STEP: AGENT CONFIDENCE ESTIMATION FROM EXECUTION TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent confidence estimation (ACE) estimates how likely a Large Language Model (LLM) agent is to complete its task and is crucial for reliable agent deployment (*e.g.*, web agents placing orders or clinical agents assisting diagnosis, *etc.*). However, most existing ACE methods remain difficult to deploy in practice due to three limitations: (1) *Insufficient evidence*. They derive confidence from token probabilities or self-reported certainty, which reflect how confident the model is in generating its text rather than whether its actions achieved the intended effect in the environment; (2) *Biased information*. They impose excessive hand-crafted inductive biases, resulting in the loss of raw agent information; and (3) *Late estimation*. They conduct estimation only after execution ends, offering limited support for early prediction. To address these limitations, we propose *STEP* (Stepwise Trajectory Encoding for Prediction), a framework that encodes the task, observation, and action text of each step to ground confidence in agent–environment interactions, applies temporal modeling to these raw step representations instead of relying on hand-crafted features, and employs multi-prefix supervision to support early prediction. Extensive experiments on ScienceWorld, WebShop, and InterCode Bash demonstrate that under within-task evaluation, STEP improves AUROC over the strongest baseline for each prediction setting by 19.2% on end-of-trajectory prediction and 11.6% on the more challenging early prediction.[1]

1 INTRODUCTION

Agent confidence estimation (ACE) estimates how likely a Large Language Model (LLM) agent (Yao et al., 2023; Liu et al., 2024) is to complete its task and is crucial for reliable agent deployment. ACE can provide early warnings during execution or final predictions upon completion, prompting timely human intervention (*e.g.*, users confirming before web agents place an order, clinicians taking over diagnostic decision-making in clinical prediction, *etc.*) to help prevent serious harm (Figure 1a).

Despite early exploration, most existing ACE methods remain difficult to deploy in practice due to three limitations (Figure 1b): (i) *Insufficient evidence.* They derive confidence from token probabilities or self-reported certainty (Kadavath et al., 2022; Tian et al., 2023; Zhang et al., 2026). These signals reflect how confident the model is in generating its text, not whether its actions achieved the intended effect in the environment, resulting in a lack of grounding in agent–environment interactions. (ii) *Biased information.* To summarize a long execution, they compress it into a small set of hand-crafted statistics (*e.g.*, aggregates of token probabilities, step counts, and positions) (Zhang et al., 2026). Such features impose strong inductive biases and discard much of the raw information in the execution, including both implicit information in the generated text beyond its token statistics and explicit information such as the agent's observations and actions. (iii) *Late estimation.* They are designed and evaluated for end-of-trajectory prediction and provide no estimate during execution, where early prediction is harder but more critical for timely intervention. Methods that do predict early rely on hand-crafted behavioral and bag-of-words features and, when available, gold reference solutions (Shi et al., 2026).

Beyond these methodological limitations, evaluation can conflate interaction-grounded prediction with reliance on shortcut cues (Geirhos et al., 2020; Gururangan et al., 2018; McCoy et al., 2019).

[1] Code: `https://anonymous.4open.science/r/step-anonymous-submission-CD75`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

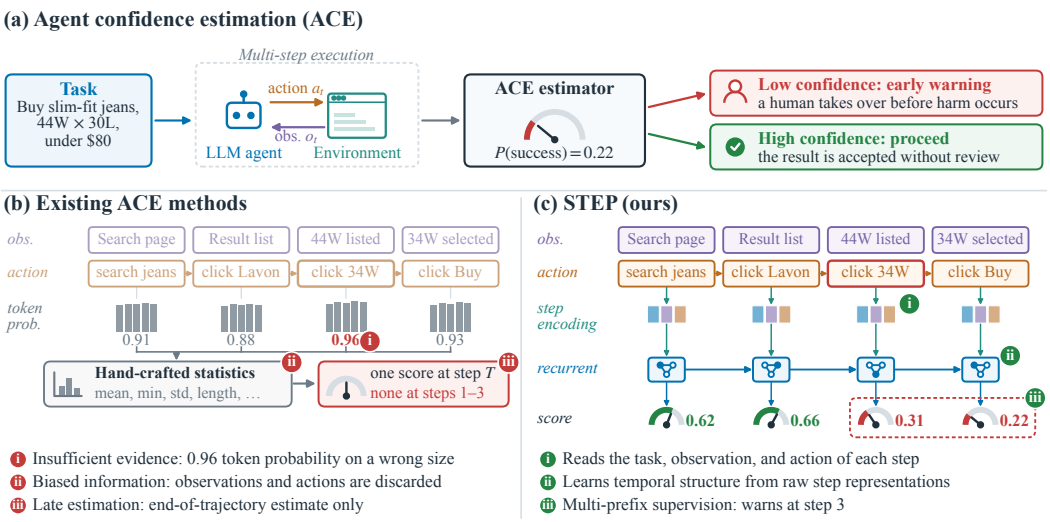

Figure 1: **ACE and STEP at a glance.** (a) ACE estimates the probability that an execution succeeds, so that a low estimate can trigger early human intervention. (b) Limitations of most existing ACE methods on an illustrative WebShop-style execution: (i) token probabilities remain high on a wrong choice, (ii) hand-crafted statistics discard the observations and actions, and (iii) a single score is produced only after the execution ends. (c) STEP's corresponding designs: it encodes the task, observation, and action of each step, applies temporal modeling to these step representations, and uses multi-prefix supervision to score every step. All panels are schematic, and the scores shown are illustrative.

Cross-task AUROC pools attempts from different tasks (Zhang et al., 2026), potentially rewarding predictors for ranking tasks by difficulty even when they cannot distinguish outcomes within a task (*e.g.*, a task-description-only predictor outperforms Holistic Trajectory Calibration (HTC; Zhang et al., 2026) on ScienceWorld; Section 4.4). Trajectory length can provide an additional shortcut when failed attempts exhaust the step budget while successful ones finish early. These issues motivate within-task evaluation with explicit length controls.

To address these limitations and support practical deployment, we propose *STEP* (Stepwise Trajectory Encoding for Prediction; Figure 1c). First, to ground confidence in agent–environment interactions, STEP encodes the task, observation, and action text of each step with a frozen text encoder, so that every action is read together with the goal it should serve and the environment state in which it was taken. Second, instead of relying on hand-crafted features, STEP models these raw step representations temporally with a recurrent encoder, which updates its state as each new step arrives. Third, STEP employs *multi-prefix supervision*: it trains a single predictor on complete trajectories and their prefixes, labeling each prefix with the attempt's final outcome, so the same model can estimate success at any point during execution. Because STEP reads only execution text and learns from final outcomes, it applies to agents whose interfaces do not expose token probabilities and requires no step-level annotation.

We evaluate STEP on ScienceWorld, WebShop, and InterCode Bash against length-based, probability-based, and prompted-LLM baselines. Under within-task evaluation, STEP improves AUROC over the strongest baseline for each prediction setting by 19.2% on end-of-trajectory prediction and 11.6% on early prediction (Figure 3). It also achieves the highest point estimates among length-matched attempts and across three additional acting models, including one that exposes no token probabilities.

Our contributions can be summarized as follows:

- We identify three design limitations in existing ACE methods that hinder reliable deployment: insufficient grounding in agent–environment interactions, information loss from hand-crafted trajectory summaries, and inadequate modeling of early-stage trajectories.
- We show that cross-task AUROC can reward predictors that exploit task difficulty without distinguishing outcomes within a task, and identify trajectory length as an additional potential confound in ACE evaluation.

- For ACE method design, we propose *STEP*, which combines interaction-grounded temporal modeling with multi-prefix supervision for early and end-of-trajectory confidence estimation; for ACE evaluation, we develop a within-task evaluation protocol with explicit length controls to reduce confounding from task difficulty and trajectory length.
- Across three benchmarks, STEP improves within-task AUROC over the strongest baseline in each prediction setting by 19.2% for end-of-trajectory prediction and 11.6% for early prediction. STEP also achieves the highest point estimates in length-matched comparisons and across three additional acting models.

2 Related Work

Confidence estimation in language models. Confidence estimation seeks to identify when a model's output is likely to be correct, while calibration concerns the agreement between predicted probabilities and empirical correctness (Guo et al., 2017). Predicted probabilities have long been used to detect classifier errors (Hendrycks & Gimpel, 2017); for language models, common sources of evidence include generation probabilities, self-assessment, and verbalized confidence (Jiang et al., 2021; Kadavath et al., 2022; Lin et al., 2022; Tian et al., 2023; Xiong et al., 2024). Learned calibrators predict when a model's answer is likely to be wrong (Kamath et al., 2020). Sampling-based methods instead assess consistency across responses, using factual agreement (Manakul et al., 2023) or uncertainty over semantically distinct answers (Kuhn et al., 2023; Farquhar et al., 2024). These approaches connect confidence to properties of generated responses. In an interactive execution, however, the environment's feedback also provides evidence about whether an action served the task. STEP instead encodes the task, observation, and action text of a single execution, rather than relying on generation probabilities or agreement across sampled answers.

Predicting outcomes from agent executions. Trajectory-level prediction extends confidence estimation to the sequence of decisions and feedback produced by an agent. Agentic Confidence Calibration (Zhang et al., 2026) proposes HTC, which summarizes completed trajectories with 48 statistics of token log-probabilities covering cross-step dynamics, intra-step stability, positional indicators, and structural attributes. EarlyEval (Shi et al., 2026) trains gradient-boosted classifiers on behavioral counts, TF-IDF text features, and, when available, overlap with gold solutions to stop agent evaluation runs early. Together, these studies motivate assessing an attempt through its history rather than its final response alone. STEP instead learns from the raw text of what the agent observed and which actions it took, without hand-crafted features. HTC is also evaluated with trajectory-level cross-validation that pools attempts across tasks, where task difficulty can act as a shortcut (Geirhos et al., 2020; Gururangan et al., 2018); we instead compare attempts within each task and use length-based controls.

Outcome verification and process supervision. Learned verifiers provide another route to assessing generated solutions. Outcome supervision trains a verifier using final correctness labels (Cobbe et al., 2021), whereas process supervision provides judgments about intermediate reasoning steps (Lightman et al., 2024). Agent process reward models further study intermediate feedback for evaluating interactions and improving policies, deriving step-level targets from Monte Carlo rollouts (Choudhury, 2025), and goal-conditioned representations have been learned for language reward models (Nath et al., 2024). STEP shares the use of final outcome supervision but takes an agent's observation–action history as its input. Through multi-prefix supervision, the same predictor assesses completed attempts and estimates eventual success during execution. Its intermediate predictions therefore concern the outcome of the attempt, not the correctness of an individual step, and require no step-level annotations.

