CB-AGENT: CONCEPT-BASED IN-CONTEXT LEARNING FOR SEQUENTIAL ACTION SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) allows large language models to use demonstrations without updating their parameters. For interactive agents, demonstrations are trajectories of reasoning, actions, and observations that illustrate how to complete a task. Reusing demonstrations in new situations is challenging: directly inputting full trajectories consumes limited prompt space and constrains the overall effectiveness, while natural-language summaries require the agent to translate general guidance into specific action choices. Therefore, we study whether expert action choices can directly supervise a reusable scoring rule. We propose CB-Agent, which extends concept-based ICL to sequential action selection. We extract environment-action decisions from expert trajectories and fit a concept vector by maximum likelihood to model the conditional distribution over candidate actions given the context. This model combines the concept with frozen language-model features and priors. During ReAct interaction, the agent uses this distribution to select environment actions without including demonstrations in the action-scoring context. Across TextWorld, ALFWorld, and ScienceWorld, CB-Agent achieves the highest cross-benchmark average among the evaluated methods at all three tested model sizes and ranks first on TextWorld and ScienceWorld at each size. ALFWorld experiments support the learned concept's contribution beyond prior-only scoring and show further gains when combined with Reflexion. Code is available at `https://anonymous.4open.science/r/CB-Agent-DFF9/`.

1 INTRODUCTION

In-context learning (ICL) enables a large language model (LLM) to perform a task using examples in its input, without updating its parameters (Brown et al., 2020). These examples, called demonstrations, typically pair an input with a desired output. For an interactive agent, demonstrations instead record trajectories of actions and observations. Few-shot ReAct includes these demonstration trajectories directly in the prompt to illustrate how reasoning and actions can be combined as new observations arrive (Yao et al., 2023b). The agent must use this experience throughout execution: each decision depends on the interaction so far and can affect the steps that follow. The goal is therefore to reuse demonstration knowledge across changing contexts to complete a new task.

Existing approaches improve how language models use experience through demonstration selection, task organization, and feedback. Fine-tuning offers another route, but collecting interaction data or human annotations and optimizing model parameters can be costly (Nakano et al., 2021). Within ICL, retrieval-based selection identifies demonstrations relevant to a query input (Liu et al., 2022). For interactive tasks, StateAct maintains explicit goals and state information to guide decisions (Rozanov & Rei, 2025). AdaPlanner refines its plans in response to environmental feedback (Sun et al., 2023). To reuse accumulated experience, ExpeL extracts natural-language insights and combines them with retrieved successful trajectories (Zhao et al., 2024). Reflexion instead maintains verbal reflections on feedback to guide subsequent attempts (Shinn et al., 2023). Demonstrations in a prompt nevertheless compete with interaction history for limited space, while reflection-based retries require additional model calls and environment interactions. For demonstration-based action selection, we investigate an alternative: using expert choices to fit a reusable scoring rule, rather than relying solely on trajectories or verbal guidance in the action-selection prompt.

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.

The central challenge is how experience should affect each action choice. A new task need not reproduce any demonstrated sequence, and applying language-based insights requires interpreting their guidance in the current situation. For example, advice to avoid repeated searches must distinguish an unproductive revisit from returning to a known location to collect a second object. Expert trajectories also provide direct supervision: the action selected in each recorded context. This motivates our central question: *can we estimate a reusable action-selection rule directly from demonstrated choices, without first expressing that experience as natural-language rules?* Fitting these choices would explicitly connect demonstration learning to the comparison of candidate actions.

Numerical task representations provide a foundation for this direction. Task Vectors and Function Vectors show that demonstrated tasks can be represented by vectors that influence a model's predictions (Hendel et al., 2023; Todd et al., 2024). More directly, concept-based ICL (CB-ICL) estimates a shared concept vector over frozen language-model features and provides theoretical analyses of label prediction for individual inputs (Tang et al., 2025). Here, a concept is a numerical vector that weights features for prediction. Adapting this framework to sequential action selection requires extracting decision examples from trajectories and handling contexts and candidates that change with interaction. The concept must express context-dependent preferences, not a fixed ranking of action strings.

We introduce CB-Agent, which converts expert trajectories into decision examples comprising a pre-action context, candidate environment actions, and the expert's choice. We model the conditional distribution over candidate actions given the context using an exponential-family model. A concept vector weights frozen LLM features to adjust the language-model prior and is fitted to the expert choices by regularized maximum likelihood. During ReAct interaction, the fitted model selects the highest-probability candidate environment action. ReAct continues to use demonstrations for reasoning and think-or-act routing, while the action scorer uses the current context without demonstrations. The concept remains fixed during execution, but action probabilities change with the context and candidates. This procedure supports both environment-supplied candidates in fixed-action-space tasks and model-generated candidates in open-action-space tasks.

Our contributions can be summarized as follows:

1. **Concept-based learning from trajectory decisions.** We adapt CB-ICL to estimate a shared concept from expert environment-action choices. The conditional model combines the learned feature weights with a language-model prior, linking demonstration fitting to candidate scoring without updating LLM parameters or requiring verbal experience summaries.

2. **Online selection with a reusable concept.** We integrate the fitted model into ReAct interaction, using one fixed concept with changing contexts and candidates in both fixed and open environment-action spaces.

3. **Empirical evaluation and analysis.** We evaluate Qwen3-4B, Qwen3-8B, and Qwen3-14B on TextWorld, ALFWorld, and ScienceWorld. CB-Agent achieves the highest cross-benchmark average among the evaluated methods at each model size and ranks first on TextWorld and ScienceWorld throughout. An ALFWorld control comparison supports the contribution of the learned concept beyond the length-adjusted language-model prior. We also examine task-related concept structure and evaluate combinations with Reflexion separately.

2 Background and Problem Setup

This section reviews ICL (Section 2.1) and concept-based prediction (Section 2.2), then defines the sequential action-selection problem (Section 2.3).

2.1 In-Context Learning

In standard few-shot ICL, input–output demonstrations accompany a query input in the prompt, guiding its prediction without updating language-model parameters (Brown et al., 2020).

For agents, demonstrations are trajectories of task instructions, actions, and observations. Few-shot ReAct includes reasoning in these trajectories to guide interaction (Yao et al., 2023b). Unlike single-output prediction, interaction requires successive decisions conditioned on the evolving history.

2.2 Concept-Based In-Context Learning

Concept-based in-context learning (CB-ICL) estimates a shared concept vector from labeled demonstrations using features from a frozen language model (Tang et al., 2025). The estimated vector weights the features of an input and its candidate labels, allowing demonstrations to inform a common prediction rule. The published CB-ICL analysis uses a linear representation of conditional probabilities.

In this work, we adopt an exponential parameterization instead of the linear form, motivated by local information geometry. Under weak dependence, where the joint distribution is close to the product of its marginals in chi-squared divergence, the linear modal decomposition admits a local exponential representation, up to higher-order terms (Huang et al., 2024). This representation motivates weighting fixed features in log-probability space. Appendix D gives the derivation and distinguishes this local approximation from the practical concept model.

