BEYOND THE BEST SKILL: ROUTER–EXPERT COEVOLUTION FOR ADAPTIVE SKILL TREES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents can improve by revising reusable textual skills without updating model weights. However, continued single-skill search can plateau despite useful local repairs. Our analysis identifies *structural interference*: repairs on some inputs are offset by regressions on others when candidates are evaluated as global replacements. We introduce RECAST (Router–Expert Coevolution for Adaptive Skill Trees), which expands an optimized Root skill into an adaptive skill tree by jointly evolving Expert skills and their routing scopes. Residual errors guide specialization, and Experts are refined through repair and harm evaluations relative to their parents on routed examples. Router selection uses end-to-end validation performance with the current Experts, while the selected Router defines the scope for further Expert refinement. Validation guides acceptance and recursive growth. At inference, the frozen tree executes the Root and follows a single routed path, returning the current node's output when routing stops. Across six benchmarks, two Qwen models, and Roots from GEPA and SkillOpt, RECAST improves all 24 paired configurations, with six-task macro-average gains of 12.82 to 15.69 percentage points. On three tasks with GPT-5.4, the macro-average gain is 15.09 percentage points. Ablations support joint adaptation of Routers and Experts, and depth analysis shows further gains from recursive specialization. Across six workflow comparisons, an upstream optimization prefix followed by RECAST achieves higher test performance with fewer optimization and validation tokens than completing upstream single-skill search.

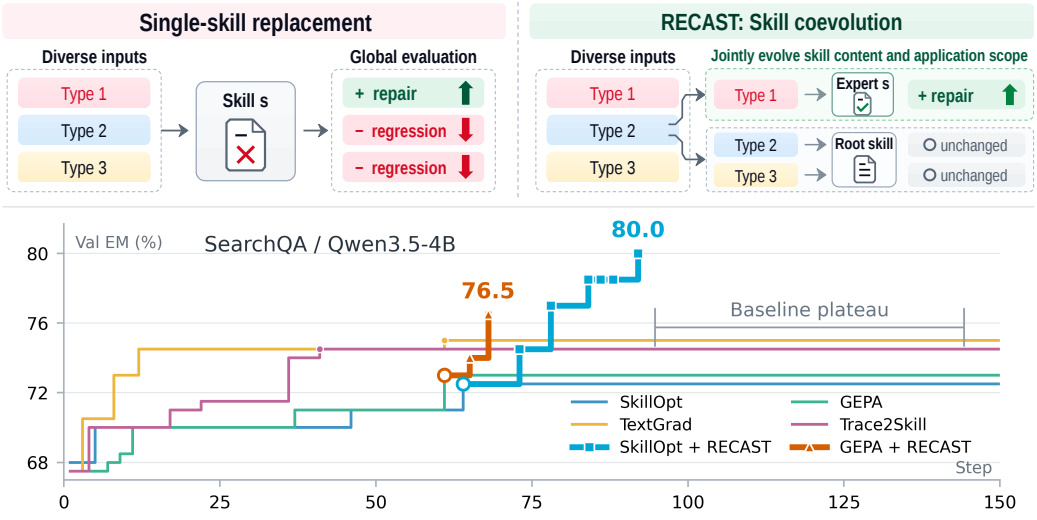

Figure 1: **From global skill replacement to conditional repair. Top:** When a revised skill is applied globally, repairs on some inputs can be offset by regressions on others. RECAST jointly adapts Expert skills and routing scopes to retain useful local repairs while leaving other inputs with the Root. **Bottom:** Validation exact-match accuracy on SearchQA with Qwen3.5-4B. Thin lines show retained best scores in single-skill search. Thick lines show continued optimization with RECAST, reaching 80.0% (SkillOpt) and 76.5% (GEPA).

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

Large language model (LLM) agents can improve by revising reusable textual skills that encode task strategies, tool-use procedures, and validation checks without updating model weights. Recent work automates this self-evolution process by learning skills from execution trajectories or refining rules through feedback (Agrawal et al., 2025; Ni et al., 2026; Yang et al., 2026; Zhang et al., 2025). These methods iteratively search over candidate textual skills, raising a central question: **does continued single-skill search yield sustained performance gains?**

To address this question, we examine the behavior of single-skill search through extended optimization runs across multiple benchmarks. Figure 1 (bottom) shows that, on SearchQA with Qwen3.5-4B, SkillOpt, GEPA, TextGrad, and Trace2Skill reach prolonged plateaus in their best validation scores despite continued search. Continued search during these plateaus consumes optimization resources without improving the best validation score. This motivates examining whether the plateaus conceal useful local improvements.

