLM that runs offline allows the system to operate effectively in resource-constrained or connectivity-limited environments. This is especially important in real-world scenarios such as medical procedures, fieldwork, or domestic tasks, where access to cloud-based models may not be feasible. Together, these two components make AI assistants more proactive, robust, and accessible - capable of delivering reliable support in a broader range of real-world conditions.

1.3 Overview of Results

rl results:
In our experiment with three large language models, we observed that some of them achieved strong results. The results demonstrate the expected influence of model size: the model with the highest number of parameters obtained superior performance. In addition, the two larger models consistently scored strong, reinforcing our hypothesis that external assistance can be used to provide additional context, thus improving the quality of the language model support suggestions.

2 Related Work

Large language models (LLMs) have become central to human-robot interaction due to their ability to combine contextual reasoning, factual knowledge retrieval, and task decomposition. The Transformer architecture (Vaswani et al., 2017) enabled these models to capture long-range dependencies efficiently, forming the basis for state of the art systems that generalize across diverse tasks. Beyond language fluency, LLMs encode relational and factual knowledge directly from text corpora, allowing them to operate as open-domain knowledge bases (Petroni et al., 2019). More recently, LLMs have been shown capable of decomposing high-level goals into actionable sub-steps, making them effective zero-shot planners for embodied agents (Huang et al., 2022). Their reliance on recent versus distant context, however, reveals both strengths and constraints in long-horizon reasoning (Khandelwal et al.,

2018). Taken together, these properties position LLMs as a promising foundation for robotic assistants that must interpret ongoing tasks, anticipate needs, and generate context-aware supportive actions.

Research on kitchen robotics underscores both the acceptance of such systems by society and the technical advances that enable their integration with language-based planning. The case of Moley Robotics illustrates how robotic tools in kitchens are increasingly perceived as essential, with growing public recognition of their role in future food preparation environments (Çakar, 2021). At the technical level, ubiquitous robotics approaches show how sensor-equipped kitchens allow robots to learn from demonstrations and autonomously perform complex tasks without excessive system complexity (Beetz et al., 2008). Extending these capabilities, recent work demonstrates how large language models, when combined with PDDL-based planning and vision-language models for ingredient recognition, can translate recipe descriptions into executable cooking behaviors in real-world settings (Sakurai et al., 2023). These studies together highlight that beyond social acceptance, the integration of LLMs with robotic platforms provides a powerful framework for context-aware assistance in kitchen environments.

Our system differs along three axes: (i) we learn a *prefix -conditioned* next -action distribution directly from demonstrations, enforcing *legality* via a subsequence -derived mask; (ii) we operate fully *offline* with compact models (both RL policy and LLM), reducing compute and connectivity requirements; and (iii) we decouple *foresight* (RL predicts the next step) from *reactive support* (LLM chooses one helper action from a small fixed vocabulary), which stabilizes behavior under plan divergence and simplifies evaluation.

2.1 Comparison to Prior Work

Prior LLM-based planners demonstrate strong performance in *action decomposition*, but they are sensitive to *plan divergence*. When execution deviates from a preplanned script by reordering steps, introducing mistakes, or completing the task via an alternative yet valid path these systems often struggle to infer the next *valid* action from the *actually executed* history. As a result, they cannot reliably anticipate what assistance will be needed downstream or how to adjust recommendations given the concrete steps already taken. Moreover, generic LLMs operate over an effectively *unbounded* action space, so they frequently propose irrelevant or unsafe steps; there is no built-in mechanism to constrain outputs to task-relevant actions only.

Kitchen–robotics integrations face related challenges. Many pipelines assume substantial sensing and perception (e.g., action and ingredient recognition, camera streams) and inherit brittleness to plan divergence from their symbolic or script-based planners. In practice, these systems also tend to rely on server-side components or high compute, complicating deployment beyond controlled environments.

Connectivity further constrains prior approaches. Access to factual knowledge bases or cloud inference is frequently required, creating a single point of failure and limiting usability in settings with unreliable or absent internet access.

In contrast, our system is grounded in *executed prefixes* rather than imagined plans. We learn a next-action policy over multiple *observed* trajectories and treat deviations as mistakes from which the agent can recover and rejoin a valid path. Given only the sequence

of actions performed so far, the RL component provides robust, long-horizon foresight, while a lightweight offline LLM selects a single supportive action from a *fixed, task-relevant action set*. A legality mask and prefix-conditioned bias table enforce these constraints at inference time, eliminating off-task suggestions that a pure LLM might produce. The entire pipeline runs locally (via `llama.cpp`) without external retrieval or online services, and it extends naturally to new procedures by appending trajectories to the dataframe no architectural changes or additional sensors are required.

