REPAIR: RUBRIC-GUIDED TURN-LEVEL PREFERENCE LEARNING FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents on long-horizon tasks keep making the same decision errors, and the two standard remedies miss them for different reasons. Supervised fine-tuning imitates expert trajectories, which never visit the states the learner reaches after its own mistakes. Reinforcement learning from a terminal reward does visit those states, but it pays for full rollouts and still cannot say which turn went wrong. We introduce **RePair** (Rubric-guided Repair as preference Pairs), which turns the learner's recurring errors into a small set of reusable behavioral rubrics. A rubric names an observable error pattern, states the rule that corrects it, and gives a criterion that a repaired turn must meet. Whenever a rubric matches a failed turn, a repair model rewrites the turn as the rule prescribes, and a validator checks the rewrite against the criterion using only what the agent could see at that turn, so no rollout is needed. The rewrite and the learner's original response then form a preference pair on the very same input, and since that input carries no rubric, the trained policy needs none at inference. On BrowseComp-Plus, rubrics induced automatically from the learner's own failures lift a supervised Qwen3-14B search agent from 42.6% to 53.2% with 2,097 pairs. The learner can also write its own repairs: with no teacher supplying the preferred responses, two rounds of self-repair still gain 7.2 points over the same baseline, and the second round adds to the first, so an agent can keep improving from its own failures.

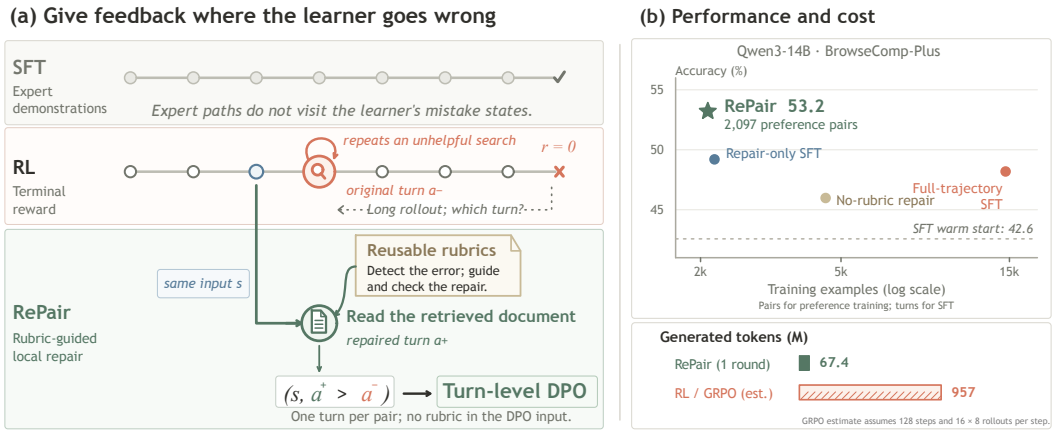


Figure 1: **RePair: rubric-guided, turn-level preference optimization.** **(a)** Expert demonstrations miss the states induced by the learner's own mistakes, while terminal rewards require full rollouts and provide weak credit assignment. RePair uses reusable behavioral rubrics to identify recurring errors, guide local repairs, and validate the replacements. Each accepted repair a^+ is paired with the original turn a^- at the same input s for turn-level DPO, without rubrics in the optimization input. **(b)** On BrowseComp-Plus, RePair improves Qwen3-14B accuracy from 42.6% to 53.2% with 2,097 preference pairs. One round generates 67.4M tokens, compared with an estimated 957M rollout tokens for 128 GRPO steps.

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.

1 INTRODUCTION

Tool-using agents often fail long-horizon tasks because of recurring decision errors. A deep-research agent that answers a question by searching and reading over many turns may reissue a search it has already run, request a document that never appeared in its evidence, or commit to an answer before the evidence supports one. Because each decision shapes every context that follows, an early mistake leaves the agent in situations it must handle many turns later. Expert demonstrations do not prepare it for such situations, since they cover the states an expert visits rather than the states the learner reaches after its own mistakes (Ross et al., 2011); reinforcement learning with verifiable rewards trains on the learner's own states, but long rollouts are expensive, and a terminal reward does not say which turn should have been different.

Local preference learning supervises the learner's own failed turns directly, but existing methods tie each correction to a single instance: Critical Step Optimization, for example, verifies each expert-proposed alternative by rolling it forward to completion, so every correction costs further rollouts and applies to one trajectory (Li et al., 2026). Rubrics, meanwhile, have entered training as scoring criteria that assess how good a response is (Gunjal et al., 2025; Ding, 2026; Shao et al., 2026), whereas corrective rules distilled from experience are supplied in the prompt at inference rather than trained into the policy (Shinn et al., 2023; Zhao et al., 2024; Fu et al., 2024).

We introduce **RePair** (Rubric-guided Repair as preference Pairs), which takes a third route: it treats rubrics as decision rules, induces them from the learner's recurring errors, and uses them at training time to build and locally validate repairs. A RePair rubric couples an observable error pattern with a corrective rule and an acceptance criterion that can be checked from the information available at that turn. Because it names a pattern of behavior rather than a question, one rubric applies to turns from unrelated trajectories, so a handful of rubrics can supervise thousands of decisions. When a rubric fires at a failed turn, a repair model proposes a new decision under its guidance, a validator checks the replacement against the criterion, and the accepted replacement is paired with the learner's original response at the same input (Figure 2). Acceptance is a local judgment rather than evidence of eventual task success; in exchange it requires no rollouts, and the rules remain inspectable. The rubric guides data construction only: the policy is trained on pairs whose input contains no rubric, and the trained agent needs none at inference.

Our contributions are the following.

- **A rollout-light rubric-driven training method.** RePair turns the learner's recurring errors into reusable behavioral rubrics, induced automatically from its own failed rollouts, and uses them to convert individual failed turns into validated preference pairs (§2). On BrowseComp-Plus, an induced codebook lifts a supervised Qwen3-14B agent from 42.6% to 53.2% with 2,097 pairs, outperforming a hand-written codebook across three training seeds (§3.2). Because every repair is written and checked at a single turn, a full repair round, failure rollouts included, consumes about as many tokens as nine GRPO steps of a recent search-agent recipe with 16 questions $\times$ 8 rollouts per step (Pan et al., 2026; §4.3).
- **Self-repair that improves across rounds.** The learner can repair its own failures under the same codebook and validator; two rounds gain 7.2 points over the same baseline, and the second round improves on the first across three seeds (§3.2).

Ablations show where the gain comes from: without rubrics, the same teacher repairing the same failed turns reaches only 46.0%, imitating seven times as many turns of the teacher's successful trajectories reaches 48.2%, imitating the accepted repairs without the rejected responses reaches 49.2%, and prompting with the rubrics instead of training on them reaches at most 46.2%, all below the 53.2% of RePair (§4.1).

2 METHOD

Setting. A tool-using agent solves a task over turns. At turn t the policy π receives an input s_t and produces a response $a_t \sim \pi(\cdot \mid s_t)$, a complete assistant turn consisting of reasoning, a structured state update, and an action; the environment then returns an observation o_t. Rather than appending the raw message history, the harness re-renders the next input from a compact state,

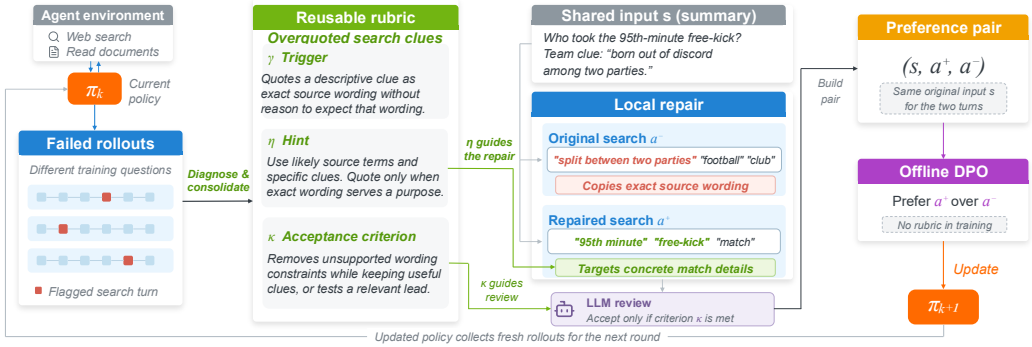


Figure 2: RePair turns recurring learner errors into local preference pairs. Rubrics (trigger γ, hint η, criterion κ) are induced from failed rollouts; the hint guides a replacement a^+ for a failed turn a^-, kept if it passes checks against κ. Both turns share the input s, and DPO uses no rubric.

$s_{t+1} = f(s_t, a_t, o_t)$, where f folds the trajectory so far into a static system prompt, the goal and its satisfied and unsatisfied constraints, a compacted action history, the evidence gathered so far, and the latest observation (Appendix A). Context length therefore depends on this state, not on the raw transcript. Each s_t is therefore a self-contained snapshot of one decision problem: a recorded input can later be given to a different model without replaying the trajectory that produced it, which is what allows a single turn to be repaired in place.

RePair runs in rounds. A round collects failed rollouts from the current policy, turns individual failed turns into preference pairs under reusable rubrics, and updates the policy on those pairs (Figure 2). This section describes which turns are eligible for repair (§2.1), the rubrics (§2.2), how a rubric turns a failed turn into a pair (§2.3), and the objective (§2.4).

2.1 Failed rollouts and eligible turns

At round k the policy π_k collects rollouts on training questions $q \sim \mathcal{D}_{\text{train}}$, and an outcome judge labels each trajectory correct or incorrect against the reference answer. RePair keeps only incorrect trajectories and records every input s_t and response a_t verbatim, because repair later poses a new decision at exactly the input the policy observed, and the stored response becomes the rejected side of the pair. Within a failed trajectory, a turn is *eligible* if its response parses into a well-formed action and is not a forced termination, a transport error, a refusal, or a truncated generation. Guided repair and validation, described below, decide which eligible turns become training examples, and one trajectory may contribute several pairs.

2.2 Reusable rubrics

A rubric $r = (\gamma, \eta, \kappa)$ specifies when a behavioral error is present and what would count as correcting it. The trigger γ describes the observable pattern, the hint η states a general decision rule, and the acceptance criterion κ specifies what a replacement must satisfy using only information available at that turn. The restriction on κ matters: a rubric is written with hindsight over whole failed trajectories, but a repair is produced and judged at the turn itself, so the criterion cannot presuppose anything the policy could not yet know. None of the three components names a particular question, which is what lets one rubric apply to turns from unrelated trajectories. A *codebook* is a set of rubrics; a labeling model reads each eligible turn and assigns whichever rubrics fire there, and each assignment (s_t, a_t, r) poses one repair problem.

Automatic induction. A codebook can be written by hand from a sample of failures, or induced from the failure pool in three stages. **Diagnosis**: a model reads one complete failed trajectory, knowing only that the final answer was judged incorrect, and proposes a bounded list of errors, each attached to a turn with a provisional trigger, hint, and criterion. **Consolidation**: findings from the whole pool are clustered by similarity, each cluster is rewritten into shared rules, a final pass merges duplicates across clusters, and specificity checks discard rules that encode a question-dependent

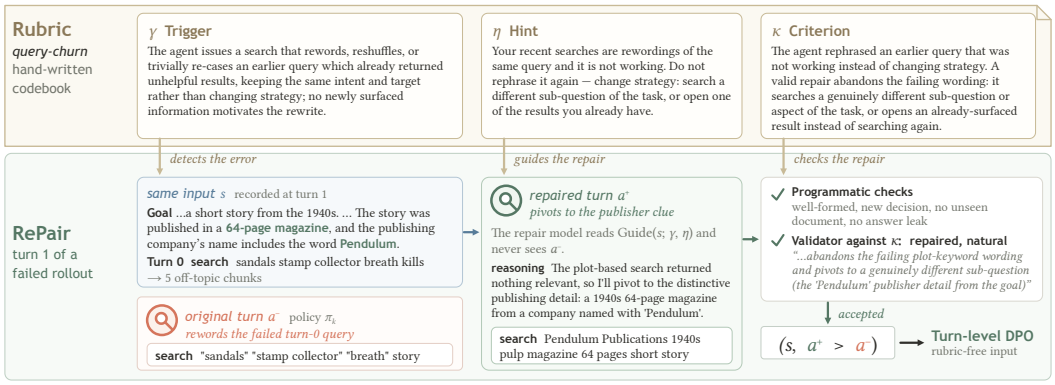


Figure 3: One repair under the hand-written QUERY-CHURN rubric (Qwen3-14B). The trigger flags turn 1 of a failed rollout, the hint guides the repair at the same input, and the validator checks the repair against the criterion. Text is quoted from the recorded pair; Appendix C gives the full rubric.

clue. **Reassignment**: the consolidated codebook is applied to the eligible turns exactly as a hand-written one would be. No finding or rule is edited by hand. Because rule discovery is separate from assignment, the same codebook can be applied unchanged to a fresh pool of failures. Clustering, budgets, and the specificity checks are detailed in Appendix B; the three codebooks used in our experiments are reproduced in Appendices C and D.