Figure 2(a) shows that 37 of 39 SkillOpt candidates on SearchQA with Qwen3.5-4B repair Root errors, yet none outperforms the Root. Regressions elsewhere offset these repairs, revealing *structural interference* under global skill replacement. Figure 2(b) further shows that an offline oracle selecting candidates per input outperforms the Root by 10 percentage points, revealing complementary repair capabilities across candidates. While existing frameworks retain diverse candidates during search (Agrawal et al., 2025), collapsing them into a single deployed skill forfeits these complementary strengths. These findings motivate applying specialized skills where they can repair errors and retaining the Root elsewhere, as illustrated in Figure 1 (top).

To realize this transition, we introduce RECAST (*Router–Expert Coevolution for Adaptive Skill Trees*), which expands an optimized Root skill into a tree of specialized textual skills. Its central idea is to jointly evolve *what a skill does* and *when it is applied*. Detecting a failure alone does not establish which Expert can repair it, and an Expert's repair capabilities change as its skill evolves. Routing scopes and Expert skills must therefore be adapted jointly. A Router identifies potential failures from execution trajectories, and recurring error mechanisms guide Expert specialization. Experts evolve through parent-relative repair and harm evaluations on routed examples. Router candidates are ranked by end-to-end validation performance with the current Experts, and the selected Router defines the scope for further Expert refinement. This process recurses within accepted Experts, with validation controlling candidate acceptance and further growth. At inference, the agent executes the Root and follows a single routed path, returning the current node's output whenever routing abstains. The SkillOpt and GEPA results in Figure 1 (bottom) illustrate validation gains beyond the single-skill search plateaus.

Across six benchmarks spanning question answering, document understanding, spreadsheet manipulation, mathematical reasoning, and interactive planning, RECAST consistently improves upon all 24 evaluated task–model–optimizer configurations using two Qwen models and Roots derived from GEPA or SkillOpt. Six-task macro-average gains over paired Roots range from 12.82 to 15.69 percentage points, while a three-task GPT-5.4 extension yields a 15.09-point macro-average gain. Full coevolution outperforms one-shot construction and single-component updates in macro-average performance. Across six workflow comparisons, an upstream optimization prefix followed by RECAST achieves higher test scores with fewer optimization and validation tokens than completing upstream search.

Our contributions are threefold:

- We identify prolonged performance plateaus under continued single-skill search and reveal structural interference, where local repairs are offset by regressions when candidates are evaluated as global replacements.
- We introduce RECAST, which jointly evolves Expert skills and their routing scopes to grow adaptive skill trees from optimized Roots. It couples parent-relative Expert refinement with Router selection guided by end-to-end validation performance.
- We demonstrate consistent gains across tasks, target models, and upstream optimizers. Coevolution ablations establish the benefits of jointly adapting Routers and Experts, while workflow comparisons show higher test scores with fewer optimization and validation tokens.

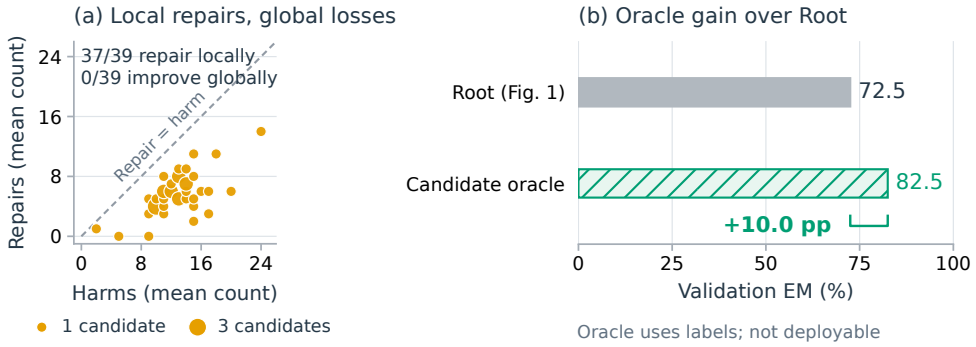

Figure 2: **Complementary repairs among SkillOpt candidates** on SearchQA with Qwen3.5-4B. **(a)** Repairs and harms relative to the optimized SkillOpt Root, with equality on the diagonal. **(b)** Validation accuracy of the same Root as in Figure 1 (72.5%) and an offline oracle selecting candidates per input (82.5%).

2 Related Work

Prompt and agent optimization. Automatic prompt optimization improves LLM behavior by searching over external text with task feedback. OPRO and Promptbreeder explore candidate instructions, while ProTeGi uses textual critiques to guide prompt updates (Yang et al., 2024; Fernando et al., 2024; Pryzant et al., 2023). DSPy optimizes LM programs, and TextGrad propagates language feedback through computational graphs (Khattab et al., 2024; Yüksekgönül et al., 2025). GEPA combines trajectory reflection with diverse candidate search, retaining candidates with complementary strengths and combining useful lessons from the search history (Agrawal et al., 2025). These approaches establish how execution feedback can improve the textual components of an agent. Building on this foundation, we examine the limits of continued search under single-skill deployment.