Let x be an input text, $\mathcal{Y}$ a finite, nonempty set of candidate labels, and $y \in \mathcal{Y}$ a label. Let m be the positive integer feature dimension, $\boldsymbol{\phi}(x, y) \in \mathbb{R}^m$ a fixed feature vector, and $Q_0(y) > 0$ a fixed base weight. A concept vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ defines the conditional distribution $P_{\boldsymbol{\alpha}}$ by multiplying each base weight by a feature-dependent factor and normalizing over labels:

$$P_{\boldsymbol{\alpha}}(y \mid x, \mathcal{Y}) = \frac{Q_0(y) \exp\{\boldsymbol{\alpha}^\top \boldsymbol{\phi}(x, y)\}}{\sum_{b \in \mathcal{Y}} Q_0(b) \exp\{\boldsymbol{\alpha}^\top \boldsymbol{\phi}(x, b)\}}. \tag{1}$$

Here $\top$ denotes transpose, and b indexes candidate labels. The concept is estimated by maximizing the conditional likelihood of the demonstrated labels and is then shared across inputs. While prior work on CB-ICL considers label prediction for individual inputs, we adapt concept-based prediction to sequential action selection for interactive agents (Section 3).

2.3 Sequential Action Selection and Problem Formulation

Let inst denote the task instruction and o_0 the initial observation provided by the environment before any action is taken. If no initial observation is provided, we treat o_0 as empty. At interaction step $t = 0, 1, \ldots$, the action string a_t is either an environment action or a `Think` action containing reasoning text. The recorded feedback o_{t+1} is an environment observation for the former and the fixed response `OK` for the latter, which leaves the environment unchanged. Both action types count as interaction steps. For a positive integer interaction length T, a complete trajectory is

$$\tau = (\mathrm{inst}, o_0, a_0, o_1, \ldots, a_{T-1}, o_T). \tag{2}$$

Let $\mathcal{D}_{\mathrm{traj}}$ denote the collection of expert trajectories, whose lengths may differ.

At the start of a new task, the agent receives the task instruction inst and initial observation o_0. Before selecting action a_t, it constructs a text context x_t from the task instruction, the initial observation, and all preceding actions and feedback. The agent either generates a `Think` action or selects an environment action. At an environment-action step, it selects a_t from a finite, nonempty candidate collection $\mathcal{A}_t$. In both cases, it records o_{t+1} and updates the context. The trajectory is thus generated through interaction rather than predicted in advance.

Given $\mathcal{D}_{\mathrm{traj}}$ and a frozen language model, our objective is to solve new tasks from the same task distribution through this interaction process, without updating the model parameters. Success is determined by whether the interaction achieves the goal specified by inst, rather than by action prediction accuracy alone. Section 3.2 describes how expert trajectories provide the decision examples used to learn the action-selection rule.

3 CB-Agent: Concept Estimation and Action Selection

In this section, we present CB-Agent. Section 3.1 specifies the conditional action model. Section 3.2 constructs decision examples from expert trajectories and defines the concept-estimation objective.

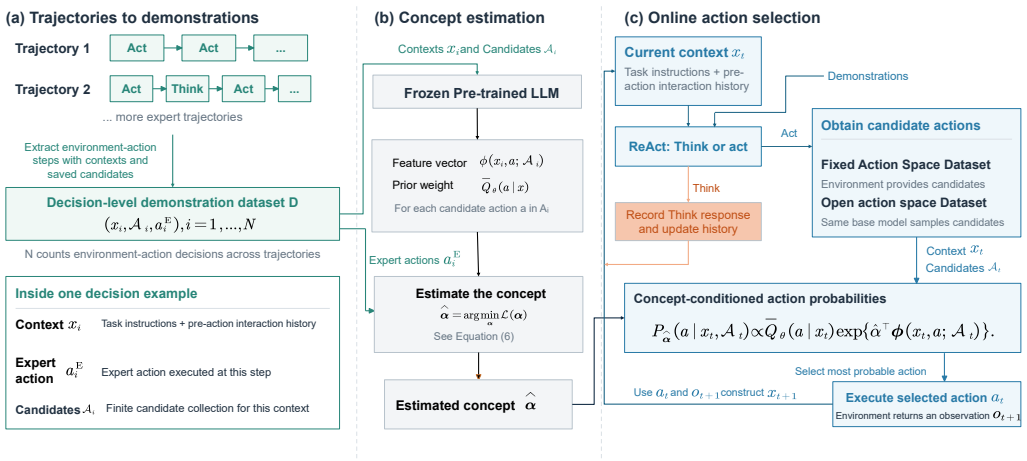


Figure 1: **Overview of CB-Agent.** (a) Only environment-action steps in expert trajectories provide decision examples comprising a pre-action context, candidate actions, and the expert action. Preceding thoughts remain in the context. (b) A frozen language model supplies action features and length-adjusted prior weights; a shared concept is estimated by regularized maximum likelihood. (c) A ReAct response determines whether to think or act. A `Think` response is recorded in the interaction history. On the action branch, the agent constructs candidate actions and selects one using the estimated concept and prior weights. The interaction history is updated after either branch, while the concept and language model remain fixed.

Finally, Section 3.3 describes online action selection with the fitted concept. Algorithm 1 summarizes the complete procedure. Figure 1 illustrates how environment-action steps in expert trajectories provide demonstrations for concept estimation and how the fitted concept guides action selection on the ReAct action branch.

3.1 Concept-Conditioned Action Probabilities

We instantiate Equation (1) with interaction contexts as inputs and candidate environment actions as labels. For one environment-action decision, we omit the step index and write x for the pre-action context, $\mathcal{A}$ for the finite, nonempty candidate collection, and $a \in \mathcal{A}$ for a candidate environment action. Unlike a fixed-label prediction problem, both the candidates and their language-model weights can change across decisions.

Let θ denote the frozen language-model parameters. Retaining the feature dimension m and concept vector $\boldsymbol{\alpha}$ from Section 2.2, let $\boldsymbol{\phi}(x, a; \mathcal{A}) \in \mathbb{R}^m$ be the feature vector of action a. We construct it from the model's last-layer representation at the final action token, with coordinate-wise standardization across candidates. The features therefore depend on the context and candidate collection. Appendix B specifies the construction.

Let $Q_\theta(a \mid x) > 0$ be the model probability of the action's token sequence, $L(a)$ its positive integer token count, and $\gamma \in [0, 1]$ a fixed length-correction exponent. Following the power-law length normalization discussed by Wu et al. (2016), we use the length-adjusted prior weight $\overline{Q}_\theta(a \mid x) = Q_\theta(a \mid x)^{1/L(a)^\gamma}$. Replacing the base weight in Equation (1) with this context-dependent weight gives the action distribution:

$$P_{\boldsymbol{\alpha}}(a \mid x, \mathcal{A}) = \frac{\overline{Q}_\theta(a \mid x) \exp\{\boldsymbol{\alpha}^\top \boldsymbol{\phi}(x, a; \mathcal{A})\}}{\sum_{b \in \mathcal{A}} \overline{Q}_\theta(b \mid x) \exp\{\boldsymbol{\alpha}^\top \boldsymbol{\phi}(x, b; \mathcal{A})\}}. \tag{3}$$

Here b indexes candidate actions. The concept adds the score $\boldsymbol{\alpha}^\top \boldsymbol{\phi}(x, a; \mathcal{A})$ to the prior log-weight. Let $\mathbf{0}$ denote the zero vector in $\mathbb{R}^m$. Setting $\boldsymbol{\alpha} = \mathbf{0}$ recovers the normalized length-adjusted prior. The same model is used for concept estimation and action selection.