2.3 GUIDED REPAIR AND LOCAL VALIDATION

Repair. Fix an assignment and write $s = s_t$ and $a^- = a_t$. A repair model ρ_k makes a new decision at s under the guidance of r,

$$\hat{a} \sim \rho_k\big(\cdot \mid \mathrm{Guide}(s; \gamma, \eta)\big), \tag{1}$$

where Guide appends the trigger and the hint to the system prompt of s and leaves the rest unchanged. The repair model sees neither a^- nor anything after turn t, so hindsight from diagnosis reaches it only through the generic rule. It may be a stronger teacher or the learner itself; the latter choice, $\rho_k = \pi_k$, is *self-repair*, in which the learner proposes replacements for its own failures while diagnosis, labeling, and validation remain external.

Validation. A candidate must pass two screens. Programmatic checks require a well-formed turn in the policy's own format that changes the decision taken at s and uses no information unavailable there, such as an unseen document identifier or leaked answer content; a candidate that commits a final answer must also agree with the reference and be grounded in the recorded context. A validator model then reads s, a^-, $\hat{a}$, and κ and accepts only if the flagged error is repaired in the sense of κ and the replacement reads as a natural decision at that point. Candidates are sampled up to a fixed budget, the first accepted one is kept, and an assignment with no accepted candidate yields no pair. Acceptance is a local judgment against κ; no candidate is ever rolled forward, so each attempt costs one generation and one validation. Reference answers enter only through the outcome judge and the final-answer check; no other role sees them. The full list of checks is given in Appendix B. On 150 sampled candidates, the validator agrees with human labels 88.7% of the time (Appendix G).

Pairs. The accepted replacement, re-rendered in the policy's ordinary turn format, becomes a^+, and (s, a^+, a^-) joins the round's pair set $\mathcal{P}_k$. Both completions are scored against the same s, which contains neither the rubric nor any repair instruction: the rubric guides data construction and is absent from policy optimization, rollouts, and evaluation.

Example. In Figure 3, a 14B rollout repeats a failed plot-keyword search with its words rearranged, and the labeler assigns QUERY-CHURN. The repair model instead searches the one distinctive clue in the goal, a publisher named "Pendulum", which the validator accepts as a genuinely different sub-question.

2.4 PREFERENCE OPTIMIZATION

The round closes with an offline update on $\mathcal{P}_k$ against a reference policy $\pi_{\mathrm{ref},k}$ frozen at the policy that started the round. Let $\ell_\theta(a \mid s)$ be the log-probability that π_θ assigns to the supervised tokens of a completion (its reasoning, state update, and action, with prompt tokens masked), averaged over those tokens (Appendix F), and let $\ell_{\mathrm{ref},k}$ be the same quantity under the reference. The reference-adjusted margin of a pair is

$$\Delta_\theta = \big[\ell_\theta(a^+ \mid s) - \ell_{\mathrm{ref},k}(a^+ \mid s)\big] - \big[\ell_\theta(a^- \mid s) - \ell_{\mathrm{ref},k}(a^- \mid s)\big]. \tag{2}$$

Label-smoothed DPO (Rafailov et al., 2023; Mitchell, 2023) increases this margin,

$$L_{\mathrm{cDPO}} = -(1-\varepsilon)\log\sigma(\beta\Delta_\theta) - \varepsilon\log\sigma(-\beta\Delta_\theta), \tag{3}$$

where the smoothing rate ε treats each label as wrong with probability ε, since acceptance is a validator's judgment rather than ground truth. Because this term is purely relative, a likelihood term $L_{\mathrm{NLL}} = -\ell_\theta(a^+ \mid s)$ anchors the chosen response, and a token-averaged KL penalty L_{KL} to $\pi_{\mathrm{ref},k}$ over both completions limits drift from the policy whose failures produced the pairs. The objective of a round is the weighted sum of the three terms,

$$L_{\mathrm{RePair}} = L_{\mathrm{cDPO}} + \alpha_{\mathrm{NLL}} L_{\mathrm{NLL}} + \lambda_{\mathrm{ref}} L_{\mathrm{KL}}. \tag{4}$$

3 EXPERIMENTS

The experiments ask three questions, each removing more human or teacher effort than the last: whether rubric-guided preference learning improves a tool-using policy when the codebook is written by hand (RQ1), whether the codebook can instead be induced automatically from the learner's own failures (RQ2), and whether the learner can also supply the repairs, removing the external teacher from the loop (RQ3). Section 3.2 answers them, and §4 locates the gain, tests whether part of the codebook suffices, and measures what the pipeline costs.

3.1 SETUP

Task and data. BrowseComp-Plus (Chen et al., 2025) asks an agent to answer a question over a fixed corpus by searching and reading across many turns, through three tools: `search` returns ranked chunks for a keyword query, `get_document` returns evidence that a reader model extracts from one document with respect to the agent's stated goal, and `finish` commits an answer; a rollout ends at `finish` or when the context budget is exhausted, and a hard question can take tens of tool calls. Each question comes with a reference answer. We use 830 questions split 199/465/166: the 199 warm-start questions serve only for supervised initialization, all failure pools and preference pairs come from rollouts on the 465 training questions, and the 166 test questions are held out. Retrieval is BM25 over the official corpus, the reader behind `get_document` is `deepseek-v4-flash`, and rollout and decoding settings are listed in Appendix A. Failure pools use one rollout per training question for the teacher-repair arms and four for self-repair.

Warm start. Qwen3-14B and Qwen3-8B (Yang et al., 2025) are each fine-tuned on the same 573 teacher trajectories (8,168 turns) collected by `deepseek-v4-pro` in the same environment: 199 on the warm-start split and 374 on three auxiliary document-search corpora (a patent-search corpus and two web corpora; 150, 149, and 75 trajectories), none of whose questions occurs in the test split. The warm start teaches the tool interface and basic task-solving behavior (Table 1); the resulting policy π_0 at each scale is the starting point and the reference for every arm, so all comparisons at a scale share one initialization.

Pair construction. We use `kimi-k3` for diagnosis, per-cluster rewriting, reassignment, and teacher repair, `gpt-5.6-sol` for the final codebook consolidation, and `deepseek-v4-flash` as the validator. The outcome judge, `gpt-oss-120b`, labels training rollouts to define the failure pools and scores test rollouts. The prompts are in Appendix E; per-arm pool sizes, assignments, and accepted pairs are in Appendix B.

Table 1: Base models and warm start on the BrowseComp-Plus test split.

Model	mean@3	Pass@3	Turns	CoT	Search	GetDoc
Qwen3-14B (base)	11.65	21.08	5.75	3,759	3.49	1.27
π_0, 14B (SFT warm start)	42.57	56.02	33.44	6,132	27.86	4.58
Qwen3-8B (base)	9.24	21.08	6.20	4,450	3.95	1.25
π_0, 8B (SFT warm start)	38.55	53.01	33.35	5,935	28.19	4.16

Table 2: Main results on the BrowseComp-Plus test split.

Method	Rub.	Pairs	mean@3	Pass@3	Turns	CoT	Search	GetDoc
Qwen3-14B								
π_0 (SFT warm start)	–	–	42.57	56.02	33.44	6,132	27.86	4.58
Hand-written codebook	3	3,713	50.80 (+8.23)	65.06	29.62	9,146	22.78	5.83
Induced codebook	25	2,097	**53.21** (+10.64)	**66.87**	32.20	5,855	26.64	4.57
Self-repair, round 1	25	6,239	46.99 (+4.42)	60.24	34.44	4,874	29.29	4.15
Self-repair, round 2	25	7,064	49.80 (+7.23)	63.25	35.26	4,189	29.94	4.32
Qwen3-8B								
π_0 (SFT warm start)	–	–	38.55	53.01	33.35	5,935	28.19	4.16
Hand-written codebook	3	6,128	**44.38** (+5.83)	**62.05**	27.72	8,672	20.75	5.97
Induced codebook	14	4,200	43.78 (+5.23)	59.04	32.49	5,938	26.80	4.69

Rub.: rubrics in the codebook; **Pairs**: pairs used for training; parentheses: gain over the same-scale π_0; tinted: RePair arms. Appendix G gives three-seed results for the 14B RePair arms and paired bootstrap intervals.

Training and evaluation. Every RePair arm updates all parameters of π_0 with the objective of Eq. 4 for four epochs; a pair whose rendered input and longer completion exceed 20,480 tokens is dropped, and the pair counts we report are those within this budget (hyperparameters in Appendix F). At test time each of the 166 questions is run three times at the rollout decoding settings, and a run that fails to complete counts as incorrect. We report **mean@3**, the judged accuracy averaged over the three runs, and **Pass@3**, the fraction of questions answered correctly at least once, both in percent. Because the method targets behavior, we also track turns, `search` and `get_document` calls per trajectory, and reasoning length in characters per turn.

3.2 Main results

Three hand-written rubrics add 8.2 points (RQ1). Inspecting failures of the 14B π_0 on the training split yielded three rubrics, one per recurring pattern: QUERY-CHURN (the agent rephrases an unproductive query instead of changing strategy), SEARCH-WITHOUT-READ (it issues search after search without opening any candidate it has surfaced), and OVER-SPECIFIED-QUERY (it stacks quoted phrases and constraints until nothing can match). The codebook was frozen before any pair was built (Appendix C). Applied to the 185 failed trajectories from one rollout per training question, it yields 3,713 pairs within budget, and training on them raises accuracy from 42.57 to 50.80 and Pass@3 from 56.02 to 65.06 (Table 2), even though no repair was ever rolled forward. The same codebook on the 8B warm start adds 5.83 points.

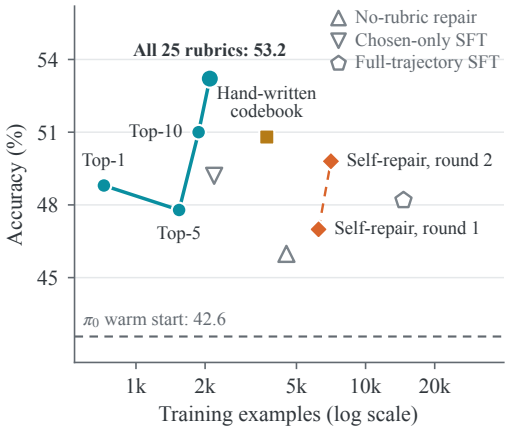


Figure 4: Accuracy against training-set size for every Qwen3-14B arm (pairs, or turns for SFT).

An induced codebook does better with fewer pairs (RQ2). Running the induction of §2.2 on the 187 failed 14B trajectories yields 25 rubrics (Appendix D) and 2,097 pairs within budget; no rule was edited by hand. This codebook reaches 53.21 accuracy and 66.87 Pass@3, 10.64 points above π_0 and

2.41 above the hand-written codebook with 44% fewer pairs (Figure 4). Across three training seeds the margin holds, 53.35 ± 0.62 against 49.40 ± 1.32 (Welch's t-test, p=0.02; Appendix G). Twenty-five rules against three is not a controlled comparison, since coverage and pairs change together, but it answers RQ2: taking the human out of rule-writing costs nothing and in fact helps. On Qwen3-8B, rerunning the pipeline from scratch on that model's own 232 failures induces a separate codebook of 14 rubrics, which adds 5.23 points with 4,200 pairs, on par with the hand-written codebook (44.38) while using a third fewer pairs.

Rubrics shape behavior. What a rule prescribes is what the policy learns to do. The hand-written codebook moves the 14B policy along exactly the axes its three rules name: with QUERY-CHURN and SEARCH-WITHOUT-READ trained in, searches per trajectory fall by 18% while document reads rise by 27%, and trajectories shorten by four turns (Figure 5, Table 2). A codebook is thus a direct lever on how an agent behaves, and the size of the shift reflects what its rules say rather than how good the policy is: the induced codebook spreads supervision over twenty-five patterns, leaves every aggregate close to π_0, and still scores higher. Section 4.2 tests which rubrics carry the gain.

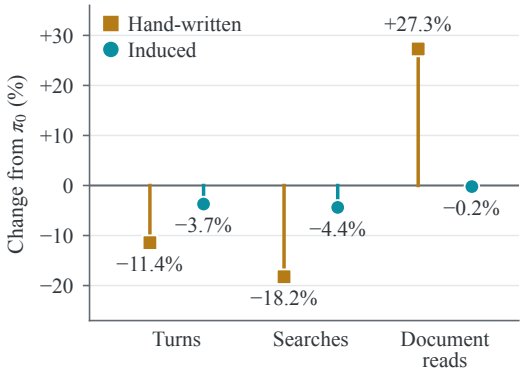


Figure 5: Behavior change from π_0 under each codebook (Qwen3-14B).