3 STEP: Stepwise Trajectory Encoding for Prediction

STEP estimates whether an agent will accomplish its task from the available execution history. During execution, the agent generates tokens to produce an action, and the environment returns an observation for the next interaction. STEP encodes each step from its task, observation, and action text, then updates a recurrent state as each new step arrives and reads out a success estimate from that state (Figure 2).

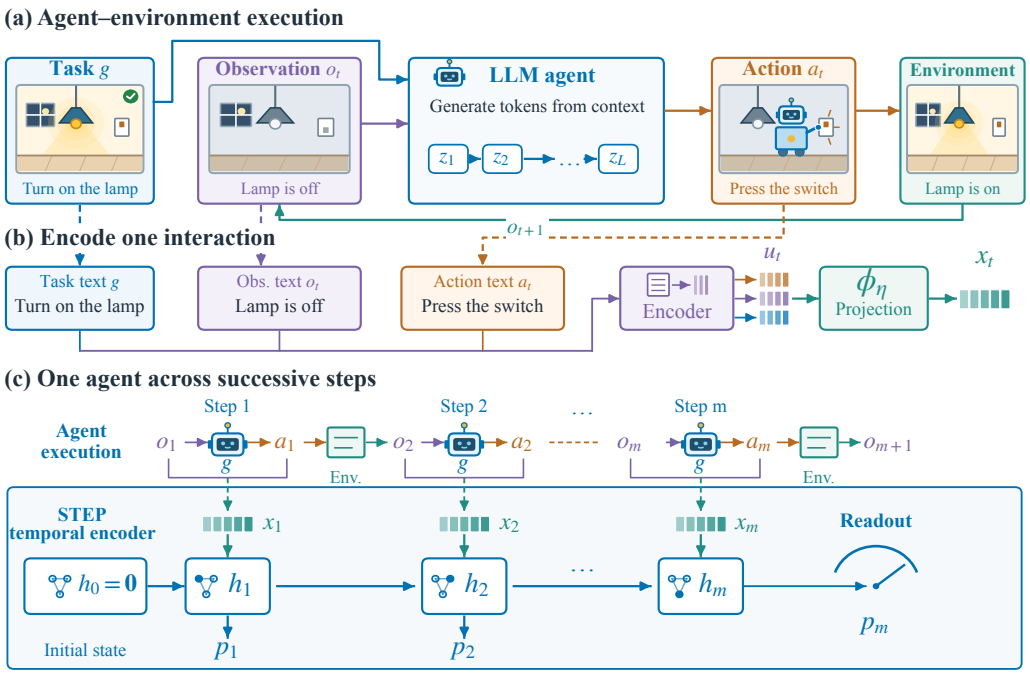

Figure 2: **Overview of STEP.** (a) At each step, the agent generates tokens from the task and the current observation to produce an action, and the environment returns the next observation. (b) Semantic step encoding: the task, observation, and action texts are embedded separately by a frozen text encoder, concatenated into u_t, and projected by a learned ϕ_η into the step representation x_t. (c) A recurrent encoder with shared weights updates its state from $h_0 = \mathbf{0}$ as each step arrives and outputs a success estimate at every step. Multi-prefix supervision applies the attempt's final outcome y to the complete trajectory and its prefixes, so a single predictor serves every observation budget. The example is schematic.

Let i index a task with description g_i, and j index an attempt at that task. The attempt has $T_{i,j}$ actions and outcome $y_{i,j} \in \{0, 1\}$. At step t, $o_t^{(i,j)}$ denotes the observation available before the agent takes action $a_t^{(i,j)}$; thus $o_1^{(i,j)}$ is the initial observation. The history after m recorded actions is

$$H_m^{(i,j)} = \left((g_i, o_t^{(i,j)}, a_t^{(i,j)})\right)_{t=1}^{m}. \tag{1}$$

We learn a predictor f_θ whose output $p_m^{(i,j)} = f_\theta(H_m^{(i,j)})$ estimates eventual success from this history. Complete records use $m = T_{i,j}$; partial records use $m < T_{i,j}$. We omit the attempt indices below when describing a single execution.

3.1 Semantic step encoding

The three fields in an execution record serve different roles. The task specifies the desired outcome, the observation describes the situation in which a decision is made, and the action specifies the agent's response. STEP combines these fields at each step so that an action can be interpreted relative to both the request and the available environmental evidence.

A shared frozen text encoder E (Reimers & Gurevych, 2019) maps each field to a normalized embedding in $\mathbb{R}^d$. We concatenate the embeddings and learn a joint step representation:

$$u_t = [E(g); E(o_t); E(a_t)] \in \mathbb{R}^{3d}, \qquad x_t = \phi_\eta(u_t) \in \mathbb{R}^{d_s}. \tag{2}$$

Here $[\,;\,;\,]$ denotes concatenation and ϕ_η is a learned nonlinear projection. Independent field embeddings preserve the distinction between a task requirement, an observed condition, and a selected action. Their fixed positions in u_t let the projection assign different weights to the three sources and learn combinations relevant to the outcome. Thus, x_t represents an action together with the task it should serve and the observation under which it was taken.

The frozen encoder provides reusable text features, while the projection adapts them to success assessment. The task embedding is shared across all records of an attempt; observation and action embeddings are computed as new records become available. This separation permits caching the text features and training the prediction layers without updating the language encoder.

3.2 Temporal encoding of execution history

A single interaction provides only local evidence. An observation may reveal the consequence of a preceding action, and an action may repeat or revise an earlier decision. STEP therefore applies a temporal encoder F_ψ to the ordered step representations rather than scoring each record independently:

$$h_m = F_\psi(x_1, \ldots, x_m), \qquad p_m = \sigma(w^\top h_m + b). \tag{3}$$

The history representation $h_m \in \mathbb{R}^{d_h}$ depends only on the first m records, and σ is the sigmoid function. In the recurrent realization used here, the encoder computes this representation through successive updates,

$$h_t = U_\psi(h_{t-1}, x_t), \qquad h_0 = \mathbf{0}, \tag{4}$$

where U_ψ combines the accumulated history with the current interaction. Figure 2(c) aligns these updates with the agent's execution steps. A new observation can change how the predictor assesses preceding behavior, while the current action supplies evidence about the agent's next decision.

The same update and readout parameters are used throughout an attempt. At deployment, STEP can retain h_{t-1} and incorporate x_t as the next record arrives, producing an updated success estimate without re-encoding earlier records. For a completed attempt, the readout uses h_T; for an ongoing attempt, it uses the latest available state. Section 4.1 specifies the temporal encoder and projection used in our experiments.

3.3 Multi-prefix supervision

Complete and partial histories expose different amounts of evidence about the same outcome. Training only on complete trajectories does not directly supervise the intermediate predictions used during execution. We therefore train the shared predictor on both complete records and partial histories, using the eventual outcome of each attempt as the target. At a partial history, this target concerns whether the attempt will ultimately succeed, rather than whether its latest action is locally correct.

Let $\mathcal{D}_{\text{train}}$ contain training attempts indexed by (i, j), and let $\mathcal{K}$ be a set of observation budgets. For budget K, the number of available records is $m_{i,j}(K) = \min(K, T_{i,j})$. With binary cross-entropy (BCE) $\ell(y, p) = -y \log p - (1 - y) \log(1 - p)$, we optimize

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(i,j)\in\mathcal{D}_{\text{train}}} \left[\lambda \ell\Big(y_{i,j}, p^{(i,j)}_{T_{i,j}}\Big) + \frac{1-\lambda}{|\mathcal{K}|} \sum_{K\in\mathcal{K}} \ell\Big(y_{i,j}, p^{(i,j)}_{m_{i,j}(K)}\Big) \right]. \tag{5}$$

Here $\theta = (\eta, \psi, w, b)$ contains the trainable parameters and λ balances complete-record and partial-history supervision. The first term trains assessment of completed attempts; the second directly trains the predictions made with limited execution evidence. Averaging over budgets gives each attempt equal total weight.

All budget-specific losses update the same projection, temporal encoder, and readout. They can be evaluated at the corresponding states of a single forward pass through each training trajectory, with gradients propagated through the shared prediction layers and the text encoder kept frozen. This supplies supervision at multiple execution lengths using one outcome label per attempt. At inference, one fitted predictor serves every observation budget, including the complete trajectory.

4 Experiments

4.1 Experimental Setup

Datasets. ScienceWorld involves scientific procedures (Wang et al., 2022), WebShop requires purchasing products that satisfy user requests (Yao et al., 2022), and InterCode Bash requires

executing shell commands to fulfill instructions (Yang et al., 2023). Our primary collections use Qwen3.7-Flash with a fixed ReAct-style prompt (Yao et al., 2023); repeated attempts restart the same task instance. We evaluate tasks with both successful and failed attempts: 100 tasks and 500 attempts each for ScienceWorld and WebShop, and 59 tasks and 195 attempts for InterCode Bash. All attempts remain in the evaluation at every observation budget. Appendix B specifies collection, success criteria, and sample selection. Length-based prediction measures how informative simple length cues are in each collection.

Baselines. (i) The *length-based predictor* uses visible step count and the sum, mean, and standard deviation of generated-token counts per step. The *mean token probability predictor* averages chosen-token probabilities over the visible generations. *HTC* (Zhang et al., 2026) uses 48 probability and token-count features from the agent's generated text. All three use regularized logistic regression; Appendix C.5 adds nonlinear variants. (ii) *Direct Prompt* uses Qwen3-8B (Yang et al., 2025) to judge success from the same task and visible execution text as STEP, without task-specific fitting (Appendix C.6). A *task-only* predictor applies a one-hidden-layer MLP to the frozen embedding of the task description, using the same text encoder and projection width as STEP, and is trained to predict each task's success rate. Because it assigns the same score to every attempt at a task, its within-task AUROC is 0.5 by construction; we report this value in the main tables and use the trained predictor to analyze cross-task evaluation (Section 4.4).

Evaluation. (i) *Data splits and observation budgets.* Five task-grouped outer folds hold out every attempt at each test task from training and model selection. All fitted predictors share these folds. Full evaluates all $T_{i,j}$ records; Prefix@K uses the first $\min(K, T_{i,j})$ for $K = 1, \ldots, 10$, retaining short attempts and fixed sample counts. Records follow Equation 1, excluding the response to the last included action (Appendix B), because this response can reveal the outcome directly (Appendix D). InterCode's ten-action limit makes Prefix@10 identical to Full. (ii) *Metrics.* Our primary metric is within-task AUROC: standard AUROC computed at each task and averaged equally over tasks. We report Full as the primary assessment, curves over the ten budgets, and their arithmetic average as mean prefix AUROC; we refer to Full as end-of-trajectory prediction and to mean prefix AUROC as early prediction. At Full, Brier score, expected calibration error (ECE), and negative log-likelihood (NLL) additionally assess probability quality (Gneiting & Raftery, 2007; Guo et al., 2017). STEP results average five training seeds; Appendix C.7 reports probability-quality results, pointwise 95% bootstrap intervals, and paired differences.