3 Data

3.1 Description & Source

We use a curated subset of EPIC -KITCHENS -100 (EK -100) comprising ∼500 videos. The official CSV provides time-aligned annotations per video: `video_id`, `verb` (the action performed), and `noun` (the object of the action). We compose each atomic action as `verb` + `noun`, yielding a vocabulary of >10,000 distinct actions across the subset.

3.2 Preprocessing

1. **Cleaning.** Lowercasing, punctuation stripping, and removal of unintelligible entries. We filter out semantically irrelevant kitchen nouns (e.g., *packaging*, *wrapper*) and collapse *immediate consecutive* duplicates to reduce trivial repetitions.
2. **Sequence construction.** For each `video_id`, actions are grouped in temporal order to form a trajectory. We prepend a `START` token and append an `END` token to mark episode boundaries and termination.
3. **Embedding.** Actions are indexed to integer IDs for efficient use in the GRU policy and RL loop.
4. **Prefix tree.** We build a trie whose root is `START`. Each root-to-leaf path encodes a *legal & correct* trajectory observed in EK -100; the children of any node are the next *ground-truth* actions from that prefix.
5. **Subsequence→next map.** To avoid scoring logits over the entire >10k action space, we construct a dictionary that maps bounded-length subsequences to the *empirically observed* set of next actions in the dataset.

3.3 Union-of-Legal from Suffixes of a Given Prefix

At training time, for the current prefix $\pi = [a_1, \ldots, a_t]$, we derive the set of *allowed* next actions by inspecting the **suffixes of** π up to a backoff length L (e.g., last 1 -L actions), querying the subsequence→next map for each suffix, and taking the union:

$$\mathcal{A}_{\text{legal}}(\pi) \;=\; \bigcup_{k=1}^{\min(L,t)} \text{Next}\big([a_{t-k+1}, \ldots, a_t]\big).$$

This captures locally valid continuations even when the global prefix did not appear from the very start of a video.

Worked example (IDs). Let the current prefix be $\pi = [1, 5, 6, 23]$ and $L = 4$. We consider the suffixes [23], [6, 23], [5, 6, 23], [1, 5, 6, 23]. Suppose the map returns:

$$\begin{aligned} [23] &\to \{8, 9, 76, 25, 79, 150\}, \\ [6, 23] &\to \{8, 9, 76, 150\}, \\ [5, 6, 23] &\to \{8, 9, 150\}, \\ [1, 5, 6, 23] &\to \{8, 9\}. \end{aligned}$$

Then

$$\mathcal{A}_{\text{legal}}(\pi) = \{8, 9, 76, 25, 79, 150\}, \qquad \mathcal{A}_{\text{correct}}(\pi) = \{8, 9\}.$$

Here, $\{8, 9\}$ are the *ground-truth next actions* (tree children), while $\{76, 25, 79, 150\}$ are *legal but incorrect* for this prefix. In practice, even with bounded backoff, $\mathcal{A}_{\text{legal}}(\pi)$ can contain *hundreds* of actions and typically dwarfs $\mathcal{A}_{\text{correct}}(\pi)$, making the task highly imbalanced.

3.4 Legal vs. Correct Actions (Class Imbalance)

Formally,

$$\left|\mathcal{A}_{\text{legal}}(\pi)\right| \gg \left|\mathcal{A}_{\text{correct}}(\pi)\right|, \quad \text{with} \quad \mathcal{A}_{\text{correct}}(\pi) \subseteq \mathcal{A}_{\text{legal}}(\pi).$$

Because the allowed set often contains many more *legal but incorrect* options than true next actions, naive exploration is unlikely to hit the correct continuation by chance.

Action taxonomy at a prefix. Using only dataset-derived structure, we partition next actions at prefix π into:

1. **Correct** (*tree children*): actions that appear as direct children of π in the ground-truth prefix tree (i.e., next actions actually observed in EK -100 for π).
2. **Incorrect but legal** (*union-legal only*): actions in the union-of-legal set computed from suffixes of π via the subsequence→next map, *but not* children of π in the tree.
3. **Illegal**: all remaining actions outside the union-of-legal set.

This partition cleanly distinguishes the dataset's true continuations at π, alternative yet observed continuations from related contexts, and actions never observed as valid next steps.

4 Methods

4.1 Approach

We pair a reinforcement learning (RL) next -action predictor with a lightweight offline LLM:

- **RL for foresight.** Given the actions already performed (a prefix), the RL policy predicts the *next* correct action. This provides long-horizon guidance: it anticipates what will likely happen next.
- **LLM for reactive support.** Conditioned on the RL next action and a fixed helper-action list, the offline LLM proposes a short, *supportive* step that best prepares for the predicted next action. Thus, the LLM stays reactive, while the RL model injects foresight.

