TEXTUALRL: TEXTUAL REINFORCEMENT LEARNING FOR CONTEXT MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents can adapt to task feedback by revising the textual context that guides their behavior, without changing their model parameters. However, turning execution rewards into reusable context updates is challenging: individual trajectories offer limited comparative evidence about what to change, while task-specific fixes may overgeneralize when incorporated into a shared context. We introduce TextualRL, a gradient-free reinforcement learning framework that treats a group of rollouts from the same task under the same context, rather than an isolated trajectory, as the unit of policy critique. *Groupwise Policy Critique* contrasts successful and failed rollouts within a task; when outcomes are uniform, it draws on cross-task evidence to identify behaviors worth preserving or formulate repair hypotheses. *Cross-Group Policy Update* reviews the resulting proposals against evidence pooled across groups, including observations that yield no proposed edit, to refine their content and applicability. Reviewed proposals form a bounded textual patch, with adoption of the candidate context determined by held-out validation rewards. Together, these components establish a reward-driven policy optimization loop in which multiple attempts jointly inform reusable context updates. We evaluate TextualRL with five target agents on six benchmarks. Averaged over the six benchmarks, TextualRL improves test accuracy over the base model by 24.4 and 15.3 percentage points for GPT-5.5 and Qwen3.8-27B, respectively.

1 INTRODUCTION

Reinforcement learning (RL) improves an agent's behavior through interaction and reward (Kaelbling et al., 1996). For LLM agents whose parameters are unavailable for optimization, textual context offers an alternative interface for adaptation: instructions and procedures condition how a fixed model interprets tasks, uses tools, and produces outputs. Reflexion (Shinn et al., 2023) incorporates verbal feedback into subsequent attempts, while Agentic Context Engineering (ACE) (Zhang et al., 2026) evolves contextual playbooks from execution experience. These developments motivate policy optimization in context space: the objective remains higher task reward, but the optimization variable is the context rather than the model weights.

Existing work provides several mechanisms for optimizing textual artifacts. ProTeGi (Pryzant et al., 2023) and TextGrad (Yuksekgonul et al., 2025) use natural-language feedback to guide updates, while GEPA (Agrawal et al., 2026) combines reflection on execution traces with evolutionary prompt search. For agent skills, Trace2Skill (Ni et al., 2026) extracts trajectory-local lessons and hierarchically consolidates them into reusable instructions. SkillOpt (Yang et al., 2026) instead analyzes successful and failed trajectory minibatches separately, merges their proposed edits, and validates bounded skill updates. These methods establish the value of execution-driven adaptation, while highlighting an important design choice: how should trajectories be organized before their feedback is converted into a shared context update?

Group-based RL provides a useful organizing principle. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples multiple responses to the same question and uses their relative rewards to guide parameter updates. For textual adaptation, a group of same-task rollouts likewise provides a comparison unavailable from an isolated execution: under the same task and context, which behavioral differences accompany different outcomes? Comparative experience learning is also explored by ExpeL (Zhao et al., 2024) and ReasoningBank's parallel MaTTS (Ouyang et al.,

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

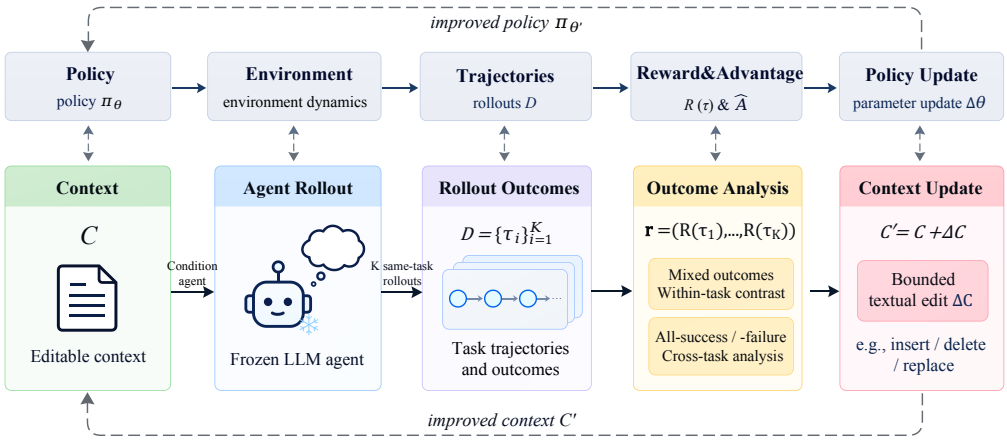

Figure 1: Conceptual comparison of parameter-space and context-space policy optimization. TextualRL uses groups of same-task rollouts and their rewards to propose textual edits ΔC to context C, while the target model remains frozen.

2026), which extract reusable insights or memories from alternative executions. Building on this perspective, we ask: *how can groups of rewarded executions drive policy optimization when updates are made to shared textual context rather than model parameters?* The goal is not merely to select a successful attempt, but to use multiple attempts to improve the policy governing future executions.

Turning grouped experience into a context update presents two challenges. First, rewards identify outcomes, not the instructions that should change. Within-task success–failure comparisons provide evidence about alternative behaviors, but this contrast disappears when every sampled rollout succeeds or every rollout fails. An analysis procedure must therefore distinguish evidence supporting preservation from evidence suggesting an unverified repair. Second, a context is shared across tasks, whereas a proposed modification may be supported only under particular conditions. Merging proposals alone cannot reveal every exception: a successful execution may require no edit while still contradicting another group's proposed rule. Effective updates therefore require both outcome-aware policy critique and a review of the evidence that supports or limits each modification.

We introduce TextualRL, a gradient-free reinforcement learning algorithm for context management that treats same-task rollout groups as the basic unit of policy critique. As illustrated in Figure 1, each iteration collects multiple executions per task under the current context while keeping the target model frozen. *Groupwise Policy Critique* compares successful and failed rollouts within mixed-outcome tasks; for all-success or all-failure tasks, it uses cross-task evidence to identify behaviors worth preserving or formulate repair hypotheses, respectively. The critics return edit proposals and trajectory evidence separately, retaining observations even when no edit is proposed. *Cross-Group Policy Update* reviews consolidated proposals against evidence pooled across analysis groups, retaining, narrowing, or revising instructions according to their support and applicability. The reviewed edits form a bounded textual patch. A step-level validation gate accepts the candidate context only if its reward on held-out tasks exceeds the recorded reference score. The selected context then conditions subsequent rollouts, closing a reward-driven policy optimization loop without model-parameter updates. We evaluate TextualRL with five target agents across six benchmarks. Averaged over the six benchmarks, TextualRL improves test accuracy over the no-skill baseline by 24.4 and 15.3 percentage points for GPT-5.5 and Qwen3.8-27B, respectively. The corresponding gains over the best baseline are 0.9 and 3.2 percentage points.

Our contributions are threefold:

- We introduce TextualRL, a group-based algorithm for gradient-free policy optimization over the shared textual context of a frozen LLM agent.
- We develop Groupwise Policy Critique, which uses the available reward structure to combine within-task comparisons with cross-task preservation and repair analysis.
- We develop Cross-Group Policy Update, which retains evidence independently of edit proposals to review their content and applicability before bounded, validation-gated updates.

2 Related Work

Prompt and Context Optimization. Automatic prompt optimization searches for text that improves task performance. AutoPrompt (Shin et al., 2020) uses gradient-guided search over discrete trigger tokens, whereas APE (Zhou et al., 2023) generates and evaluates instructions from demonstrations. DSPy (Khattab et al., 2024) compiles modular language-model pipelines through demonstration bootstrapping and selection, while MIPRO (Opsahl-Ong et al., 2024) jointly optimizes instructions and demonstrations across modules. Context and skill optimization extends this scope to reusable guidance for future executions. Dynamic Cheatsheet (Suzgun et al., 2026) maintains adaptive memory across test queries. Trace2Skill (Ni et al., 2026) independently extracts trajectory-local patches and hierarchically consolidates them into agent skills. EvoSkill (Alzubi et al., 2026) proposes skills from failed trajectories and evaluates candidate skill repositories through validation. SkillOpt (Yang et al., 2026) separately analyzes successful and failed trajectory minibatches, merges their proposals, and selects bounded skill updates using held-out validation. TextualRL shares the goal of feedback-driven textual adaptation but changes how executions inform an update: same-task rollouts under the same context are preserved for joint critique before proposal consolidation. Moreover, trajectory evidence is retained independently of proposed edits. This allows observations that warrant no new instruction to constrain other proposals, making evidence about a rule's applicability available beyond the editing suggestions themselves.