The learner's own repairs recover 7.2 of the 10.6 points in two rounds (RQ3). Self-repair keeps the induced codebook and its external diagnosis, labeling, and validation, but lets the learner propose the replacement at each flagged turn with its own rollout decoding settings. The accepted responses come from the learner, but not from its ordinary rollout distribution: they are the samples that a rubric steered and a validator accepted, which leaves room for improvement even though the learner writes both sides of each pair. From four rollouts per training question, 59% of the learner's assignments yield an accepted pair against 80% of the teacher's, giving 6,239 pairs within budget; the first round reaches 46.99 accuracy and 60.24 Pass@3, 4.42 points above π_0. The second round starts from the round-1 policy as both policy and reference, collects fresh rollouts on the same questions, and is built only from the new failures under the same codebook: 66% of its assignments now yield a pair, 7,064 pairs fit the budget, and accuracy rises to 49.80 with 63.25 Pass@3, 2.81 points over round 1 and within a point of the hand-written codebook repaired by the teacher. Across three seeds, round 2 averages 50.14 ± 1.14 against 46.25 ± 0.95 (p=0.01). Self-repair needs about three times as many pairs as teacher repair, and its policy still searches more (Table 2).

4 Analysis

4.1 Where does the gain come from?

RePair supervises the learner's own failed turns with a rubric-guided teacher repair, scored against the learner's response. Table 3 removes one of these elements at a time from the same warm start and teacher, and Figure 4 plots every 14B arm against the size of its training set: no arm trained on more data matches RePair.

The rubric carries most of the gain. Without it, the same teacher repairing the same failed turns (checked by the same validator, minus the criterion) yields more pairs, 4,521 against 2,097, but recovers only 3.41 of the 10.64 points, and its trajectories stretch to 52.9 turns. Given the

Table 3: Removing one element of RePair at a time (Qwen3-14B, induced codebook).

Arm	Data	mean@3	Pass@3
RePair	2,097	**53.21**	**66.87**
rubric-guided teacher repair vs. learner's response			
No-rubric repair	4,521	45.98 (−7.23)	58.43
same failed turns, unguided teacher, no criterion			
Full-trajectory SFT	14,624	48.19 (−5.02)	60.24
imitates the teacher's successful trajectories			
Chosen-only SFT	2,190	49.20 (−4.01)	63.25
imitates the accepted repairs, no rejected side			
Rubrics in prompt	–	40.96 (−12.25)	53.61
codebook in π_0's prompt at test time, no training			

Data: DPO pairs or SFT turns.

same teacher and the same failures, unguided repair recovers less than a third of the gain: the rubric, not the teacher, turns a local repair into useful supervision.

Supervising the learner's own failed states beats imitating the teacher. Full-trajectory SFT imitates 14,624 turns of successful teacher trajectories, seven times RePair's pair count, and reaches 48.19, half the gain. Successful teacher paths never visit the states the learner reaches after its own mistakes, where every RePair pair is built, and the imitated policy inherits the teacher's behavior rather than a repaired version of its own: 2.43 document reads per trajectory against 4.57.

The preference objective preserves reasoning that imitation collapses. Chosen-only SFT trains on the same accepted repairs and reaches 49.20. Its reasoning shrinks to 2,136 characters per turn, near the 1,932 of the corrections it imitates; RePair on the same corrections keeps 5,855 characters. Scoring each repair against the learner's own rejected response teaches the decision without copying the teacher's style.

Rules must be trained, not prompted. Written into π_0's prompt, the hand-written and induced rubrics reach 46.18 and 40.96; used to build training pairs, they reach 50.80 and 53.21.

4.2 No small subset of the codebook reproduces its gain

Assignments are concentrated: FAILED-SEARCH-REPEAT, the top induced rubric, takes 1,379 of the 3,782 assignments, and the ten most frequent rubrics cover 89% of the pairs within budget. Retraining on the pairs of the top 1, 5, and 10 rubrics alone, with the same recipe and no new repairs, gives 48.80, 47.79, and 51.00 (Table 4); the remaining fifteen rubrics, 223 pairs in all, bring the policy to 53.21. No frequent subset reproduces the full codebook: a codebook has to cover the failure modes, not only the frequent ones.

Table 4: Retraining on the pairs of the most frequently assigned rubrics (Qwen3-14B).

Rubrics	Pairs	mean@3	Pass@3
Top-1	725	48.80	62.65
Top-5	1,541	47.79	61.45
Top-10	1,874	51.00	65.06
All 25	2,097	**53.21**	**66.87**

4.3 Cost

Pairs are built from single turns, not rollouts. Table 5 breaks down the cost of one round of the induced-codebook arm. For comparison, it also counts the rollouts that outcome-based GRPO (Shao et al., 2024) would generate in the same environment under a recent search-agent recipe, 16 questions with 8 rollouts each per step for 128 steps (Pan et al., 2026). The only full trajectories RePair generates are the 465 rollouts of the failure pool; every later stage consists of single-turn generations on recorded inputs. A repair candidate averages 1,585 generated tokens, against 58,431 for a π_0 rollout, so one rollout costs as much as 37 repair attempts, and validation judges each candidate in place rather than executing it. Including those rollouts, a round generates 67M tokens, about as much as nine GRPO steps; the 128-step recipe generates fourteen times as much, and Search-R1 trains GRPO for 500 steps (Jin et al., 2025). The comparison is not accuracy-matched, and most RePair tokens come from models larger than the 14B policy. Training on the 2,097 pairs took 82 accelerator-hours on 16 H200s, and each pair is reused over four epochs, whereas a policy-gradient step uses each rollout only once. Self-repair has the same per-turn cost, with the learner writing the candidates.

Table 5: Token cost of one RePair round and of GRPO rollouts (Qwen3-14B, induced codebook).

	Tokens (M)	
Stage	generated	prefill
Failure pool (465 π_0 rollouts)	27.3	188.5
Codebook induction	6.8	40.1
Rubric assignment	4.0	17.0
Repair candidates (teacher)	13.1	112.9
Validation	16.2	67.1
RePair, one round	67.4	425.5
GRPO, one step (128 rollouts)	7.5	51.8
GRPO, 128 steps	957	6,630

Priced from measured π_0 rollouts (58,431 generated and 404,681 prefill tokens each). Judge calls fall in the failure-pool row, reader calls are excluded, and GRPO rows count rollouts only.

5 Related Work

Post-training for agentic search. Search agents are commonly trained with on-policy RL under an outcome reward (Jiang et al., 2025; Jin et al., 2025; Gao et al., 2025). Rollouts can exceed 100 tool calls (Gao et al., 2025), and a terminal reward assigns every action in a failed rollout the same negative advantage as the decisive mistake (Tao et al., 2026). Turn-level credit schemes (Wei et al., 2025; Tao et al., 2026) assign reward or credit to individual turns but still learn through full rollouts, and a scalar credit never says what the turn should have done instead, which is exactly the contrast a preference pair provides. Supervised fine-tuning on expert trajectories provides dense supervision, but on the expert's states rather than the learner's (Ross et al., 2011). Agent variants of step-level preference learning (Lai et al., 2024) take the expert's own step as the preferred action (Xiong et al., 2024; Chen & Yuille, 2026) or accept one that leads to a successful trajectory or continuation (Song et al., 2024; Li et al., 2026). Each such judgment is instance-level; RePair accepts a repair by a local check against a reusable rule, which requires no rollout and applies to every trajectory in which the pattern recurs. LEAP (Choudhury & Sodhi, 2024) and Agent-R (Yuan et al., 2025) also correct the learner's own trajectories, but one instance at a time.

Rubrics in training. Rubrics are mostly used in training as rewards that score responses (Gunjal et al., 2025; Huang et al., 2025; Shao et al., 2026), or as scores from which preference pairs are formed: AdaRubric generates scoring dimensions from a task description and pairs whole trajectories by score margin (Ding, 2026). In these approaches the rubric ranks responses that already exist, so the chosen response is bounded by what the policy samples; a RePair rubric instead locates a failed turn, guides the construction of a replacement, and accepts or rejects it. Constitutional AI also trains on revisions made under hand-written principles (Bai et al., 2022). Closest in spirit is RuscaRL, which places checklist rubrics in the task instruction during rollouts and decays them so that the model internalizes the pattern (Zhou et al., 2025); its rubrics describe what an answer should contain, whereas ours describe a recurring error of the learner, fire only where that error occurs, and validate the repair. Rules of the latter kind have also been distilled from experience but applied in the prompt or at decoding time without weight updates (Shinn et al., 2023; Zhao et al., 2024; Fu et al., 2024; Wu et al., 2025); RePair trains the rule into the policy, so no rule is needed at inference.

6 Limitations

Our evidence comes from one setting. All experiments use deep-research search on BrowseComp-Plus, a single and comparatively simple tool environment with three tools, and two sizes of one open model family, each from a supervised warm start. Further evaluation in richer tool environments and across additional model families would help clarify how broadly the findings generalize. Induced rubrics are written by a model from a sample of failures and inherit some of that model's uncertainty, so their content and emphasis may vary across failure samples and induction runs. Examining rubric consistency across repeated samples and different inducing models would help characterize this variation and its influence on subsequent repair.

7 Conclusion

RePair turns the recurring errors of a tool-using agent into reusable behavioral rubrics and uses them to build turn-level preference pairs at the learner's own failed states, with local validation instead of rollouts. On BrowseComp-Plus, a codebook induced automatically from failed rollouts lifts a supervised Qwen3-14B agent by 10.6 points with about two thousand pairs, outperforms a hand-written codebook, and carries over to repairs written by the learner itself; removing the rubric from the same pipeline loses most of the gain. Rubrics that prescribe behavior, rather than score it, are a compact and inspectable form of supervision for long-horizon agents.

AI use statement

In this work, we used generative AI tools to generate synthetic datasets, to provide feedback on the research methodology and experiments, and to clean and reformat datasets. All training data of

RePair are produced by large language models as part of the method itself: the warm-start trajectories are collected by a teacher model, and every preference pair is constructed by the diagnosis, consolidation, labeling, repair, and validation pipeline of §2, run by the models listed in §3.1 or, in self-repair, by the learner itself, with the outcome judge defining the failure pools. General-purpose AI assistants provided feedback on the design of the method and the experiments, and helped clean and reformat the datasets used in our experiments. We have not used generative AI tools to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to implement methods, to assist with translation, to support qualitative or thematic data analysis, or to interpret results; formulating mathematical claims, providing critical ingredients for their proofs, and assisting in the writing of proofs are not applicable to this work.

We have reviewed all AI-assisted work. Synthetic preference pairs enter training only through the screens that are part of the method, the programmatic checks and validator of §2.3 and Appendix B; all three codebooks are reproduced in full in Appendices C and D, and the authors inspected the codebooks, the failure pools, and samples of the accepted pairs. All methodological and experimental decisions were made by the authors, and every cleaned or reformatted dataset was checked by the authors before use. We take responsibility for the final content of this work, including the text, claims, and artifacts produced with the aid of generative AI.

Reproducibility statement

The code for the harness, pair construction, training, and evaluation is at the anonymous repository `https://anonymous.4open.science/r/RePair-DFC9`. Its README links the data of the main experiments: the question splits, the codebooks, the warm-start trajectories, the preference pairs of each arm, and the BM25 index over the BrowseComp-Plus corpus. The environment, rollout, and evaluation settings are given in Appendix A, the construction pipeline and its checks in Appendix B, all three codebooks in Appendices C and D, all prompts in Appendix E, and the optimization hyperparameters in Appendix F.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022. doi: 10.48550/arXiv.2212.08073. URL https://arxiv.org/abs/2212.08073.

Yixiong Chen and Alan Yuille. Agentic-DPO: From imitation to agentic policy optimization on expert trajectories. *arXiv preprint arXiv:2607.10601*, 2026. doi: 10.48550/arXiv.2607.10601. URL https://arxiv.org/abs/2607.10601.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025. doi: 10.48550/arXiv.2508.06600. URL https://arxiv.org/abs/2508.06600.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. *arXiv preprint arXiv:2410.05434*, 2024. doi: 10.48550/arXiv.2410.05434. URL https://arxiv.org/abs/2410.05434.

Liang Ding. AdaRubric: Task-adaptive rubrics for reliable LLM agent evaluation and reward learning. *arXiv preprint arXiv:2603.21362*, 2026. doi: 10.48550/arXiv.2603.21362. URL https://arxiv.org/abs/2603.21362.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. *arXiv preprint arXiv:2403.08978*, 2024. doi: 10.48550/arXiv.2403.08978. URL https://arxiv.org/abs/2403.08978.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025. doi: 10.48550/arXiv.2508.07976. URL https://arxiv.org/abs/2508.07976.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025. doi: 10.48550/arXiv.2507.17746. URL https://arxiv.org/abs/2507.17746.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025. doi: 10.48550/arXiv.2508.12790. URL https://arxiv.org/abs/2508.12790.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. DeepRetrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*, 2025. doi: 10.48550/arXiv.2503.00223. URL https://arxiv.org/abs/2503.00223.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. doi: 10.48550/arXiv.2503.09516. URL https://arxiv.org/abs/2503.09516.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-DPO: Step-wise preference optimization for long-chain reasoning of LLMs. *arXiv preprint arXiv:2406.18629*, 2024. doi: 10.48550/arXiv.2406.18629. URL https://arxiv.org/abs/2406.18629.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. Verified critical step optimization for LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. doi: 10.48550/arXiv.2602.03412. URL https://arxiv.org/abs/2602.03412.