4.2 RL Design

States, actions, rewards. A state s_t is the current prefix $\pi_t = [a_1, \ldots, a_t]$. Actions are discrete atomic actions from the shared vocabulary and are partitioned *at the current prefix* into:

1. **Correct**: children of π_t in the ground-truth prefix tree;
2. **Incorrect but legal**: present in the union-of-legal set from suffixes of π_t, but not tree children;
3. **Illegal**: all others.

We assign $r_t = +1$ for a correct action and $r_t = -0.5$ for an incorrect-but-legal action. Illegal actions are masked out (see below). A correct action extends the prefix; an incorrect action leaves the prefix unchanged (enabling recovery to the correct branch). Episodes terminate on `END` or after 100 steps.

Legality mask. Let $\mathcal{A}_{\text{legal}}(\pi)$ be the union of next actions derived from suffixes of the prefix (bounded backoff length L). We add a mask $M \in \mathbb{R}^{1\times V}$ to the logits, where $M_i = 0$ for $i \in \mathcal{A}_{\text{legal}}(\pi)$ and $M_i = -10^9$ otherwise.

Prefix×Action bias table. We maintain a sparse table $B \in \mathbb{R}^{|\mathcal{P}|\times V}$ (init. zeros). At each step we update the cell (π, a) with a clipped additive rule:

$$B[\pi, a] \leftarrow \text{clip}\Big(B[\pi, a] + \begin{cases} \eta_+, & r_t > 0 \\ -\eta_-, & r_t < 0 \end{cases}, [-c, c]\Big).$$

Example. If $\pi = [1, 5, 6]$ and $a = 9$ is correct ($r_t = +1$), with $\eta_+ = 0.5$, $c = 2.0$: $B[\pi, 9] \leftarrow \min\{B[\pi, 9] + 0.5,\ 2.0\}$. If $a = 76$ is incorrect ($r_t = -0.5$), then $B[\pi, 76] \leftarrow \max\{B[\pi, 76] - 0.5,\ -2.0\}$. At inference we add the row $B[\pi, \cdot]$ to the logits, biasing toward historically correct choices without forbidding exploration.

4.3 Policy Network (Why GRU)

The policy/value network is a GRU over embedded action IDs (128-dim embeddings, 256-dim hidden). GRUs are well-suited here because:

- They natively model *procedural sequences* and next-step prediction;

- Compared to LSTMs, GRUs are *lighter* (fewer parameters), important for offline/low-compute settings;
- Gated dynamics mitigate vanishing/exploding gradients while retaining long-range dependencies.

Given the final hidden state h_t:

$$\text{logits} = W_p h_t + b_p \in \mathbb{R}^V, \qquad V_\theta(s_t) = W_v h_t + b_v \in \mathbb{R}.$$

We add the legality mask and bias row, then compute probabilities with a (temperature-scaled) softmax:

$$p(a \mid s_t) \;=\; \text{softmax}\Big(\tfrac{1}{\tau}\left(\text{logits} + M + \alpha\, B[\pi_t, \cdot]\right)\Big),$$

where $\tau > 0$ is the temperature (see below).

GRU equations (completeness). For embedding x_t and previous state h_{t-1}:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) && \text{(update)} \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) && \text{(reset)} \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \end{aligned}$$

4.4 Training and Losses

We train for 50,000 episodes, each capped at 100 steps, sampling start states from random prefixes. Optimization uses Adam and gradient clipping. We compute advantages with GAE:

$$\delta_t \;=\; r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t), \qquad \hat{A}_t \;=\; \delta_t + \gamma\lambda\, \hat{A}_{t+1},$$

with per-episode advantage normalization.

What each loss does.

$$\mathcal{L}_{\text{policy}}(\theta) = -\,\mathbb{E}\Big[\log \pi_\theta(a_t \,|\, s_t)\, \hat{A}_t\Big]$$

(increase prob. of actions with positive advantage; decrease if negative)

$$\mathcal{L}_{\text{value}}(\theta_v) = \tfrac{1}{2}\,\mathbb{E}\Big[(V_{\theta_v}(s_t) - \hat{R}_t)^2\Big]$$

(train a calibrated critic to reduce policy gradient variance)

$$\mathcal{H}(\pi_\theta) = \mathbb{E}\left[-\sum_a \pi_\theta(a \,|\, s_t) \log \pi_\theta(a \,|\, s_t)\right]$$

(promote exploration by discouraging premature peaking)

The total loss is

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{policy}} + c_v\, \mathcal{L}_{\text{value}} - c_e\, \mathcal{H}.$$

Why this balance: the policy term learns the control, the value term stabilizes learning by providing a low-variance baseline, and the (negative) entropy term maintains sufficient exploration in the presence of many legal-but-incorrect options.