Learning from Trajectory Feedback. Trajectory feedback supports both task execution and learning across attempts. ReAct (Yao et al., 2023) interleaves reasoning and actions with environment observations, and Self-Refine (Madaan et al., 2023) iteratively critiques and revises an output. LATS (Zhou et al., 2024) combines tree search, self-reflection, and environment feedback to explore alternative trajectories. Voyager (Wang et al., 2024) uses execution feedback to improve programs and accumulate reusable code skills. Comparative experience learning is also established: ExpeL (Zhao et al., 2024) extracts insights from same-task success–failure pairs and shared patterns across successful tasks, while ReasoningBank (Ouyang et al., 2026) uses parallel MaTTS to curate reusable memory by comparing multiple executions of the same query. In parameter-space RL, GRPO (Shao et al., 2024) uses relative rewards within groups of same-question responses to guide policy updates. TextualRL instead performs gradient-free policy optimization over a shared context, coupling comparative experience with an explicit proposal–review–validation loop. *Group-wise Policy Critique* adapts its analysis to the available reward structure: mixed outcomes support within-task contrast, whereas uniform outcomes require cross-task evidence for preservation or repair hypotheses. *Cross-Group Policy Update* then reviews both the content and applicability of proposed instructions using independently pooled evidence, before edits undergo held-out validation. Together, these mechanisms connect outcome-aware comparison to cross-task evidence review, turning groups of attempts into reward-selected updates to a frozen agent's shared policy context.

3 TextualRL

3.1 Context-Space Policy Optimization

Let $C \in \mathcal{C}$ denote the editable textual context containing instructions and procedures shared across tasks. For a task x, the context induces a trajectory distribution $\pi_\theta(\tau \mid x, C)$ through the target agent, where a trajectory τ comprises actions, available observations, and the final output. The target parameters θ remain fixed, and TextualRL optimizes the context to maximize expected task reward:

$$\max_{C \in \mathcal{C}} J(C) = \mathbb{E}_{x \sim p_{\text{task}}} \mathbb{E}_{\tau \sim \pi_\theta(\cdot|x,C)}[R(\tau)], \tag{1}$$

where p_{task} is the task distribution and $R(\tau) \in \{0, 1\}$ indicates task success. Policy optimization thus operates through textual changes to C rather than updates to θ.

At step t, we sample a training batch B_t and collect K rollouts per task under the current context C_t. For each $x \in B_t$, we retain its indexed rollout group and reward vector:

$$D_x = \{\tau_{x,i}\}_{i=1}^{K}, \qquad \mathbf{r}_x = (r_{x,1}, \ldots, r_{x,K}), \tag{2}$$

where $r_{x,i} = R(\tau_{x,i})$. We omit the step index from D_x and $\mathbf{r}_x$ for clarity. Keeping same-task rollouts together preserves the behavioral and outcome comparisons used to propose an update.

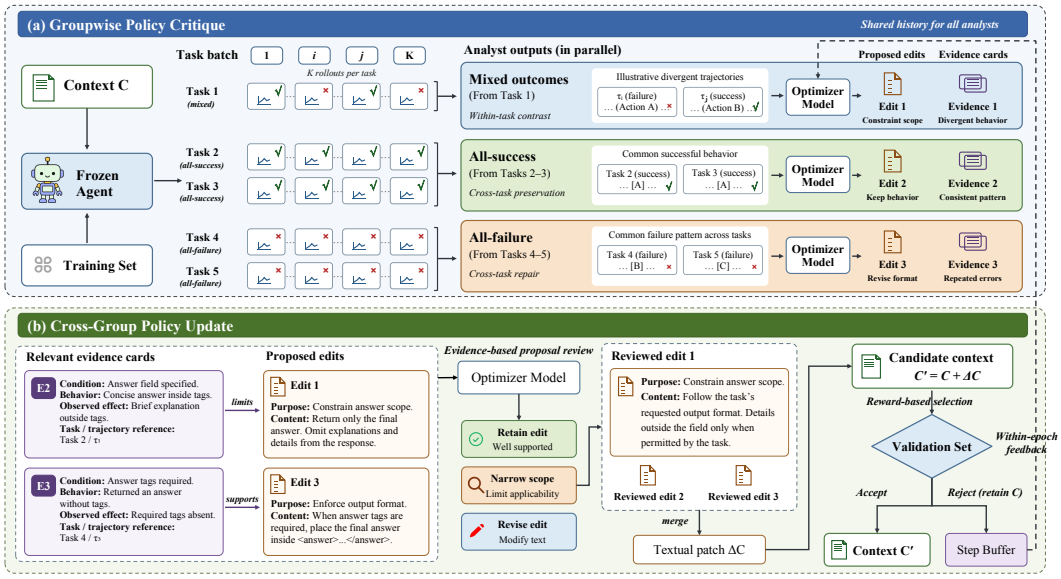

Figure 2: Overview of TextualRL. (a) Groupwise Policy Critique preserves same-task rollout groups and uses within-task contrast for mixed outcomes, cross-task preservation for all-success outcomes, and cross-task repair for all-failure outcomes. Parallel critics return edit proposals and evidence cards separately. (b) Cross-Group Policy Update reviews proposals against pooled evidence, constructs a bounded patch, and selects the candidate context through held-out validation. Within-epoch feedback informs subsequent critique calls.

As shown in Figure 2, each step comprises two modules. *Groupwise Policy Critique* uses the available outcome structure to produce edit proposals and trajectory evidence. *Cross-Group Policy Update* reviews these proposals against pooled evidence, constructs a bounded textual patch, and uses held-out validation rewards to select the next context. Subsequent rollouts are conditioned on the selected context, forming a gradient-free policy optimization loop.

3.2 Groupwise Policy Critique

The comparison available within a rollout group depends on its reward structure. Let $n_x = \sum_{i=1}^{K} r_{x,i}$ be the number of successful rollouts for task x. We route each group according to

$$\text{route}(\mathbf{r}_x) = \begin{cases} \text{mixed}, & 0 < n_x < K, \\ \text{all-success}, & n_x = K, \\ \text{all-failure}, & n_x = 0. \end{cases} \tag{3}$$

Before critique, we consolidate duplicate trajectories while preserving their outcome labels, original rollout references, and multiplicities. All analysis groups retain complete task blocks; multiple rollouts of one task do not count as support from multiple distinct tasks.

Within-task contrast. Each mixed-outcome task forms a separate analysis group. The critic compares successful and failed rollouts to identify the earliest observable behavioral divergence, such as a different action or use of available evidence. It proposes at most one atomic repair or preservation rule supported by this contrast, and returns no edit when no supported distinction is observable.

Cross-task preservation. When all rollouts succeed, there is no within-task success–failure contrast. We therefore group all-success task blocks into analysis minibatches containing at least two distinct tasks. The critic identifies a shared, observable behavior worth preserving and proposes an instruction only when that behavior is not already covered by C_t.

Cross-task repair. All-failure task blocks are grouped in the same manner. The critic identifies a shared, observable failure mechanism across at least two distinct tasks and proposes a corresponding

repair hypothesis. Failure alone neither supplies a correct answer nor verifies the proposed repair. Both homogeneous routes return no edit when the required common behavior or mechanism is unsupported, and a route with fewer than two task blocks is skipped.

Edit proposals and trajectory evidence. Let $g \subseteq B_t$ be the tasks assigned to one critique call and $\mathcal{G}_t$ the collection of these analysis groups. A mixed group contains one task, whereas a homogeneous group contains at least two tasks from the same outcome category. Each call returns proposed edits P_g and evidence cards E_g:

$$(P_g, E_g) = \mathrm{Critique}\big(C_t, H_t, \{(x, D_x, \mathbf{r}_x) : x \in g\}\big). \tag{4}$$

Here, $\mathrm{Critique}$ invokes the optimizer model with route-specific instructions. Calls run in parallel and share the current context C_t and within-epoch history H_t.

An edit proposal specifies a modification to C_t. An evidence card records an observable condition, behavior, and effect, together with a short supporting quote and references to the corresponding task and rollout. Unobserved conditions or effects are marked as unknown. Cards may be returned even when P_g is empty: behavior already handled correctly by the current context can still constrain a proposal from another group. We therefore pool proposals and evidence independently:

$$P_t = \bigcup_{g \in \mathcal{G}_t} P_g, \qquad E_t = \bigcup_{g \in \mathcal{G}_t} E_g. \tag{5}$$

3.3 Cross-Group Policy Update

A modification supported by one group may restrict behavior that succeeds elsewhere. Cross-Group Policy Update therefore reviews the content and applicability of local proposals before evaluating their combined effect on the shared context.

Evidence-based proposal review. The optimizer first consolidates P_t through hierarchical merging to remove redundancy and resolve conflicting instructions. It then reviews the consolidated candidates using C_t and the independently pooled evidence E_t. For each edit, it compares the prescribed behavior and scope with relevant observations, including the behavior being regulated, the failure mechanism being addressed, or an exception within the proposed scope. The review records whether these observations *support*, *limit*, or *contradict* the instruction.

Based on these relations, the optimizer retains the edit, narrows its applicability using an observable condition, or revises its content. The resulting edits form $\widetilde{P}_t$, with each decision accompanied by a rationale and references to the evidence used. Insufficient relevant evidence leaves the proposal's original scope and uncertainty unchanged. The review neither requires evidence from every outcome category nor introduces unrelated edits to account for unused cards. Figure 2(b) illustrates this distinction. Card E2 records a successful response with the answer inside the required field and an explanation outside it, limiting Edit 1's blanket answer-only restriction. The revised instruction follows the requested format and allows details outside the field only when permitted by the task. Card E3 instead supports Edit 3 by documenting missing required answer tags. The pool E_t contains only current-step evidence, so this review requires no additional target-agent rollouts.