Eric Mitchell. A note on DPO with noisy preferences & relationship to IPO, 2023. URL https://ericmitchell.ai/cdpo.pdf.

Zhuoshi Pan, Qizhi Pei, Junru Lu, Honglin Lin, H. Vicky Zhao, Di Yin, and Xing Sun. ContextPilot: Teaching agents for proactive context management via fine-grained RL. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. doi: 10.48550/arXiv.2608.28476. URL https://arxiv.org/abs/2608.28476.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2338. URL https://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen-tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR Tulu: Reinforcement learning with evolving rubrics for deep research. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2511.19399. URL https://arxiv.org/abs/2511.19399.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2403.02502.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents. *arXiv preprint arXiv:2607.13988*, 2026. doi: 10.48550/arXiv.2607.13988. URL https://arxiv.org/abs/2607.13988.

Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via fine-grained reward structure and credit assignment. *arXiv preprint arXiv:2505.11821*, 2025. doi: 10.48550/arXiv.2505.11821. URL https://arxiv.org/abs/2505.11821.

Chunlong Wu, Ye Luo, Zhibo Qu, and Min Wang. Meta-Policy Reflexion: Reusable reflective memory and rule admissibility for resource-efficient LLM agent. *arXiv preprint arXiv:2509.03990*, 2025. doi: 10.48550/arXiv.2509.03990. URL https://arxiv.org/abs/2509.03990.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://arxiv.org/abs/2406.11176.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025. doi: 10.48550/arXiv.2501.11425. URL https://arxiv.org/abs/2501.11425.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2308.10144.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, Hengtong Lu, Wei Chen, Yan Xie, and Mingli Song. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general LLM reasoning. *arXiv preprint arXiv:2508.16949*, 2025. doi: 10.48550/arXiv.2508.16949. URL https://arxiv.org/abs/2508.16949.

A Environment, rollout, and evaluation settings

Retrieval. `search` scores the official BrowseComp-Plus corpus with BM25 (Lucene scoring, k_1=1.5, b=0.75, English stop words and stemming) over 969,157 chunks of 100,195 documents; documents are split at headings into 1,024-token chunks with an overlap of 128 tokens, each prefixed with its heading. A search returns the five top-ranked chunks with snippets of at most 256 tokens; when a result is later compacted into the action history, its snippet is cut to 250 characters. `get_document` passes up to 24,000 tokens of the parent document to the reader, which extracts the evidence entries described below.

Context construction. The rendering f of §2 rebuilds the policy input at every turn from a static system prompt and eight fields: the goal; the required output format of `finish`; the satisfied constraints, each with the document identifiers that support it, and the unsatisfied constraints, both maintained by the policy through its state update; the action history, in which the harness lists every past action with a compacted result; the evidence, the facts with document identifiers that the reader extracted from opened documents, at most 300 characters each and 24 entries in total, with the oldest entries not cited by a satisfied constraint evicted first; the policy's notes, at most 400 characters; and the last observation, the full result of the most recent tool call, carried into the next turn only. Under this rendering the input of a test turn of π_0 has a median of 11,384 tokens, a 95th percentile of 24,147, and a maximum of 28,685 (system prompt and chat template included); the hand-written and induced arms have medians of 10,376 and 10,863 tokens with the same upper tail.

Rollout settings. Table 6 lists the settings used to collect failure pools on the training split and to evaluate on the test split. Apart from the questions and the number of rollouts, the two differ only in the per-generation token caps of the policy and the reader and in the wall-clock limit of an evaluation job. The judge, `gpt-oss-120b`, runs at temperature 0 and top-p 1 and is given the final answer only. Its prompt (Appendix E) asks whether the extracted final answer is semantically equivalent to the reference and names the same answer: differences in capitalization, whitespace, diacritics, non-semantic punctuation, and common spelling or transliteration variants are ignored; additional detail is allowed only when it is correct and does not contradict the answer; a different entity, an ambiguous answer, or a substantive inconsistency is incorrect; and numerical answers may differ by a small margin. The grounding check on a candidate final answer (§2.3) is programmatic: of the words of at least four characters in the answer, at least half, and at least one, must occur in the recorded input of the turn. A policy turn that reaches the context limit is regenerated up to five times with a fresh seed; the repetition penalty stays at 1.1 for the first two retries and rises to 1.2, 1.3, and 1.4 for the remaining three. Evaluation uses seed 12345. A run whose retries are exhausted, or that exceeds the six-hour wall-clock limit of an evaluation job, is scored as incorrect and excluded from behavior averages; this affects at most nine of the 498 runs of every policy except the 8B hand-written arm, 63 of whose runs did not complete.

Table 6: Rollout settings for failure-pool collection and test evaluation.

Setting	Failure-pool collection	Test evaluation
Questions	465 training	166 test
Rollouts per question	1 (self-repair: 4)	3
Policy tokens per generation	4,096	no cap
Reader tokens per generation	10,240	no cap
Job wall-clock limit	none	6 hours
Shared		
Policy decoding	thinking on; temperature 1.0, top-p 0.8, repetition penalty 1.1	
Context window	32,768 tokens	
Forced finish	once the policy input reaches 26,000 tokens	
Turn cap	200 turns	
Truncation retries	5	
Reader (`get_document`)	`deepseek-v4-flash`, thinking on, temperature 0	
Outcome judge	`gpt-oss-120b`, temperature 0, final answer only	

A generation with no token cap is bounded only by the context window.

B Data construction details

This appendix follows the pipeline of §2 stage by stage. Each stage is a separate model call with its own prompt, made by the models listed in §3.1; the diagnosis, reassignment, and consolidation prompts are reproduced in Appendix E.

Failure pools. For the 14B teacher-repair arms, π_0 ran once on each of the 465 training questions and the outcome judge marked 188 rollouts incorrect. A model API's content filter blocked one

of these trajectories, leaving 187 for induction and reassignment; the hand-written arm lost two more the same way and used 185. The 8B π_0 likewise ran once per question; 7 of its 239 incorrect rollouts were blocked, leaving 232, on which both 8B arms run. The 8B induced arm builds its own codebook from these 232 rather than reusing the 14B one. For self-repair, π_0 ran four times per question (1,860 rollouts); 821 failed rollouts entered reassignment, and 805 received at least one assignment. In round 2 the round-1 policy ran the same four rollouts per question, and 798 failed rollouts entered reassignment under the same codebook after 15 were blocked. Trajectories that receive no verdict from the outcome judge are discarded; eligible turns are defined in §2.1.

Diagnosis. The diagnosis model reads the question, the complete failed trajectory, its final answer marked as incorrect, the list of eligible turns, and the policy's input at the last turn. It returns a bounded number of findings, ordered by its judgment of how much each mistake mattered. A finding names a turn and gives an issue name, a trigger, a hint, and an acceptance criterion, all phrased generically. Specificity checks reject a finding that contains a term from the question, a quoted phrase, a date, a long number, or an identifier-like string. Findings serve only to build the codebook: the turns a rubric is applied to are chosen afterwards by reassignment, which the finding budget does not limit.

Clustering and consolidation. Findings from the whole failure pool are grouped by textual similarity. The text of a finding, its issue name followed by its hint, is vectorized with word unigram and bigram and character 3- to 5-gram TF–IDF features (sublinear term frequency), reduced to 128 dimensions by truncated SVD, ℓ_2-normalized, and partitioned with K-means (K=12, ten initializations, findings weighted by the frequency of their issue name). `kimi-k3` rewrites each cluster into draft rules, and `gpt-5.6-sol` consolidates the drafts of all clusters into the final codebook, merging duplicates; the specificity checks are applied again to the surviving rules. For the 14B pool, 938 findings with 683 distinct issue names formed 12 clusters, 47 draft rules, and 25 final rules; for the 8B pool, 1,000 findings with 702 issue names formed 12 clusters, 45 drafts, and 14 rules.

Reassignment. The labeling model reads each failed trajectory with the codebook and assigns to each eligible turn whichever rubrics fire there; a hand-written codebook is applied likewise. A turn may receive several rubrics: an assignment is a (trajectory, turn, rubric) triple, each is repaired and validated on its own, and each accepted assignment becomes its own pair. In the 14B induced arm the 3,782 assignments fall on 3,160 distinct turns, 556 of which carry more than one rubric.

Repair. Repair poses a new decision at the recorded input s_t of the assigned turn, reused verbatim rather than re-rendered, with the rubric's trigger and hint appended to its system prompt (Eq. 1); both completions of the resulting pair are scored against s_t without them. A teacher candidate is sampled at temperature 1.0 and top-p 0.95 with at most 12,288 generated tokens. In self-repair the learner writes the candidates at its rollout decoding settings, with the same guidance and the same external validation. At most four candidates are drawn per assignment, and the first to pass validation is kept; an assignment with no accepted candidate yields no pair.

Validation. Programmatic checks reject a candidate with invalid structure, a missing action, reasoning longer than 10,000 characters, or prompt scaffolding echoed into its text. A candidate with a nonterminal action is also rejected if it leaks answer content, repeats the original action, or cites a document identifier absent from s_t, and a final-answer candidate must match the reference answer and be grounded in s_t (Appendix A). The validator model then reads s_t and compares the original and candidate actions against the rubric's trigger and acceptance criterion, returning two binary decisions: whether the flagged error is repaired, and whether the replacement is a natural decision at that turn. A candidate is accepted only if both are yes. The validator is `deepseek-v4-flash` for every arm.

No-rubric repair. The no-rubric arm of §4.1 repairs the same failed turns as the induced-codebook arm under the learner's original prompt, with no rubric, hint, or reference answer. Its candidates pass the same programmatic checks and the same validator, whose input omits the rubric criterion.

Teacher trajectories for full-trajectory SFT. `kimi-k3` attempts each of the 465 training questions four times in the environment of Appendix A, and the outcome judge marks 1,534 of

the 1,860 trajectories correct. Every well-formed turn of a correct trajectory is a training example, rendered like a policy turn; 14,624 of the 14,989 turns fit the 20,480-token limit.

Construction funnel. Table 7 gives, for each arm, the size of the failure pool, the number of rubric assignments on its eligible turns, the accepted pairs, and the pairs within the training length budget.

Table 7: Construction funnel per arm.

Arm	Model	Pool	Assignments	Accepted	In budget	CoT$^+$	CoT$^-$
Hand-written codebook	14B	185	6,507	4,982	3,713	981	5,065
Induced codebook	14B	187	3,782	3,021	2,097	1,932	5,356
Self-repair, round 1	14B	821	16,412	9,745	6,239	4,604	5,562
Self-repair, round 2	14B	798	16,385	10,842	7,064	4,028	4,627
Chosen-only SFT	14B	187	3,782	3,021	2,190	1,932	–
Full-trajectory SFT	14B	1,534	–	14,989	14,624	1,568	–
No-rubric repair	14B	187	–	6,702	4,521	1,683	5,635
Top-1 rubric	14B	187	1,379	1,132	725	–	–
Top-5 rubrics	14B	187	2,713	2,214	1,541	–	–
Top-10 rubrics	14B	187	3,390	2,713	1,874	–	–
Hand-written codebook	8B	232	9,798	8,040	6,128	777	5,394
Induced codebook (14 rubrics)	8B	232	6,869	5,589	4,200	1,115	5,759

Pool: failed trajectories processed (successful teacher trajectories for full-trajectory SFT); **CoT$^+$** and **CoT$^-$**: mean reasoning characters of the chosen and rejected responses over accepted pairs, or of the imitated turns for SFT. Rubric subsets reuse the induced arm's pairs; the no-rubric arm has no assignments.

C HAND-WRITTEN CODEBOOK

The three hand-written rubrics of §3.2 are listed below in the form used during pair construction: for each rubric, the trigger γ that the labeling model matches against an eligible turn, the hint η appended to the system prompt during repair, and the acceptance criterion κ that the validator checks. Each entry also reports the number of eligible turns to which the labeling model applied the rubric and the accepted pairs it produced, with the pairs within the training length budget.