Entropy: start-from-beginning vs. random-prefixes. If we *start from the beginning* of trajectories, the goal is to *discover* the breadth of legal branches; we therefore emphasize *exploration* (larger c_e, higher entropy). In contrast, when we *start from random prefixes*, after many steps the correct continuation is often highly specific (empirically, after $\sim$30 steps there is usually a single correct action). In that regime we prioritize *exploitation* (smaller c_e), because the task is to *home in* on the unique correct next step rather than to broaden coverage.

Temperature for exploration control. We also apply temperature τ in softmax:

$$p(a \mid s_t) = \frac{\exp\big((\ell_a)/\tau\big)}{\sum_j \exp\big((\ell_j)/\tau\big)}, \quad \ell = \text{logits} + M + \alpha B[\pi_t, \cdot].$$

Lower $\tau < 1$ *sharpens* the distribution (more exploitative; concentrates mass on top actions). Higher $\tau > 1$ *flattens* it (more exploratory; spreads probability mass). This gives a simple, continuous knob to modulate exploration without changing the network or rewards.

4.5 Operational Summary

At each step, the GRU encodes the prefix, produces logits and a value; we add legality mask and the bias row, apply temperature-scaled softmax, sample an action, apply reward and bias updates, and (if correct) extend the prefix. Over time the agent learns to separate *legal* from *actually correct* continuations, delivering reliable next-action foresight for the offline LLM to exploit.

4.6 LLM Dynamic Prompting and Model Setup

To inject RL foresight into a reactive helper, we use *dynamic prompting*: the LLM receives the *current action* (mandatory), the RL *predicted next action* (optional), and a fixed *helper-action list* from which it must choose a single supportive step.

Models (offline-capable). We evaluate three lightweight, locally runnable instruction models:

- **Mistral-7B** (7B parameters),
- **Phi-3-mini** (3B),
- **Llama-3.2** (1B).

Prompting interface. Each model uses a minimal, model-specific template but the same semantics:

- **Inputs:** current action (required), predicted next action (if available), fixed helper-action list.

- **Instruction:** "Act as a robotic kitchen assistant. Choose *one* helper action from the list that best prepares for the predicted next action. Respond in $\leq$10 words."
- **Constraint:** the output must be *one* item from the closed helper-action list (no free-form actions).

Runtime setup (local inference). To execute large batches, we host each model sequentially via `llama.cpp` on a dedicated local server. This lightweight C++ runtime enables fully offline inference and efficient throughput on constrained hardware. We stream RL outputs (prefix $\rightarrow$ predicted next action) into the prompts and write model responses to CSV for evaluation.

4.7 Second Stage of the Experiment (LLM Evaluation Protocol)

We assess how well offline LLMs propose *supportive* actions when given RL foresight.

Environment & deployment. Experiments run in Google Colab (Pro) for stable GPU access. Each LLM is served locally (via `llama.cpp`) and queried in a separate pass to avoid contention.

Batch inference. We generate $\sim$40,000 RL predictions (current action, RL next action). For each triplet $\langle$current, predicted next, helper-action list$\rangle$, the LLM must return exactly one helper action from the list. Outputs are logged to CSV alongside inputs.

Sampling & filtering for human evaluation. From the full set, we sample 400 rows per model (total $N = 1200$). This size follows standard finite-population guidelines (95% confidence, $\pm$5% margin). We remove rows where the RL next action is `END` or the context is too trivial to admit a meaningful supportive step.

Scoring protocol. Two authors score each item on a 1–10 scale:

- **1:** incoherent/irrelevant,
- **10:** concise, contextually correct, and practically useful for the predicted next action.

We report per-model *mean* and *median* scores as the primary metrics. These summarize central tendency while being robust to occasional outliers (e.g., terse refusals or overconfident but inapplicable suggestions).

.