For two candidates $a, b \in \mathcal{A}$, their log-probability ratio separates the prior preference from the learned adjustment:

$$\log \frac{P_{\boldsymbol{\alpha}}(a \mid x, \mathcal{A})}{P_{\boldsymbol{\alpha}}(b \mid x, \mathcal{A})} = \log \frac{\overline{Q}_\theta(a \mid x)}{\overline{Q}_\theta(b \mid x)} + \boldsymbol{\alpha}^\top [\boldsymbol{\phi}(x, a; \mathcal{A}) - \boldsymbol{\phi}(x, b; \mathcal{A})]. \tag{4}$$

One concept vector is fitted to the expert trajectories selected for a task and shared across its environment-action decisions, giving a common scoring rule. This does not impose a fixed preference over action strings: their features change with the context and candidates. The learned adjustment can reinforce or reverse the prior ranking.

This exponential family models expert action preferences; its scope and limitations are discussed in Appendix F.

3.2 Concept Estimation from Expert Trajectories

For a new task, let $\mathcal{D}_{\text{traj}}$ contain its selected expert trajectories. We construct one decision example for each environment-action step in these trajectories. At such a step t, the input is the pre-action context x_t defined in Section 2.3, the prediction target is the recorded expert action a_t, and a finite candidate collection $\mathcal{A}_t$ is stored with that step. Each candidate collection must contain the recorded action. The context includes only information available before a_t, excluding the action itself and its subsequent feedback. `Think` steps do not supply fitting targets, but their reasoning text and `OK` feedback remain in the contexts of later decisions. We then pool the examples across trajectories and index them by $i = 1, \ldots, N$, where $N \geq 1$ is the total number of decision examples. Writing a_i^{E} for the expert action in example i, the resulting dataset is

$$\mathcal{D} = \{(x_i, \mathcal{A}_i, a_i^{\text{E}})\}_{i=1}^N. \tag{5}$$

Thus, $\mathcal{D}_{\text{traj}}$ contains complete trajectories, whereas $\mathcal{D}$ contains their environment-action decisions. The index t denotes a position within an original trajectory, including `Think` steps; i enumerates only the pooled fitting examples.

The fixed/open distinction refers to the environment-action space. For fixed-action-space tasks, the environment supplies environment-action candidates. For open-action-space tasks, we construct candidates by sampling multiple replies from the acting model under the same ReAct context and extracting environment-action strings. This produces a finite candidate collection without enumerating the full action space. During expert data collection, the expert selects from the constructed candidates, which are saved with the trajectory for concept estimation. Appendix B provides the candidate-construction settings and preprocessing details.

We estimate the concept by regularized maximum likelihood. Let $\lambda > 0$ be the regularization strength, $\mathcal{L}$ the fitting objective, and $\widehat{\boldsymbol{\alpha}}$ the estimated concept. With $\|\cdot\|_2$ denoting the Euclidean norm, we minimize the average negative log-probability of expert actions plus a squared-norm penalty:

$$\begin{aligned} \widehat{\boldsymbol{\alpha}} &= \arg\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}), \\ \mathcal{L}(\boldsymbol{\alpha}) &= -\frac{1}{N} \sum_{i=1}^N \log P_{\boldsymbol{\alpha}}(a_i^{\text{E}} \mid x_i, \mathcal{A}_i) + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_2^2. \end{aligned} \tag{6}$$

Only the concept vector is optimized. Candidate features and prior weights are computed before fitting and held fixed, along with θ. The fitted concept can be reused across test tasks that use the same expert demonstrations, backbone, and feature construction. Each loss term compares the expert action with its alternatives, and pooling these terms fits one concept across trajectories. Decisions are weighted equally, so trajectories with more environment-action decisions contribute more terms.

With fixed features, positive prior weights, and $\lambda > 0$, the objective in Equation (6) is strongly convex and has a unique finite minimizer, which we approximate numerically. Appendix C proves these optimization properties and states the conditions for interpreting the data term as a trajectory negative log-likelihood.

3.3 Online Action Selection

At each test step, a ReAct response determines whether the agent reasons or takes an environment action. A `Think` response is recorded with fixed feedback `OK`, updating the history for the next step.

Algorithm 1 CB-Agent

Inputs: Expert trajectories $\mathcal{D}_{\mathrm{traj}}$ selected for the task; frozen model θ; regularization $\lambda > 0$; task instruction inst and environment.

(a) Decision-dataset construction

for each environment-action step t in each trajectory of $\mathcal{D}_{\mathrm{traj}}$ **do**
  Reconstruct x_t from the instruction and preceding history, retaining thoughts.
  Retrieve the saved candidates $\mathcal{A}_t$ and expert action $a_t \in \mathcal{A}_t$.
end for
Pool these decisions as $\mathcal{D} = \{(x_i, \mathcal{A}_i, a_i^{\mathrm{E}})\}_{i=1}^{N}$.

(b) Concept estimation

Cache $\boldsymbol{\phi}(x_i, a; \mathcal{A}_i)$ and $\overline{Q}_\theta(a \mid x_i)$ for all i and $a \in \mathcal{A}_i$.
Fit $\widehat{\boldsymbol{\alpha}}$ by minimizing Equation (6) with the cached quantities fixed.

(c) Online action selection

Receive o_0 (empty if unavailable); initialize the interaction record and $t \leftarrow 0$.
while the interaction has not terminated and the budget remains **do**
  Generate a ReAct response using demonstrations and current history.
  if the response is `Think` **then**
    Set a_t to the `Think` response and $o_{t+1} \leftarrow$ `OK`.
  else
    Construct x_t without demonstrations.
    Obtain environment-action candidates $\mathcal{A}_t$ (Section 3.2).
    Compute $\boldsymbol{\phi}(x_t, a; \mathcal{A}_t)$ and $\overline{Q}_\theta(a \mid x_t)$ for $a \in \mathcal{A}_t$.
    Select a_t by Equation (7); execute it to receive o_{t+1}.
  end if
  Append (a_t, o_{t+1}) to the interaction record; set $t \leftarrow t + 1$.
end while

Fixed during online action selection: $\widehat{\boldsymbol{\alpha}}$ and θ.

On the environment-action branch, the agent constructs x_t and $\mathcal{A}_t$, recomputes candidate features and prior weights, and selects

$$a_t = \arg\max_{a \in \mathcal{A}_t} P_{\widehat{\boldsymbol{\alpha}}}(a \mid x_t, \mathcal{A}_t). \tag{7}$$

Ties are resolved by a fixed rule. The selected environment action is executed and its observation is appended to the history. The concept $\widehat{\boldsymbol{\alpha}}$ remains fixed throughout the task, and θ is frozen throughout fitting and execution.

The ReAct prompt includes demonstrations to guide reasoning and routing. The action scorer instead uses x_t, containing the current task and its interaction history, including preceding thoughts, but no expert demonstrations. Thus, the fitted concept influences environment-action selection, while reasoning remains generated by the language model.

Candidate generation determines which actions can be considered; Equation (7) determines which one is selected. Only the selected action is executed, and unselected proposals are not tested in the environment. Algorithm 1 summarizes the procedure for nonempty candidate collections; parsing and fallback details are discussed in Appendix B.