QUERY-CHURN — 4,625 assignments; 3,740 accepted, 2,712 within budget

Trigger. The agent issues a search that rewords, reshuffles, or trivially re-cases an earlier query which already returned unhelpful results, keeping the same intent and target rather than changing strategy; no newly surfaced information motivates the rewrite. It is not merely stacking too many constraints (that is over-specified) and not ignoring an unread surfaced result (that is search-no-read). **Hint.** Your recent searches are rewordings of the same query and it is not working. Do not rephrase it again — change strategy: search a different sub-question of the task, or open one of the results you already have. **Criterion.** The agent rephrased an earlier query that was not working instead of changing strategy. A valid repair abandons the failing wording: it searches a genuinely different sub-question or aspect of the task, or opens an already-surfaced result instead of searching again.

SEARCH-WITHOUT-READ — 1,367 assignments; 957 accepted, 766 within budget

Trigger. The agent issues another search while one or more already-surfaced results plausibly relevant to an unresolved constraint remain unopened, continuing to search instead of calling get_document on candidates it has already retrieved. The fault is not rewording a prior query nor over-constraining it. **Hint.** You have been searching repeatedly without reading. Some results already returned are likely to contain what you need. Stop searching and open the most promising document you have already surfaced with get_document. **Criterion.** The agent kept issuing searches without reading any of the candidates it had already surfaced. A valid repair stops searching and opens one of the already-surfaced documents with get_document, choosing one plausibly relevant to an unresolved constraint.

OVER-SPECIFIED-QUERY — 515 assignments; 285 accepted, 235 within budget

Trigger. The agent's search stacks multiple quoted phrases and/or many simultaneous constraints so the query is too narrow for any document to match; the primary fault is excessive specificity in a single query, distinct from merely rewording a prior failure or from leaving surfaced results unread. **Hint.** Your query stacks too many exact constraints, so nothing matches. Drop the quotes and the least essential terms, and search for

the one or two most distinctive keywords instead. **Criterion.** The query stacked quoted phrases or so many constraints at once that nothing relevant could match. A valid repair simplifies: it keeps only the one or two most distinctive terms, drops the rest, and does not just reshuffle the same over-constrained wording.

D INDUCED CODEBOOK

The 25 rubrics induced from the 14B failure pool in §3.2 are listed below in the same form, exactly as they left the final consolidation step; none was edited afterwards. The codebook of 14 rubrics induced separately from the 8B failure pool follows them. Ranked by how often reassignment applied them, the five most frequent rubrics are FAILED-SEARCH-REPEAT, QUOTED-QUESTION-PROSE, DERIVED-CONSTRAINT-QUERY, UNGROUNDED-CANDIDATE-DETAIL, and REFUTED-CANDIDATE-PURSUIT, and the next five are ACTIONABLE-RESULT-IGNORED, EXTRACTION-GOAL-OVERREACH, DISTORTED-QUESTION-CONSTRAINT, REDUNDANT-DOCUMENT-READ, and PLAN-ACTION-MISMATCH (§4.2). The listing also reports, for each rubric, the number of eligible turns to which reassignment applied it and the number of accepted pairs it produced.

QWEN3-14B CODEBOOK (25 RUBRICS)

FAILED-SEARCH-REPEAT

1,379 assignments; 1,132 accepted, 725 within budget

Trigger. After an unhelpful result, the agent repeats the same effective retrieval request with no changed information need, evidence, or plausible retrieval mechanism. Similar wording alone is insufficient: a synonym, phrase boundary, or scope change may retrieve different evidence, and a prior tool failure may justify a retry. **Hint.** Check what the earlier request actually established. Change vocabulary, scope, source type, or evidence target for a concrete reason, or inspect a relevant result. Repeat a request only when a changed condition or failed execution explains why another attempt could help. **Criterion.** The action addresses an unresolved need with a plausible change in retrieval or reading, or gives a concrete reason to retry. A synonym change may qualify; a cosmetically novel query with no plausible information benefit does not. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

QUOTED-QUESTION-PROSE

461 assignments; 396 accepted, 338 within budget

Trigger. The search treats a long descriptive or analytic phrase from the question as exact source wording without a reason to expect that wording, making the query unnecessarily restrictive. Quoting a title, proper name, distinctive source phrase, or explicitly testing a plausible wording is not itself an error. **Hint.** Translate descriptive clues into likely source vocabulary. Keep useful names and discriminative terms; use quotes when exact wording has a clear purpose. A phrase from the question can be tested as a hypothesis, but failure of that phrase does not establish absence of the underlying fact. **Criterion.** The query removes an unjustified wording restriction while retaining useful clues, or the action tests a relevant lead. Exact phrases are allowed when their purpose is clear; merely removing all quotes is not sufficient. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

DERIVED-CONSTRAINT-QUERY

358 assignments; 276 accepted, 204 within budget

Trigger. The agent mistakes a derived count, threshold, date bound, or range for a fact sources must state verbatim, or promotes a sampled value within a range into an established constraint. Searching a clearly tentative value or subset is not itself an error. **Hint.** Retrieve underlying facts and apply the question's calculation or range yourself. A focused search may test one value or subset, provided it remains a search hypothesis and other allowed values stay possible. Use explicit source numbers when they are meaningful retrieval anchors. **Criterion.** The action retrieves underlying facts or tests a clearly tentative value or subset. Reasoning and state retain the original range and its open or closed boundaries; failure of one test does not eliminate untested values. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

UNGROUNDED-CANDIDATE-DETAIL

299 assignments; 235 accepted, 168 within budget

Trigger. The agent presents an unverified candidate, name expansion, date, place, or attribute as established, and relies on it to confirm a condition or exclude alternatives. Proposing a candidate from background knowledge or a range as an explicit, testable hypothesis is not an error. **Hint.** Separate candidate generation from verification. You may propose a name or detail not yet retrieved, including from background knowledge, when you label it tentative and use the next action to test it. Do not convert that proposal into evidence, a satisfied condition, or a fabricated source. **Criterion.** The completion removes the unsupported assertion or turns it into

an explicit hypothesis with a relevant verification action. Candidate hypotheses may appear in search terms, but remain unverified until evidence supports the relevant entity and relationship. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

REFUTED-CANDIDATE-PURSUIT 216 assignments; 175 accepted, 106 within budget

Trigger. Evidence tied to the correct entity and time explicitly contradicts a required condition, yet the agent treats that candidate as valid without addressing the contradiction. Missing search hits, an empty extraction, an unsupported note, or a list of uncertain completeness is not sufficient refutation. **Hint.** Separate contradiction, lack of support, and low information yield. Exclude a candidate when reliable evidence rules it out. Keep unsupported candidates uncertain, though you may deprioritize them. If sources conflict or identity or date scope is ambiguous, test that conflict before making a categorical exclusion. **Criterion.** The completion excludes a demonstrably contradicted candidate or takes a specific action to resolve a genuine evidence conflict. Mere lack of support leads to uncertainty or lower priority, not elimination; a list omission counts only when relevant completeness is established. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

ACTIONABLE-RESULT-IGNORED 179 assignments; 118 accepted, 73 within budget

Trigger. A visible document, candidate, or list offers a concrete way to resolve an uncertainty that could change the answer, but the agent bypasses it for a less relevant action or unsupported commitment without a reason. An unread result alone does not require reading when the answer is already sufficiently supported or a better test is available. **Hint.** Compare available leads by which unresolved question they can answer. Read a useful page, test a candidate, or inspect a list when that is the best next step. You may choose a better test or finish once evidence is sufficient; explain the decision without treating unread leads as already verified. **Criterion.** The action tests the useful lead, performs a justified better test of the uncertainty, or finishes on already sufficient evidence. It does not simply repeat broad retrieval or assert that the untested lead has been confirmed or refuted. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

EXTRACTION-GOAL-OVERREACH 151 assignments; 107 accepted, 71 within budget

Trigger. The requested extraction has no plausible connection to the document's visible subject or available content, or asks the reader to assume facts it has not found. A title, genre, or short snippet omitting a detail is not sufficient to prove the document cannot contain it. **Hint.** Use the page's visible scope to form a plausible extraction request, without assuming its answer. A project article can contain background facts and a biography can contain specific dates. Seek a dedicated source when the current page has no reasonable route to the needed information. **Criterion.** The goal asks for information the source could plausibly contain, or the action seeks a more suitable source. If it switches to another task, the unverified detail stays unresolved; skipping the read must not mark that condition satisfied. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

DISTORTED-QUESTION-CONSTRAINT 128 assignments; 100 accepted, 70 within budget

Trigger. The agent changes an explicit subject, relationship, temporal scope, range boundary, unit, or quantifier from the question and reasons as though the changed condition were the original one. A query focused on only some conditions, or a stated temporary interpretation of ambiguity, is not itself distortion. **Hint.** Keep each condition attached to its subject and preserve direction, time, units, and quantifiers. A query may test one part at a time. Treat ambiguous readings as alternatives to resolve, and revise candidate hypotheses rather than rewriting the question to fit them. **Criterion.** The completion preserves the question's actual conditions and tests a relevant part or explicitly unresolved interpretation. It does not silently alter a condition, attach it to another subject, or delete it merely because a favored candidate fails it. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

REDUNDANT-DOCUMENT-READ 121 assignments; 90 accepted, 57 within budget

Trigger. The agent repeats a successful read or equivalent extraction of already sufficient evidence without a new question, conflict, or reason to expect additional information. A changed goal, incomplete extraction, relevant unexamined content, or failed prior tool call may justify another read. **Hint.** Check what was actually extracted and what remains missing. Re-read for a distinct detail, to resolve a conflict, or to recover from an identified extraction problem. A prior empty result is not a refutation, but repeating the same successful empty extraction needs a reason to expect a different outcome. **Criterion.** The action obtains missing information

from a suitable source, or a reread states a concrete unmet goal or recovery reason. It does not duplicate sufficient evidence merely because the document remains salient. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

PLAN-ACTION-MISMATCH — 98 assignments; 84 accepted, 62 within budget

Trigger. The action's actual tool or arguments do not implement the agent's stated information need or intended test, and no coherent revision explains the difference. Departing from an outdated or flawed plan in favor of a better justified action is not an error. **Hint.** Compare the tool parameters with the question you intend this action to answer. Execute that test, or briefly revise the plan when current evidence makes another action preferable. Keep the action, reasoning, and any carried notes consistent. **Criterion.** The action carries out a coherent information need supported by its reasoning. It may implement the earlier plan or an explicitly justified alternative; preserving a bad plan is not required. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

IRRELEVANT-LEAD-PURSUIT — 96 assignments; 94 accepted, 70 within budget

Trigger. The agent pursues a result whose visible content has no plausible direct or intermediate connection to an unresolved need, relying only on name overlap, boilerplate, or frequency. A short snippet's failure to state the target fact is not enough to label a lead irrelevant. **Hint.** Ask what this source could plausibly establish or which useful lead it could reveal. A biography, list, or related page may provide an intermediate link without answering the whole question. Prefer a better connected source when the current result is only superficial noise. **Criterion.** The action targets a plausible direct or intermediate evidence need. Exploratory reading is acceptable with a concrete connection; lack of an answer in the snippet alone is not grounds to reject the source or its candidate. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

PARTIAL-EVIDENCE-OVERCLAIM — 59 assignments; 46 accepted, 34 within budget

Trigger. The agent promotes partial support, a suggestive match, or an uncertain inference into a fully established candidate fact or satisfied condition. A transparent inference that follows from supported premises about the same entity is valid even if no source states the conclusion verbatim. **Hint.** Track which part of a condition is supported and what still needs testing. Use valid calculations and logical combinations of evidence, making their premises clear. Keep plausible but non-entailing inferences tentative rather than silently promoting them to facts. **Criterion.** The completion narrows the claim to its support, makes a valid evidence-based inference explicit, or investigates the missing part. Reasoning, notes, citations, and satisfy updates agree on what is established; a topic match alone does not verify a relationship. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

MALFORMED-TOOL-CALL — 57 assignments; 44 accepted, 21 within budget

Trigger. The chosen tool receives missing, mistyped, or cross-wired arguments that prevent it from expressing the intended operation, such as document fields passed to search or query text passed as a document identifier. Judge validity by the actual tool contract. **Hint.** Choose the operation first: discovery, evidence extraction, or final submission. Then supply the fields and types that operation requires. Formatting a call correctly should preserve its useful purpose. **Criterion.** The replacement satisfies the selected tool's schema and serves a relevant purpose. Search has query text, get_document has a valid document handle and extraction goal, and finish has the required answer fields. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