5 Conclusion

5.1 RL Result & Evaluation

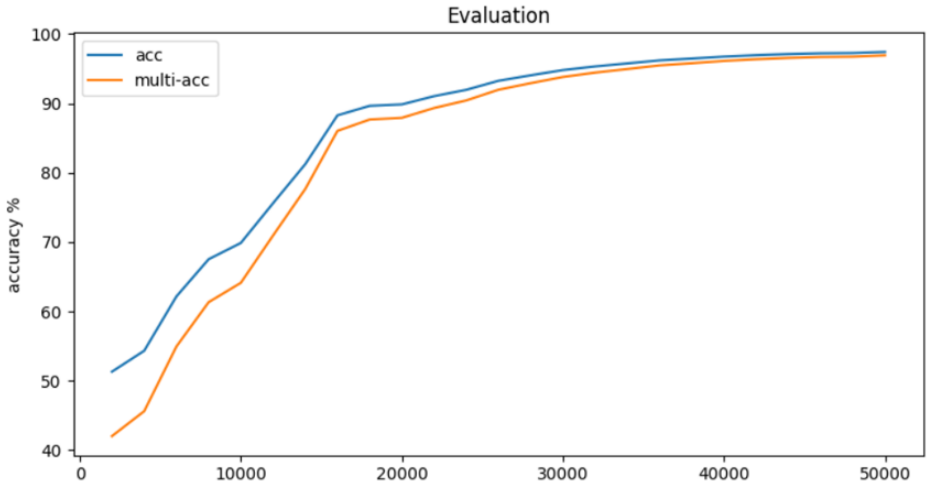

Figure 1: Prefix next action prediction accuracy

Our primary metric was *next–action accuracy*: given a prefix (the actions completed so far), how often does the policy predict a ground-truth next action? After 50,000 episodes, the policy reaches **97.7%** overall next–action accuracy. Restricting to prefixes where more than one legal continuation exists (*multi-choice* prefixes), accuracy is **97.5%**. For reference, when we evaluate against a setting that allows only the ground-truth next actions (i.e., no legal-but-incorrect distractors), accuracy reaches **98.4%**.

Taken together, these results indicate that the agent has learned the task's dynamics almost entirely: it reliably follows the correct trajectory even when multiple legal continuations are present, and its performance approaches the empirical upper bound defined by the dataset's ground truth.

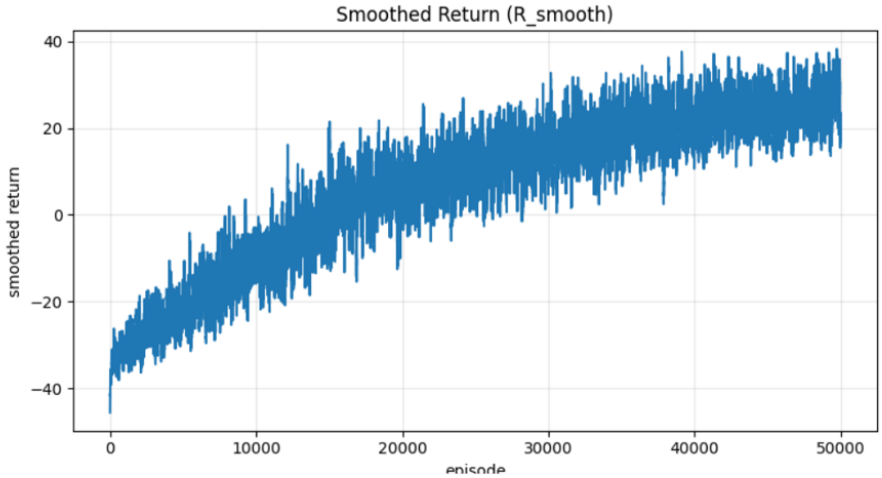

Figure 2: Smoothed Returns vs episodes

R_{smooth} rises steadily from about -45 (near the worst-case -50) to oscillate in the 20–40 band; since our reward is $+1$ (correct) and -0.5 (incorrect), $R_{\text{smooth}} > 25$ roughly indicates that more than half of the steps are correct on average. Although this still trails the near–ground-truth next-action accuracy, it highlights how the prefix×action *bias table* accelerates learning by nudging logits toward historically correct continuations.

Why the gap? Coverage is the main bottleneck. We face up to $\sim$ 40,000 distinct prefixes, but only 50,000 episodes with 100 steps each insufficient to fully learn many rare prefixes. Moreover, some prefixes admit 200–300 *legal* next actions (via the union-of-legals), often ending in common actions followed by many alternatives. A few mistakes in such hard states, coupled with the "stay-in-state" dynamics, can depress episode returns even when most other steps are correct. Given the persistent upward trend, we expect R_{smooth} to settle higher (empirically in the $\sim$ 80–100 range) with additional training as more rare prefixes are reinforced.