4 Experiments

Section 4.1 describes the evaluation setting, followed by cross-benchmark results (Section 4.2), concept visualizations (Section 4.3), combination with Reflexion (Section 4.4), and a control comparison for the learned concept (Section 4.5).

4.1 Experimental Setup

Benchmarks and metrics. We evaluate on TextWorld, a framework for text-based games (Côté et al., 2018); ALFWorld, which expresses household tasks as text-based interaction (Shridhar et al., 2021); and ScienceWorld, which requires agents to carry out scientific tasks in an interactive environment (Wang et al., 2022). In our TextWorld and ALFWorld settings, the environment supplies

environment-action candidates. ScienceWorld uses an open-action-space setting in which the model proposes a finite candidate collection at each step. We report normalized scores for TextWorld and ScienceWorld and success rate over 134 tasks for ALFWorld, all on a 0–100 scale. Avg. is the unweighted mean of these three metrics, computed before rounding. It is a cross-benchmark summary, not an overall success rate.

Models and baselines. We use Qwen3-4B, Qwen3-8B, and Qwen3-14B. ReAct interleaves reasoning and actions, whereas Act-only omits explicit reasoning (Yao et al., 2023b). Self-Consistency provides a comparison based on aggregating sampled responses (Wang et al., 2023). StateAct makes goals and states explicit (Rozanov & Rei, 2025), while ReflAct reflects on the current state relative to the goal (Kim et al., 2025). ExpeL reuses natural-language insights and retrieved trajectories (Zhao et al., 2024). We also include DeltaMem (Tan et al., 2026) and a Trajectory SFT reference that updates model parameters using trajectory data. Reflexion configurations with two attempts are evaluated separately.

Demonstrations and concept fitting. Qwen3-32B generates the expert trajectories. Each task type uses a fixed set of two trajectories for ALFWorld and ten for TextWorld and ScienceWorld, shared across its test tasks and demonstration-based baselines. The test task's own expert trajectory is excluded from fitting and parameter selection. For each backbone, CB-Agent fits one concept per task type using Equation (6) and reuses it without test-time updates, keeping the language model frozen. ScienceWorld candidates are sampled by the agent's own backbone. Appendix B provides collection, fitting, and execution details.

Evaluation scope. The main comparison evaluates single-trial task performance. We report the two-trial Reflexion comparison separately. Computational costs and the scope of the empirical evidence are discussed in Appendix F.

4.2 Main Results

Table 1 shows that CB-Agent achieves the highest cross-benchmark average among the evaluated methods at every model size: 66.6, 72.5, and 75.9 for 4B, 8B, and 14B, respectively. It leads in six of nine model–benchmark combinations, and its scores increase with model size on all three benchmarks.

On TextWorld, CB-Agent ranks first throughout, including at 8B where it exceeds the strongest baseline, DeltaMem, by 2.1 points. On ScienceWorld, ExpeL is the strongest baseline at every size, but CB-Agent consistently surpasses it. Thus, the leading results span both environment-supplied and model-generated candidate collections. The experience-based baselines are less consistent across environments: ExpeL is strongest on ScienceWorld, whereas DeltaMem is competitive on TextWorld at 8B. ALFWorld shows a different pattern: StateAct leads at 4B and 8B, and ReflAct leads at 14B. CB-Agent improves from 76.9 to 81.3 but remains below these leaders. One possible limitation is that better individual action choices may still leave errors that prevent task completion. Since ALFWorld measures complete-task success, such partial improvements would not increase its reported score.

Finally, relative to ReAct, CB-Agent improves eight settings and matches one, with average gains of 2.4, 5.1, and 2.5 points at the three model sizes, computed before rounding.

4.3 Visualization of Estimated Concepts

For each of ALFWorld's six task types, we repeatedly sample five complete expert trajectories at random and fit one concept per set using Section 3.2. We apply t-SNE (van der Maaten & Hinton, 2008) separately for each backbone. This diagnostic uses five trajectories per concept, rather than the two used in the main ALFWorld evaluation. Figure 2 shows recurring same-type clusters across all three backbones, with some dispersion and outliers. This provides qualitative evidence of task-related structure across sampled trajectory sets, suggesting that the estimated concepts capture patterns shared by tasks of the same type.

Table 1: Results across three benchmarks and model sizes. All metrics use a 0–100 scale: normalized scores for TextWorld and ScienceWorld, and success rate for ALFWorld. Avg. is the unweighted mean across the three benchmarks. Bold marks the best result within each model block.

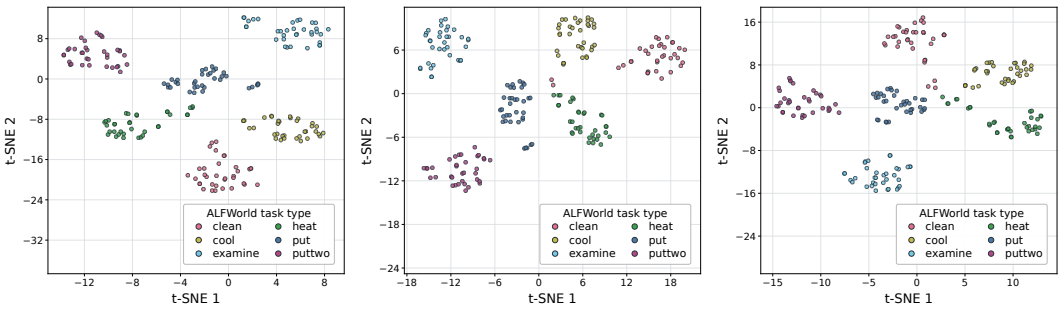
		Fixed action space		Open action space	
Model	Method	TextWorld	ALFWorld	ScienceWorld	Avg.
Qwen3-4B	ReAct	61.8	76.9	53.9	64.2
	Act-only	52.0	54.5	48.6	51.7
	Self-Consistency	57.4	69.4	35.4	54.1
	StateAct	43.3	**79.1**	28.7	50.4
	ExpeL	37.0	67.2	58.7	54.3
	ReflAct	57.2	50.0	28.7	45.3
	DeltaMem	49.7	41.8	31.4	41.0
	Trajectory SFT	26.6	22.4	4.9	18.0
	CB-Agent (ours)	**63.5**	76.9	**59.3**	**66.6**
Qwen3-8B	ReAct	74.6	72.4	55.1	67.4
	Act-only	62.4	68.7	36.6	55.9
	Self-Consistency	54.7	79.9	44.0	59.5
	StateAct	38.5	**82.8**	17.6	46.3
	ExpeL	45.5	68.7	58.7	57.6
	ReflAct	61.3	56.0	35.8	51.0
	DeltaMem	75.4	62.7	36.2	58.1
	Trajectory SFT	49.2	11.2	8.3	22.9
	CB-Agent (ours)	**77.5**	78.4	**61.6**	**72.5**
Qwen3-14B	ReAct	80.6	78.4	61.4	73.5
	Act-only	75.9	74.6	59.5	70.0
	Self-Consistency	57.1	81.3	50.8	63.1
	StateAct	46.0	73.1	26.4	48.5
	ExpeL	39.1	79.9	63.5	60.8
	ReflAct	72.6	**84.3**	46.1	67.7
	DeltaMem	67.0	74.6	49.4	63.7
	Trajectory SFT	52.9	32.1	11.3	32.1
	CB-Agent (ours)	**82.3**	81.3	**64.1**	**75.9**

4.4 Combining CB-Agent with Reflexion

We evaluate whether concept-based action selection can be combined with Reflexion, which uses feedback from earlier attempts to guide subsequent ones (Shinn et al., 2023). On ALFWorld, we compare Reflexion with CB-Agent + Reflexion under a two-trial setting for each backbone. The concept remains fixed: reflection changes the interaction context rather than refitting the concept from test feedback. This comparison is separate from the main table because additional attempts introduce a different evaluation budget. Table 2 shows success-rate gains of 4.5, 7.5, and 5.2 percentage points at 4B, 8B, and 14B, respectively. These results support combining a fixed concept-based scoring rule with reflections that update the context across attempts.