Agent skill learning and evolution. LLM agents retain execution experience as reflective memory, executable skill libraries, and evolving contextual playbooks (Shinn et al., 2023; Wang et al., 2024; Zhang et al., 2025). Recent work turns this experience into reusable procedural knowledge: Trace2Skill distills trajectory-local lessons, while EvoSkill constructs and revises skills through failure analysis (Ni et al., 2026; Alzubi et al., 2026). CoEvoSkills uses verification feedback to guide skill improvement, and FlowEvo accumulates executable skills from successful workflows (Zhang et al., 2026; Ren et al., 2026). SkillCAT evaluates candidate patches through replay and organizes accepted knowledge for task-specific use (Chen et al., 2026). SkillOpt treats a persistent skill document as optimizable state, using bounded textual edits, held-out validation, and feedback from rejected updates to control the optimization process (Yang et al., 2026). Our work studies continued improvement from an already optimized skill. In the single-skill deployment setting studied here, a candidate can repair some errors yet fail as a global replacement because it introduces regressions elsewhere. RECAST preserves these complementary local repairs as specialized skills and learns where they improve upon an optimized parent, allowing useful improvements to accumulate beyond the best single skill.

3 Motivating Analysis

We examine why continued single-skill search plateaus and whether candidate revisions retain useful local repairs.

3.1 Prolonged plateaus under continued single-skill search

Figure 1 (bottom) shows prolonged plateaus in the best validation scores of SkillOpt, GEPA, TextGrad, and Trace2Skill on SearchQA with Qwen3.5-4B. For SkillOpt and GEPA, candidate scores continue to fluctuate without improving the retained best (Appendix Figure 5), so continued search consumes

resources without aggregate gains. Similar plateaus across other tasks and models are summarized in Appendix Table 8.

Aggregate scores can mask local repairs offset by regressions elsewhere. We therefore examine candidate behavior on individual examples.

3.2 Structural interference and the value of conditional selection

To examine what aggregate scores conceal, we compare candidate and Root behavior on individual inputs. We analyze all 39 distinct skills in a SkillOpt candidate pool on SearchQA / Qwen3.5-4B, evaluated on the same 200 validation examples. The optimized SkillOpt Root has 72.50% validation accuracy, as in Figure 1, and serves as the reference for paired repair and harm counts. For each candidate, repeated evaluations are averaged per example before computing repairs, harms, and overall performance.

Figure 2(a) contrasts repairs on Root failures with harms on Root successes. Of the 39 candidates, 37 repair at least one Root error, yet none improves the overall score over the Root. This *structural interference* arises when local repairs are offset by regressions elsewhere, causing global selection to discard skills that remain useful on specific inputs.

Figure 2(b) evaluates an offline oracle that uses reference outcomes to select the candidate with the highest averaged score for each input. It reaches 82.50% validation accuracy, 10 percentage points above the optimized Root, revealing complementary repairs that single-skill deployment leaves unrealized.

These findings motivate evaluating Experts on the inputs where they improve upon their parent. Routing must identify these inputs from deployment-visible evidence. As Expert capabilities evolve, their useful routing scopes can change, motivating joint evolution of Expert skills and Routers.

4 Method

RECAST expands an optimized Root skill into an adaptive skill tree through coevolution of Router and Expert skills, jointly adapting Expert capabilities and their routing scopes. As summarized in Figure 3(a), offline optimization uses Root rollouts to identify residual errors and guide specialization, followed by alternating Expert refinement and Router reselection. Expert updates provide *realized utility* for choosing Routers, while Router updates define the *routed scope* for further Expert refinement. Figure 3(b) illustrates inference with the resulting frozen skill tree. The Executor model M_E and Router model M_R retain fixed weights throughout, while textual skills and the tree structure evolve during offline optimization.

Appendix B gives pseudocode for offline construction and frozen-tree inference, together with the candidate-selection procedure.

4.1 Skill trees and conditional repair

Let $F(x; s)$ denote the output of M_E executing task x with textual skill s, and let $m(F(x; s), y) \in [0, 1]$ measure task performance against reference y. An upstream optimizer $\mathcal{O}$, such as GEPA or SkillOpt, produces the initial skill $s_0 = \mathcal{O}(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$. RECAST uses this optimized skill as the Root of a tree T. Each node v stores a skill s_v, while an internal node also has a textual Router skill r_v that guides M_R in selecting a child or abstaining based on the task and that node's visible execution trajectory. Thus, a skill's contribution depends on both its behavior and the inputs routed to it.