Implementation details. STEP uses frozen 384-dimensional `all-MiniLM-L6-v2` embeddings (Wang et al., 2020), a 16-dimensional step projection, and a GRU (Cho et al., 2014) with hidden size 16. The predictor has 22,401 trainable parameters excluding the frozen encoder (overhead in Appendix C.8). Training uses task-complete batches and Equation 5. Validation selects λ and one checkpoint per fold and seed, shared across all budgets. The logistic baselines use the same multi-prefix objective, data splits, and validation criterion, with features computed from the visible records. No predictor is refitted on validation tasks. Appendix C gives preprocessing, architecture, and fitting details; supplementary experiments specify their settings alongside their results.

4.2 Main Results

4.2.1 Full-trajectory prediction

STEP reaches full-trajectory AUROC of 0.720, 0.789, and 0.680 in ScienceWorld, WebShop, and InterCode Bash (Figure 3(b)), the highest among all methods in every environment. Its gains over length-based prediction are 0.182, 0.199, and 0.082, respectively, with paired 95% intervals excluding zero in ScienceWorld and WebShop, supporting discrimination beyond these length summaries.

Relative to HTC, STEP gains 0.139, 0.123, and 0.091 in the three environments, with positive paired intervals in ScienceWorld and WebShop. Averaged over the three environments, STEP's AUROC is 19.2% higher than that of HTC, the strongest end-of-trajectory baseline (0.730 versus 0.612).

STEP also exceeds Direct Prompt by 0.085, 0.187, and 0.192, respectively, with positive paired intervals in all three environments. Appendix C.6 gives the prompting protocol.

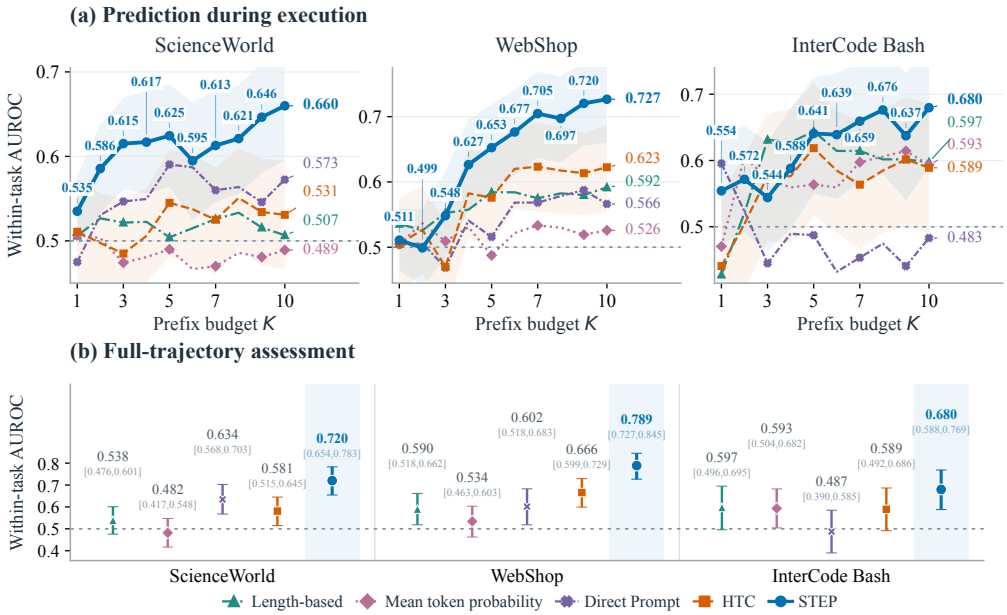

Figure 3: **Prediction from partial and complete executions.** (a) Within-task AUROC after the first $\min(K, T)$ records; blue labels give STEP's values. Shading shows pointwise 95% bootstrap intervals for STEP and HTC. Vertical scales differ across environments. (b) Full-trajectory within-task AUROC with 95% bootstrap intervals, also printed beside each estimate. Dashed lines mark chance. Each learned predictor is shared across budgets. In InterCode Bash, $K = 10$ and Full coincide.

4.2.2 Prediction during execution

Figure 3(a) traces prediction from partial to complete execution histories. STEP's mean prefix AUROC is 0.611, 0.636, and 0.619 across the three environments, exceeding all baselines. Averaged over the three environments, its mean prefix AUROC is 11.6% higher than that of length-based prediction, the strongest early-prediction baseline (0.622 versus 0.558). Paired gains over length-based prediction exclude zero in ScienceWorld and WebShop, as does the gain over HTC in ScienceWorld.

Performance need not rise with every record: at Prefix@3, length-based prediction remains competitive in WebShop (0.553 versus 0.548) and exceeds STEP in InterCode (0.631 versus 0.544).

4.2.3 Additional acting models

Table 1 evaluates the same training recipe with separately fitted predictors on three additional ScienceWorld agent collections. STEP achieves the highest AUROC for all three acting models, exceeding length-based prediction by 0.096, 0.094, and 0.033 for GPT-4.1, DeepSeek-V4-Flash, and Gemini-3.5-Flash-Lite. STEP remains applicable to Gemini's text-only records, for which probability-based baselines are unavailable. Appendix F details the collection and evaluation protocol.

Table 1: **Additional acting models in ScienceWorld.** Full within-task AUROC ↑; each collection contains 100 tasks and 500 attempts, with predictors fitted separately. STEP averages five seeds. Dashes indicate unavailable token probabilities.

Predictor	GPT-4.1	DeepSeek-V4-Flash	Gemini-3.5-Flash-Lite
Length-based	0.538	0.553	0.593
Mean token probability	0.579	0.554	–
HTC	0.592	0.642	–
STEP	**0.634**	**0.647**	**0.625**

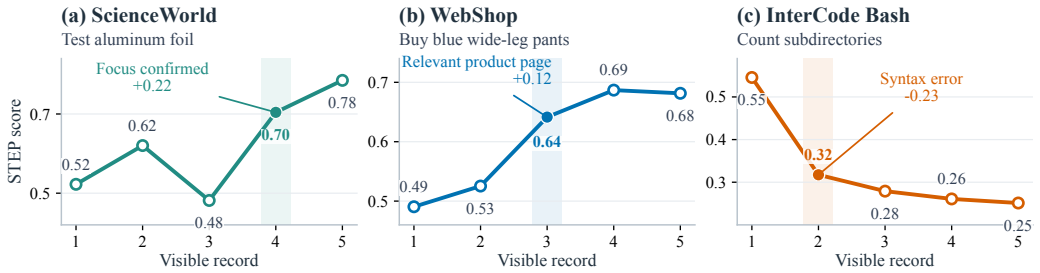

Figure 4: **Selected STEP score updates across three environments.** Curves show five-seed mean held-out scores from the predictors used in the main comparison. Shaded columns mark an observed event; in-panel annotations give its score change from the preceding record. Each panel shows five visible records, with a separate vertical scale.

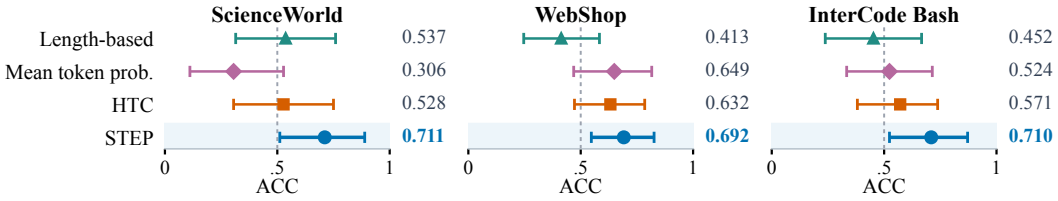

Figure 5: **Length-matched discrimination.** Task-averaged accuracy (ACC) on same-task success–failure pairs with equal step counts and at most 10% token-count difference. Mean token prob. denotes the mean token probability predictor. Bars show 95% bootstrap intervals; the dashed line marks chance. Pair counts and matching details appear in Appendix G.

4.3 Ablation Study

Training objective. Table 4 in Appendix A compares Full-only BCE with multi-prefix supervision under the same architecture, task batches, optimizer, and validation criterion. In WebShop, mean prefix AUROC rises from 0.616 to 0.636 while Full performance stays similar. In ScienceWorld, the prefix change is small (0.609 to 0.611), while Full decreases from 0.738 to 0.720. Appendix A also compares budget-specific, fixed-K, and prefix-trained regimes.

Temporal aggregation. We replace recurrent history aggregation with a simple average of record representations (mean pooling) or a learned weighted average (attention pooling). Under the same multi-prefix supervision, mean pooling raises ScienceWorld's mean prefix AUROC from 0.611 to 0.643, while STEP has higher Full AUROC (0.720 versus 0.691). Attention pooling reaches prefix means of 0.618 and 0.635 in ScienceWorld and WebShop, comparable to STEP (0.611 and 0.636). The results support modeling execution content without establishing recurrence as its sole source of predictive value; recurrence additionally allows STEP to update its estimate online as each step arrives. Appendix E.1 reports the implementation and paired intervals; Appendix E examines content and order controls.

Text encoder. Replacing MiniLM with frozen BGE-small, E5-base, or Qwen3-Embedding-0.6B preserves competitive performance, with no replacement consistently improving both partial- and full-trajectory prediction across environments. ScienceWorld Full AUROC remains between 0.719 and 0.722; E5 improves mean prefix AUROC in WebShop but reduces Full AUROC. These results extend the evaluation beyond a single text encoder and support retaining MiniLM as a lightweight default. Appendix E.2 reports the complete comparison and curves.

4.4 Deep Analysis

Execution behavior. Figure 4 shows three illustrative attempts. STEP's score rises when focus on the target foil is confirmed in ScienceWorld and when a relevant product page appears in WebShop. In a failed InterCode Bash attempt, an explicit shell syntax error accompanies a sharp decrease, and the score remains low as the error recurs. These examples illustrate local responses, not a general progress or error detector. Appendix E.4 gives the annotation procedure and event-level summaries.

Table 2: **Task information in cross-task evaluation.** Full AUROC on 200 fixed ScienceWorld tasks (81 with mixed outcomes), from task-disjoint out-of-fold predictions. Cross-task AUROC pools attempts from all tasks; within-task AUROC averages per-task AUROCs. Mixed-task columns re-evaluate the same predictions. Task-only applies a small MLP to the frozen task-description embedding.