UNVERIFIED-CONSTRAINT-FINISH — 57 assignments; 45 accepted, 37 within budget

Trigger. The agent commits while overlooking or falsely asserting resolution of a condition that could change the answer, when a feasible useful check remains. A transparent budget-required best estimate is not an error merely because some facts remain unknown, and an entailed condition need not be quoted verbatim by a source. **Hint.** Before finishing, ask which unresolved condition could change the candidate or requested field. Check it when feasible. If the evidence is sufficient, submit; if you must end with uncertainty, distinguish the best-supported estimate from a verified match and do not claim unverified conditions are satisfied. **Criterion.** The replacement checks the material uncertainty when feasible, finishes on sufficient evidence, or gives an honest budget-required estimate. Its explanation and confidence reflect actual support; a generic hedge does not justify skipping an available decisive check. Preserve question constraints and current tool/finish requirements. Keep

hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

UNSUPPORTED-CITATION-CLAIMS — 46 assignments; 29 accepted, 25 within budget

Trigger. The agent attributes a claim to visible source material that does not support it, binds evidence to the wrong entity or time, or presents an inference as a source quotation. Faithful paraphrase and valid, clearly explained inference from cited premises are allowed. **Hint.** Match each cited claim to the source passage, entity, relation, and time it actually supports. Keep quotations separate from deductions. A metadata field or snippet can support what it explicitly shows when the task permits that evidence type; it does not establish unseen body content. **Criterion.** The completion removes or narrows unsupported claims, obtains needed evidence, or explains a valid inference from the cited premises. Citations use available sources and comply with the task's evidence requirements; sharing keywords with a claim is not enough. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

LOW-SELECTIVITY-ENTITY-SEARCH — 25 assignments; 16 accepted, 13 within budget

Trigger. The agent searches using only common attributes when a clearly more discriminative clue, useful candidate test, or manageable source list is available, without a reason for broad exploration. An initial vocabulary probe or deliberate broadening after an overrestrictive query is not an error. **Hint.** Choose clues by their ability to distinguish candidates and by how sources express them. Use rare relations or a small relevant list when helpful. Broad searches remain useful for discovering terminology or recovering from an overly narrow hypothesis. **Criterion.** The action uses a discriminative clue, examines a useful list, tests a tentative candidate, or deliberately broadens retrieval for an explained information need. It need not include every constraint or always choose a roster. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

PREMATURE-NEGATIVE-FINISH — 21 assignments; 17 accepted, 12 within budget

Trigger. The agent gives a noncommittal or not-found finish despite a specific promising test that could materially improve the answer and is feasible under the visible remaining budget. Also applies when it abstains despite an already sufficiently supported answer. Unknown hypothetical strategies or a mandatory finish instruction do not by themselves justify more tool use. **Hint.** Before giving up, identify whether a concrete useful test remains and whether you can perform it. Use it when feasible; finish directly if the evidence already supports an answer. Do not try to exhaust every possible strategy, violate a finish requirement, or infer corpus-wide absence from unsuccessful retrieval. **Criterion.** The replacement performs a feasible useful test or submits an answer already sufficiently supported. If the visible budget requires ending, it reports the best-supported conclusion and remaining uncertainty without inventing a candidate or overstating what the corpus lacks. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

EXTRACTED-ANSWER-UNCOMMITTED — 8 assignments; 0 accepted, 0 within budget

Trigger. Retrieved evidence sufficiently establishes the candidate's identity and requested answer field, with no unresolved condition or credible conflict that could change the answer, yet the agent continues retrieval without a concrete verification need. One partial profile match or the answer field alone is insufficient to trigger this rule. **Hint.** Check identity, the requested field, and any remaining uncertainty that could change the answer. When these are settled, submit the supported answer instead of looking for another confirmation. Additional research is justified only by a specific unresolved issue, not by a general wish for more sources. **Criterion.** When the trigger holds, the replacement calls finish with the supported answer and a faithful explanation. It neither invents support for an unchecked condition nor keeps searching without identifying a material unresolved issue. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

INVENTED-DOCUMENT-REFERENCE — 7 assignments; 7 accepted, 2 within budget

Trigger. The agent uses a document handle that was not surfaced or otherwise made available by the tool interface, invents a placeholder, or performs an unsupported conversion between identifiers. A parent or chunk identifier conversion explicitly allowed by the interface is valid. **Hint.** Treat document identifiers as tool-provided handles. Reuse a visible handle or an explicitly supported equivalent; surface a new document through search before reading it. A candidate hypothesis does not authorize inventing a document ID. **Criterion.** The action reads an available document using a valid handle or searches to obtain the needed document. Any parent/chunk conversion follows the tool contract, and the reading goal addresses a relevant information need. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

MALFORMED-QUERY-SYNTAX 4 assignments; 4 accepted, 3 within budget

Trigger. An accidental quote, broken operator, truncation, or stray fragment makes the query fail to express the intended request under the visible search interface. Abbreviations, prefixes, names, or unusual punctuation are not errors merely because they are not ordinary full words. **Hint.** Check the literal query against the supported interface. Repair accidental damage and unsupported syntax while keeping useful technical terms, names, and intentional shorthand. Use plain keywords when an operator's behavior is unknown. **Criterion.** The action uses supported syntax and valid arguments and expresses the intended information need. A different tool is acceptable when it serves that need; a well-formed but unrelated call is not a repair. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

CONFIRMED-CLUE-RECHECKING 4 assignments; 3 accepted, 3 within budget

Trigger. The agent spends an action reconfirming sufficiently supported information with no identified reliability or identity issue, while a material uncertainty remains or the answer is ready. Checking weak support, conflicting sources, or an ambiguous entity match is not redundant confirmation. **Hint.** Direct effort toward the uncertainty most likely to change the answer. Recheck a settled-looking clue when there is a specific reason to doubt its evidence or entity binding. Otherwise investigate the remaining field or finish once the required support is sufficient. **Criterion.** The action targets a material uncertainty, resolves a stated evidence-quality issue, or submits the sufficiently supported answer. It does not seek another copy of an already adequate fact while leaving the decisive question untouched. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

NOISY-TITLE-ACCEPTANCE 3 assignments; 1 accepted, 1 within budget

Trigger. The agent treats apparent site furniture, a display artifact, or an unresolved variant as the authoritative name of a work or entity, and lets that choice distort retrieval or the answer. Testing a variant as a search hypothesis or using a supported alternate title is not an error. **Hint.** Separate names from navigation text and presentation artifacts. Compare visible forms and their surrounding context when the difference matters. Search plausible variants when useful, keeping their identity and spelling unresolved until the evidence supports a choice. **Criterion.** The completion uses a supported name, tests relevant variants, or checks the disputed form. It neither presents unverified display noise as canonical nor removes a meaningful title component merely because a shorter string looks cleaner. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

COMPOSITE-EXACT-ANSWER 3 assignments; 1 accepted, 1 within budget

Trigger. The exact-answer field combines distinct candidates or incompatible name forms so that it does not identify one answer in the requested form. A documented alias, transliteration, or original-language name for the same entity is not an error solely because it appears in parentheses. **Hint.** Choose the entity and answer form the question asks for. Resolve substantive identity differences and use a supported name. Keep uncertainty or alternatives in the explanation; include a same-entity alias in the answer only when it clarifies identity and the output format allows it. **Criterion.** The finish identifies one supported entity or value in the requested form. It does not concatenate rival answers or invent a hybrid name; a permitted, documented same-entity alias remains valid. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

EXTRACTION-GOAL-TOO-NARROW 2 assignments; 1 accepted, 1 within budget

Trigger. The agent's extraction unnecessarily excludes relevant alternatives or an essential answer field that the visible source could provide, causing a material information gap. A focused request is valid when source size, reader limits, or a deliberate staged investigation make broader extraction less useful. **Hint.** Request the entries and fields needed for the current comparison, including the answer field when available. Avoid restricting a list to an unverified favorite. For a large source, use a focused batch with an explicit unresolved remainder instead of demanding everything at once. **Criterion.** The goal covers the useful candidates and fields at a feasible scope, or explicitly stages the extraction. It does not silently drop relevant alternatives or claim the requested answer field is settled before it has been obtained. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

CONVERGENT-EVIDENCE-DISMISSAL 0 assignments; 0 accepted, 0 within budget

Trigger. Visible evidence links the same candidate across the relevant subclues, but the agent demands one all-in-one source and repeats settled identification work without a remaining uncertainty. Entity ambiguity, conflicting facts, or uncertainty about source dependence can justify further checking. **Hint.** Combine evidence across sources after checking identity, time, and the links between claims. Distinguish complementary support from repeated copies of the same claim. Use the convergent candidate as working context and focus on what could still change the answer. **Criterion.** The action addresses a material unresolved link, identity conflict, or requested field, or finishes when support is sufficient. It does not demand a single source solely to restate a chain already established across compatible evidence. Preserve question constraints and current tool/finish requirements. Keep hypotheses and unknowns explicit; neither unsupported satisfy updates nor treating missing evidence alone as a refutation is valid.

QWEN3-8B CODEBOOK (14 RUBRICS)

REPEATED-EXHAUSTED-RETRIEVAL 4,112 assignments; 3,310 accepted, 2,390 within budget

Trigger. With no visible hard contradiction, the schema-valid step repeats an exhausted retrieval: search reuses the same anchor and term family after prior calls produced only already-seen or irrelevant results, or get_document revisits a document with an extraction goal already covered. **Hint.** Treat fixed-corpus retrieval as a ledger. Once a search axis or document goal has exposed what it can, surface rewrites and overlapping rereads add no evidence. Change the unresolved clue, anchor, or scope; open a relevant unread result; or revisit a document only for a clearly distinct field not covered before. **Criterion.** The replacement opens a relevant unread result, searches a materially different unresolved clue or anchor, or rereads only for a distinct uncovered field. A finish is acceptable only if visible evidence resolves all hard constraints; no retrieval is a surface repeat.

UNSUPPORTED-HYPOTHESIS-ANCHORING 1,327 assignments; 1,173 accepted, 922 within budget

Trigger. A schema-valid, nonrepeated search or extraction treats an unestablished candidate, sought value, or inferred connection as fact. It is absent from the question and retrieved text, or supported only by resemblance, existence, or an isolated clue; the fault is not transcription, wrong-entity assignment, or direct refutation. **Hint.** Separate discovery from confirmation. Narrow with constraints and terminology already visible, and frame document goals to extract all relevant facts rather than to confirm a favored story. A hypothesis may be tested, but the action must expose disconfirming evidence and must not supply the unknown as though it were established. **Criterion.** The replacement uses attested constraints and asks neutrally for evidence, or explicitly tests the unsupported link without assuming it. A finish relies only on visible support and does not present the unattested hypothesis as fact.

PURSUING-REFUTED-PREMISE 319 assignments; 275 accepted, 201 within budget

Trigger. Retrieved evidence already directly refutes a working candidate or premise under a hard constraint, establishes the sought construct as nonexistent under that reading, or marks relied-on details as nonfactual. The step nevertheless makes a valid search or read, or finishes in the requested form, while treating it as viable. **Hint.** Distinguish contradiction from missing evidence. A hard conflict disqualifies the current interpretation unless a relevant independent source can resolve the conflict; gathering unrelated supporting clues cannot repair it. Focus next on the conflicting constraint or widen to alternative candidates or interpretations. **Criterion.** The replacement does not continue ordinary confirmation. Its search or document goal targets the visible conflict or pursues an unrefuted interpretation with question-grounded constraints; a finish relies only on visibly supported, unrefuted evidence.

LITERAL-PARAPHRASE-QUERY 235 assignments; 194 accepted, 178 within budget

Trigger. With no visible refutation, a schema-valid, nonrepeated search uses a long narrative sequence copied from the question or invented as an exact source phrase, although nothing shows that wording occurs in the corpus. Its anchor is descriptive prose, not a transcription error, unsupported hypothesis, wrong-entity assignment, or derived condition. **Hint.** Assume question prose summarizes source material rather than transcribes it. Translate the clue into short, distinctive content terms that a relevant source would naturally use, and avoid making invented syntax the retrieval key. Opening a promising result is preferable to betting on an unsupported exact phrase. **Criterion.** The replacement search uses short, discriminative source-side terms rather than the long paraphrase, or get_document opens a relevant unread result. A finish is acceptable only if visible evidence already resolves the question.

SOURCE-GOAL-MISMATCH 203 assignments; 166 accepted, 136 within budget

Trigger. The step makes a schema-valid, nonrepeated get_document call on an unread result. The extraction goal is grounded and attached to the correct entity, but the result's visible subject, word sense, source type,

or scope cannot plausibly contain that fact; no hard-constraint conflict or earlier refutation is involved. **Hint.** Predict likely content from the result's visible subject, scope, and source type before opening it. Choose a document that could actually contain the requested fact and is consistent with the hard constraints. Recurrence or keyword overlap does not make an incompatible source useful. **Criterion.** The replacement opens a visible unread result whose subject and source type could contain the requested fact, or searches for such a source. A finish is acceptable only if visible evidence already resolves the grounded need.

LITERAL-DERIVED-CONDITION-QUERY 178 assignments; 114 accepted, 104 within budget

Trigger. With no visible refutation, a schema-valid, nonrepeated search uses a task-stated calculated, aggregate, comparative, or positional condition as a literal retrieval anchor, although it must be evaluated from underlying records. Its other terms are accurately grounded and the condition is attached to the correct entity. **Hint.** Separate retrieval from evaluation. Search for the record type and topical entities likely to contain the raw components, then inspect candidate records and evaluate the condition yourself. Do not assume a source will state a value or relationship that exists only after computation. **Criterion.** The replacement searches for sources holding the underlying components or opens a relevant surfaced record and evaluates the condition from its contents. A finish is acceptable only when the visible components support the evaluation.