For a parent node p and a candidate Expert c, we define the paired advantage as

$$a_{p\to c}(x, y) = m(F(x; s_c), y) - m(F(x; s_p), y). \tag{1}$$

For binary success metrics, this yields $+1$ for a repair, -1 for a harm, and zero otherwise. On the validation scope routed to c, its net contribution is $\sum a_{p\to c} = W - L$, where W and L count repairs and harms, respectively. This applies the instance-level diagnosis from Section 3.2 directly to skill evolution. Because an Expert is evaluated relative to its parent within its assigned scope, a locally effective skill can be retained even if it is unsuitable as a global replacement.

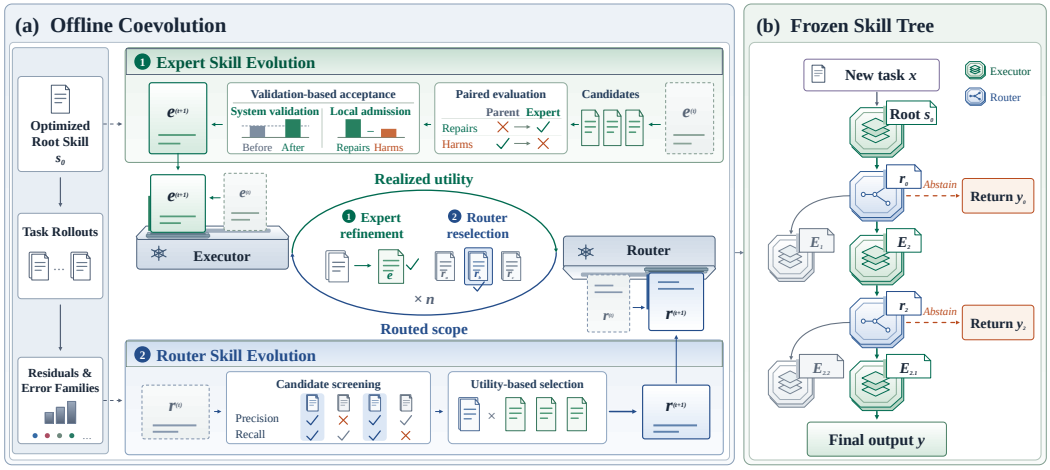

Figure 3: **Overview of RECAST. (a)** Offline coevolution: Expert refinement provides realized utility for Router reselection (green arrow), while the selected Router defines the routed scope for further refinement (blue arrow). **(b)** The frozen tree follows the selected path $s_0 \to E_2 \to E_{2.1}$, leaving other branches inactive. Routing abstention returns the current node's output.

4.2 RESIDUAL DISCOVERY AND INITIAL ROUTING

For the current parent, which is initially the Root, we collect execution trajectories and outputs on training and validation examples. Training scores and diagnostics identify residual errors and guide textual Router proposals. Validation scores rank the candidates. Initial Router selection favors high error recall subject to a validation precision floor, yielding a scope of potential failures alongside concise diagnoses derived from execution evidence. The Router uses only task inputs and deployment-visible trajectories, while reference outcomes serve as offline supervision and selection signals.

Recurring error mechanisms in routed training failures are then summarized into error families that organize Expert specialization. Each family captures an actionable failure mechanism, such as incorrect evidence selection or value normalization. Family induction uses training diagnostics, while assigning an example to a family relies only on deployment-visible information. Families with insufficient sample support remain unexpanded. Because routing may also include examples already solved by the parent, each Expert is evaluated on these examples as part of its assigned scope to account for potential harms.

4.3 ROUTER–EXPERT COEVOLUTION AND RECURSIVE GROWTH

Figure 3(a) shows two coupled updates: *Expert skill evolution* refines behavior on the routed examples, and *Router skill evolution* selects where the current Experts should be applied. The central loop alternates these updates so that routing scopes adapt as Expert capabilities change.

Expert skill evolution. For each family, revisions at round t use the current Expert skill $e^{(t)}$ and family-specific training diagnostics, with fixed parent rollouts providing the reference behavior. As shown in the upper panel, candidate revisions undergo paired evaluation against the parent on routed validation examples. With support n, repairs W, and harms L, local admission requires

$$\begin{aligned} n &\geq n_{\min}, \\ W - L &\geq \max\{g_{\min}, \lceil \rho_{\text{gain}} n \rceil\}, \\ L &\leq \max\{\ell_{\min}, \lfloor \rho_{\text{loss}} n \rfloor\}. \end{aligned} \tag{2}$$

These conditions require sufficient evidence and net benefit while bounding regressions. System validation then checks that the candidate does not reduce the composed tree's validation performance. A revision that passes both checks becomes $e^{(t+1)}$. The thresholds and task-specific search budgets are detailed in Appendix C.