Predictor	Cross-task		Within-task
	All tasks	Mixed tasks	Mixed tasks
Task-only	0.816	0.569	0.500
Length-based	0.497	0.424	0.420
Mean token probability	0.646	0.546	0.534
HTC	0.696	0.635	0.645
STEP	0.871	0.702	**0.711**

Task information in cross-task evaluation. Why evaluate within tasks? We collect a separate ScienceWorld cohort of 200 fixed tasks with five attempts each, without selecting tasks by outcome; 81 tasks have mixed outcomes. Table 2 shows that a predictor reading only the task description reaches 0.816 cross-task AUROC, above HTC (0.696) and the other existing baselines, yet is at chance within each task. Cross-task AUROC can therefore reward recognizing task difficulty rather than assessing the execution. STEP remains the strongest predictor under within-task evaluation (0.711). Its own cross-task AUROC falls from 0.871 on all tasks to 0.702 on the mixed tasks, close to its within-task value, so its advantage there also largely reflects task composition. The size of this effect depends on how task difficulty varies: in a WebShop cohort built the same way, task-only reaches 0.556 (Appendix H).

Discrimination beyond length. Does STEP still distinguish outcomes when successful and failed attempts have similar lengths? We pair attempts within each task, requiring equal step counts and at most 10% relative difference in generated-token counts. We then measure how often each full-trajectory predictor ranks the successful attempt higher. STEP reaches ranking accuracies of 0.711, 0.692, and 0.710 (Figure 5), the highest among all predictors in every environment. Its paired gains over length-based prediction exclude zero in WebShop and InterCode Bash, supporting discrimination beyond these matched length cues. STEP also retains a higher point estimate than length-based prediction at looser 20% and 30% tolerances in every environment. Appendix G gives the matching protocol, paired intervals, and tolerance sensitivity.

5 Conclusion

We studied agent confidence estimation and showed that most existing methods derive confidence from generation signals rather than the execution, and that cross-task evaluation can reward recognizing task difficulty rather than assessing the execution. STEP encodes the task, observation, and action text of each step, applies temporal modeling to these step representations, and uses multi-prefix supervision to estimate success at any point during execution, without token probabilities or step-level annotation. Under within-task evaluation with length controls, it achieves the highest AUROC among length-based, probability-based, and prompted-LLM baselines in all three environments for both end-of-trajectory and early prediction.

Limitations

Our study focuses on text-based agent environments with binary task outcomes. Real-world agents may operate over multimodal observations, receive graded or delayed feedback, and face tasks whose success criteria are only partially specified; extending ACE to these settings remains an open problem. In addition, like other learned estimators, STEP relies on outcome-labeled executions from the target setting, and how to adapt such estimators efficiently as environments and agents evolve is an important direction for future work.

REPRODUCIBILITY STATEMENT

Sections 3–4 specify prediction time, sample selection, metrics, splits, and predictors. Appendix C provides preprocessing, hyperparameters, split seeds, and fitting procedures; the remaining appendices report sampling and auxiliary analyses. Collection manifests, fold assignments, and held-out predictions were retained for result verification. The reported numerical results and figures were checked against these saved outputs.

AI USE STATEMENT

Generative AI assisted the authors in verifying code accuracy and refining the language during the manuscript writing process; it also aided in revising the paper. AI involvement was strictly supplementary, and the authors double-checked every step. The finalization of all presented content was carried out by the authors themselves.

REFERENCES

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025. URL https://arxiv.org/abs/2502.10325.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/ARXIV.2110.14168. URL https://arxiv.org/abs/2110.14168.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, June 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07421-0. URL http://dx.doi.org/10.1038/s41586-024-07421-0.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, November 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-00257-z. URL http://dx.doi.org/10.1038/s42256-020-00257-z.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 107–112. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-2017. URL https://aclanthology.org/N18-2017/.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hkg4TI9xl.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. doi: 10.1162/tacl_a_00407. URL https://aclanthology.org/2021.tacl-1.57/.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. doi: 10.48550/ARXIV.2207.05221. URL https://arxiv.org/abs/2207.05221.

Amita Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5684–5696. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.503. URL https://aclanthology.org/2020.acl-main.503/.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=8s8K2UZGTZ.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3428–3448. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1334. URL https://aclanthology.org/P19-1334/.

Vaskar Nath, Dylan Slack, Jeff Da, Yuntao Ma, Hugh Zhang, Spencer Whitehead, and Sean Hendryx. Learning goal-conditioned representations for language reward models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d46f127a80dc58cbc0732a717285c43a-Abstract-Conference.html.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Yuling Shi, Zhensu Sun, Junsen Dong, Chengcheng Wan, David Lo, and Xiaodong Gu. EarlyEval: Cheaper agent evaluation via early outcome prediction. *arXiv preprint arXiv:2609.02783*, 2026. doi: 10.48550/ARXIV.2609.02783. URL https://arxiv.org/abs/2609.02783.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjeQKFxFpZ.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 23826–23854. Curran Associates, Inc., 2023. doi: 10.52202/075280-1035. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract-Datasets_and_Benchmarks.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.15778.

A Training Alternatives

Table 3 and Figure 6 compare full-trajectory, fixed-K=10, prefix-trained, and budget-specific training. The matched loss ablation in Table 4 holds task-complete batches and the common validation criterion fixed. The full-trajectory regime instead uses minibatches of attempts and Full validation loss. In the prefix-trained regime, STEP samples prefixes and HTC enumerates budgets 1–10.

Table 3: **Training regimes.** STEP within-task AUROC ↑ on all three cohorts, averaged over five seeds. Budget-specific models do not share a predictor across prefixes. Dashes denote unavailable full-trajectory evaluations. Full-trajectory training uses attempt minibatches and selects by Full validation BCE. Matched objective controls are reported separately in Table 4. In InterCode Bash, the ten-step cap makes fixed-K=10 and full-trajectory training identical.

Training regime	Prefix mean	Prefix@3	Prefix@10	Full
ScienceWorld				
Budget-specific	0.610	0.665	0.654	—
Full-trajectory	0.600	0.590	0.669	0.735
Fixed $K = 10$	0.610	0.619	0.654	—
Prefix-trained	0.604	0.616	0.634	—
WebShop				
Budget-specific	0.637	0.627	0.704	—
Full-trajectory	0.619	0.531	0.710	0.784
Fixed $K = 10$	0.631	0.561	0.704	—
Prefix-trained	0.625	0.589	0.686	—
InterCode Bash				
Budget-specific	0.617	0.539	0.670	—
Full-trajectory	0.589	0.507	0.670	0.670
Fixed $K = 10$	0.589	0.507	0.670	0.670
Prefix-trained	0.627	0.547	0.695	0.695

Table 4: **Training-objective controls.** Within-task AUROC ↑ with the same architecture, task batches, and validation criterion. All rows use one model across budgets; hyperparameter search sizes differ.

Training objective	Prefix mean	Prefix@3	Full
ScienceWorld			
Full-only BCE	0.609	0.581	0.738
Full + multi-prefix BCE	0.611	0.615	0.720
WebShop			
Full-only BCE	0.616	0.448	0.785
Full + multi-prefix BCE	0.636	0.548	0.789
InterCode Bash			
Full-only BCE	0.594	0.535	0.669
Full + multi-prefix BCE	0.619	0.544	0.680

Tables 3 and 4 cover all three environments. Multi-prefix supervision changes mean prefix AUROC from 0.609 to 0.611 in ScienceWorld, 0.616 to 0.636 in WebShop, and 0.594 to 0.619 in InterCode Bash. Table 4 also reports the associated Full scores. Because InterCode Bash attempts contain at most ten action records, fixed-K=10 and full-trajectory training use identical inputs and share a checkpoint under the same recipe. Figure 6 visualizes the ScienceWorld and WebShop training-regime comparisons.

B Task Collection and Prefix Construction

Rollouts use the DashScope API model identifier `qwen3.7-flash`, a fixed ReAct-style prompt, temperature 0.7, top-p 0.9, a 768-token generation limit per turn, and reasoning mode disabled. Action limits are 100, 50, and 10 for ScienceWorld, WebShop, and InterCode Bash, respectively.

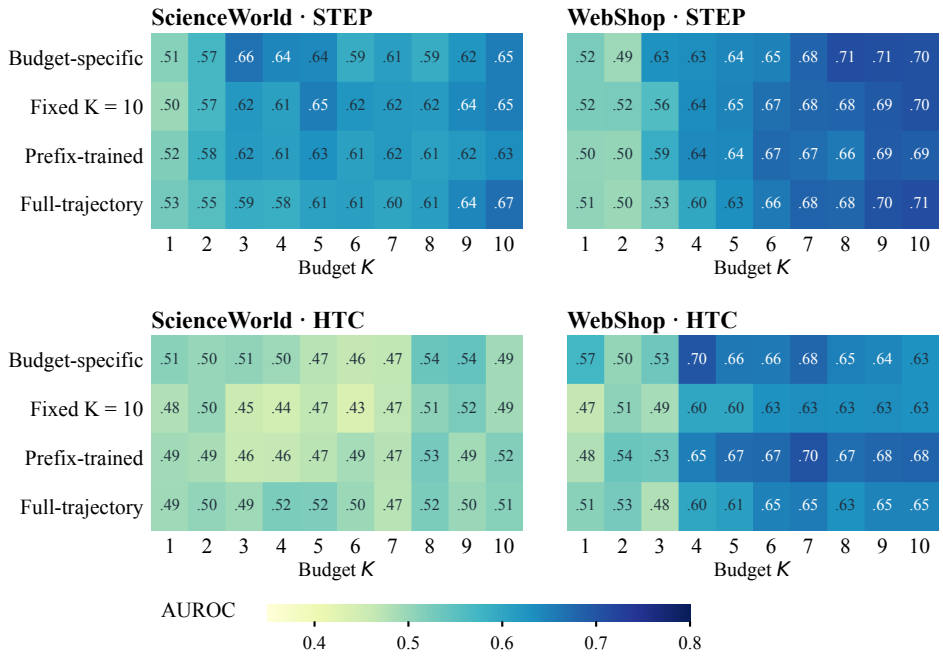

Figure 6: **STEP and HTC under different training protocols.** Heatmaps report within-task AUROC at budgets 1–10 using a common color scale and the same task cohorts and outer folds. Budget-specific training fits one predictor per position; other regimes share a predictor across positions. Cells show point estimates, averaged over five seeds for STEP. Table 3 also reports full-trajectory scores.

Success requires a ScienceWorld score of 100, WebShop terminal reward 1 within tolerance 10^{-8}, or InterCode submit reward 1.

ScienceWorld and WebShop each contain 100 tasks with five attempts per task, including both successful and unsuccessful attempts. Task selection combines filtering collected rollouts for mixed outcomes with adaptive screening: candidate tasks are sampled without replacement under a fixed random seed, two attempts are collected, and tasks with differing outcomes are completed to five attempts. All attempts use the same agent prompts and sampling parameters. The resulting samples target within-task outcome discrimination rather than benchmark-wide success rates.