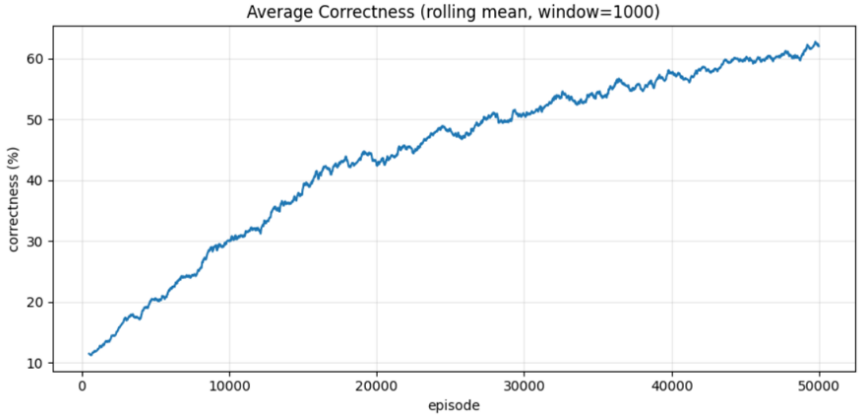

Figure 3: Step correctness average

The per-episode correctness fraction climbs from $\sim$ 10% to $\sim$ 65%, mirroring the upward trend in R_{smooth}. This again underscores the benefit of the prefix×action *bias table*, which steers probabilities toward historically correct continuations. The metric still lags behind the near–ground-truth next-action *accuracy* because correctness is averaged over full episodes: a small number of hard prefixes with 200–300 legal options can incur several mistakes and depress the episode's mean, even when most other steps are right. As coverage improves (more episodes visiting rare prefixes), we expect the curve to keep rising; empirically it trends toward the high 80s–90s, and with sufficient training can approach the accuracy curve.

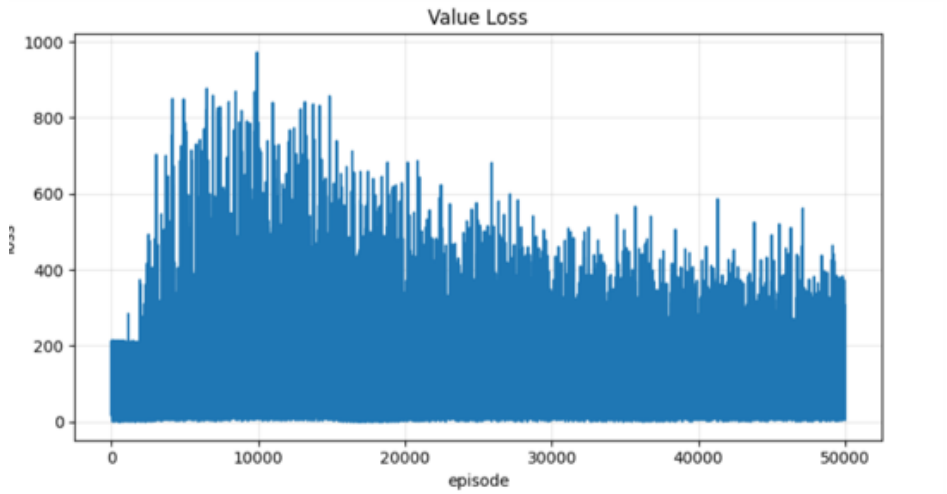

Figure 4: Average Value loss per episodes

Perhaps the strongest indicator of the bias table's impact is the steady drop in value loss, which decreases from roughly $8 \times 10^2 - 10^3$ down to $\sim 4 \times 10^2$. Since the value loss is an MSE between predicted returns and empirical returns, MSE ≈ 400 implies an RMSE of ~ 20 reward units. On a 100-step horizon with $(+1/-0.5)$ rewards, this corresponds very roughly to being off by on the order of 10–20 step-equivalents in expected return. In other words, the critic is getting substantially better calibrated, but there remains meaningful headroom. Given the continuing downward trend (and broader prefix coverage with more episodes), we expect further reductions as the bias table concentrates probability mass on historically correct continuations and the value head learns the long-horizon return structure more precisely.

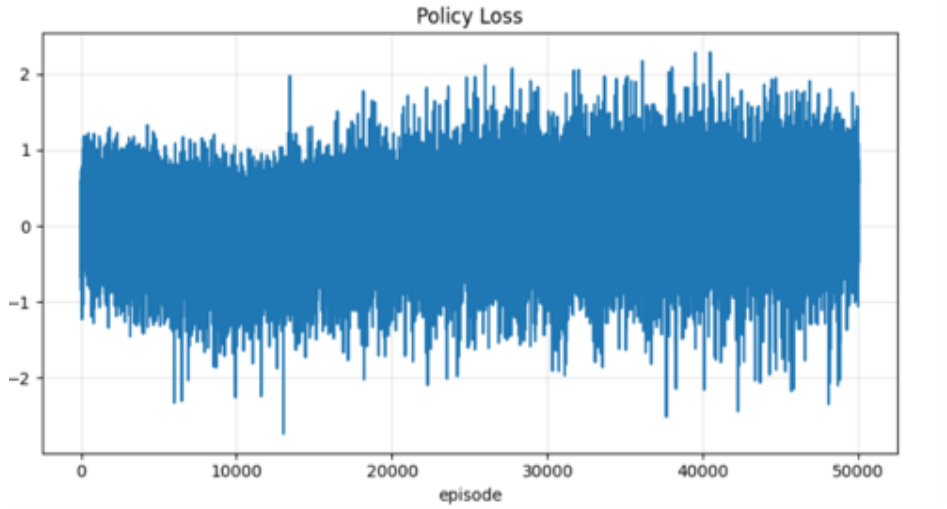

Figure 5: Average Policy loss per episodes

Throughout training the policy loss remains in a narrow band (approximately -1 to 1):

$$\mathcal{L}_{\text{policy}} \;=\; -\,\mathbb{E}\Big[\log \pi_\theta(a_t \mid s_t)\, \hat{A}_t\Big]\,.$$