Textual patch. Each reviewed edit specifies an operation—`append`, `insert_after`, `replace`, or `delete`—together with target text when required and replacement or insertion content. We select at most L_t edits, ranking candidates with the optimizer when they exceed this budget. We also remove edits with explicit dependencies on information unavailable during target execution. The remaining operations form a textual patch ΔC_t, yielding

$$C'_t = C_t + \Delta C_t \equiv \mathrm{Apply}(C_t, \Delta C_t), \tag{6}$$

where $+$ denotes applying textual operations rather than numerical addition.

Reward-based selection. We evaluate C'_t on a held-out validation set $\mathcal{V}$ using the benchmark's binary success metric. Let $\widehat{V}(C)$ denote mean validation success under the evaluation protocol and s_t the trainer's recorded reference score. The step-level selection rule is

$$(C_{t+1}, s_{t+1}) = \begin{cases} (C'_t, \widehat{V}(C'_t)), & \widehat{V}(C'_t) > s_t, \\ (C_t, s_t), & \text{otherwise.} \end{cases} \tag{7}$$

Thus, an accepted candidate updates both the active context and the reference score; a rejected candidate leaves both unchanged. Validation evaluates the complete candidate context, not the individual effects of its constituent edits.

Within-epoch feedback. After validation, the shared step buffer augments H_t with observed rollout failure patterns. For rejected candidates, it also records the attempted edit combination and its validation score. Subsequent critique calls in the same epoch receive this history to reduce repeated ineffective proposals. Rejection concerns the evaluated combination and does not imply that every constituent edit is harmful. This history is distinct from E_t, which contains only the current step's evidence. Algorithm 1 summarizes the step-level procedure. Appendix C.3 provides the optimizer prompt blocks.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

Benchmarks. We evaluate TextualRL on six benchmarks spanning question answering, document understanding, mathematical reasoning, and interactive task execution. SearchQA (Dunn et al., 2017) requires answering questions from provided search snippets. Spreadsheet (Ma et al., 2024) evaluates workbook manipulation through executable Python programs. OfficeQA (Opsahl-Ong et al., 2026) requires reasoning over enterprise documents using local file tools and supplied source-page evidence. DocVQA (Mathew et al., 2021) pairs questions with document images. LiveMathematicianBench (He et al., 2026), abbreviated as LiveMath, tests mathematical reasoning through multiple-choice questions. ALFWorld (Shridhar et al., 2021) evaluates sequences of textual actions for completing household tasks. We use the released benchmark splits from SkillOpt (Yang et al., 2026), with dataset details, split sizes, and evaluation metrics provided in Appendix B.1. The training split supplies trajectories for context optimization, while validation scores determine step-level update acceptance and checkpoint selection. The test split is used only to evaluate the selected contexts.

Baselines. We compare TextualRL with seven baselines in Table 1. **Base Model** evaluates the frozen target using the original LLM. **Human skill** uses manually written task guidance, while **LLM skill** uses a skill generated in a single pass. These baselines measure the benefit of adding static instructions without iterative optimization. **Trace2Skill** (Ni et al., 2026) extracts lessons from individual training trajectories and consolidates them into skills. **TextGrad** (Yuksekgonul et al., 2025) updates text using language-model feedback as textual gradients. **GEPA** (Agrawal et al., 2026) uses reflective evolutionary search to improve prompts from execution feedback. **SkillOpt** (Yang et al., 2026) aggregates edit proposals from mini-batch and selects bounded skill updates through validation. These main comparisons use the benchmark-specific direct-chat setting, including the tools or environment interactions required by each task.

Implementation Details. We use five frozen target models: GPT-5.5, GPT-5.4, Qwen3.5-4B, Qwen3.6-35B-A3B, and Qwen3.8-27B. A separate context is optimized for every target–benchmark pair. Unless otherwise specified, GPT-5.5 serves as the optimizer. The target model's parameters remain unchanged during both optimization and evaluation. Starting from the benchmark-specific initial context, we optimize for four epochs with up to 40 tasks per batch and $K = 4$ rollouts per task. A full batch therefore contains 160 trajectories. Analysis and merge minibatches have size eight, subject to the task-group constraints in Section 3.2. The edit budget follows a cosine schedule from four to two edits. The within-epoch buffer provides feedback to subsequent critique calls. Additional implementation details appear in Appendix B.

4.2 MAIN RESULTS

Table 1 compares TextualRL with the seven baselines across all five target models. TextualRL achieves the highest six-benchmark average for every target, with gains of 11.4–24.4 percentage points over No skill, 0.9–3.2 points over the strongest baseline SkillOpt. It outperforms all baselines in 29 of the 30 target–benchmark settings and ties SkillOpt on GPT-5.5 with ALFWorld. For GPT-5.5, the average score increases from 58.8% without a skill to 83.2% with TextualRL. For Qwen3.8-

Table 1: Main test results (%). Bold and underlining mark the highest and second-highest distinct scores in each column within each target model. Subscripts on TextualRL scores show percentage-point gains over the corresponding Base Model.

Model	Skill Source	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld
GPT-5.5	Base Model	77.7	41.8	33.1	78.8	37.6	83.6
	Human skill	81.8	72.9	66.9	90.1	38.4	91.8
	LLM skill	80.9	43.2	51.7	89.6	40.0	93.3
	Trace2Skill	82.4	49.6	65.7	90.6	52.0	87.3
	TextGrad	81.4	41.1	42.0	87.2	49.2	82.8
	GEPA	84.8	73.6	63.9	89.1	43.2	85.8
	SkillOpt	87.3	80.7	72.1	91.2	66.9	**95.5**
	TextualRL	**88.9** +11.2	**82.5** +40.7	**72.7** +39.6	**92.5** +13.7	**67.2** +29.6	**95.5** +11.9
GPT-5.4	Base Model	76.9	41.4	50.0	77.6	36.8	75.4
	Human skill	80.1	59.3	59.3	88.5	41.6	74.6
	LLM skill	79.4	46.1	20.4	90.4	36.8	83.6
	Trace2Skill	81.2	53.2	51.2	89.3	40.0	73.1
	TextGrad	82.5	38.6	54.7	85.3	36.6	78.4
	GEPA	82.4	61.1	60.4	88.3	41.6	74.6
	SkillOpt	83.1	62.5	62.8	91.2	44.0	91.0
	TextualRL	**85.5** +8.6	**63.6** +22.2	**64.5** +14.5	**91.7** +14.1	**47.2** +10.4	**91.8** +16.4
Qwen3.5-4B	Base Model	68.1	9.3	14.5	86.9	22.4	30.6
	Human skill	66.3	16.4	22.7	87.8	18.4	28.4
	LLM skill	65.0	10.4	20.9	88.0	28.8	55.2
	Trace2Skill	68.5	19.3	16.3	88.0	27.2	64.9
	TextGrad	60.7	13.9	20.9	85.6	10.6	53.7
	GEPA	68.6	16.9	22.7	85.1	28.8	60.4
	SkillOpt	71.2	23.9	29.7	89.0	52.0	81.3
	TextualRL	**73.9** +5.8	**27.1** +17.8	**32.0** +17.5	**89.3** +2.4	**56.0** +33.6	**82.1** +51.5
Qwen3.6-35B-A3B	Base Model	72.7	38.2	45.9	87.6	31.2	59.7
	Human skill	74.1	44.3	41.9	88.4	29.6	44.8
	LLM skill	72.6	42.9	46.5	88.4	24.8	60.4
	Trace2Skill	75.4	33.2	32.0	90.4	29.6	70.9
	TextGrad	76.4	22.9	33.7	84.5	7.2	67.9
	GEPA	75.8	45.4	43.6	88.0	31.2	73.9
	SkillOpt	80.3	47.5	47.1	91.4	41.6	82.1
	TextualRL	**81.2** +8.5	**52.1** +13.9	**49.4** +3.5	**92.8** +5.2	**44.8** +13.6	**83.6** +23.9
Qwen3.8-27B	Base Model	73.6	43.6	43.0	88.5	34.4	50.8
	Human skill	74.9	48.6	43.6	89.0	32.8	47.8
	LLM skill	74.5	46.4	42.4	88.8	32.0	59.7
	Trace2Skill	76.9	45.0	43.0	90.4	37.6	72.4
	TextGrad	77.6	39.3	36.1	88.2	35.2	56.7
	GEPA	78.3	50.7	44.2	89.0	40.0	75.4
	SkillOpt	80.4	56.4	46.5	93.3	48.0	82.1
	TextualRL	**82.7** +9.1	**60.7** +17.1	**47.7** +4.7	**94.4** +5.9	**56.0** +21.6	**84.3** +33.6

27B, the corresponding scores are 55.6% and 71.0%. The improvements therefore extend across both GPT and Qwen targets, with a separately optimized context for each target and benchmark. Specifically, TextualRL raises Qwen3.5-4B from 30.6% without a skill to 82.1% on ALFWorld,, a gain of 51.5 percentage points and the largest improvement over No skill. This result shows that optimizing context can substantially improve task completion even for the smallest frozen target. On Qwen3.8-27B, TextualRL reaches 56.0% on LiveMath and 60.7% on Spreadsheet, exceeding the strongest baseline by 8.0 and 4.3 points, respectively. The advantage is more modest on DocVQA, ranging from 0.3 to 1.4 points over the strongest baseline across the five targets.