InterCode Bash contributes 59 tasks with mixed observed outcomes and 195 attempts (101 successful and 94 failed), selected from a 200-task, 900-attempt collection. Fifty selected tasks have three attempts and nine have five. Collection is adaptive, so these counts do not estimate benchmark-wide prevalence. Each attempt is capped at ten model-generated actions. If the agent has not submitted by the cap, the environment's standard submit operation obtains the outcome label; that framework-generated operation is not an action record. The predictor uses the same observation-before-action boundary as in the other environments. All 1,342 selected action records have aligned token-probability traces.

B.1 Exact prefix construction

For each attempt, we first form the ordered records (g, o_t, a_t) and then take the first $\min(K, T)$ records. We do not append the response to the last included action. For example, a two-action attempt at Prefix@3 supplies (g, o_1, a_1) and (g, o_2, a_2), not the feedback following a_2. The same attempt supplies these two records at larger budgets. Padding only batches these variable-length inputs and is excluded by packed-sequence computation. Restricted-input models remove a text field without changing this record selection.

The input includes no explicit total-length or termination field. Nevertheless, retaining short records exposes their visible length. Excluding the last response does not remove this cue or guarantee the absence of outcome-revealing language in earlier text or actions. The protocol measures prediction

over a fixed set of attempts using the available prefix; it does not assert that all of them are still executing. Full follows the same field exclusions but removes the budget cap.

C Predictor Implementation and Fitting

C.1 Text representation and recurrent predictor

Each task description, observation, and action is independently tokenized to at most 256 tokens. A frozen `all-MiniLM-L6-v2` encoder produces token states. We average non-padding states and normalize the result to unit Euclidean length. We cache the frozen encoder's normalized embeddings across folds and budgets.

With all inputs present, each action record has dimension $3 \times 384 = 1152$. Layer normalization has 2,304 trainable parameters, the 1152-to-16 projection has 18,448, the GRU has 1,632, and the output layer has 17: 22,401 in total. The projection uses GELU and dropout 0.1. Packed sequences exclude padding from the recurrent computation. The initial-state model uses two text fields and sequence length one; restricted-input models also change the input projection size. The comparisons thus hold the modeling recipe, but not the exact parameter count, fixed.

For the primary model, one checkpoint supplies all prefix and Full predictions. The common validation score is $0.5\mathcal{L}_F + 0.5|\mathcal{K}|^{-1}\sum_K \mathcal{L}_K$, independent of the candidate training λ. AdamW uses learning rate 10^{-3}, weight decay 10^{-4}, and gradient clipping at 5. Training lasts at most 100 epochs, with patience 20. Each task-complete batch contains all attempts from up to three tasks, at most 15 attempts. Repeated states from short attempts retain their weights in Equation 5. Budget-specific controls use batch size 16 and select checkpoints by their budget's validation BCE. Neither neural protocol refits on validation tasks.

C.2 Data splits and training seeds

The primary datasets use five outer stratified, task-grouped folds with split seed 42. For outer fold f, the first of four stratified, task-grouped inner folds, with seed $4200 + f$, supplies validation tasks. This allocates approximately 60%, 20%, and 20% of tasks to training, validation, and testing. The stored fold assignments are reused across compared recurrent predictors.

Reported training seeds are 42–46; within a fold, the optimization seed is the reported seed plus $100f$. Every trajectory receives a held-out score at every budget, including short attempts. Training and evaluation both use the prefix rule in Section 4.1.

C.3 Uncertainty and score checks

For recurrent results, task-by-seed arrays of within-task AUROC are resampled along both axes.

Main comparisons use 10,000 bootstrap draws; budget-specific analyses use 20,000 draws. Bootstrap intervals are pointwise, without adjustment across budgets.

C.4 Length and token-probability baselines

HTC computes 48 statistics from chosen-token, top-1, and top-k probability traces over the agent's complete generated text at each visible step. The features include probability gradients, distribution shape, first–last changes, and token-count summaries. For prefix prediction, features are recomputed using only generations from the available steps.

For generated-token counts $n_1, \ldots, n_m$ at the visible steps, Length-based uses

$$\left(m,\ \sum_{t=1}^{m} n_t,\ \frac{1}{m}\sum_{t=1}^{m} n_t,\ \sqrt{\frac{1}{m}\sum_{t=1}^{m}(n_t - \bar{n})^2} \right).$$

Counts come from the generated-token arrays without their probability values. The mean token probability predictor uses the arithmetic mean of chosen-token probabilities across the visible

generations. Its logistic coefficient can have either sign: this is a learned one-feature predictor, not necessarily a monotone calibrator of the raw token-probability average.

Each linear baseline uses a training-only scaler, assigning weight 0.05 to each prefix and 0.5 to Full. Scaling is fixed across candidates. The fitting objective is Equation 5 plus $\alpha\|\theta\|_2^2/(2N_{\text{train}})$, including the intercept in the penalty. L-BFGS-B uses at most 2,000 iterations; convergence is checked. Validation selects $\lambda \in \{0.25, 0.5, 0.75\}$ and

$$\alpha \in \{0.001, 0.01, 0.1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50\}.$$

There are 45 candidates per baseline and fold and three λ candidates per STEP seed and fold. All use the common validation BCE without refitting on validation tasks. For the alternative regimes in Figure 6, HTC uses four-fold inner cross-validation followed by refitting on the outer training pool.

C.5 Nonlinear probability and length predictors

The main comparison changes both the input representation and the predictor family: the summary baselines are linear, whereas STEP is nonlinear. We therefore strengthen the summary baselines with a small MLP on the same 48 HTC features, four length features, or their 52-dimensional concatenation. Features use the complete generated text at each visible step, as in Appendix C.4; no execution-text embeddings are supplied. Each network is $D \to 16 \to 1$ with GELU and no dropout, giving 801, 97, or 865 trainable parameters, respectively. One network shares weights across Full and all prefixes, without a budget-ID input.

Fitting and selection. We reuse the main cohorts, stored training/validation/test task splits, five seeds, and $\min(K, T)$ prefix rule. Scaling uses training tasks only, with weight 0.5 for Full and 0.05 for each prefix. We optimize the multi-prefix BCE objective with $\lambda \in \{0.25, 0.5, 0.75\}$ and an L2 penalty $\alpha\|\theta\|_2^2/(2N_{\text{train}})$ on all weights and biases, using the same 15-value α grid as the linear baselines. Adam uses learning rate 10^{-3}, batches of three complete tasks, gradient clipping at 5, at most 100 epochs, and patience 20 with minimum improvement 10^{-6}. Early stopping and candidate selection both use $0.5\mathcal{L}_F + 0.5|\mathcal{K}|^{-1}\sum_K \mathcal{L}_K$, with no validation refit. Each input family has 45 candidates per fold and seed. Summary-MLP additionally selects among the three input families by this same validation score, not test AUROC. This gives the summary predictors a larger search budget than STEP's three candidates per fold and seed; it is not a compute-matched comparison.

Table 5: **Nonlinear probability and length baselines.** Within-task AUROC ↑ under the main task splits and validation protocol. Prefix mean averages $K = 1, \ldots, 10$. Neural methods average five seeds. Summary-MLP selects the input family by validation BCE within each fold and seed. Bold marks the highest point estimate, not significance.

	ScienceWorld		WebShop		InterCode Bash	
Method	Full	Prefix mean	Full	Prefix mean	Full	Prefix mean
HTC-linear	0.581	0.522	0.666	0.576	0.589	0.564
HTC-MLP	0.593	0.533	0.685	0.591	0.568	0.545
Length-linear	0.538	0.518	0.590	0.567	0.597	0.588
Length-MLP	0.560	0.536	0.591	0.566	0.614	0.573
HTC-Length-MLP	0.620	0.541	0.687	0.589	0.577	0.541
Summary-MLP	0.586	0.529	0.691	0.591	0.576	0.540
STEP	**0.720**	**0.611**	**0.789**	**0.636**	**0.680**	**0.619**

Results. Table 5 compares the MLPs with the main STEP predictor, not the Full-only specialists. Paired differences support STEP's Full advantage over all four nonlinear comparisons in ScienceWorld and WebShop, and its prefix-mean advantage in ScienceWorld. Against the concatenated HTC–length MLP, the Full gains are 0.100 [0.011, 0.187] and 0.101 [0.020, 0.185], respectively. ScienceWorld's prefix-mean gain is 0.070 [0.003, 0.136]. WebShop's prefix-mean comparisons with HTC-MLP, the concatenated MLP, and Summary-MLP remain uncertain, as do all Full and prefix-mean comparisons in InterCode Bash. WebShop exceeds Length-MLP on the prefix mean, but this alone does not establish an advantage over probability summaries.

These results support predictive value in execution text beyond the tested nonlinear summary predictors, rather than attributing the main Full gains solely to a linear readout restriction. They do not isolate a semantic or temporal mechanism, nor rule out stronger summary predictors. The architecture and learning rate were fixed; 32 of 225 selected MLPs have their best epoch at 95–100, and 14 select the largest α, so the search is not an exhaustive optimization study. Intervals condition on the existing cohorts and folds and do not account for repeated exploratory use of these data.

C.6 Direct Prompt baseline

Direct Prompt uses Qwen3-8B in non-thinking mode with 4-bit NF4 inference. The prompt supplies the task followed by the visible sequence of observation-before-action and executed-action text. It excludes action responses beyond the boundary, rewards, outcome labels, termination metadata, and token probabilities.

Table 6: **Direct Prompt baseline.** Within-task AUROC ↑ using the same action-aligned execution text as STEP. Direct Prompt uses Qwen3-8B without task-specific training. STEP values average five training seeds.

	Prefix@3		Full	
Dataset	Direct Prompt	STEP	Direct Prompt	STEP
ScienceWorld	0.547	**0.615**	0.634	**0.720**
WebShop	0.469	**0.548**	0.602	**0.789**
InterCode Bash	0.445	**0.544**	0.487	**0.680**

The model scores the next-token alternatives `Success` and `Failure`. Both labels are single tokens under its tokenizer. We reverse their display order in a second prompt and average success-versus-failure log odds before computing AUROC. No examples, task-specific training, validation selection, or probability calibration are used.

Full records are supplied without truncation. The longest prompts contain 5,440, 30,563, and 4,543 tokens in ScienceWorld, WebShop, and InterCode Bash, respectively. Peak allocated GPU memory is 10.5, 22.9, and 10.4 GiB. This evaluator assesses direct prompting with an 8B-parameter model, while STEP learns outcome prediction from the training tasks using a frozen sentence encoder and a compact prediction head.