4.5 Effect of the Learned Concept

To isolate the contribution of the fitted concept, we compare CB-Agent with a control that sets $\boldsymbol{\alpha} = \mathbf{0}$ on ALFWorld using all three backbones. Both configurations retain the same ReAct routing, candidate construction, language-model prior, length correction, and interaction budget. With the concept term removed, the control selects actions using only the length-adjusted prior weights in Equation (3); it is neither random selection nor standard ReAct. Table 3 reports success rates, with CB-Agent values reproduced from Table 1. The learned concept improves success rates by 10.4, 6.0, and 14.9 percentage points at 4B, 8B, and 14B, respectively, computed before rounding. Gains across all three backbones support its contribution beyond the length-adjusted language-model prior in this ALFWorld setting.



(a) Qwen3-4B (b) Qwen3-8B (c) Qwen3-14B

Figure 2: t-SNE visualizations of concept vectors on ALFWorld for (a) Qwen3-4B, (b) Qwen3-8B, and (c) Qwen3-14B. Each point represents a concept estimated from five randomly sampled expert trajectories of a single task type, with colors denoting the six task types. Concepts from the same task type tend to cluster across all three backbones. Each panel is projected independently.

Table 2: ALFWorld success rate (%) over 134 tasks with Qwen3 backbones. Both methods use two trials. Bold marks the higher value.

Method	4B	8B	14B
Reflexion	85.1	82.1	85.8
CB-Agent + Reflexion	**89.6**	**89.6**	**91.0**

Table 3: ALFWorld success rate (%) over 134 tasks with Qwen3 backbones. Single-trial concept control. Bold marks the higher value.

Configuration	4B	8B	14B
$\boldsymbol{\alpha} = \mathbf{0}$	66.4	72.4	66.4
CB-Agent	**76.9**	**78.4**	**81.3**

These gains address a different question from the comparison with ReAct. The main table compares complete agent configurations, whereas this control tests whether learning from expert choices improves the same candidate-scoring pipeline. Although CB-Agent does not lead on ALFWorld, its gains over prior-only selection across all three backbones support the contribution of the learned concept within this pipeline. Together with the Reflexion results, this supports using a fixed concept as a reusable component: expert choices inform the scoring rule, while interaction feedback and reflections change the context in which it is applied.

5 Related Work

ICL research explores numerical representations of demonstration knowledge, including Task Vectors and Function Vectors (Hendel et al., 2023; Todd et al., 2024). CB-ICL estimates shared concepts for individual-input prediction (Tang et al., 2025); we adapt this framework to sequential action selection with changing contexts and candidates. For language agents, ReAct interleaves reasoning and actions (Yao et al., 2023b), while ExpeL reuses language-based insights and trajectories (Zhao et al., 2024), and Reflexion uses verbal feedback across attempts (Shinn et al., 2023). CB-Agent retains ReAct interaction but fits a numerical concept from expert choices to score actions, without requiring verbal experience summaries or updating LLM parameters. Appendix E provides an extended discussion.

6 Conclusion

We introduced CB-Agent, which adapts concept-based in-context learning to sequential action selection. It fits a reusable concept from expert environment-action decisions and uses frozen language-model features to score candidates in the evolving interaction context, without first converting experience into verbal rules. Across three benchmarks and three Qwen3 model sizes, CB-Agent achieves the highest average score among the evaluated methods. An ALFWorld control comparison supports the contribution of the learned concept beyond prior-only action scoring. Combining CB-Agent with Reflexion further improves ALFWorld success rates, while concept visualizations provide qualitative evidence of task-related structure. Future work will examine candidate coverage and transfer to unseen task types.

AI USE STATEMENT

In this work, we used OpenAI Codex to assist with interpreting author-provided experimental results and improve the language and readability of the manuscript. We have not used generative AI tools to develop theoretical models, formulate mathematical claims, write proofs, propose hypotheses, or implement the proposed method. As part of the experimental workflow, we used Qwen3-32B to generate the expert demonstration trajectories used in this study. The roles of language models in demonstration generation and agent execution are described in the method and experimental sections. The authors retain responsibility for all research decisions and for the accuracy, originality, and integrity of the final manuscript, including its mathematical claims, experimental interpretations, references, and AI-assisted artifacts.

ETHICS STATEMENT

This work evaluates language agents in existing benchmarks: TextWorld, ALFWorld, and ScienceWorld. The experiments are conducted in simulated environments and do not involve human participants, personal data collection, or real-world deployment.

REPRODUCIBILITY STATEMENT

Section 3 specifies the action model, concept-fitting objective, and interaction procedure. Section 4.1 describes the benchmarks, metrics, backbones, and shared-demonstration protocol. Appendix B documents feature construction, length-adjusted prior weights, trajectory processing, optimization, and candidate generation. Appendix C provides derivations of the objective's properties under the stated assumptions. The source code is available at `https://anonymous.4open.science/r/CB-Agent-DFF9/`.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. doi: 10.48550/arXiv.2005.14165. URL https://arxiv.org/abs/2005.14165.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9354–9366. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.557. URL https://aclanthology.org/2024.findings-acl.557/.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018. doi: 10.48550/arXiv.1806.11532. URL https://arxiv.org/abs/1806.11532.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1107–1128. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.64. URL https://aclanthology.org/2024.emnlp-main.64/.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

Shao-Lun Huang, Anuran Makur, Gregory W. Wornell, and Lizhong Zheng. Universal features for high-dimensional learning and inference. *Foundations and Trends in Communications and Information Theory*, 21(1–2):1–299, 2024. doi: 10.1561/0100000107. URL https://doi.org/10.1561/0100000107.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. ReflAct: World-grounded decision making in LLM agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33433–33465. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1697. URL https://aclanthology.org/2025.emnlp-main.1697/.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10/.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. doi: 10.48550/arXiv.2112.09332. URL https://arxiv.org/abs/2112.09332v1.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In *Advances in Neural Information Processing Systems*, volume 37, pp. 114843–114871, 2024. doi: 10.52202/079017-3646. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d032263772946dd5026e7f3cd22bce5b-Abstract-Conference.html.

Nikolai Rozanov and Marek Rei. StateAct: Enhancing LLM base agents via self-prompting and state-tracking. *arXiv preprint arXiv:2410.02810*, 2025. URL https://arxiv.org/abs/2410.02810v3. Version 3.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2655–2671. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.191. URL https://aclanthology.org/2022.naacl-main.191/.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768v2.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. AdaPlanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 58202–58245. Curran Associates, Inc., 2023. doi: 10.52202/075280-2537. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b5c8c1c117618267944b2617add0a766-Abstract.html.