Router skill evolution. The lower panel separates candidate screening from utility-based selection. Candidate Router skills are first screened by their ability to detect parent errors. Let $P(r)$ and $R(r)$ denote validation precision and recall. Given precision floor τ and recall tolerance ϵ, the eligible Router pool is

$$\mathcal{R}_{\text{safe}} = \{r : P(r) \geq \tau,\ R(r) \geq R^* - \epsilon\}, \tag{3}$$

where R^* is the highest recall among precision-feasible candidates. Let $T[r, \mathcal{E}]$ denote the composed tree with candidate Router r and active Expert pool $\mathcal{E}$ installed at the parent, holding the rest of the tree fixed. Writing $F_T(x)$ for the tree's returned output, Router selection maximizes end-to-end validation utility

$$r^* = \arg\max_{r \in \mathcal{R}_{\text{safe}}} \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(x,y) \in \mathcal{D}_{\text{val}}} m\big(F_{T[r,\mathcal{E}]}(x), y\big). \tag{4}$$

Selection updates the current Router $r^{(t)}$ to $r^{(t+1)} = r^*$. Screening preserves error coverage, while utility-based selection accounts for what the current Experts actually repair or harm.

Coevolution loop. The central loop in Figure 3(a) alternates *Expert refinement* and *Router reselection*. The updated Experts provide *realized utility* for comparing Router candidates, shown by the green arrow. The selected Router changes the examples assigned to each Expert, providing the *routed scope* shown by the blue arrow. Routed training examples guide the next round of Expert refinement. The loop runs for a bounded number of rounds and retains the best validated state.

Validation-gated recursive growth. An accepted Expert can become a parent for further specialization. Within its assigned scope, RECAST repeats residual discovery, family induction, and Router–Expert coevolution. A proposed split must meet minimum parent and child support requirements and improve on the unsplit parent. Local admission is followed by validation of the composed tree, with unsuccessful changes reverted to the previously accepted state.

Recursion continues within the configured depth and search budgets when validation supports further improvement. The resulting structure can retain the Root alone, a single layer of Experts, or deeper branches of different depths. After evolution, the accepted skills, Routers, and family assignments are frozen. Training data provide proposal evidence and validation data control selection and growth; Test data are used only for final evaluation.

4.4 INFERENCE WITH A FROZEN SKILL TREE

For a new input x, M_E first executes the Root skill, producing output y_0 and trajectory τ_0. The local Router uses M_R to inspect this execution and either abstains or selects an available child. If a child is selected, M_E executes its Expert skill and routing proceeds from the new trajectory. At a leaf or upon abstention, the current node's output is returned.

In Figure 3(b), Router r_0 selects E_2 after the Root executes, and Router r_2 selects $E_{2.1}$ after E_2 executes. The selected path returns the output of $E_{2.1}$, while E_1 and $E_{2.2}$ remain inactive. Alternatively, abstention by r_0 returns y_0, and abstention by r_2 returns y_2. Each input therefore follows one path through the tree and returns the current node's output when routing stops.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on SearchQA (Dunn et al., 2017), SpreadsheetBench (*Sheet*; Ma et al., 2024), OfficeQA (Opsahl-Ong et al., 2026), DocVQA (Mathew et al., 2021), LiveMathematicianBench (*LiveMath*; He et al., 2026), and ALFWorld (Shridhar et al., 2021). These benchmarks cover question answering, spreadsheet manipulation, document understanding, mathematical reasoning, and interactive planning. We follow the train/validation/test protocol of SkillOpt (Yang et al., 2026), with split sizes detailed in Appendix A.

Evaluation metrics. We report task success rates or exact-match accuracy (%) on held-out Test sets. Macro-averages weight all six benchmarks equally, and gains are reported in percentage points.

Table 1: Test performance (%). † Results from Yang et al. (2026), Table 1. Green subscripts show gains over the corresponding Root in percentage points. Bold and underline mark the best and second-best local results per target model.