At Full, STEP exceeds the prompted evaluator by 0.085 (95% CI: 0.002–0.165), 0.187 (0.089–0.285), and 0.192 (0.045–0.336) in ScienceWorld, WebShop, and InterCode Bash. The Prefix@3 point-estimate gains are 0.069, 0.079, and 0.099, respectively, but none has a paired interval excluding zero. These results treat direct prompting as a high-compute reference, not as a formal upper bound.

C.7 Evaluation metrics, uncertainty, and probability scores

Within-task discrimination. For task i with R_i attempts, the evaluated task set is

$$\mathcal{I} = \left\{ i : 0 < \sum_{j=1}^{R_i} y_{i,j} < R_i \right\}. \tag{6}$$

Let P_i and N_i contain successful and failed attempts, respectively, and let $s_{i,j}$ be a predictor score at a fixed observation budget. Standard AUROC at task i is

$$\mathrm{AUC}_{\mathrm{task}}(i; s) = \frac{\sum_{u \in P_i} \sum_{v \in N_i} [\mathbb{1}(s_{i,u} > s_{i,v}) + \frac{1}{2}\mathbb{1}(s_{i,u} = s_{i,v})]}{|P_i||N_i|}. \tag{7}$$

We report $|\mathcal{I}|^{-1} \sum_{i \in \mathcal{I}} \mathrm{AUC}_{\mathrm{task}}(i; s)$. Task-only scores tie within every task, yielding 0.5; cross-task AUROC need not share this property (Appendix H). All attempts remain in the comparison at every budget. The metric describes the selected mixed-outcome tasks, and the length-based control measures the contribution of the specified length features rather than ruling out all structural cues.

Uncertainty estimates. We use 10,000 bootstrap draws to obtain pointwise 95% percentile intervals. STEP estimates resample tasks and training seeds independently; deterministic baselines resample tasks. Paired differences use the same task and seed draws, pairing each STEP seed with the deterministic baseline scores. These intervals condition on the collected attempts and fixed folds, excluding uncertainty from adaptive collection, new executions, and fold selection. They are not multiplicity-adjusted bands.

Full-trajectory probability quality. We compute Brier score $\mathbb{E}[(p-y)^2]$, NLL $\mathbb{E}[-y\log p - (1-y)\log(1-p)]$, and ECE on held-out Full-trajectory predictions. ECE divides predicted success probabilities into ten equal-width bins and sums each bin's fraction of attempts times the absolute gap between its mean predicted probability and observed success rate (Guo et al., 2017). Lower is better for each metric, but ECE alone can favor a weak predictor that stays near the cohort success rate; we interpret it alongside Brier and NLL.

Table 7: **Success prediction with a single model across budgets.** Within-task AUROC ↑, equally averaged over 100 ScienceWorld tasks, 100 WebShop tasks, or 59 InterCode Bash tasks; STEP additionally averages five seeds. Prefix mean averages budgets 1–10. Full uses the same fitted model on the complete execution record. In InterCode Bash, Prefix@10 equals Full. Bold marks the highest point estimate, not statistical significance.

Method	Prefix mean	Prefix@3	Prefix@10	Full
ScienceWorld				
Task-only	0.500	0.500	0.500	0.500
Length-based	0.518	0.522	0.507	0.538
Mean token probability	0.485	0.474	0.489	0.482
HTC	0.522	0.485	0.531	0.581
STEP	**0.611**	**0.615**	**0.660**	**0.720**
WebShop				
Task-only	0.500	0.500	0.500	0.500
Length-based	0.567	**0.553**	0.592	0.590
Mean token probability	0.520	0.509	0.526	0.534
HTC	0.576	0.470	0.623	0.666
STEP	**0.636**	0.548	**0.727**	**0.789**
InterCode Bash				
Task-only	0.500	0.500	0.500	0.500
Length-based	0.588	**0.631**	0.597	0.597
Mean token probability	0.574	0.572	0.593	0.593
HTC	0.564	0.576	0.589	0.589
STEP	**0.619**	0.544	**0.680**	**0.680**

AUROC averages task-level rankings. Brier, ECE, and natural-log NLL are computed across attempts, separately for each STEP seed and then averaged across seeds. ScienceWorld and WebShop have five attempts per task; InterCode Bash has unequal attempt counts, so its probability scores are attempt-weighted whereas its AUROC remains task-averaged. The scores describe the selected mixed-outcome cohorts, not calibration under the benchmarks' original success prevalences. STEP has lower Full Brier score than HTC in ScienceWorld and InterCode Bash (0.214 versus 0.240, and 0.224 versus 0.240), while HTC is lower in WebShop (0.234 versus 0.240). STEP also has higher ECE than HTC in WebShop (0.065 versus 0.028). Better within-task ranking therefore does not imply uniformly better probability forecasts.

C.8 Prediction overhead relative to agent execution

We compare the cost of updating each predictor after every action on the 195 InterCode Bash attempts. All methods use their trained held-out-fold models; STEP uses seed 42. We replay each attempt on a single CPU thread (ARM64 macOS 27.0, PyTorch 2.12.0), with models already loaded. STEP includes task encoding once per attempt, observation and action encoding at every step, and incremental GRU updates. The baselines include probability-signal extraction, feature computation, normalization, and prediction. HTC caches each step's generated probability signals and recomputes its 48 statistics for each prefix. No method receives precomputed text embeddings or prefix features.

Table 8: **Full-trajectory discrimination and probability quality.** AUROC is within-task; Brier, ten-bin ECE, and NLL are attempt-weighted. STEP entries average the metric across five seeds. All predictions are out-of-fold on the selected mixed-outcome cohorts.

Method	AUROC ↑	Brier ↓	ECE ↓	NLL ↓
ScienceWorld				
Length-based	0.538	0.248	**0.018**	0.690
Mean token probability	0.482	0.252	0.027	0.698
HTC	0.581	0.240	0.072	0.675
STEP	**0.720**	**0.214**	0.062	**0.613**
WebShop				
Length-based	0.590	0.252	0.042	0.720
Mean token probability	0.534	0.249	0.029	0.691
HTC	0.666	**0.234**	**0.028**	**0.659**
STEP	**0.789**	0.240	0.065	0.675
InterCode Bash				
Length-based	0.597	0.240	**0.037**	0.672
Mean token probability	0.593	0.253	0.073	0.699
HTC	0.589	0.240	0.063	0.673
STEP	**0.680**	**0.224**	0.070	**0.639**

Table 9: **Within-task AUROC intervals.** On 100 ScienceWorld tasks, 100 WebShop tasks, and 59 InterCode Bash tasks. Brackets give pointwise 95% bootstrap intervals, resampling tasks and seeds for STEP and tasks for deterministic baselines. Prefix mean averages budgets 1–10; Full uses the same fitted model. In InterCode Bash, Prefix@10 equals Full.

Method	Prefix mean [95% CI]	Full [95% CI]
ScienceWorld		
Length-based	0.518 [0.469, 0.568]	0.538 [0.476, 0.601]
Mean token probability	0.485 [0.430, 0.540]	0.482 [0.417, 0.548]
HTC	0.522 [0.478, 0.567]	0.581 [0.515, 0.645]
STEP	0.611 [0.567, 0.656]	0.720 [0.654, 0.783]
WebShop		
Length-based	0.567 [0.506, 0.629]	0.590 [0.518, 0.662]
Mean token probability	0.520 [0.459, 0.579]	0.534 [0.463, 0.603]
HTC	0.576 [0.523, 0.627]	0.666 [0.599, 0.729]
STEP	0.636 [0.588, 0.684]	0.789 [0.727, 0.845]
InterCode Bash		
Length-based	0.588 [0.504, 0.666]	0.597 [0.496, 0.695]
Mean token probability	0.574 [0.499, 0.647]	0.593 [0.504, 0.682]
HTC	0.564 [0.486, 0.641]	0.589 [0.492, 0.686]
STEP	0.619 [0.555, 0.680]	0.680 [0.588, 0.769]

We exclude disk I/O and model loading, warm up each predictor, and take the median of three timed replays per attempt.

For attempt r, let C_r be the cumulative prediction time and R_r its recorded rollout duration, including agent model calls and environment actions but excluding environment reset. We report $100C_r/R_r$ as the estimated additional cost relative to execution. This pairs local replay timings with recorded rollout durations, rather than measuring an integrated deployment. The rollout median is 7.85 s. Although STEP costs more than probability-based predictors, its median cumulative cost is only 36.07 ms, including the encoder. The median per-attempt overhead is 0.431%, and the ratio of total prediction time to total rollout time is 0.458% (Table 10).

C.9 Execution input ablation

We compare predictors using actions, observations, or both, each with the task description. All three variants use the main task cohorts, folds, five seeds, and multi-prefix supervision with the

Table 10: **Prediction overhead on InterCode Bash.** Cumulative time to update predictions throughout each of 195 attempts, including STEP's text encoder. Relative cost is computed against each attempt's recorded rollout duration. P95 denotes the 95th percentile across attempts.

	Prediction time (ms)		Relative cost (%)	
Method	**Median**	**P95**	**Median**	**P95**
Length-based	0.146	0.200	0.0018	0.0037
Mean token probability	0.041	0.062	0.0005	0.0009
HTC	3.232	5.292	0.0303	0.0553
STEP	36.066	95.759	0.4306	1.2889

Table 11: **Execution input ablation under the main training protocol.** Every variant includes the task description. One model trained with multi-prefix supervision serves all budgets; values are mean within-task AUROCs over five seeds. Prefix mean averages budgets 1–10. Removing a field changes the projection size.

Execution text	**Prefix mean**	**Prefix@3**	**Full**
ScienceWorld			
Actions	0.608	0.611	0.716
Observations	0.597	0.585	0.692
Actions + observations (STEP)	0.611	0.615	0.720
WebShop			
Actions	0.626	0.544	0.770
Observations	0.609	0.546	0.727
Actions + observations (STEP)	0.636	0.548	0.789
InterCode Bash			
Actions	0.560	0.510	0.585
Observations	0.589	0.522	0.651
Actions + observations (STEP)	0.619	0.544	0.680

same validation criterion and λ candidates. A single selected model supplies every prefix and Full prediction. Removing a field reduces the input projection size but leaves the frozen encoder, projection width, and GRU width unchanged. Observations are always taken before the associated action; none of these variants receives the response to its last included action. Table 11 reports both partial- and full-trajectory prediction rather than a separately trained Prefix@3 model.

At Full, actions alone reach 0.716, 0.770, and 0.585 in ScienceWorld, WebShop, and InterCode Bash; observations alone reach 0.692, 0.727, and 0.651. Using both gives the highest point estimate in each environment (0.720, 0.789, and 0.680). The paired advantage over observations alone is supported in WebShop (0.062; 95% CI: 0.001–0.123); the other Full differences include zero. The input roles therefore carry overlapping predictive information, with no uniform evidence that both are necessary.