Haoran Tan, Zeyu Zhang, Zhicheng Cao, Rui Li, and Xu Chen. DELTAMEM: Incremental experience memory for LLM agents via residual trees. *arXiv preprint arXiv:2606.03083*, 2026. doi: 10.48550/arXiv.2606.03083. URL https://arxiv.org/abs/2606.03083.

Huaze Tang, Tianren Peng, and Shao-lun Huang. On theoretical interpretations of concept-based in-context learning. *arXiv preprint arXiv:2509.20882*, 2025. URL https://arxiv.org/abs/2509.20882v2. Version 2.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://jmlr.org/papers/v9/vandermaaten08a.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

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/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. URL https://arxiv.org/abs/1609.08144.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=RdJVFCHjUMI.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023a. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract.html.

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

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

APPENDIX CONTENTS

A Notation 15
B Representation and Execution Details 15
C Properties of the Conditional Objective 17
D From Weak Dependence to the Exponential Concept Model 17
E Extended Related Work 18
F Limitations 20

A NOTATION

Table 4 summarizes the principal notation used in the paper. The trajectory index t and pooled-example index i refer to different units: t includes reasoning steps, whereas i indexes only environment-action decisions used for fitting.

Table 4: Principal notation for CB-ICL and CB-Agent.

Symbol	Meaning
Prediction and interaction	
x, y, $\mathcal{Y}$	Input text, candidate label, and finite nonempty label set in CB-ICL.
$Q_0(y)$	Positive base weight for label y in the CB-ICL model.
inst	Task instruction specifying the goal of an interaction.
τ, T	Complete interaction trajectory and its number of steps, including both environment and `Think` actions.
t, a_t, o_{t+1}	Interaction-step index, environment or `Think` action, and recorded feedback.
o_0	Initial observation before any action; empty if the environment provides none.
x_t	Pre-action text context containing the instruction, initial observation, and preceding actions and feedback.
$\mathcal{A}_t$	Finite nonempty candidate collection at environment-action step t.
x, a, $\mathcal{A}$	Context, candidate environment action, and candidate collection for one decision, with its index omitted.
Expert data and concept estimation	
$\mathcal{D}_{\mathrm{traj}}$	Collection of selected complete expert trajectories.
$\mathcal{D}$, N	Pooled dataset of expert environment-action decisions and its number of examples.
i, $(x_i, \mathcal{A}_i, a_i^{\mathrm{E}})$	Pooled-example index and its context, saved candidates, and recorded expert environment action.
θ	Frozen language-model parameters.
m, $\boldsymbol{\phi}(x, a; \mathcal{A})$	Feature dimension and fixed action-feature vector in $\mathbb{R}^m$ for a given context and candidate collection.
$\boldsymbol{\alpha}$, $\widehat{\boldsymbol{\alpha}}$	Concept parameter vector in $\mathbb{R}^m$ and its fitted estimate.
$P_{\boldsymbol{\alpha}}$	Concept-conditioned distribution over candidate labels or environment actions, as specified by its arguments.
$\mathcal{L}$, λ	Regularized negative log-likelihood objective and positive regularization strength.
Language-model weights and feature construction	
$Q_\theta(a \mid x)$	Frozen model probability of the complete candidate action's token sequence.
$L(a)$, γ	Number of action tokens and length-correction exponent in $[0, 1]$.
$\overline{Q}_\theta(a \mid x)$	Length-adjusted prior weight, $Q_\theta(a \mid x)^{1/L(a)^\gamma}$.
p_θ, a_ℓ, $a_{<\ell}$	Next-token distribution, action token at position ℓ, and preceding action tokens.
d, $\boldsymbol{u}_{i,a}$	Hidden width and last-layer hidden vector at the final token of candidate a in example i; $m = d + 1$ in the implemented feature construction.
M_i	Number of candidate entries, $\lvert\mathcal{A}_i\rvert$.
$\mu_{i,j}$, $\sigma_{i,j}$	Mean and standard deviation of hidden coordinate j across candidates in example i.
$\widetilde{\boldsymbol{u}}_{i,a}$, ϵ	Coordinate-wise standardized hidden vector and positive numerical floor used when dividing by standard deviations.

B REPRESENTATION AND EXECUTION DETAILS

Candidate features. Let d be the language model's hidden width, so that $m = d+1$ in our feature construction. For decision i and candidate $a \in \mathcal{A}_i$, let $\boldsymbol{u}_{i,a} \in \mathbb{R}^d$ be the last-layer hidden vector at

the final token of a, obtained by encoding the context followed by the action prefix and candidate string. The ALFWorld implementation uses plain-text serialization with the prefix `\n> `, rather than applying a chat template. Let $M_i = |\mathcal{A}_i|$ be the number of candidate entries. For coordinate $j \in \{1, \dots, d\}$, let $u_{i,a,j}$ be coordinate j of $\boldsymbol{u}_{i,a}$, and let $\mu_{i,j}$ and $\sigma_{i,j}$ be its mean and standard deviation across candidates. Choose a numerical floor $\epsilon > 0$ to avoid division by zero. Denote the standardized coordinate by $\widetilde{u}_{i,a,j}$ and the vector collecting these coordinates by $\widetilde{\boldsymbol{u}}_{i,a}$:

$$\mu_{i,j} = \frac{1}{M_i} \sum_{a \in \mathcal{A}_i} u_{i,a,j}, \qquad \sigma_{i,j} = \sqrt{\frac{1}{M_i} \sum_{a \in \mathcal{A}_i} (u_{i,a,j} - \mu_{i,j})^2}, \tag{8}$$

$$\widetilde{u}_{i,a,j} = \frac{u_{i,a,j} - \mu_{i,j}}{\max(\sigma_{i,j}, \epsilon)}, \qquad \boldsymbol{\phi}(x_i, a; \mathcal{A}_i) = \begin{bmatrix} \widetilde{\boldsymbol{u}}_{i,a} \\ M_i^{-1/2} \end{bmatrix}. \tag{9}$$

We recompute the means and standard deviations for each candidate collection, including at test time, without using expert labels. The appended coordinate $M_i^{-1/2}$ is a normalized bias feature shared by all candidates at decision i, following the feature construction in CB-ICL (Tang et al., 2025).

Language-model weights. For token index $\ell \in \{1, \dots, L(a)\}$, let a_ℓ be token ℓ of candidate action a and $a_{<\ell}$ its preceding tokens. Let p_θ be the frozen model's next-token distribution. The model probability $Q_\theta(a \mid x)$ is the product of these token probabilities, so its length-adjusted log-weight is

$$\log \overline{Q}_\theta(a \mid x) = \frac{1}{L(a)^\gamma} \sum_{\ell=1}^{L(a)} \log p_\theta(a_\ell \mid x, a_{<\ell}). \tag{10}$$

When computing token probabilities, the input includes the model's reply prefix. The length $L(a)$ counts only action tokens, not context or role-prefix tokens. Fitting and execution use the same text formatting and token-counting convention. All candidates are assumed to have positive model probability. The reported implementation first normalizes these prior weights over candidates, then adds their logarithms to the concept scores. Ignoring numerical stabilization, prior normalization divides every weight by the same number, which cancels between the numerator and denominator of Equation (3). Using the unnormalized weights therefore gives the same action distribution.