Because we (i) normalize advantages and (ii) use a moderate softmax temperature, the advantages $\hat{A}_t$ are $\mathcal{O}(1)$, which keeps gradient magnitudes well scaled and prevents unstable updates. The near–zero mean with small oscillations indicates that the policy tracks the sign of the advantage closely up–weighting actions with positive $\hat{A}_t$ and down–weighting those with negative $\hat{A}_t$ without exploding or vanishing gradients. Concretely, this reflects:

- **GRU credit assignment:** the GRU backbone provides smooth, history–aware updates that stabilize long–horizon credit assignment.
- **Sharper logits:** the legality mask and prefix×action bias table sharpen action logits, keeping policy gradients informative.
- **Variance control:** batching updates (every 100 episodes) further damps gradient variance, promoting steady adaptation to reward signals.

Net effect: the policy adapts promptly to the observed advantages while maintaining stable learning dynamics over the full training run.

5.2 LLM Results

Table 1: Average and median evaluation scores for the three language models.

Model	Average Score	Median Score
Mistral-7B	8.28	10
Phi-3-mini (3B)	7.09	7
Llama-3.2 (1B)	4.97	5

The evaluation results highlight clear differences in the performance of the three models. The Mistral-7B model achieved the strongest outcomes, with an average score of 8.28 and a median score of 10, indicating that most of its responses were coherent, contextually appropriate, and closely aligned with human reasoning in kitchen tasks. The Phi-3-mini model obtained moderate results, with an average of 7.09 and a median of 7, reflecting a consistent but somewhat less accurate level of performance. In contrast, the Llama-3.2 (1B) model scored significantly lower, with an average of 4.97 and a median of 5, suggesting limited capability to produce contextually relevant or useful outputs.

These findings further reinforce the expected effect of model size on performance: larger models not only achieved higher average scores but also demonstrated greater reliability, as reflected in their higher median values. The results strongly support the hypothesis that leveraging larger language models, particularly in combination with RL-generated contextual information, substantially improves the quality of supportive action recommendations.

When the models were provided only with the current action but not the predicted next action, the results, as expected, dropped sharply. For example, in the Llama-3.2 model, the absence of the predicted action led to a decrease of 2.18 points in the average score and 2 points in the median score.

5.3 RL Conclusions and Takeaways

Our experiments demonstrate that a lightweight sequence policy, trained on EK–100 prefixes with a legality mask and a prefix×action bias table, can deliver reliable long-horizon guidance.

- **Foresight for planning.** Given sufficient episodes and diverse trajectories, the RL policy attains high next-action accuracy (e.g., ∼97.7% overall; ∼97.5% on multi-branch cases), providing the foresight that downstream LLMs can leverage to anticipate and adjust future assistance.
- **Bias table accelerates learning.** The prefix×action bias term sharpens logits, reduces sample complexity, and helps the policy track ground-truth branches more quickly, yielding near–ground-truth behavior.
- **Recovery from divergence.** The environment design (rewarding only correct continuations and staying in place on mistakes) allows the agent to recover from off-trajectory actions and rejoin the correct path instead of cascading into failure.
- **Action constraints keep the agent on task.** The legality mask prunes impossible actions and focuses exploration on task-relevant options, improving stability and speeding recovery while avoiding irrelevant or infeasible steps.

5.4 LLM Conclusion and Takeaways

Several notable trends and behaviors were observed across the three models. The Llama-3.2 (1B) model exhibited a bias toward selecting the "Wash" action, possibly because this was the first item in the action list. Some of the "medium" scores reflected cases where the model selected the same action predicted by the RL model rather than a supportive action. In some contexts, this substitution was reasonable, while in others it was less appropriate.

Interestingly, there were cases in which all three models received a perfect score of 10, despite choosing different actions (see Figure 1). The highest-quality responses (score = 10) were typically obtained when the predicted future action introduced a new component not present in the current action. In such cases, the models correctly inferred that the most helpful step would be to fetch the required item.

current_action	predicted_action	mistral_7b_model_reply	mistral_7b_model_score	phi_3_mini_model_reply	phi_3_mini_model_score	llama_3.2_model_reply	llama_3.2_model_score
throw-off dirt	wipe surface	I'd clean the surface before my chef wipes it.	10	Fetch cleaning supplies to assist chef.	10	I'll wash the surface.	10

Figure 6: Examples where all three models received a score of 10, despite selecting different actions.