4.3 Ablations

To evaluate the contribution of each module, we conduct ablation studies on Qwen3.8-27B and report the results in Table 2. For Groupwise Policy Critique, *w/o outcome routing* pools all trajectories collected in a training step and partitions them directly into analyst minibatches of eight trajectories, without first separating them into mixed-outcome, all-success, and all-failure groups. This com-

Table 2: Ablations on Qwen3.8-27B (%). Full TextualRL is the shared reference for both groups.

Setting	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld
TextualRL (full)	**82.7**	**60.7**	**47.7**	**94.4**	**56.0**	**84.3**
Groupwise Policy Critique						
w/o outcome routing	81.8	58.9	46.5	92.5	53.6	83.6
Single rollout ($K = 1$)	80.7	57.5	46.5	93.3	50.4	82.1
Cross-Group Policy Update						
Intra-group cards only	82.1	59.3	47.1	93.0	53.6	83.6
Remove cards without edits	80.9	58.2	45.9	93.6	54.4	82.8
w/o proposal review	81.4	57.1	45.4	92.8	50.4	82.8

Table 3: Mean ± sample standard deviation over three independent runs on Qwen3.8-27B (%).

Method	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld	Overall
SkillOpt	79.93±0.47	56.55±0.54	45.93±0.59	92.87±0.41	47.73±1.22	81.59±0.86	67.43±0.33
TextualRL	**82.62**±0.16	**60.24**±0.55	**47.09**±0.58	**94.03**±0.41	**56.00**±0.80	**84.08**±1.14	**70.68**±0.26

Table 4: Effect of training rollout count K on Qwen3.8-27B (%).

Method	K	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld	Overall
TextualRL	1	80.7	57.5	46.5	93.3	50.4	82.1	68.4
	2	81.4	58.9	47.7	93.6	53.6	82.8	69.7
	4	82.7	**60.7**	47.7	**94.4**	**56.0**	84.3	71.0
	8	**82.9**	60.0	**48.8**	94.1	**56.0**	**85.1**	**71.1**

pares outcome-aware critique with direct minibatch analysis. We also evaluate a single rollout per task ($K = 1$). For Cross-Group Policy Update, *intra-group cards only* restricts the review of each merged edit to evidence cards from its originating critique groups. We additionally remove cards produced without edit proposals or disable cross-group proposal review.

The result shows that removing outcome routing reduces the six-benchmark average by 1.5 percentage points, while using $K = 1$ reduces it by 2.5 points. These results support the value of both organizing critique by outcome patterns and collecting multiple rollouts per task in Groupwise Policy Critique. For Cross-Group Policy Update, restricting review to cards from the originating critique groups reduces the average by 1.2 points, while removing cards without edits reduces it by 1.7 points. These results support using evidence beyond a proposal's originating groups and retaining observations that do not directly produce an edit. Disabling cross-group proposal review reduces the average by 2.7 points, highlighting the value of using evidence to assess and refine proposals before updating the context. Full TextualRL achieves the highest score on every benchmark in both groups.

4.4 Additional Experiments

Run stability. To assess variation across runs, we repeat context optimization three times for TextualRL and SkillOpt on Qwen3.8-27B, keeping the data splits and evaluation protocol fixed. Each run independently selects its checkpoint by validation score before test evaluation. Table 3 reports the mean and sample standard deviation across runs. TextualRL achieves an overall mean of 70.68%, compared with 67.43% for SkillOpt, and has a higher mean score on all six benchmarks. The largest mean gain is on LiveMath, at 8.27 points. The overall advantage ranges from 3.18 to 3.36 points across the three runs, showing that the improvement persists across repeated optimization runs.

Rollout count. To examine the effect of trajectory sampling, we vary the number of rollouts per training task over $K \in \{1, 2, 4, 8\}$ on Qwen3.8-27B. Table 4 shows that the overall score increases from 68.4% at $K = 1$ to 71.0% at $K = 4$. Increasing K from four to eight doubles the rollout count per task but adds only 0.1 percentage points to the overall score. The additional rollouts do not improve every benchmark: Spreadsheet and DocVQA decrease by 0.7 and 0.3 points, respectively, while LiveMath remains unchanged. In this setting, $K = 4$ captures most of the observed improvement with half the rollouts per task required by $K = 8$, so we use it as the default setting.

Agent harnesses. To evaluate TextualRL beyond direct chat, we run GPT-5.5 in the Codex and Claude Code harnesses on five benchmarks, excluding ALFWorld. Figure 3 compares the resulting overall scores across skill sources. TextualRL leads both comparisons, exceeding Base Model by 26.1 and 20.8 points and SkillOpt by 1.3 and 1.7 points, respectively. These gains show that context optimization complements the capabilities provided by existing agent harnesses, improving performance within both tested execution interfaces.

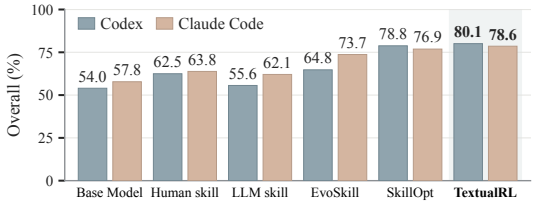

Figure 3: Harness Overall performance (%).

Skill transfer. To test whether learned context transfers across target models, we apply skills learned with Qwen3.8-27B as target and GPT-5.5 as optimizer to Qwen3.6-35B-A3B and Qwen3.5-4B without further optimization. Table 5 compares these transferred skills with skills optimized independently for each receiving model (Origin). Transfer yields slightly higher overall scores on both models. This supports sharing context across the tested models to avoid separate optimization runs.

Table 5: Transfer Overall (%).

Target model	Skill	Overall
Qwen3.6-35B-A3B	Origin	67.3
	Transferred	67.8
Qwen3.5-4B	Origin	60.1
	Transferred	60.3

Optimizer choice. To examine how optimizer choice affects context optimization, we fix Qwen3.8-27B as the target and compare GPT-5.5 with Qwen3.8-27B as the optimizer (Table 6). GPT-5.5 achieves the higher overall score and produces higher scores on all six benchmarks. The Qwen optimizer still improves over Base Model on every benchmark, averaging a 14.1 point gain. Thus, optimizer choice influences the quality of learned context, while using the same model for both roles remains effective. Appendix C.5 provides full benchmark results for these three experiments.

Table 6: Optimizer (%).

Optimizer	Overall
GPT-5.5	71.0
Qwen3.8-27B	69.7

Training dynamics. To examine when context optimization yields gains and whether additional epochs continue to improve unseen performance, we track training, validation, and test scores across four epochs on Qwen3.8-27B. Figure 4 shows that most test improvement occurs early, with different gains across benchmarks. By epoch 2, DocVQA improves from 88.5% to 93.9%, and LiveMath from 34.4% to 54.4%. Their validation scores then plateau, while test scores fluctuate within 93.6–94.4% and 54.4–56.0%, respectively. OfficeQA reaches 47.1% on test after the first epoch and shows no sustained improvement thereafter, despite a validation score of 62.5%. Thus, further optimization does not consistently improve unseen performance. On OfficeQA and LiveMath, validation scores improve much more than test scores. This highlights a generalization challenge, instructions favored by a fixed validation set may be less effective on unseen tasks.

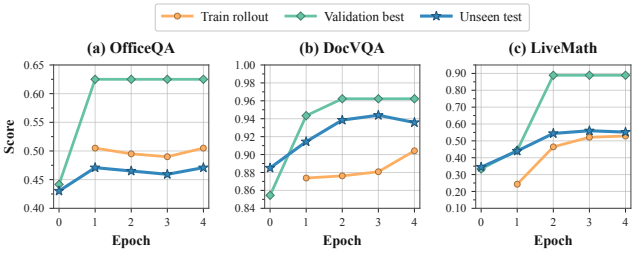

Figure 4: Epoch curves for TextualRL on Qwen3.8-27B.

5 Conclusion

We introduce TextualRL, a gradient-free reinforcement learning framework for optimizing the textual context of a frozen LLM agent through task rewards. Groupwise Policy Critique uses same-task rollout comparisons and cross-task analysis to propose context edits according to the available outcome patterns. Cross-Group Policy Update reviews these proposals against pooled trajectory evidence, including observations that produce no edit, to refine their content and applicability before validation-based selection. Across five target models and six benchmarks, TextualRL achieves the highest six-benchmark average for every target. Ablation studies and additional experiments support the effectiveness of both modules and the consistency of the observed improvements.