Think actions and trajectory construction. A `Think` action contains the reasoning text as part of its action string and receives the fixed feedback `OK`. The ReAct response determines the branch: a response beginning with `think:` is recorded directly; otherwise, the concept scorer selects an environment action from its candidates. The initial action proposed by the routing response is not the executed action unless the scorer selects it. `Think` responses contribute no likelihood terms. Their text and feedback are included only in subsequent contexts. A `Think` step advances the interaction index without changing the environment.

The expert Qwen3-32B model uses native thinking, but content inside its `<think>` tags is discarded. The stored reasoning is the explicit agent-level `Think` response outside those tags. The evaluated 4B, 8B, and 14B agents use explicit `Think` responses without native model thinking.

Demonstration selection and fitting scope. As specified in Section 4.1, demonstrations are fixed by task type, and each backbone's fitted concept is reused across test tasks of that type. It also remains fixed during reflection-based retries. Test-time feedback updates the interaction context but is not used to refit the concept. During expert collection, candidate actions and the expert's choice are saved together. Each evaluated backbone recomputes features and prior weights on these saved contexts and candidates. Only environment-action steps contribute fitting targets; preceding `Think` responses remain in their contexts.

Optimization and execution settings. We use the following settings across TextWorld, ALFWorld, and ScienceWorld. We initialize the concept at zero and use Adam with learning rate 0.01, at most 1,500 iterations, and gradient-norm tolerance 10^{-4}. Its squared-norm penalty has coefficient 0.5, corresponding to $\lambda = 1$ in Equation (6). Candidate features and prior weights are cached before fitting. Each trial allows at most 50 interaction steps, including `Think` steps. We filter `look` and `inventory` from the environment-action candidates, with `look` as the empty-collection fallback. We use length exponent $\gamma = 0.8$ and unit weight on the prior log-probability.

Open-action-space candidate construction. ScienceWorld requests 20 independently sampled replies to the same ReAct context at temperature 0.7, rather than asking for a list of 20 actions in one reply. Replies may repeat, so this does not guarantee 20 distinct candidates. Each retained candidate is a single environment-action string; `Think` is handled by routing, not by the concept scorer. During expert collection, Qwen3-32B proposes and selects candidates. During evaluation, the agent's own backbone proposes candidates. Fitting reuses the saved expert candidate collections and recomputes their features and prior weights with the evaluated backbone. The recorded expert target is therefore already among its saved candidates. Only the selected environment action is executed, and candidates are resampled at the next environment-action decision.

C Properties of the Conditional Objective

This appendix establishes properties of the fitting objective for a given dataset, with features and prior weights held fixed. The analysis concerns the optimization objective; predictive accuracy and task performance are evaluated experimentally.

Existence and uniqueness. Let $\nabla^2\mathcal{L}$ denote the Hessian of the objective with respect to $\boldsymbol{\alpha}$, I_m the m-dimensional identity matrix, and Cov the covariance under the indicated candidate distribution. With fixed features and positive prior weights,

$$\nabla^2\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Cov}_{a\sim P_{\boldsymbol{\alpha}}(\cdot|x_i,\mathcal{A}_i)}[\boldsymbol{\phi}(x_i,a;\mathcal{A}_i)] + \lambda I_m. \tag{11}$$

Each covariance is positive semidefinite, so the objective is λ-strongly convex. The negative log-likelihood is nonnegative, and the quadratic penalty makes the objective grow without bound as $\|\boldsymbol{\alpha}\|_2$ grows. Consequently, a unique finite minimizer exists.

Trajectory interpretation. The objective weights expert environment-action decisions equally; a trajectory with more such decisions contributes more terms. No independence assumption between consecutive decisions is needed to define this empirical objective. To interpret the data loss as a trajectory likelihood, assume that the initial conditions do not depend on $\boldsymbol{\alpha}$. Also assume that, given the preceding interaction record and selected action, feedback does not depend on $\boldsymbol{\alpha}$. Candidate generation and any external stopping rule must also be independent of $\boldsymbol{\alpha}$ conditional on the preceding record. The same condition applies to ReAct routing and reasoning generation. A `Think` response and its deterministic `OK` feedback then contribute only factors independent of $\boldsymbol{\alpha}$. Under the action model, the joint probability of a recorded trajectory and its saved candidate collections factors into the conditional environment-action probabilities and terms independent of $\boldsymbol{\alpha}$. Taking negative logarithms, summing over trajectories, and dividing by N gives the data term in Equation (6), up to an additive term independent of $\boldsymbol{\alpha}$. This interpretation does not require assuming that the expert policy is exactly realizable by the fitted model.

D From Weak Dependence to the Exponential Concept Model

The exponential form in Equation (1) is motivated by a local representation of conditional distributions. We give the derivation using the modal decomposition and weak-dependence framework of Huang et al. (2024), then connect it to the concept parameterization in Section 2.2.

Weak dependence as a small conditional correction. For this local analysis, let X and Y be random variables on fixed finite sets $\mathcal{X}_0$ (inputs) and $\mathcal{Y}$ (labels), each with at least two elements. Write P_{XY} for their joint distribution and P_X, P_Y for its marginals. Consider a family of such distributions indexed by $\delta > 0$, with marginal probabilities bounded below by a constant $c_0 > 0$ as $\delta \to 0$. We suppress this dependence in the notation. The weak-dependence condition is

$$\chi^2(P_{XY}\|P_XP_Y) := \sum_{x\in\mathcal{X}_0}\sum_{y\in\mathcal{Y}} \frac{\big(P_{XY}(x,y) - P_X(x)P_Y(y)\big)^2}{P_X(x)P_Y(y)} \le \delta^2. \tag{12}$$

Thus, weak dependence concerns the relation between the input X and label Y. Define their relative departure from independence by

$$r(x,y) := \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} - 1, \qquad P(y \mid x) = P_Y(y)\big[1 + r(x,y)\big]. \tag{13}$$

Equation (12) implies $|r(x,y)| \le \delta/\sqrt{P_X(x)P_Y(y)} \le \delta/c_0$. Also, $\sum_y P_Y(y)r(x,y) = 0$ for every x by conditional normalization. All order statements below are uniform over x, y as $\delta \to 0$.

From the linear correction to an exponential form. Let $K = \min\{|\mathcal{X}_0|, |\mathcal{Y}|\}$. The modal decomposition supplies functions $f_j : \mathcal{X}_0 \to \mathbb{R}$ and $g_j : \mathcal{Y} \to \mathbb{R}$, and coefficients $\sigma_j \ge 0$, for $j = 1, \dots, K-1$, such that

$$r(x,y) = \sum_{j=1}^{K-1} \sigma_j f_j(x) g_j(y). \tag{14}$$

The functions are mean-zero and orthonormal under their respective marginals; the coefficients are the nonconstant singular values of the matrix with entries $P_{XY}(x,y)/\sqrt{P_X(x)P_Y(y)}$ (Huang et al., 2024, Proposition 2.2). Since $r = O(\delta)$, Taylor expansion of $\log(1+r)$ gives

$$\log \frac{P(y \mid x)}{P_Y(y)} = \sum_{j=1}^{K-1} \sigma_j f_j(x) g_j(y) + O(\delta^2). \tag{15}$$

This is the local exponential representation: to first order, the same features that describe a linear change in probability also describe an additive change in log-probability. The second-order remainder here follows directly from the stated finite-support and marginal lower-bound conditions.