Method	SearchQA	Spreadsheet Bench	OfficeQA	DocVQA	LiveMath	ALFWorld	Avg.
Qwen3.5–4B							
Non-iterative baselines†							
No Skill	68.10	9.30	14.50	86.90	22.40	30.60	38.63
Human Skill	66.30	16.40	22.70	87.80	18.40	28.40	40.00
One-shot LLM Skill	65.00	10.40	20.90	88.00	28.80	55.20	44.72
Skill optimizer: GPT-5.5†							
Trace2Skill	68.50	19.30	16.30	88.00	27.20	64.90	47.37
TextGrad	60.70	13.90	20.90	85.60	10.60	53.70	40.90
GEPA	68.60	16.90	22.70	85.10	28.80	60.40	47.08
SkillOpt	71.20	23.90	29.70	89.00	52.00	81.30	57.85
Skill optimizer: Qwen3.5-4B							
GEPA	68.93	19.29	20.35	86.90	43.55	45.52	47.42
GEPA + RECAST	76.86$_{+7.93}$	**37.14**$_{+17.86}$	36.05$_{+15.70}$	90.64$_{+3.74}$	66.94$_{+23.39}$	64.93$_{+19.40}$	62.09$_{+14.67}$
SkillOpt	71.50	12.86	22.09	87.97	50.00	57.46	50.31
SkillOpt + RECAST	**77.64**$_{+6.14}$	36.79$_{+23.93}$	**42.44**$_{+20.35}$	**94.39**$_{+6.42}$	**70.16**$_{+20.16}$	**73.88**$_{+16.42}$	**65.88**$_{+15.57}$
Qwen3.6–35B-A3B							
Non-iterative baselines†							
No Skill	72.70	38.20	45.90	87.60	31.20	59.70	55.88
Human Skill	74.10	44.30	41.90	88.40	29.60	44.80	53.85
One-shot LLM Skill	72.60	42.90	46.50	88.40	24.80	60.40	55.93
Skill optimizer: GPT-5.5†							
Trace2Skill	75.40	33.20	32.00	90.40	29.60	70.90	55.25
TextGrad	76.40	22.90	33.70	84.50	7.20	67.90	48.77
GEPA	75.80	45.40	43.60	88.00	31.20	73.90	59.65
SkillOpt	80.30	47.50	47.10	91.40	41.60	82.10	65.00
Skill optimizer: Qwen3.6-35B-A3B							
GEPA	76.50	37.50	43.60	86.63	36.29	56.72	56.21
GEPA + RECAST	**82.14**$_{+5.64}$	61.79$_{+24.29}$	**52.33**$_{+8.72}$	93.05$_{+6.42}$	**63.71**$_{+27.42}$	78.36$_{+21.64}$	71.90$_{+15.69}$
SkillOpt	78.64	35.71	37.21	90.64	45.97	66.42	59.10
SkillOpt + RECAST	81.79$_{+3.14}$	**65.00**$_{+29.29}$	50.00$_{+12.79}$	**94.39**$_{+3.74}$	60.48$_{+14.52}$	**79.85**$_{+13.43}$	**71.92**$_{+12.82}$

Baselines. We use optimized Roots from GEPA (Agrawal et al., 2025) and SkillOpt (Yang et al., 2026), comparing each Root with its RECAST continuation. Table 1 additionally includes Trace2Skill (Ni et al., 2026) and TextGrad (Yüksekgönül et al., 2025), alongside No Skill, Human Skill, and One-shot LLM Skill baselines. Published reference results are taken from Yang et al. (2026).

Implementation details. The main study evaluates Qwen3.5-4B (Qwen Team, 2026a) and Qwen3.6-35B-A3B (Qwen Team, 2026b), yielding 24 paired comparisons. The target model executes tasks and proposes Expert revisions, while gpt-oss-120b (OpenAI, 2025) performs routing and proposes Router revisions. We additionally evaluate GPT-5.4 (OpenAI, 2026) on three tasks. Training data provide proposal evidence, while validation data guide selection and tree growth. Model weights remain frozen throughout optimization, and the resulting skills and tree structure are fixed before test evaluation. Search budgets, generation settings, and acceptance thresholds appear in Appendix C.

5.2 MAIN RESULTS

Table 1 shows that RECAST improves over its paired Root in all 24 configurations. Six-task macro-average gains range from 12.82 to 15.69 percentage points across the two models and two upstream optimizers. Notably, our skill trees, self-optimized using open-source backbone models, even

outperform strong baselines optimized by more capable models such as GPT-5.5, indicating that conditional skill specialization can bridge gaps in model capability. Moreover, these improvements span static answer generation and interactive planning tasks, demonstrating the broad applicability of RECAST.

The gains also persist across different starting skills. Both GEPA and SkillOpt Roots improve on every task, while their relative strengths vary across benchmarks. On the larger model, their RECAST continuations reach nearly identical macro-average performance despite different task-level profiles. The size of the gain varies across tasks and starting skills. For example, improvements on SpreadsheetBench range from 17.86 to 29.29 points, while DocVQA improves from already high Root scores. Together, these results support RECAST as a continuation of existing skill optimization across both target-model scales and upstream optimizers.

Extension to GPT-5.4. Table 2 evaluates RECAST with GPT-5.4 on SearchQA, OfficeQA, and LiveMath. Starting from paired SkillOpt Roots, RECAST improves performance on all three tasks and increases the macro-average score by 15.09 percentage points. These improvements span question answering with retrieved evidence, document reasoning, and mathematical reasoning.

The largest gains occur on OfficeQA and LiveMath, while SearchQA improves further from an already strong baseline. As in our primary study, each optimized Root is compared with its resulting skill tree under the same evaluation setup. This extension demonstrates that conditional skill specialization remains highly effective even when deployed on highly capable, state-of-the-art models, confirming that RECAST provides complementary benefits beyond base model scaling.

Table 2: GPT-5.4 test performance (%). Δ: gain over SkillOpt in percentage points.