AI USE STATEMENT

We used generative AI tools to polish the language and improve the clarity of the manuscript.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A describe the optimization procedure, including task grouping, evidence review, and textual patch application. Section 4 and Appendix B report the benchmark splits, evaluation metrics, model configurations, optimization budgets, generation settings, and validation-based checkpoint selection. We provide the method-specific optimizer prompts in Appendix C.3 and a complete update example in Appendix C.4, illustrating how trajectory evidence informs proposals, review, and the final context update.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026. URL https://arxiv.org/abs/2603.02766.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017. URL https://arxiv.org/abs/1704.05179.

Linyang He, Qiyao Yu, Hanze Dong, Baohao Liao, Xinxing Xu, Micah Goldblum, Jiang Bian, and Nima Mesgarani. LiveMathematicianBench: A live benchmark for research-level mathematical reasoning with proof sketches. *arXiv preprint arXiv:2604.01754*, 2026. URL https://arxiv.org/abs/2604.01754.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. doi: 10.1613/jair.301. URL https://doi.org/10.1613/jair.301.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f1cf02ce09757f57c3b93c0db83181e0-Paper-Conference.pdf.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 94871–94908. Curran Associates, Inc., 2024. doi: 10.52202/079017-3007. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ac840df270ac537dd74530a15c332684-Paper-Datasets_and_Benchmarks_Track.pdf.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad

Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. DocVQA: A dataset for VQA on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2200–2209, January 2021. URL https://openaccess.thecvf.com/content/WACV2021/html/Mathew_DocVQA_A_Dataset_for_VQA_on_Document_Images_WACV_2021_paper.html.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026. URL https://arxiv.org/abs/2603.25158.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning. *arXiv preprint arXiv:2603.08655*, 2026. URL https://arxiv.org/abs/2603.08655.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, pp. 94327–94354, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/980ea04d23d1f6908964eba2a74afe45-Abstract-Conference.html.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

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.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106. Association for Computational Linguistics, March 2026. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026. URL https://arxiv.org/abs/2605.23904.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y. Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

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*, volume 38, pp. 19632–19642. AAAI Press, March 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=92gvk82DE-.

A Method Implementation Details

Algorithm 1 A step of TextualRL with fixed target parameters

Require: Context C, recorded validation score s, training batch B, trajectories per task K, validation set $\mathcal{V}$, edit budget L, shared history H

For each $x \in B$, collect D_x under C and compute $\mathbf{r}_x$ (Equation 2)
Route each D_x using $\mathbf{r}_x$, then consolidate duplicate trajectories within the task
Form groups $\mathcal{G}$: one per mixed task and multiple tasks per homogeneous group
In parallel, obtain (P_g, E_g) for each $g \in \mathcal{G}$ from $\{(x, D_x, \mathbf{r}_x) : x \in g\}$, C, and H
Pool proposals P and valid current-step cards E independently
Consolidate P hierarchically and review the candidates against E to obtain $\widetilde{P}$
Select at most L edits from $\widetilde{P}$ and filter dependencies on unavailable runtime information
if no edits remain **then**
    return C, s ▷ No candidate update
end if
Assemble ΔC and apply it to obtain $C' \leftarrow C + \Delta C$
Evaluate $v' \leftarrow \widehat{V}(C')$ on $\mathcal{V}$
if $v' > s$ **then**
    Adopt C', v' and update the best validation-scored snapshot when improved
else
    Retain C, s and record the rejected edit combination and its score
end if
Add rollout and rejection feedback to H for subsequent proposals

Proposal consolidation. Mixed-task and all-failure proposals enter the corrective branch, while all-success proposals enter the preservation branch. The optimizer recursively merges proposal batches within each branch before combining the two branches. The final merge prioritizes corrective edits unless they conflict with a supported success pattern.

Edit application. Patch operations are applied deterministically while preserving protected scaffold regions. The per-edit dependency filter uses explicit patterns with limited negation handling to detect dependencies on information unavailable during target execution, such as gold answers, grader feedback, or validation scores. It removes an affected edit as a whole and retains other valid edits from the same proposal.

Evidence provenance. We retain evidence cards whose task and rollout references resolve to the displayed trajectories and whose supporting quotes occur in the referenced trace. Repeated cards for the same task and trajectory are consolidated. References attached to reviewed edits are checked against the pooled cards. If no valid cards remain, the optimizer uses ordinary proposal merging.

Repeated proposals. The step buffer screens exact or near-equivalent repetitions of a rejected edit combination at the combination level. An individual edit family is blocked only after it appears in at least three distinct rejected candidates and no accepted candidate.

B Implementation and Data Notes

B.1 Data Splits and Metrics

Table 7 summarizes the data splits and evaluation metrics for all six benchmarks, with counts referring to the experimental subsets. We report binary task accuracy or success rate in percentages. SpreadsheetBench requires all test cases for a task to pass. DocVQA uses hard accuracy, with ANLS ≥ 0.999 counted as correct. Overall performance is the unweighted mean across the six benchmarks.

For ALFWorld, we use the official SkillOpt release with 39 training, 18 validation, and 134 test environments. The released validation manifest selects 18 environments from the full `valid_seen`

pool of 140, whereas the SkillOpt (Yang et al., 2026) describes 140 validation environments. Our update acceptance and checkpoint selection use the 18 environments in the released manifest.

Table 7: Benchmark splits and evaluation metrics.

Benchmark	Train	Validation	Test	Metric
SearchQA	400	200	1,400	Normalized exact match
SpreadsheetBench	80	40	280	All-case success
OfficeQA	50	24	172	Normalized exact match
DocVQA	107	53	374	Hard accuracy
LiveMath	35	17	125	Choice accuracy
ALFWorld	39	18	134	Task success

B.2 Critique and Merge Batches

In full TextualRL, the critique minibatch size of eight counts distinct task blocks, whereas the merge minibatch size counts proposal bundles. Each mixed task is analyzed separately with all of its unique sibling trajectories. Within each homogeneous route, task blocks are grouped into batches of eight. A final batch containing two to seven tasks is processed as is. If the final batch contains only one task, it is combined with the preceding batch, producing a batch of nine. A homogeneous route containing only one task is skipped.

Outcome routing uses the original K rollout outcomes before deduplication. Duplicate trajectories are consolidated only within the same task and outcome label, using the recorded conversation with whitespace normalized and timing and request metadata removed. Each retained trajectory keeps its multiplicity and original rollout references. Deduplication reduces the trajectories shown to the critic but does not change K, the task's outcome category, or the number of distinct tasks supporting a proposal.

Hierarchical merging processes up to eight proposal bundles per call, and each bundle may contain multiple edits. A final batch of two to seven bundles is merged normally. A singleton bundle passes directly to the next level without a merge call. These batch sizes are separate from the edit budget, which bounds the number of edits applied to form the candidate context.

For *w/o outcome routing*, we pool the step's trajectories and form analyst minibatches of eight trajectories. We adapt the mixed-outcome analyst prompt by removing the requirements that trajectories belong to a single task and include both successes and failures. The resulting unified prompt asks the analyst to produce edit proposals and evidence cards from the available trajectories and outcomes.

In *intra-group cards only*, each critique group $g \in \mathcal{G}_t$ corresponds to one analyst call. Separate calls within the same outcome route remain distinct groups. The ablation retains hierarchical consolidation followed by evidence review during the final merge. For each merged edit, review is restricted to the union of evidence cards from the critique groups that contributed its source proposals. For example, an edit formed from proposals by groups A and B can use $E_A \cup E_B$, while cards from other groups are excluded.

B.3 Generation and Evaluation

Training rollouts use a temperature of 0.7, while validation and test inference use a temperature of 0. Top-p is left at the serving backend's default. The random seed is set to 42. GPT-5.5 and GPT-5.4 use `medium` reasoning effort.

For each target–benchmark pair, context optimization collects $K = 4$ rollouts per training task under the same current context, except in the rollout-count ablation. Validation selects the context used for test evaluation. The TextualRL scores in Table 1 are from individual optimization runs. The run-stability experiment separately repeats the complete optimization and validation-based checkpoint selection three times for TextualRL and SkillOpt on Qwen3.8-27B, with fixed data splits. Its reported variation is across independently optimized contexts, rather than repeated evaluations of a single context.

Table 8: Token usage under the Qwen3.8-27B as target model with GPT-5.5 as optimizer. Iteration usage includes candidate validation. Inference usage sums input and visible output text per task.