Normalization and the concept parameterization. To express this representation in the notation of Equation (1), take $m = K - 1$, set $\phi_j(x,y) = f_j(x)g_j(y)$, and collect the modal coefficients in $\boldsymbol{\alpha}_\star = (\sigma_1, \dots, \sigma_m)^\top$. Then $\boldsymbol{\alpha}_\star^\top \boldsymbol{\phi}(x,y) = r(x,y)$. Define the normalizing factor $Z_\star(x) = \sum_{b \in \mathcal{Y}} P_Y(b) \exp\{r(x,b)\}$. The zero-mean identity above and the expansion $e^r = 1 + r + O(\delta^2)$ imply $Z_\star(x) = 1 + O(\delta^2)$. Consequently, the normalized exponential distribution satisfies

$$\frac{P_Y(y)\exp\{\boldsymbol{\alpha}_\star^\top \boldsymbol{\phi}(x,y)\}}{\sum_{b \in \mathcal{Y}} P_Y(b)\exp\{\boldsymbol{\alpha}_\star^\top \boldsymbol{\phi}(x,b)\}} = P(y \mid x) + O(\delta^2). \tag{16}$$

This gives Equation (1) with base weight $Q_0 = P_Y$ as a first-order accurate model. Normalization ensures a positive distribution whose probabilities sum to one for any finite concept vector, including outside the local regime.

The concept model retains this normalized form while using a chosen positive base weight Q_0 and fixed language-model features in place of the distribution-dependent modal features. These choices define the model family; demonstration likelihood determines its concept vector. The $O(\delta^2)$ approximation above applies to the full modal representation, while the practical model's approximation quality depends on its chosen features and base weights. Section 3 applies this construction to decisions with changing contexts and candidate actions, using the length-adjusted language-model weights as the context-dependent baseline.

E Extended Related Work

E.1 In-Context Learning and Numerical Task Representations

ICL uses demonstrations to specify a task without updating the language model's parameters (Brown et al., 2020; Dong et al., 2024). One line of work improves the selection and ordering of examples in the prompt (Rubin et al., 2022; Lu et al., 2022). For instance, Liu et al. (2022) retrieve semantically similar demonstrations for each input. Another line investigates how a model infers or represents the task conveyed by demonstrations (Xie et al., 2022). Hendel et al. (2023) show that ICL behavior can often be represented using a task vector computed from examples. Todd et al. (2024) identify activation vectors that encode task functions and can elicit corresponding behavior when inserted

into a model's forward pass. These two studies motivate numerical representations of demonstration knowledge, but their activation-based mechanisms differ from fitting weights for an explicit candidate-scoring model.

CB-ICL connects concept estimation over language-model features with theoretical analyses of label prediction (Tang et al., 2025). This framework is the starting point of CB-Agent. We use expert choices from multiple trajectories to fit a shared concept, then apply it to context-dependent action features during execution. The key change in setting is that an action affects the feedback and thus the context of later decisions; the inputs are not simply a fixed collection of prediction queries. Candidate collections can also differ from one step to the next. Accordingly, the prediction analysis of prior CB-ICL does not by itself establish a task-success guarantee for our agent. Our objective analysis concerns fitting with fixed features and prior weights, while the sequential behavior is assessed experimentally.

E.2 Reasoning, State Tracking, and Action Selection

Chain-of-thought prompting elicits intermediate reasoning (Wei et al., 2022), while tree-based search explores alternative reasoning paths (Yao et al., 2023a). ReAct provides a framework for alternating language-based reasoning with actions and environmental observations (Yao et al., 2023b). Subsequent methods change how this interaction is organized. StateAct uses self-prompting and state tracking to maintain task goals and relevant state information (Rozanov & Rei, 2025). ReflAct emphasizes repeated reflection on the relation between the current state and the goal (Kim et al., 2025). AdaPlanner revises plans using environmental feedback and also reuses successful plans as few-shot exemplars (Sun et al., 2023). These are overlapping approaches to improving decisions, rather than mutually exclusive categories: a method can organize the current interaction while also retaining useful experience.

Self-Consistency offers a different approach to inference: it samples multiple reasoning paths and selects an answer by aggregating their outcomes (Wang et al., 2023). CB-Agent's open-action-space procedure also uses multiple sampled replies, but sampling serves to propose candidates for the current decision. Selection uses a concept fitted from expert choices together with language-model prior weights, rather than agreement among sampled answers. Only the selected action is executed. Thus, candidate proposal and candidate selection are distinct parts of our procedure; the proposal mechanism is not a standard component of ReAct. CB-Agent uses ReAct contexts without introducing a separate planning or state-tracking architecture.

E.3 Learning and Reusing Agent Experience

ExpeL and Reflexion show how agents can reuse execution experience through language. ExpeL extracts insights from training-task experience and retrieves relevant past trajectories for subsequent decisions (Zhao et al., 2024). Reflexion converts feedback into verbal reflections retained in episodic memory to guide later trials (Shinn et al., 2023). Their common use of text should not obscure the difference between transferring experience across tasks and adapting across repeated attempts at a task. Nor does freezing the LLM distinguish CB-Agent from these approaches, since they also avoid updating its parameters. Other work reuses executable skills or synthesizes task and state knowledge from trajectories (Wang et al., 2024; Qiao et al., 2024).

CB-Agent differs in what is extracted from experience and how it enters the decision rule. Expert action choices supervise a numerical concept through a likelihood objective, and that concept weights action features when comparing candidates. The method does not require an intermediate natural-language summary of the expert experience. In contrast to trajectory-based LLM fine-tuning (Zeng et al., 2024; Chen et al., 2024), the optimized parameters belong to the concept model, not the LLM. This distinction is about the representation and use of experience, not a general claim of superiority over language-based memory or fine-tuning. The concept remains fixed within each test task, while new observations and any retained reflections change the context used for scoring. This permits the combination with Reflexion studied in Section 4.4, without refitting the concept from test-time feedback.

F LIMITATIONS

Modeling and transfer. CB-Agent models expert action preferences using frozen language-model features. Our theoretical analysis characterizes the concept-fitting objective, while end-to-end task performance is evaluated empirically. The current setting uses demonstrations from the same task type; performance may depend on their quality and coverage. Transfer to unseen task types remains an important direction for future work.

Candidate selection and computation. In open-action-space tasks, action selection depends on the coverage and quality of generated candidates. Candidate generation, scoring, and concept estimation introduce additional computation, and our comparisons assess task performance rather than efficiency under matched computational budgets. Demonstrations are omitted from the action-scoring context but remain in ReAct reasoning and routing, so the complete agent remains subject to context-window limits.

Evaluation scope. Our experiments cover three text-interaction benchmarks and three model sizes within the Qwen3 family. The prior-only control provides evidence for the learned concept's contribution on ALFWorld; extending this analysis to the other benchmarks would clarify its broader applicability. Evaluation across additional model families and tasks, together with repeated-run uncertainty estimates, would further characterize the robustness of the observed gains.