BYPASSED-VISIBLE-LEAD 132 assignments; 71 accepted, 58 within budget

Trigger. With no visible hard contradiction, context contains an unread result or untested alternative that bears directly on an unresolved constraint and is comparable to or stronger than the chosen route. The step takes an otherwise grounded, schema-valid, nonrepeated search or read elsewhere, or narrows without comparing that lead. **Hint.** Maintain a lead ledger across result pages and rank leads by expected evidentiary yield, not ranking position or familiarity. Before launching another viable but weaker route, inspect the best matching unread result or directly compare unresolved alternatives. Visible source terminology may also support a focused follow-up when opening is not appropriate. **Criterion.** The replacement acts on the visible lead by opening it, testing the unexamined alternative against the unresolved constraint, or using its attested terms for focused comparison. A finish requires visible evidence resolving the alternatives and all hard constraints.

GENERIC-DESCRIPTOR-ONLY-QUERY 89 assignments; 69 accepted, 63 within budget

Trigger. With no visible refutation, a schema-valid, nonrepeated search uses only broadly shared descriptors although a distinctive clue is visible. It has no target-specific anchor and does not rely on a long literal paraphrase or a condition that must be derived from records. **Hint.** Broad descriptors retrieve broad collections. Anchor discovery on the rarest visible clue or attested source-side term, then add only the shared descriptors needed to express the unresolved condition. If no anchor is available, inspect a relevant result for vocabulary rather than stacking more generic terms. **Criterion.** The replacement search includes a distinctive visible clue or attested target-specific term, or get_document opens a relevant unread result to obtain such an anchor. A finish requires visible support for all hard constraints.

MALFORMED-TOOL-CALL 83 assignments; 69 accepted, 41 within budget

Trigger. The visible tool schema and prior results make the required argument type clear, but the step supplies another tool's fields, omits substantive query text, or gives get_document free text instead of an exact surfaced identifier. The call cannot perform the stated retrieval. **Hint.** Match the tool to the retrieval intent before calling it. Use search for free-text discovery and get_document only for an exact surfaced result with an extraction goal that its content could satisfy. Treat an input error or irrelevant output from an empty call as a signal to repair the call, not as evidence about the question. **Criterion.** The replacement's parameters fit its tool and stated need: search has substantive query text, get_document uses an exact visible result identifier with a plausible goal, or finish is used only when visible evidence already resolves all hard constraints.

FINISH-WITH-UNSUPPORTED-CLAIMS 68 assignments; 49 accepted, 33 within budget

Trigger. The step calls finish with the requested answer form and no visible direct contradiction, but presents a candidate or hard constraint as confirmed while the visible evidence and notes leave that constraint unverified or support only part of it. **Hint.** Audit every final claim against retrieved text. Treat each hard constraint as unresolved until visible evidence supports all of its distinctive parts; existence or a partial match is not confirmation. Seek the missing evidence, or state the limitation without recasting the clue as optional. **Criterion.** The replacement searches or opens a relevant source for the unresolved constraint, or finishes only if every claimed hard constraint is visibly supported. Any remaining gap is explicit and is not presented as confirmed.

WRONG-ENTITY-CLUE-TARGET 59 assignments; 45 accepted, 33 within budget

Trigger. The visible question structure assigns an unresolved property to a linked entity, but a schema-valid, nonrepeated step with accurately grounded terms searches for or extracts it as though it belonged to a different entity. No retrieved evidence already refutes the underlying candidate. **Hint.** Map each clue to its subject before

retrieving. Track which entity carries each property and which relation links it to the requested answer. Aim the action at the property-bearing entity; evidence about a related entity does not satisfy the clue. **Criterion.** The replacement's search or extraction targets the entity visibly assigned the unresolved property and preserves its relation to the requested answer. A finish is acceptable only if visible evidence resolves that correctly attached clue.

PREMATURE-INABILITY-FINISH — 44 assignments; 43 accepted, 31 within budget

Trigger. The step calls finish without a purported answer and declares the task or corpus unable to supply it, although visible snippets already contain the requested information, a plausibly relevant result remains unread, or prior searches cover only the same retrieval axis. **Hint.** Treat absence as a conclusion that requires an audit, not as the default meaning of failed wording. Recheck visible snippets, inspect plausible unread results, and vary the clue, terminology, and scope materially. Only an exhausted set of grounded avenues can support an inability claim. **Criterion.** The replacement answers from visible evidence, opens a plausibly relevant unread result, or searches a materially different unresolved clue. It does not repeat the inability claim while those visible avenues remain.

MISTRANSCRIBED-EVIDENCE-TERM — 17 assignments; 10 accepted, 9 within budget

Trigger. With no visible hard contradiction, a schema-valid, nonrepeated search attempts to reuse a distinctive term visible in prior results but changes its spelling, and the altered form becomes the query anchor. **Hint.** Copy evidence-derived terms exactly rather than reconstructing them from memory. Small transcription changes can redirect literal retrieval and create a false impression that evidence is absent. If the spelling remains uncertain, use other distinctive attested terms instead. **Criterion.** The replacement search reproduces the visible distinctive term exactly or uses other attested, discriminative terms. A get_document call may open a relevant visible result; a finish requires already sufficient visible evidence.

WRONG-ANSWER-FORM — 3 assignments; 1 accepted, 1 within budget

Trigger. The step calls finish and purports to answer, but its output does not satisfy the representation explicitly requested by the question, or merely repeats an incomplete identifier already supplied there. The issue is the answer form, not a contradiction about the underlying candidate. **Hint.** Treat the exact requested representation as a hard constraint. Compare the question's requested form with every supported form visible in the evidence; if the needed form is not yet evidenced, investigate it rather than submitting a partial form as complete. Repeating information from the question adds no answer. **Criterion.** The replacement finish uses the requested representation and is supported by visible evidence, or the replacement search or read specifically seeks that representation. It does not submit the question's partial identifier as a completed answer.

E PROMPTS

The prompts of the three codebook-construction roles and of the outcome judge are reproduced below exactly as used; braces mark fields filled at run time.

Diagnosis: system prompt

```
You review one failed trajectory of a deep-search agent. The agent answers a hard question
     with three tools: `search` (keyword search over a fixed corpus), `get_document` (open one
      already-surfaced result and extract evidence), and `finish` (commit to an exact final
     answer). This trajectory's final answer was judged incorrect.

You are given no rubric of known error patterns. Read the trajectory and find the steps where
     the agent's decision was poor -- where a different action at that step could plausibly
     have changed what was found. You do not know the correct answer and must not guess it;
     judge each decision only by what the agent could see at that moment.

For each faulty step emit:
- "step": the step number, from the valid list.
- "issue": a short kebab-case name you coin for this kind of mistake (e.g. "ignored-promising-
     result", "constraint-dropped-from-state").
- "trigger": 1-3 sentences naming the observable situation that makes this the right label for
      this step: what the agent could see at that moment and what it did with it. Write it
     about the step AS TAKEN, never about a replacement -- a later labeler matches this text
     against a step, so wording like "the replacement does X" cannot be matched at all. It
     must be checkable from the step alone.
- "hint": 2-4 sentences shown to an agent re-deciding this one step. It must teach the general
      decision skill the agent lacked, phrased so the identical text would help on any other
     question where this mistake occurs.
- "success_criteria": 1-3 sentences a reviewer uses to decide whether a replacement action at
     this step fixes the mistake. The reviewer sees only the agent's context and the
     replacement turn and also does not know the correct answer, so the criteria must be
     checkable from the action itself -- which tool it picks and how its argument relates to
     what is visible -- never from whether some answer is right.
```

```
STRICT GENERALITY RULES for "trigger", "hint" and "success_criteria"; violating any of them
     voids the finding:
- Never mention any entity from this question or trajectory: no person, place, work or
     organization names, no titles, no dates or years, no numbers, no document ids.
- Never propose concrete query wording, and never use double-quoted phrases.
- Never state, imply, or narrow down what the final answer might be.
- The text must remain fully sensible if pasted into a different trajectory that shows the
     same kind of mistake.

Label only steps where the mistake is clear and consequential; return at most {max_findings}
     findings, ordered by how much the mistake mattered. A step may appear at most once. If
     nothing qualifies, return an empty list.

Return STRICT JSON only:
{{"findings":[{{"step": <int>, "issue": "<kebab-case>", "trigger": "<text>", "hint": "<text>",
      "success_criteria": "<text>"}}]}}
```

Diagnosis: user template

```
QUESTION:
{question}

TRAJECTORY (chronological; each action the agent took, then the full tool result it saw,
     verbatim):
{trajectory}

FINAL ANSWER GIVEN (judged incorrect): {final_answer}

Valid step numbers you may label: {valid_steps}

AGENT'S VIEW AT ITS LAST STEP (the prompt the agent itself saw at its final turn, verbatim;
     ACTION HISTORY summarizes the actions taken and EVIDENCE holds what the agent had
     extracted from documents):
{agent_view}

Find the faulty steps per the rules. Return strict JSON only.
```

Reassignment: system prompt

```
You label the faulty steps of one failed trajectory of a deep-search agent. The agent answers
     a hard question with three tools: `search` (keyword search over a fixed corpus), `
     get_document` (open one already-surfaced result and extract evidence), and `finish` (
     commit to an exact final answer). This trajectory's final answer was judged incorrect.

You are given a fixed CODEBOOK of error families, each with the criteria that define it. Your
     job is only to decide, for each step in the valid list, whether it commits one of these
     errors. Do not invent new error names and do not label a step whose mistake is not in the
      codebook -- an unlabeled step is the correct output when nothing in the codebook fits.

You do not know the correct answer and must not guess it. Judge each decision only by what the
      agent could see at that moment, and only against the codebook criteria -- not by whether
      the trajectory eventually succeeded.

Label a step only where the error is clear and consequential: a different action at that step
     could plausibly have changed what was found. A step may carry more than one family if it
     genuinely commits both. Label EVERY step in the valid list that meets that bar -- the
     same mistake repeated across ten steps is ten labels, not one. Do not summarise, do not
     keep only the worst, and do not stop early.

Return STRICT JSON only:
{{"labels":[{{"step": <int>, "family": "<family-name from the codebook>"}}]}}

Every "family" must appear verbatim in the codebook and every "step" must come from the valid
     list. Return {{"labels":[]}} if no step commits a codebook error.
```

Reassignment: user template

```
CODEBOOK ({n_families} families):

{codebook}

QUESTION:
{question}

TRAJECTORY (chronological; each action the agent took, then the full tool result it saw,
     verbatim):
{trajectory}

FINAL ANSWER GIVEN (judged incorrect): {final_answer}
```

```
WHAT THE AGENT SAW WHEN IT ANSWERED:
{agent_view}

VALID STEPS TO LABEL: {valid_steps}
```