Benchmark	Method	Final tokens	Recorded iteration tokens (M)			Inference text tokens / task
			Target	Optimizer	Total	
SearchQA	SkillOpt	937	31.33	7.67	39.00	2,698.2
	TextualRL	2,422	66.26	13.57	79.83	5,035.4
Spreadsheet	SkillOpt	2,908	49.62	4.99	54.60	9,044.2
	TextualRL	1,397	32.99	8.35	41.34	4,145.6
OfficeQA	SkillOpt	1,589	44.18	4.70	48.88	105,784.5
	TextualRL	974	142.87	7.21	150.08	137,714.1
DocVQA	SkillOpt	415	4.60	0.63	5.22	741.0
	TextualRL	659	10.76	0.79	11.56	1,132.2
LiveMath	SkillOpt	1,534	1.59	1.29	2.88	7,889.8
	TextualRL	1,913	4.98	4.34	9.32	8,410.2
ALFWorld	SkillOpt	516	8.55	0.28	8.83	24,929.9
	TextualRL	1,600	25.21	0.83	26.05	43,786.2

Each rollout starts with a fresh interaction history and the same task inputs, with separate sample identifiers for its saved outputs. Spreadsheet rollouts operate on independent copies of the input workbook. OfficeQA rollouts start new conversations over the supplied documents. ALFWorld rollouts initialize and reset separate environments using the task's game file, with environment seeds derived from the batch seed and environment index.

B.4 Checkpoint Selection

For each run, we select the context with the highest recorded validation score for test evaluation. We track this checkpoint separately from the active context after the final update. Checkpoint selection uses validation scores exclusively, and the selected context is then supplied to the frozen LLM agent for test tasks.

B.5 Token Accounting

The settings and selected skills for Table 8 follow the corresponding Qwen3.8-27B experiments in Table 1, with GPT-5.5 as optimizer. Final refers to the skill selected for inference. Table 8 reports recorded iteration subtotals. Target counts sum input and output tokens for training rollouts and candidate validation within saved steps. Optimizer counts cover optimization calls in those same steps. Totals include all saved steps, including those after the selected checkpoint. The accounting scope excludes subsequent unseen evaluation, initial validation outside these steps, and unrecorded recovery or retry usage. M denotes one million tokens. Totals are calculated from the original integer counts, and each displayed column is rounded independently.

Inference counts use the Qwen3.8-27B tokenizer with `add_special_tokens=False`. Input message text is accumulated across all saved decisions for each task, with repeated history counted at each decision. Output counts include the saved visible model text. Their sum is calculated before rounding and averaged over tasks with saved trajectories. Each token contributes equally to this measure of recorded text volume, independent of token prices, caching, or computational cost.

TextualRL records 2.05–3.24 times SkillOpt's iteration tokens on five benchmarks. Collecting $K = 4$ trajectories per task incurs sampling cost to obtain comparative feedback, analogous to group-based RL such as GRPO (Shao et al., 2024). Trajectory analysis, evidence review, and candidate validation also consume optimization tokens. Review reuses existing rollouts. At inference, these optimization procedures are absent, and token usage depends on the learned context and execution history. The corresponding increases are smaller, at 6.6–86.6%. Spreadsheet instead uses 24.3% fewer recorded iteration tokens and 54.2% fewer inference text tokens.

B.6 EVIDENCE REVIEW AUDIT

To examine how review uses recorded evidence, we audit run with Qwen3.8-27B as target and GPT-5.5 as optimizer across all six benchmarks. We inspect all 76 saved steps, containing 464 merged edits and 578 raw evidence references. We randomly sample up to 20 reviewed edits per benchmark with seed 42, including all 13 available LiveMath edits, yielding 113 edits and 153 references. Two separate GPT-6 Astra sessions independently assess the same sample using an identical rubric, without seeing the original review labels, validation outcomes, or each other's judgments. Each assesses card faithfulness to the available runtime text, the card's relation to the final reviewed edit, and the grounding of the edit's scope.

Table 9: Evidence review audit on Qwen3.8-27B runs.

Audit item	Result	Measurement
Reference traceability	99.13% (573/578)	Card resolves and its quote matches the source trajectory.
LLM relation agreement	94.77% (145/153)	Agreement between two GPT-6 Astra assessments, $\kappa = 0.715$.
Reported review changes	8.48% (37/436)	Optimizer reports `narrow` or `revise`.

Table 9 separates source traceability from model assessment. References are checked against the current-step card pool after the existing card validation procedure, and all raw review references remain in the denominator. Relation labels are *supports*, *limits*, *contradicts*, *irrelevant*, and *unclear*, always evaluated against the final reviewed edit. The two assessments label 134/153 card statements as supporting the rule. Agreement measures consistency between two evaluations by the same model, rather than correctness against human judgments.

These findings support the auditability of Cross-Group Policy Update. Traceable references allow readers to inspect the observations cited for individual rules, while the two assessments agree on a supporting relation for most sampled references (87.58%). The review therefore exposes links between evidence and edits that can be examined beyond the final task score. The optimizer reports narrowing or revision for 37 of the 436 edits with review metadata. This pattern is consistent with review serving as a selective refinement stage, where proposals can be retained after evidence comparison and changes can target their content or applicability.

Effect of reviewed updates. To examine whether evidence review improves task performance beyond proposal consolidation, we evaluate three context variants for one selected update each on SearchQA, OfficeQA, and DocVQA, using Qwen3.8-27B as the target and GPT-5.5 as optimizer. *Old skill* is the context before the selected update. *New skill w/o review* applies the consolidated proposal without evidence review, while *New skill* applies the corresponding reviewed edit. Each comparison uses the same historical context and unseen-test split.

Table 10: Unseen-test performance (%) of three context variants for selected updates.

Benchmark	Old skill	New skill w/o review	New skill
SearchQA	81.57	82.07	**82.71**
OfficeQA	45.35	45.93	**47.09**
DocVQA	89.57	90.37	**91.44**

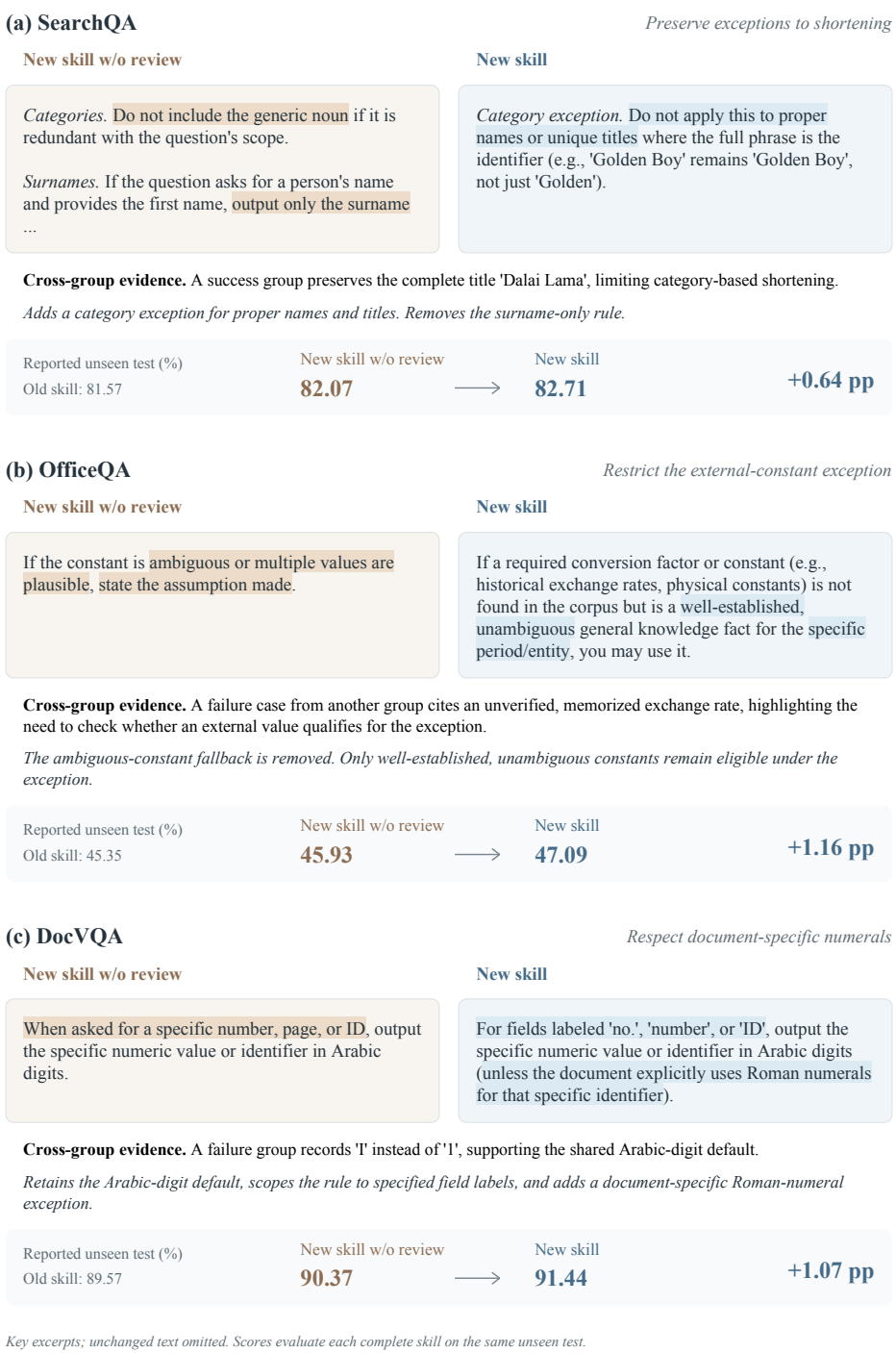

Figure 5: Key rule changes before and after evidence review for the updates in Table 10.