Dataset	SkillOpt	+ RECAST	Δ
SearchQA	84.29	88.36	+4.07
OfficeQA	22.67	41.28	+18.60
LiveMath	42.74	65.32	+22.58
Avg.	49.90	64.99	+15.09

5.3 ABLATION STUDY

To evaluate the dynamic coupling between Expert capabilities and Router scopes, Table 3 compares full coevolution with three ablated variants initialized from the same SkillOpt Roots. The *One-shot* variant limits Expert search to one round with a fixed seed Router. The *Router-frozen* variant allows Expert evolution under a fixed initial scope, while the *Expert-frozen* variant updates the Router with a fixed Expert pool.

Table 3: Router–Expert coevolution ablation: test performance (%).

Method	SearchQA	Spreadsheet Bench	OfficeQA	DocVQA	LiveMath	ALFWorld	Avg.
Qwen3.5–4B							
One-shot	73.64	13.57	35.47	93.05	40.32	61.94	53.00
Router-frozen	73.57	19.29	33.14	92.78	53.23	64.18	56.03
Expert-frozen	74.07	16.43	38.37	93.05	56.45	65.67	57.34
Full coevolution	**77.64**	**36.79**	**42.44**	**94.39**	**70.16**	**73.88**	**65.88**
Qwen3.6–35B-A3B							
One-shot	79.43	34.64	48.26	92.78	51.61	64.93	61.94
Router-frozen	79.36	39.64	48.26	93.58	48.39	70.15	63.23
Expert-frozen	79.00	38.93	50.00	92.78	59.68	73.88	65.71
Full coevolution	**81.79**	**65.00**	**50.00**	**94.39**	**60.48**	**79.85**	**71.92**

Full coevolution achieves the highest macro-average across both models and is best or tied for best in all 12 configurations. Updating either component alone improves the macro-average over *One-shot*, confirming that both feedback directions provide distinct value. However, neither single-component variant matches joint adaptation, as even the stronger *Expert-frozen* variant lags behind full coevolution by 8.54 percentage points on 4B and 6.21 on 35B. These results highlight the benefit of repeatedly aligning routing scopes with evolving Expert capabilities.

5.4 Analysis

5.4.1 Effect of Tree Depth

We vary the deployment depth of a tree grown from a SkillOpt Root on SearchQA with Qwen3.5-4B. Depth 0 runs only the Root, depth 1 permits a flat Expert layer, and depths 2–3 enable recursive branches (Appendix Table 10).

Figure 4 shows validation accuracy rising from 72.50% to 80.00% and test accuracy from 71.50% to 77.64%. Recursive branches improve on the flat pool, with diminishing gains.

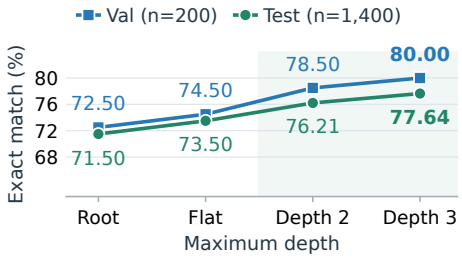

Figure 4: **Deployment depth** on SearchQA with Qwen3.5-4B and a SkillOpt Root.

5.4.2 Cost Analysis

Optimization cost. Motivated by the plateaus in Section 3, Table 4 compares completing upstream single-skill optimization with an upstream prefix followed by RECAST across six workflows on SearchQA and LiveMath. Token totals include optimization and validation calls, including the prefix that generates the starting skill. Scores are final test results at each workflow's actual token consumption. Detailed accounting appears in Appendix E.1.

Table 4: Optimization cost and test performance. Tokens are in millions, and scores are percentages.

Cell	Upstream	Baseline tokens	RECAST tokens	Baseline Test	RECAST Test
SearchQA / 4B	SkillOpt	25.285	14.303	71.50	77.64
SearchQA / 4B	GEPA	23.390	17.248	68.93	76.86
LiveMath / 4B	SkillOpt	3.071	2.733	50.00	70.16
LiveMath / 4B	GEPA	11.042	6.409	43.55	66.94
LiveMath / 35B	SkillOpt	2.425	2.072	45.97	60.48
LiveMath / 35B	GEPA	19.230	9.075	36.29	63.71

Across all six comparisons, the prefix-plus-RECAST workflow achieves higher test performance with fewer optimization and validation tokens. Token reductions range from 11.0% to 52.8%, with test accuracy gains of 6.14 to 27.42 percentage points. On SearchQA with 4B, for example, the SkillOpt-based workflow uses 43.4% fewer tokens and gains 6.14 points. These results favor allocating continued optimization to conditional specialization within the evaluated workflows.

Inference cost. RECAST adds Router calls and sequential Expert executions at inference. Appendix E.2 reports Router and Executor tokens separately. On SearchQA with Qwen3.6-35B-A3B, they average 5,168 and 9,691 tokens per example, respectively, totaling 14,859 versus 4,268 for the frozen Root ($3.48\times$). While inference uses more tokens, the workflows above achieve higher final test performance with fewer optimization and validation tokens than continued single-skill search. This trade-off can be acceptable in applications that prioritize task accuracy.