D Input Availability and Post-Execution Signals

Table 12: **Input availability and outcome recognition.** Within-task AUROC on the same 100 ScienceWorld, 100 WebShop, and 59 InterCode Bash tasks as the main evaluation. Neural results average per-seed AUROCs over five seeds. Final observations, termination reasons, and recorded scores are diagnostic post-execution inputs, not STEP inputs.

Available information	**ScienceWorld**	**WebShop**	**InterCode Bash**
Task + initial observation	0.498	0.500	0.500
Full record, excluding last feedback	0.720	0.789	0.680
Step count + termination reason	1.000	0.739	1.000
Final observation, score text removed	0.695	0.811	0.668
Final observation, original text	0.695	0.947	0.668
Recorded final score	1.000	1.000	1.000

Table 12 compares input availability on the same task populations and folds as the main evaluation. The full-record row is the primary STEP model evaluated at Full, excluding the last action's response. The initial-state row uses the task and first observation. The two final-observation rows use the task and the response to the last model-generated action, either unchanged or with explicit score/reward lines removed. These text controls use the same encoder, projection/GRU widths, optimizer, task batches, validation BCE, and five seeds, with one input record per attempt. Since their input is static, they are trained once per fold and seed without a prefix-weight search.

The step-count-plus-termination predictor uses total action count and a one-hot termination reason, with scaling and category vocabulary fitted on training tasks only. Its regularization is selected by validation BCE over the same 15 candidates as the main linear baselines. The recorded-score row directly ranks attempts by the logged final environment score or reward; it is not a learned predictor. All rows use within-task AUROC, with neural AUROCs computed separately for each seed before averaging.

The post-execution rows are diagnostic controls, not deployable early predictors. Removing explicit score text does not remove every outcome-revealing phrase. In InterCode Bash, any framework-generated forced-submit response is excluded from the final-observation text; its resulting reward is included only in the recorded-score row.

The initial-state predictors remain near chance (0.498, 0.500, and 0.500), whereas step count plus termination reason reaches 1.000, 0.739, and 1.000. In WebShop, removing explicit score text lowers terminal-observation AUROC from 0.947 to 0.811; the same removal rule changes no terminal text in ScienceWorld or InterCode Bash, so their raw and cleaned rows coincide. The recorded final score separates outcomes perfectly in all three cohorts. These contrasts show why completed-log outcome recognition must be distinguished from prediction using the action-aligned execution record.

E Representation Controls and Behavior Diagnostics

The aggregation, text-encoder, and content controls cover ScienceWorld, WebShop, and InterCode Bash, using the main task cohorts and outer folds. The qualitative behavior analysis uses examples from all three environments.

E.1 Aggregation across observation budgets

We replace recurrent accumulation with mean or attention pooling over the projected visible records. Both controls pass the pooled vector through one zero-initialized GRU update and the shared output architecture. Attention assigns each record a learned linear score and normalizes weights only over visible, non-padding records. It adds 16 parameters to the 22,401-parameter architecture; mean pooling preserves the parameter count. Neither pooling control uses positional encoding or a recurrent-history bypass. All methods use the same frozen embeddings, five seeds, task folds, multi-prefix supervision, validation criterion, and $\lambda \in \{0.25, 0.5, 0.75\}$ selection.

Prediction from partial executions. Mean prefix AUROC for STEP, mean pooling, and attention pooling is 0.611/0.643/0.618 in ScienceWorld, 0.636/0.623/0.635 in WebShop, and 0.619/0.621/0.594 in InterCode Bash. Mean pooling improves on STEP in ScienceWorld by 0.031 (paired 95% CI: 0.009–0.056). STEP–attention differences include zero in all three environments, as do STEP–mean-pooling differences in WebShop and InterCode Bash.

Full-trajectory prediction. The corresponding Full AUROCs are 0.720/0.691/0.672, 0.789/0.687/0.708, and 0.680/0.672/0.657. STEP exceeds attention pooling in WebShop by 0.081 [0.030, 0.134], while the corresponding intervals include zero in ScienceWorld and InterCode Bash. Figure 7 reports all paired comparisons. Aggregation affects the prefix–full-trajectory trade-off, without establishing a uniform advantage for recurrence.

E.2 Text encoder sensitivity

We replace the frozen MiniLM encoder with BGE-small-en-v1.5, E5-base-v2, and Qwen3-Embedding-0.6B, producing 384-, 768-, and 1,024-dimensional field embeddings, respectively.

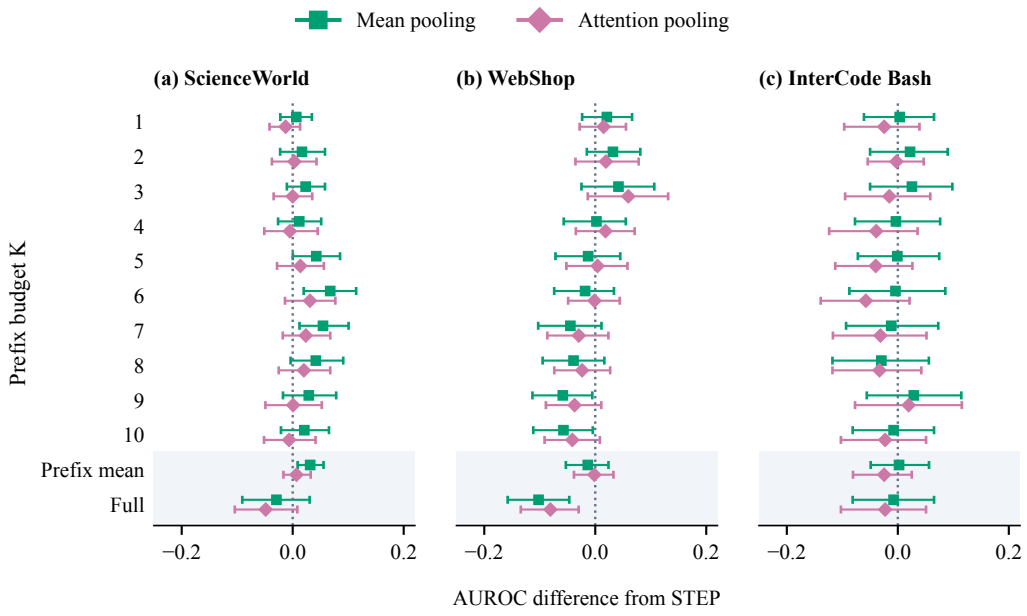

Figure 7: **Aggregation ablation under multi-prefix supervision.** Points show pooling minus STEP within-task AUROC; positive values favor pooling. Bars are paired pointwise 95% task-by-seed bootstrap intervals. Each environment retains its full cohort at every budget: 100 tasks and 500 attempts in ScienceWorld and WebShop; 59 tasks and 195 attempts in InterCode Bash. The shaded rows summarize the prefix mean and Full.

The step projection and GRU widths remain 16. All variants use the original task cohorts, five outer folds, five seeds, multi-prefix supervision, and validation selection protocol. Only the frozen text representation and the input width of the projection change; the larger embeddings therefore increase the projection parameter count.

Each encoder receives the same original-text substring ending at the 254th MiniLM content token, with shorter fields unchanged. This preserves the primary model's visible text rather than giving alternative encoders longer observations. We use each encoder's native pooling: CLS for BGE, masked mean pooling with the `query:` prefix for E5, and the last valid token for Qwen. Qwen embeds fields without an instruction. Embeddings are normalized to unit length and cached in the same precision as the primary model.

Table 13: **Text encoder ablation.** Within-task AUROC averaged over five seeds. Mean denotes the average over prefix budgets $K = 1, \ldots, 10$; Full uses the complete execution record. Bold marks the highest point estimate in each column, not a significant difference.

	ScienceWorld		WebShop		InterCode Bash	
Frozen encoder	Mean	Full	Mean	Full	Mean	Full
MiniLM-L6 (main)	0.6112	0.7202	0.6362	**0.7887**	0.6190	0.6797
BGE-small	0.6098	0.7212	0.6319	0.7635	**0.6258**	0.7000
E5-base	0.6076	**0.7218**	**0.6544**	0.7752	0.6248	**0.7042**
Qwen3-Embedding-0.6B	**0.6119**	0.7192	0.6371	0.7417	0.6106	0.6466

Table 13 and Figure 8 show that no replacement dominates across environments and prediction budgets. E5 raises the point estimates for WebShop mean prefix AUROC by 0.018 and InterCode Bash Full AUROC by 0.025, but neither paired gain is established. MiniLM retains the highest Full AUROC in WebShop, while ScienceWorld Full estimates differ by less than 0.003. Paired intervals for all encoder replacements include zero for both mean prefix and Full AUROC, based on 10,000 task-by-seed bootstrap draws. The comparison supports using multiple frozen encoders within STEP without identifying a consistently better replacement for MiniLM.

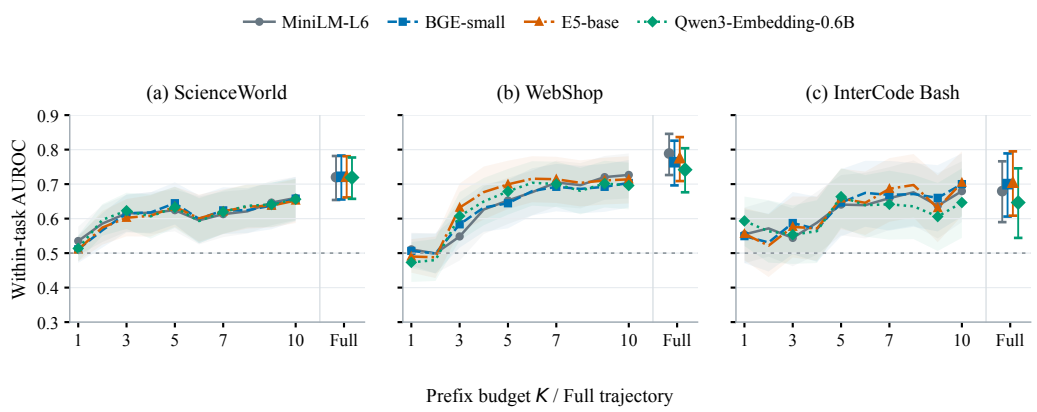

Figure 8: **Frozen text encoder comparison.** Each curve uses one selected STEP predictor per fold and seed at all budgets, retaining the first $\min(K, T)$ records. Separate points show Full predictions. Curves and points average five seeds; shading and bars give pointwise 95% task-by-seed bootstrap intervals. All variants use the same input text and task folds.