Figure 5 illustrates how review refines the applicability of consolidated proposals. SearchQA protects proper names and titles during category extraction and removes the surname-only rule. OfficeQA removes the ambiguous-constant fallback while retaining a qualified exception for external constants. DocVQA preserves the Arabic-digit default, specifies the applicable field labels, and permits document-specific Roman numerals. In all three selected updates, review improves on an already beneficial proposal (Table 10) while clarifying the conditions and exceptions in the resulting guidance.

C EXPERIMENTAL RESULTS

C.1 TRAINING DYNAMICS

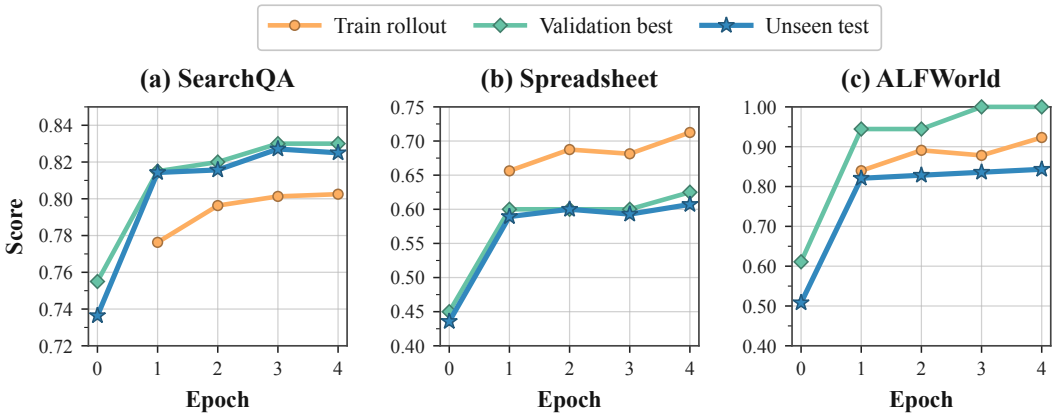

Figure 6: Epoch curves for TextualRL on SearchQA, Spreadsheet, and ALFWorld with Qwen3.8-27B. Epoch and score definitions follow Figure 4.

Epoch 0 denotes the Base Model. The training curve reports the mean rollout score within each epoch. Validation and test scores correspond to the best checkpoint selected by validation up to that epoch. Figure 6 shows that SearchQA and Spreadsheet also obtain most of their test gains in the first epoch, increasing from 73.6% to 81.4% and from 43.6% to 58.9%, respectively. Subsequent changes are smaller, with final scores of 82.5% and 60.7%. ALFWorld's reported test score increases across all four epochs, from 50.8% to 84.3%, while validation reaches 100% at epoch 3. Together with Figure 4, these curves show that the timing and magnitude of improvements vary by benchmark. A validation-selected checkpoint can remain unchanged across epochs, with repeated test evaluations yielding different scores.

C.2 Optimized Contexts

To illustrate what TextualRL encodes in the optimized context, Figure 7 presents one instruction excerpt from each of the six benchmarks for Qwen3.8-27B. These examples specify concrete operations, such as verifying source values before arithmetic and checking inventory after a failed action. They also define when a rule applies, including when to write static spreadsheet values and when a theorem supports a claimed bound.

(a) SearchQA
Exact answer extraction

Content: If a formatted document provides a *Clue* | *Answer* pair, extract the Answer span exactly.

(b) Spreadsheet
Formula handling

Content: Unless the instruction explicitly states the cell must contain a live formula that recalculates, compute the result in Python and write the static value (number, string, date) to the cell.

(c) OfficeQA
Operand verification

Content: For multi-step computations, first write out a complete operand ledger listing every value with its period and source, verify the ledger against the retrieved table, and only then perform the arithmetic.

(d) DocVQA
Table grounding

Content: For tables, identify the specific row label and column header mentioned in the question, then extract the value at their intersection.

(e) LiveMath
Theorem scope

Content: Be cautious about asserting uniform-in-time or global bounds (e.g., constant C independent of T) unless the theorem explicitly provides them.

(f) ALFWorld
State recovery

Content: If a *take* or *move* action returns "Nothing happens," immediately issue an *inventory* action to confirm what you are currently carrying.

Figure 7: Representative instructions from the optimized contexts for six benchmarks. Original wording is preserved, with formatting standardized for readability.

C.3 Optimizer Prompts

The boxes below reproduce the method-specific system-prompt blocks used by TextualRL, with line wrapping and formatting adjusted for readability. In the implementation, *skill* denotes the editable context C. Each outcome-specific block defines the corresponding critique task, and all three routes share the evidence-card instruction. The final block guides proposal review against the pooled evidence.

C.3.1 Outcome Analysis

The user message supplies the current context C_t, the edit budget, within-epoch history H_t, and the assigned task trajectories with their outcomes. The analyst returns a JSON object containing a proposed `patch` and `cross_group_cards`.

Mixed outcomes: within-task contrast

This call contains all unique trajectories from exactly one task whose K sibling rollouts include both successes and failures. Compare complete trajectories, locate the first observable behavioral divergence, and emit at most one atomic repair or preservation rule. The rule must be supported by the within-task contrast rather than by task literals or final evaluator labels. If no plausible causal divergence is visible, return an empty patch.

All-success outcomes: cross-task preservation

Each Task Block is one independent task whose sibling rollouts all succeeded. Multiple unique trajectories inside one Task Block are correlated alternatives, not independent votes. First compare Task Blocks semantically. Emit a rule only for an inference-time behavior shared by at least two distinct task_ids. Treat the result as preservation evidence, not proof that the behavior caused success. If no cross-task invariant is visible, return an empty patch. Set `evidence_kind` to `consensus` and `generality_scope` to `cross_task`.

All-failure outcomes: cross-task repair

Each Task Block is one independent task whose sibling rollouts all failed. Multiple unique trajectories inside one Task Block are correlated attempts, not independent votes. First cluster Task Blocks by a shared, observable failure mechanism. Emit a cautious repair hypothesis only when at least two distinct task_ids expose the same mechanism. Do not infer a correct answer or task-specific procedure from failure alone. If no cross-task mechanism is visible, return an empty patch. Set `evidence_kind` to `consensus` and `generality_scope` to `cross_task`.

C.3.2 Evidence and Review

The shared evidence instruction is included in all three critique routes. For cross-group review, the user message supplies C_t, the consolidated candidate patches, and the pooled current-step cards E_t. The review status `insufficient_evidence` retains the proposal's original scope and uncertainty, as described in Section 3.3.

Shared instruction: trajectory evidence cards

Keep the current route's proposal rules and edit budget. In the same JSON response, alongside patch, also return cross_group_cards, a list of objects: task_id, rollout_id, quote, condition, behavior, observed_effect. For each displayed unique trajectory, report at most ONE salient behavioral observation, even when no edit is warranted (return patch with an empty edits list in that case). quote must be a short verbatim runtime action/observation span, up to 800 characters. Describe observable preconditions, the action, and its visible effect in at most 300 characters each. Use "unknown" when the trace does not establish a condition or effect. Do not invent a successful repair. Choose behavior relevant to a proposed repair or worth preserving. For mixed groups expose both sides of the divergence; for homogeneous groups describe the actual observed behavior, not a presumed cause of the final score. Only cite Task Block IDs and displayed trajectory IDs in this call. Do not cite gold answers, evaluator feedback, final grading labels, hidden siblings, or previous-step summaries as runtime evidence. Cards are evidence for later semantic comparison, not extra edits or independent votes. Do not put any card metadata or task IDs into executable skill content.

Cross-Group Policy Update: evidence-based proposal review

Alongside the candidate edits, you receive behavioral evidence from the current training step's mixed, all-success, and all-failure task groups. Associate each candidate ONLY with semantically relevant actions and observable preconditions. Benchmark names, task-ID similarity, and matching outcome labels are not evidence of relevance. Do not force all three group types to support every candidate. Mixed evidence can suggest an improvement direction. Independent failure tasks can corroborate an observed failure mechanism, not prove a repair. Successful behavior can support preservation or LIMIT a repair's scope: do not prohibit a behavior merely because it also appears in a failed task. Distinguish a genuine contradiction under comparable conditions from different valid preconditions. Use an observable condition in executable content when narrowing a rule; keep supported exceptions and unrelated successful behavior intact. Only revise an existing proposed behavior; do not invent unrelated edits to use every card. If no relevant cross-group evidence exists, keep the original evidence scope and uncertainty rather than inventing support or discarding the edit solely for missing evidence. All-success and all-failure proposals remain eligible.
Maintain the normal output patch format and edit budget. For EACH output edit, add training-only cross_group_review with decision (retain, narrow, revise, or insufficient_evidence), reason, and evidence (a list of task_id, rollout_id, relation [supports, limits, contradicts], reason). Cite only cards actually used. Correlated siblings are not independent task votes. These records do not automatically change support_count, ranking, acceptance, or causal confidence. Keep metadata out of skill content. Existing protected sections and runtime observability restrictions still apply.