Final consolidation: system prompt

```
You are finalizing an error-rubric codebook for a deep-search agent. The agent answers
     difficult questions using three tools:

- `search`: keyword search over a fixed corpus.
- `get_document`: open an already-surfaced result and extract evidence.
- `finish`: commit to an exact final answer.

The codebook has two downstream readers, and each field serves exactly one of them:

- LABELER examines a failed trajectory and selects exactly one rubric from the codebook for
     each faulty step, or leaves the step unlabeled. The LABELER uses `trigger` to judge the
     action the agent actually took.
- REVIEWER later examines a replacement action for a labeled step and uses `success_criteria`
     to judge whether the replacement fixes the mistake. The REVIEWER does not know the
     correct answer. The REVIEWER sees only the context that was visible to the agent at that
     point and the replacement action.

Therefore:

- `trigger` must describe the observable fault as it occurred and the action the agent
     actually took.
- `hint` and `success_criteria` must describe the fix.
- A `trigger` must not describe what the replacement action does, because the LABELER must
     match it against the faulty step.
- `success_criteria` must be verifiable from the replacement action itself and the context
     visible at that point, without knowing the correct answer.
- The step context includes the question, visible tool history, returned results, opened
     document content, agent notes or reasoning, and the actual tool and parameters used at
     that step.

The candidate rubrics below were drafted independently. They may contain duplicates,
     subsumption, inappropriate splits, unclear names, or inconsistencies between triggers and
      prescribed fixes. Normalize all candidates globally. Do not treat input clusters,
     provenance, original names, or ordering as final category boundaries.

Apply these rules:

1. Global merging

Compare every candidate rubric with every other candidate, including candidates from different
      source clusters.

Merge two candidates whenever their triggering situations are the same or substantively
     equivalent, even if their names, source clusters, or proposed fixes differ.

If one trigger is another trigger plus extra qualifiers, keep only the broader accurate rubric
     . The narrower situation is not a separate rubric unless the added qualifier changes both
      the nature of the mistake and the applicable corrective principle.

Different manifestations of evidence do not by themselves create different errors. For example
     , a result recurring across searches, a snippet matching distinctive clues, and an
     enumeration page may all be manifestations of the same broader situation. Split them only
      if the difference changes the nature of the agent's mistake and the corrective principle
     .

When merging, rewrite every field around the shared error and shared corrective requirement.
     Do not concatenate drafts. The merged rubric must allow a sound correction that none of
     the drafts happened to mention.

Do not make a particular tool mandatory merely because one draft proposed it. Require a tool
     in `success_criteria` only when the visible situation makes that tool necessary.

2. Splitting

Split a draft only according to the observable situation in which the error occurs, never
     merely because the recommended fixes differ.

Split only when all of the following are true:

- The situations can be distinguished from information visible before the action;
- The distinction changes the nature of the mistake;
- The distinction changes the applicable corrective principle;
- The resulting triggers can still be made mutually exclusive and verifiable.
```

```
If the situation is the same and only the tool choice, query wording, evidence source, or
    repair detail differs, merge the candidates and state their shared requirement generally.

For every split, internally check the resulting rubrics against the entire codebook for
    equivalence, subsumption, and substantive overlap. Merge or tighten boundaries if overlap
     remains.

3. Label uniqueness and mutual exclusivity

The LABELER must select at most one rubric for each faulty step. Therefore, all final `trigger
    ` fields must be mutually exclusive.

Do not manufacture artificial exclusivity by adding many incidental qualifiers. Define each
    rubric around one primary decision error.

When multiple faults can appear in one action, define boundaries around the most direct
    primary decision failure and exclude situations covered by another rubric's primary
    trigger.

Do not split one error into separate rubrics for repeating once versus repeatedly, recurring
    results versus non-recurring results, or results originating from different search groups
    , unless those situations genuinely require different corrective principles.

4. Field requirements

Each rubric must contain:

- `name`: a short, clear, kebab-case error name. The name alone should communicate the core
    mistake. It should describe the faulty behavior and its distinguishing object or
    condition. Do not use numeric suffixes such as `-2` or `-3`, repair instructions, or
    inherited names that no longer match the trigger.
- `trigger`: one to three sentences and fewer than 350 characters. It must state what was
    visible to the agent before the action and what the agent actually did. It must be
    decidable from the visible step context and action alone, without later outcomes, hidden
    intent, or the correct answer.
- `hint`: two to four sentences and fewer than 700 characters. It must teach a general
    decision rule to an agent reconsidering its next action, applicable across questions.
- `success_criteria`: one to three sentences and fewer than 350 characters. It must let the
    REVIEWER determine whether the replacement fixes the mistake from the visible context and
     replacement action alone, including the selected tool and the relationship between its
    parameters and the visible information.

5. Writing triggers

A `trigger` describes the fault, not the recommended action.

It must describe:

- what information was already visible;
- how the agent incorrectly responded to it;
- why that response belongs to this error situation.

Do not put the following in a `trigger`:

- what the replacement should do;
- whether a later action succeeds;
- the correct answer;
- hidden state that requires replaying future steps;
- conditions that depend on future tool results.

If an error depends on earlier searches, reads, or evidence, the trigger may refer to that
    history because it was visible before the action. State what the history actually showed.

6. Writing hints and success criteria

A `hint` should teach a general decision rule, not provide an answer or a fixed query template
     for a particular question.

A `success_criteria` must verify that the replacement:

- uses relevant information visible at the time;
- selects an action matched to the unresolved information need;
- avoids the original error;
- makes a substantive strategic change rather than merely changing surface wording.

When multiple sound repairs are possible, describe the requirement they share. Do not turn one
     draft's preferred repair into the only valid repair.

If the replacement uses `search`, its parameters should show that it uses materially
    discriminative information related to the unresolved clue.
```

```
If the replacement uses `get_document`, it should open a visible, unread, relevant result, and
     its extraction goal should match content that the document could plausibly contain.

If the replacement uses `finish`, the action and visible evidence should show that it does not
     ignore an explicit unmet hard constraint or an unverified condition.

7. Strict generality constraints

The following restrictions apply to every final rubric field: `name`, `trigger`, `hint`, and `
     success_criteria`.

- Do not include names of people, places, works, organizations, titles, dates, years, numbers,
     or document IDs.
- Do not include concrete query wording.
- Do not include concrete phrases enclosed in double quotation marks.
- Do not state, imply, or narrow down any particular answer.
- Use abstract wording that applies across questions.

8. Final audit

Before output, internally verify that:

- all candidates were compared globally, including across source clusters;
- no equivalent, duplicate, or fully subsumed rubric remains;
- every `trigger` describes an actual fault rather than a repair;
- all triggers are mutually exclusive, so at most one rubric can match a single step;
- every `hint` gives a general decision rule;
- every `success_criteria` is verifiable from the replacement action and visible context;
- every name is semantically clear and has no numeric suffix;
- all character limits are satisfied;
- the exact JSON schema is followed;
- the final output contains no extra explanation, Markdown, code fences, or comments.

Output strictly valid JSON only, in exactly this format:

{{"rubrics":[{{"name":"<kebab-case>","trigger":"<text>","hint":"<text>","success_criteria":"<
     text>"}}]}}
```

Final consolidation: user template

```
DRAFT RUBRICS ({n_drafts}):

{listing}
```

Outcome judge

```
Judge whether the final answer in [response] is semantically equivalent to the reference [
     correct_answer] for [question].

[question]: {question}

[response]: {response}

[correct_answer]: {correct_answer}

Your judgement must be in the format and criteria specified below:

extracted_final_answer: The final exact answer extracted from the [response]. Put the
     extracted answer as 'None' if there is no exact, final answer to extract from the
     response.

reasoning: Explain whether extracted_final_answer and [correct_answer] have the same meaning
     and identify the same answer. Semantic equivalence matters, not character-for-character
     identity. Treat differences in capitalization, whitespace, Unicode accents or diacritics,
      and non-semantic punctuation or quotation marks as equivalent unless [question]
     explicitly requires a particular spelling or format. Allow common spelling,
     transliteration, or romanization variants of the same unambiguous person, place, work, or
      entity. Additional detail is allowed only when it is correct and does not change or
     contradict the answer. Do not treat genuinely different entities, titles, values, or
     ambiguous answers as equivalent. Do not comment on background, attempt to solve the
     problem, or argue for an answer different from [correct_answer].

correct: Answer 'yes' if extracted_final_answer is semantically equivalent to [correct_answer
     ], including the non-semantic variants allowed above, or is within a small margin of
     error for numerical problems. Answer 'no' if there is a meaningful inconsistency,
     ambiguity, non-equivalency, or an incorrect answer.
```

```
confidence: The extracted confidence score between 0% and 100% from [response]. Put 100 if
    there is no confidence score available.
```

F Optimization details

Completion masks. Each target is rendered as a complete assistant turn with reasoning, structured-state, and action fields; all three are supervised, input-prompt tokens are excluded, and an end-of-sequence token is appended to both completions. Character spans are mapped to token masks separately for the chosen and rejected responses, fixing the position sets $M(a^+)$ and $M(a^-)$ used in §2.4. The score $\ell_\theta(a \mid s)$ used there is the token-averaged log-probability

$$\ell_\theta(a \mid s) = \frac{1}{|M(a)|} \sum_{u \in M(a)} \log \pi_\theta(a_u \mid s, a_{<u}). \tag{5}$$

The reference KL uses the same positions; for each candidate, the penalty is

$$K_\theta(a \mid s) = \frac{1}{|M(a)|} \sum_{u \in M(a)} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid s, a_{<u}) \,\|\, \pi_{\mathrm{ref},k}(\cdot \mid s, a_{<u}))\,. \tag{6}$$

The pair penalty in Eq. 4 is the equal-weight mean of the two candidates,

$$L_{\mathrm{KL}} = \tfrac{1}{2}\big[K_\theta(a^+ \mid s) + K_\theta(a^- \mid s)\big], \tag{7}$$

and the combined loss is averaged over the minibatch.

Preference training. Every RePair arm optimizes Eq. 4 with β=0.1, ε=0.1, α_{NLL}=0.1, and λ_{ref}=0.2, updating all parameters of π_0 under fully sharded data parallelism with activation checkpointing, using Adafactor, a learning rate of 10^{-6} with cosine decay and 10% warmup, bf16 precision, seed 12345, one pair per device on 16 devices with four gradient-accumulation steps (global batch 64), and 4 epochs. The policy and the reference are two copies of the round's starting policy (π_0 in round 1), and the number of optimizer steps follows from the pair count (Table 8). All hyperparameters were fixed before any test evaluation.

Table 8: Preference-training runs.

Arm	Model	Pairs	Steps
Hand-written codebook	14B	3,713	236
Induced codebook	14B	2,097	132
Top-1 / Top-5 / Top-10 rubrics	14B	725 / 1,541 / 1,874	48 / 100 / 120
Self-repair, round 1	14B	6,239	392
Self-repair, round 2	14B	7,064	444
No-rubric repair	14B	4,521	284
Hand-written codebook	8B	6,128	384
Induced codebook	8B	4,200	264

All arms use full-parameter updates, global batch 64, learning rate 10^{-6}, and 4 epochs.

Supervised arms. The warm start trains a LoRA adapter (r=64, α=128 for 14B; r=32, α=64 for 8B; dropout 0.05; all attention and MLP projections) with learning rate 10^{-4}, cosine decay with 10% warmup, sequence packing at 16,384 tokens, global batch 32, and 4 epochs (408 steps), and the adapter is merged to give π_0. Chosen-only SFT and full-trajectory SFT each update all parameters of π_0 with the warm start's schedule (learning rate 10^{-4}, cosine decay with 10% warmup), no packing, a 20,480-token length limit, global batch 16, and 4 epochs: 548 steps on the 2,190 chosen turns and 3,656 on the 14,624 teacher turns.

Iteration. The policy trained in round k is the starting policy and the frozen reference of round $k+1$, which uses fresh failures and preference pairs from the same training-question pool. Every round trains for its full epoch budget, and the number of rounds is chosen by the experimenter; there is no automatic stopping or convergence criterion.

G ROBUSTNESS CHECKS

Training seeds. The two 14B codebook arms and both self-repair rounds are retrained with seeds 12346 and 12347, besides the seed 12345 used everywhere else; every round-2 run starts from the round-1 policy trained with seed 12345. Table 9 lists each run. Across seeds the induced codebook stays above the hand-written one (Welch's t-test, p=0.02), and round 2 above round 1 (p=0.01).

Table 9: Test mean@3 of three training seeds (Qwen3-14B).

Arm	12345	12346	12347	Mean $\pm$ s.d.	Pass@3
Hand-written codebook	50.80	48.19	49.20	49.40 $\pm$ 1.32	64.46
Induced codebook	53.21	52.81	54.02	53.35 $\pm$ 0.62	67.47
Self-repair, round 1	46.99	45.18	46.59	46.25 $\pm$ 0.95	61.04
Self-repair, round 2	49.80	51.41	49.20	50.14 $\pm$ 1.14	64.06

s.d.: sample standard deviation over seeds; Pass@3: mean over seeds. π_0 is shared by all arms and is not retrained.

Paired bootstrap. For two arms evaluated on the same 166 test questions, we resample the questions with replacement 10,000 times and recompute the difference in mean@3 each time, using each question's accuracy over its three runs. The 95% interval spans the 2.5th to 97.5th percentiles of the resampled differences, and the two-sided p-value is twice the smaller fraction of differences on either side of zero (Table 10). All runs use seed 12345.

Table 10: Paired bootstrap over test questions (Qwen3-14B, seed 12345).

Comparison	Δ mean@3	95% interval	p
Induced codebook vs. π_0	+10.64	[+8.21, +12.68]	$< 10^{-4}$
Hand-written codebook vs. π_0	+8.23	[+5.76, +10.98]	$< 10^{-4}$
Induced vs. hand-written codebook	+2.41	[−0.13, +5.34]	0.064
Self-repair, round 2 vs. π_0	+7.23	[+5.12, +9.88]	$< 10^{-4}$
Induced codebook vs. no-rubric repair	+7.23	[+4.21, +10.37]	$< 10^{-4}$
Induced codebook vs. full-trajectory SFT	+5.02	[+2.34, +8.67]	2×10^{-4}
Induced codebook vs. chosen-only SFT	+4.01	[+2.12, +6.27]	$< 10^{-4}$

Validator agreement. We sample 150 repair candidates at random from the 14B induced arm. Two annotators judge each one on the validator's two questions, whether the flagged error is repaired and whether the replacement is a natural decision, and accept it only if both hold; their 19 disagreements are settled by adjudication. The annotators agree on 87.3% of the candidates (κ=0.74), and the validator agrees with the settled label on 88.7% (κ=0.76; Table 11).

Table 11: Validator decisions against the settled human label (150 candidates).

	Human: accept	Human: reject	Total
Validator: accept	84	14	98
Validator: reject	3	49	52
Total	87	63	150