The larger Mistral-7B model demonstrated more advanced reasoning capabilities. It was able to propose supportive actions involving items not explicitly mentioned, such as suggesting a towel when the current action was hand washing, or recommending a knife and cutting board when the predicted action was chopping. Additionally, Mistral generally respected practical limitations, avoiding unrealistic suggestions such as fetching water or a sink, mistakes that were occasionally made by the smaller models.

In situations where no clear supportive action was available, models that opted to "wait" rather than generate an irrelevant suggestion were scored relatively high, as this behavior reflected robustness. Finally, the Llama-3.2 model occasionally interpreted the presence of a knife as a potential safety hazard, refusing to provide a response in such cases. While this behavior reduced its utility in the experiment, it highlighted the model's sensitivity to risk interpretation.

In terms of processing speed, the Mistral-7B model, although the slowest among the three, still processed approximately 4 iterations per second. The other two models operated at a roughly tenfold faster rate, demonstrating that all models achieved processing speeds sufficient for near real-time performance.

5.5 RL: Future Work

We outline several directions to strengthen robustness, coverage, and generalization:

- **More compute & longer rollouts.** Increase episode length and total episodes to visit more prefixes per run, improving coverage of the large state space and reducing variance in value estimates.
- **Broader action space, lighter constraints.** Gradually relax (or ablate) the legality mask to stress-test recovery from divergence and learn to down-weight merely-legal but incorrect branches without hard pruning.
- **Generalization beyond seen paths.** Train on *all* subsequences (not only prefixes) and augment with trajectory perturbations to enable next-action prediction on unseen yet valid continuations.
- **Algorithmic upgrades.** Compare REINFORCE to PPO/GAE and other policy-gradient variants (e.g., A2C, TRPO); study entropy schedules and temperature annealing for exploration control.
- **Architecture exploration.** Evaluate deeper/wider GRUs, gated CNN/Transformer encoders with causal masking, and LSTMs; measure trade-offs in sample efficiency and stability.
- **Curriculum & sampling.** Use curriculum learning (short $\rightarrow$ long contexts), prioritized sampling of hard prefixes (large legal sets, frequent errors), and balanced correct/incorrect ratios.
- **Bias table learning.** Tune bias learning rates, clipping, and decay; try learned bias via a small prefix-conditioned adapter instead of a static table.
- **Evaluation at scale.** Expand datasets beyond EK–100, add cross-recipe and cross-domain tests, and report long-horizon metrics (success on N-step continuations) in addition to one-step accuracy.
- **Online adaptation.** Enable continual updates from user corrections at inference time (on-device), with safeguards to prevent drift.

- **Safety & constraints.** Incorporate task-specific safety rules and hierarchical options (high-level subgoals → low-level actions) to reduce error cascades.

5.6 LLM Future Work

One aspect we did not explore, due to limited computational resources and time constraints, was fine-tuning of the models. Fine tuning could potentially improve performance by adapting the models more closely to the specific vocabulary, action sequences, and decision-making requirements of the kitchen assistant domain. Such adaptation would likely reduce irrelevant or incoherent outputs and enhance the consistency of supportive action recommendations.

In addition, our evaluation covered only three models out of a wide variety of available language models across different sizes and architectures. It is possible that other models, particularly those optimized for reasoning or instruction-following, could yield superior results.

Another limitation of our experimental setup was that the models were tested without any memory of past actions, effectively treating each input as a Markov process. The evaluation was not performed in the natural sequence of recipe steps but rather on randomized samples. Furthermore, each model was provided only with the current action (and sometimes the predicted next action), rather than the full chain of preceding steps. A promising direction for future work would be to incorporate the entire process history as a dynamic input variable or to extend the models with an explicit memory mechanism. Such an approach, however, would require models capable of handling sufficiently large context windows.

6 Code

All code to reproduce this project is available at: GitHub Link.

References

Michael Beetz, Dipesh Jain, Lorenz Mösenlechner, Michael Schreiber, and Moritz Tenorth. Robots in the kitchen: Exploiting ubiquitous sensing and actuation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3510–3515, 2008.

Kamil Çakar. The use of robotics in the kitchens of the future: The example of 'moley robotics'. *Journal of Tourism and Gastronomy Studies*, 9(2):1084–1096, 2021.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning (ICML)*, pages 9118–9147, 2022.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 284–294, 2018.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2463–2473, 2019.

Ryo Sakurai, Takashi Kobayashi, and Tetsuya Ogata. Real-world cooking robot system from recipes based on food state recognition using foundation models and pddl. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11582–11589, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.