C.4 Complete Update Example

Figure 8 traces an accepted SearchQA update using GPT-5.5 as both the frozen target and the optimizer. For the same declarative clue, three trajectories identify Henrik Ibsen, while one returns `True`. The mixed analyst proposes resolving the implied entity and reserving Boolean answers for explicit verification questions. The optimizer reviews the consolidated proposal against current-step evidence from all three outcome groups and retains this rule alongside complementary extraction guidance. The resulting patch is appended to a context with no learned rules and accepted after validation accuracy rises from 69.0% to 77.0%. This improvement is measured for the complete patch.

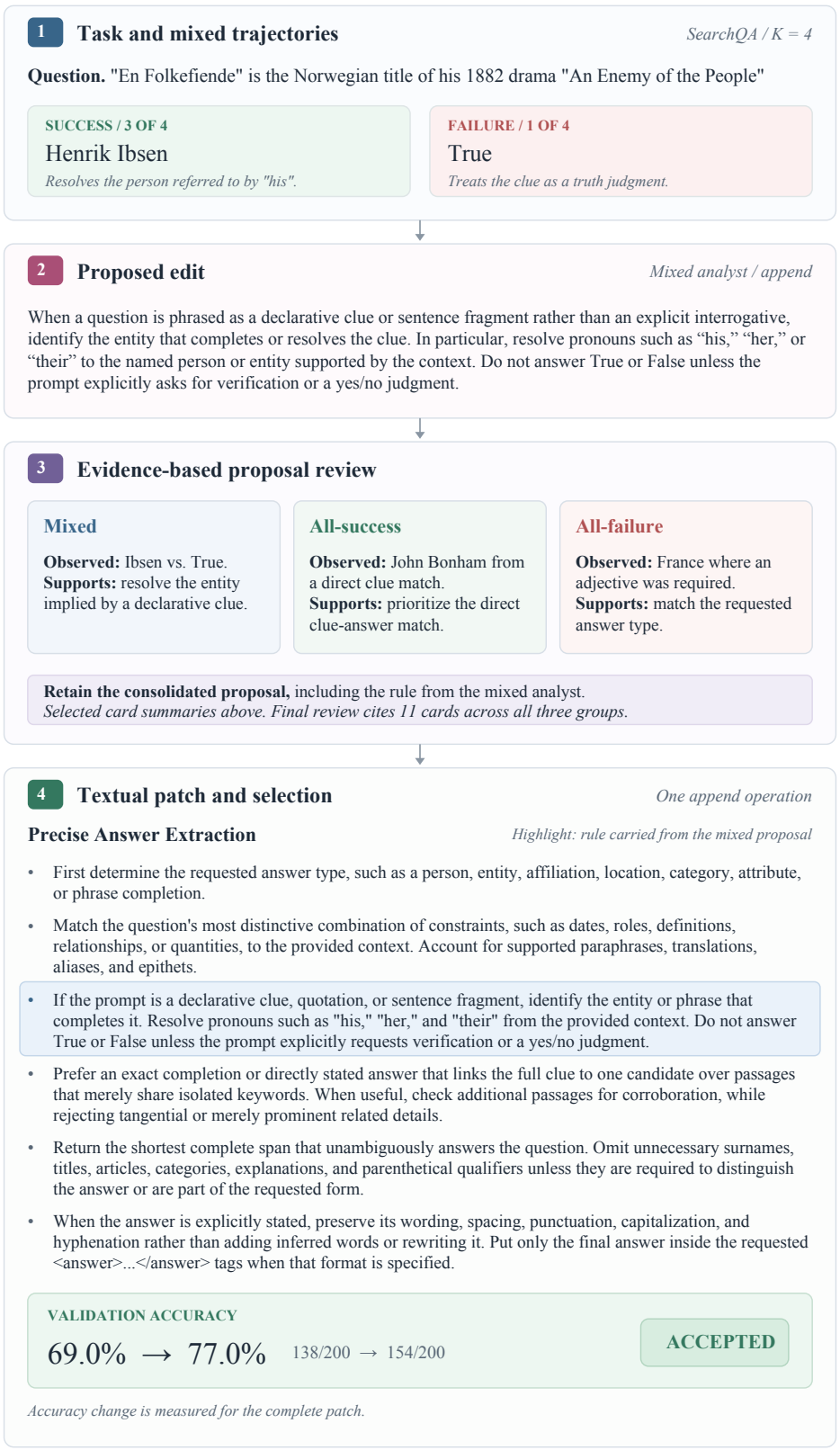

Figure 8: A complete context update example on SearchQA.

C.5 Detailed Additional Results

The following tables report the complete benchmark results underlying the overall scores in Figure 3 and Tables 5–6. Harness averages use five benchmarks, while skill transfer and optimizer comparisons use all six.

C.5.1 Agent harnesses

To evaluate TextualRL beyond direct chat, we use GPT-5.5 within the Codex and Claude Code harnesses on five benchmarks, excluding ALFWorld. Table 11 includes the harness baselines reported by SkillOpt, with EvoSkill (Alzubi et al., 2026) as an additional skill evolution baseline. TextualRL achieves five-benchmark averages of 80.1% and 78.6% under Codex and Claude Code, respectively, improving over Base Model by 26.1 and 20.8 percentage points. Relative to SkillOpt, the average gains are 1.3 and 1.7 points. TextualRL obtains the highest score in nine of the ten harness–benchmark settings, the exception is Codex LiveMath, where TextualRL scores 76.8% versus SkillOpt's 78.4%. These results support the effectiveness of context optimization across both agent harnesses.

Table 11: Test results (%) with GPT-5.5 under different agent harnesses.

Harness	Skill Source	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath
Codex	Base Model	81.8	27.5	38.3	87.2	35.2
	Human skill	84.1	50.7	40.0	88.8	48.8
	LLM skill	83.4	25.0	34.3	89.8	45.6
	EvoSkill	61.4	67.5	42.4	89.3	63.2
	SkillOpt	87.3	85.0	51.1	92.2	**78.4**
	TextualRL	**88.6** +6.8	**85.7** +58.2	**56.4** +18.1	**92.8** +5.6	76.8 +41.6
Claude Code	Base Model	81.9	22.1	57.6	86.6	40.8
	Human skill	83.7	37.1	66.3	88.0	44.0
	LLM skill	82.4	37.1	56.4	89.6	44.8
	EvoSkill	84.0	75.0	70.3	87.2	52.0
	SkillOpt	85.9	80.4	71.5	90.1	56.5
	TextualRL	**87.1** +5.2	**83.9** +61.8	**73.3** +15.7	**90.9** +4.3	**57.6** +16.8

C.5.2 Skill transfer

To test whether learned skills remain effective beyond the model used during optimization, we transfer the benchmark-specific skills learned with Qwen3.8-27B as the target and GPT-5.5 as the optimizer to Qwen3.6-35B-A3B and Qwen3.5-4B. Table 12 compares skills optimized for each receiving model (Origin) with the transferred skills, which are used without further context optimization. Transfer yields a slightly higher overall score on both receiving models and matches the original skill on ALFWorld. The gains are not uniform: OfficeQA decreases by 1.1 percentage points on Qwen3.6-35B-A3B, and LiveMath decreases by 0.8 points on Qwen3.5-4B. These results suggest that the learned guidance can be applied across the tested Qwen models while retaining its overall effectiveness without additional context optimization.

Table 12: Skill transfer from Qwen3.8-27B to two target models (%).

Target model	Skill	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld	Overall
Qwen3.6-35B-A3B	Origin	81.2	52.1	49.4	92.8	44.8	83.6	67.3
	Transferred	81.4	53.2	48.3	93.3	47.2	83.6	67.8
Qwen3.5-4B	Origin	73.9	27.1	32.0	89.3	56.0	82.1	60.1
	Transferred	74.3	27.9	32.0	90.1	55.2	82.1	60.3

C.5.3 OPTIMIZER CHOICE

To examine whether the observed gains depend on using GPT-5.5 as the optimizer, we also optimize skills with Qwen3.8-27B while keeping the target fixed as Qwen3.8-27B. Table 13 reports an overall score of 69.7% with the Qwen optimizer, compared with 71.0% with GPT-5.5. GPT-5.5 produces higher scores on all six benchmarks, with an average advantage of 1.2 percentage points. Nevertheless, the Qwen optimizer improves over the Base Model in Table 1 on every benchmark, with an average gain of 14.1 points. TextualRL therefore remains effective when the same model serves as both target and optimizer in this setting, although the optimizer choice affects the final task performance.

Table 13: Optimizer comparison with Qwen3.8-27B as the fixed target (%).

Optimizer	SearchQA	Spreadsheet	OfficeQA	DocVQA	LiveMath	ALFWorld	Overall
GPT-5.5	82.7	60.7	47.7	94.4	56.0	84.3	71.0
Qwen3.8-27B	81.5	58.6	46.5	93.9	54.4	83.6	69.7