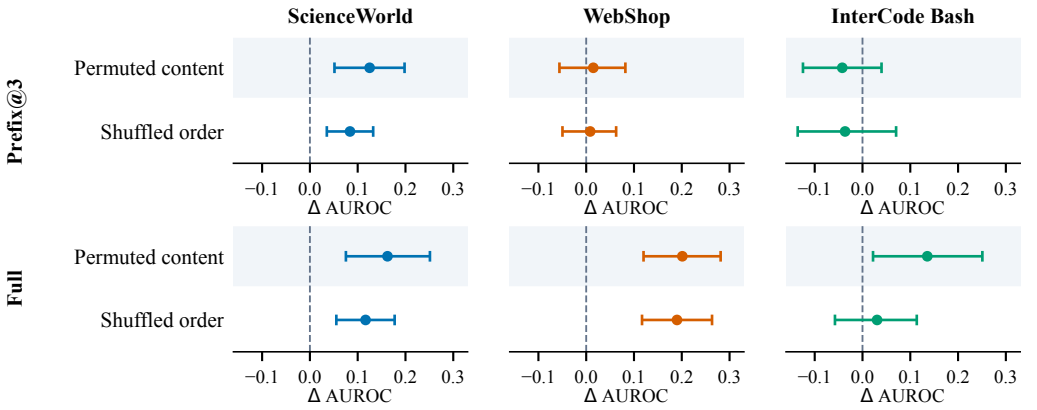

Figure 9: **Content and order controls at matched observation budgets.** Top: Prefix@3-trained specialists. Bottom: Full-trained specialists. Within each panel, STEP and both controls use identical task cohorts, folds, and training budgets. Points show STEP minus control within-task AUROC; bars are pointwise 95% intervals from 20,000 paired task-by-seed bootstrap draws. Positive values favor STEP.

E.3 Content and order controls

We test aligned content and record order at two observation budgets: Prefix@3 and Full. Each panel in Figure 9 compares STEP with two controls trained specifically for that panel's input budget. All variants use five seeds, the same task folds, trainable layers, and parameter count, attempt minibatches of 16, and validation BCE at the corresponding budget. Prefix@3 uses the first $\min(3, T)$ records; Full uses all available records. These matched specialist comparisons are distinct from the multi-prefix supervision used in the main comparison and Appendix E.1.

Content permutation exchanges observation–action embedding pairs within each training, validation, or test split and exact action-token-count buckets. It preserves task embeddings and the four action-length features, but not observation length. Order shuffling permutes the visible records within an attempt. Both controls exclude the last action's response.

Early prefixes. The Prefix@3 specialists reach 0.663, 0.627, and 0.539 in ScienceWorld, WebShop, and InterCode Bash. Aligned content and canonical order help in ScienceWorld, with paired gains

over permutation and shuffling of 0.125 [0.051, 0.198] and 0.084 [0.035, 0.132]. Both comparisons remain uncertain in WebShop and InterCode Bash, so the early-prefix controls provide less consistent evidence than the Full controls below.

Aligned content and order. Full-only STEP reaches AUROC 0.735, 0.784, and 0.670 in ScienceWorld, WebShop, and InterCode Bash, respectively. Content permutation lowers these values to 0.572, 0.583, and 0.535. The paired gains over permutation are 0.162 [0.075, 0.251], 0.201 [0.120, 0.281], and 0.136 [0.022, 0.251], supporting aligned content in all three environments. Order shuffling yields 0.618, 0.594, and 0.640. Its effect is supported in ScienceWorld and WebShop, but the InterCode Bash interval includes zero (Figure 9).

These interventions preserve some lexical and structural cues and do not isolate semantics from all such cues.

E.4 Annotated execution examples

A score-blind, purposive sample contains 12 ScienceWorld and four WebShop attempts, from distinct tasks, with 132 visible records. Rule-nominated error/repetition pools and an unfiltered pool supply candidates; the initial annotations were frozen before prediction scores were inspected. The annotations have been manually verified. They describe newly visible goal-related progress, repeated information, explicit errors, and recovery from an earlier error, not eventual correctness.

Figure 4 includes two cases from this annotated sample and a separate InterCode Bash attempt with an explicit shell error. The plotted curves use the held-out STEP predictors from the main comparison, averaged across five seeds.

Under these predictors, 14 of 21 ScienceWorld progress events and both WebShop progress events increased scores. Responses to errors, recovery, and repeated actions were mixed. A record contains feedback and a new action, so score changes cannot be attributed to feedback alone. These exploratory observations do not establish a progress or recovery detector.

F Additional Acting Models in ScienceWorld

We collect 100 mixed-outcome tasks with five attempts per task separately for GPT-4.1, DeepSeek-V4-Flash, and Gemini-3.5-Flash-Lite. Each collection uses five task-grouped outer folds and an inner validation split. STEP uses seeds 42–46 and the primary multi-prefix supervision recipe; deterministic linear baselines share its task splits and validation objective. Predictors are trained within each acting-model collection. Because task selection depends on observed outcomes, these collections need not contain identical task instances.

GPT-4.1 and DeepSeek provide token probabilities for their generated text. Gemini supplies no usable token log probabilities, so HTC and mean token probability are unavailable. For Gemini's length-based baseline, generated tokens are counted with the frozen MiniLM reference tokenizer; the other collections use provider token arrays. Full-trajectory inputs follow the same observation-before-action boundary as in the primary evaluation.

G Length-Matched Outcome Discrimination

Pair construction. We use the full-trajectory out-of-fold predictions from the main comparison, without retraining or changing task splits. For each task, we enumerate all successful–failed attempt pairs. A pair is retained when its attempts have equal numbers of execution records and relative token-count difference $|n^+ - n^-| / \max(n^+, n^-) \leq \epsilon$, where n^+ and n^- are their generated-token counts. The primary tolerance is $\epsilon = 0.10$; Table 14 also reports the prespecified tolerances 0.20 and 0.30. All methods use identical retained pairs.

Scoring and sample size. Each retained pair receives one point when the successful attempt has the higher predicted score, half a point for a tie, and zero otherwise. We average these values within each task, then equally across tasks with retained pairs and across training seeds. Thus, tasks with more qualifying pairs do not receive more weight. A pair is a comparison, not an independent

trajectory: one attempt may participate in several pairs. At the 10% tolerance, ScienceWorld has 23 pairs involving 42 distinct attempts from 18 tasks; WebShop has 52 pairs involving 77 attempts from 24 tasks; InterCode Bash has 30 pairs involving 51 attempts from 21 tasks. Unlike unrestricted within-task AUROC, this metric only compares length-matched pairs.

Table 14: **Length-matching sensitivity.** Equal step counts are required throughout; the tolerance bounds relative token-count difference. Values are task-averaged pairwise ranking accuracy. Length and Mean prob. denote the length-based and mean token probability predictors.

Dataset	Tol.	Tasks	Pairs	STEP	HTC	Length	Mean prob.
ScienceWorld	10%	18	23	0.711	0.528	0.537	0.306
	20%	22	36	0.677	0.443	0.542	0.409
	30%	25	41	0.676	0.370	0.517	0.460
WebShop	10%	24	52	0.692	0.632	0.413	0.649
	20%	32	83	0.610	0.609	0.341	0.659
	30%	33	92	0.597	0.624	0.366	0.639
InterCode Bash	10%	21	30	0.710	0.571	0.452	0.524
	20%	27	45	0.722	0.593	0.500	0.500
	30%	28	49	0.745	0.601	0.524	0.518

Paired uncertainty. We use 10,000 bootstrap draws over tasks and training seeds, retaining all qualifying pairs of each sampled task together. STEP's gains over length-based prediction are 0.174 [-0.148, 0.481], 0.278 [0.085, 0.483], and 0.257 [0.019, 0.500] in ScienceWorld, WebShop, and InterCode Bash. Its gains over HTC are 0.183 [-0.150, 0.500], 0.060 [-0.175, 0.307], and 0.138 [-0.124, 0.390]. These are exploratory pointwise intervals. Matching constrains step count and total token count, but does not equate every length feature, such as variation in tokens per record.

Tolerance sensitivity. STEP retains a higher point estimate than length-based prediction at all three tolerances in every environment. Its comparison with HTC is less consistent: in WebShop, the difference is 0.060 at 10%, 0.001 at 20%, and -0.026 at 30%. The analysis therefore supports useful discrimination among attempts with similar lengths, rather than uniform superiority over probability-based prediction under every matching rule.

H Task Information in Cross-Task Evaluation

Cohorts. The ScienceWorld cohort contains 200 task instances fixed before outcomes were observed, with five Qwen3.7-Flash attempts each (1,000 attempts). Of these tasks, 95 have five failures, 24 have five successes, and 81 have both outcomes. Unlike the main cohorts, tasks are not selected for mixed outcomes. A WebShop cohort collected in the same way contains 103 all-failure, 47 all-success, and 50 mixed tasks.

Protocol. All predictors use task-disjoint five-fold out-of-fold evaluation with identical outer folds and training/validation partitions. Task-only applies a one-hidden-layer MLP (16 units) to the frozen MiniLM embedding of the task description, sharing STEP's text encoder and projection width but using no temporal encoder; it is trained with each training task's success fraction as a soft target under BCE. Length-based, mean token probability, HTC, and STEP follow the main multi-prefix supervision protocol; only Full predictions are evaluated. Task-only and STEP average five seeds (standard deviations: task-only 0.007 cross-task on all tasks; STEP 0.003, 0.008, and 0.022 across the three columns of Table 2). Direct Prompt was not run on these cohorts. Cross-task AUROC over all 200 ScienceWorld tasks compares 329 successful with 671 failed attempts; only 410 of the 220,759 successful–failed pairs come from the same task (0.19%), so this metric is almost entirely a comparison across tasks. The two mixed-task columns evaluate the same 405 attempts.

Results. In ScienceWorld, task-only reaches 0.816 cross-task AUROC on all tasks but 0.500 within tasks. STEP's cross-task AUROC falls from 0.871 on all tasks to 0.702 on mixed tasks, close to its within-task value of 0.711, so most of its all-task advantage reflects task composition rather than a

difference between metrics on identical attempts. HTC's cross-task AUROC on all tasks exceeds that on mixed tasks by 0.060 (95% task-bootstrap interval [0.009, 0.116]), whereas its all-task cross-task minus mixed within-task difference of 0.051 has an interval that includes zero ($[-0.018, 0.117]$). We therefore do not claim that HTC is inflated by cross-task evaluation. In WebShop, task-only reaches 0.556 cross-task AUROC on all tasks, and HTC scores 0.639 cross-task on all tasks, 0.623 cross-task on mixed tasks, and 0.667 within tasks. The contribution of task information thus depends on how difficulty varies across tasks, which makes cross-task AUROC hard to compare across benchmarks; within-task AUROC removes this dependence.