6 Conclusion

Continued single-skill search can plateau despite useful local repairs. RECAST addresses this limitation through Router–Expert coevolution, jointly evolving Expert skills and their routing scopes. Expert refinement provides realized utility for Router selection, while the selected Router defines the scope for further Expert refinement. Repeating this process within accepted Experts grows adaptive skill trees under validation control. RECAST improves all 24 paired Qwen configurations across six benchmarks, and ablations support joint adaptation over one-shot construction and single-component updates. These results support jointly evolving what a skill does and when it is applied to sustain improvement beyond single-skill search plateaus.

Reproducibility Statement

Appendix A documents the benchmarks, data splits, evaluation metrics, and paired comparisons. Appendix B provides algorithms for offline tree construction, candidate selection, and frozen-tree inference. Appendix C reports model roles, generation settings, routing and admission thresholds, search budgets, and execution limits. Appendices D and E detail supplementary analyses and optimization and inference cost accounting. Appendix F presents an initial and evolved Expert skill pair, and Appendix G provides Router execution and Expert revision prompt templates. Test evaluation is separated from all training and model-selection steps.

AI Use Statement

We used general-purpose AI assistants for literature discovery, manuscript drafting and language editing, consistency checks, and preparation of figures and supporting scripts. All AI-assisted material was reviewed by the authors. Research questions, method design, experimental decisions, and reported measurements remained under the authors' control. The authors take full responsibility for the paper's technical content, results, conclusions, and correctness.

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, Alexandros G. Dimakis, Ion Stoica, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. *CoRR*, abs/2507.19457, 2025. doi: 10.48550/ARXIV.2507.19457. URL https://doi.org/10.48550/arXiv.2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *CoRR*, abs/2603.02766, 2026. doi: 10.48550/ARXIV.2603.02766. URL https://doi.org/10.48550/arXiv.2603.02766.

Kunfeng Chen, Qihuang Zhong, Juhua Liu, and Bo Du. SkillCAT: Contrastive assessment and topology-aware skill self-evolution for LLM agents. *CoRR*, abs/2606.13317, 2026. doi: 10.48550/ARXIV.2606.13317. URL https://doi.org/10.48550/arXiv.2606.13317.

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

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Linyang He, Qiyao Yu, Hanze Dong, Baohao Liao, Xinxing Xu, Micah Goldblum, Jiang Bian, and Nima Mesgarani. LiveMathematicianBench: A live benchmark for mathematician-level reasoning with proof sketches. *CoRR*, abs/2604.01754, 2026. doi: 10.48550/ARXIV.2604.01754. URL https://doi.org/10.48550/arXiv.2604.01754.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ac840df270ac537dd74530a15c332684-Abstract-Datasets_and_Benchmarks_Track.html.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pp. 2199–2208. IEEE, 2021. doi: 10.1109/WACV48630.2021.00225. URL https://doi.org/10.1109/WACV48630.2021.00225.

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. *CoRR*, abs/2603.25158, 2026. doi: 10.48550/ARXIV.2603.25158. URL https://doi.org/10.48550/arXiv.2603.25158.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. GPT-5.4 thinking system card. Technical report, OpenAI, March 2026. URL https://openai.com/index/gpt-5-4-thinking-system-card/.

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. *CoRR*, abs/2603.08655, 2026. doi: 10.48550/ARXIV.2603.08655. URL https://doi.org/10.48550/arXiv.2603.08655.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 7957–7968. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.494. URL https://doi.org/10.18653/v1/2023.emnlp-main.494.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Zeyu Ren, Ling Yue, Ran Li, Yishu Wang, Shengxiang Xu, Hanmo Liu, Shaowu Pan, and Shimin Di. FlowEvo: Self-evolving agents through the co-evolution of workflows and executable skills. *CoRR*, abs/2607.21596, 2026. doi: 10.48550/ARXIV.2607.21596. URL https://doi.org/10.48550/arXiv.2607.21596.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

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. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

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. *CoRR*, abs/2605.23904, 2026. doi: 10.48550/ARXIV.2605.23904. URL https://doi.org/10.48550/arXiv.2605.23904.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nat.*, 639(8055):609–616, 2025. doi: 10.1038/S41586-025-08661-4. URL https://doi.org/10.1038/s41586-025-08661-4.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Co-EvoSkills: Self-evolving agent skills via co-evolutionary verification. *CoRR*, abs/2604.01687, 2026. doi: 10.48550/ARXIV.2604.01687. URL https://doi.org/10.48550/arXiv.2604.01687.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *CoRR*, abs/2510.04618, 2025. doi: 10.48550/ARXIV.2510.04618. URL https://doi.org/10.48550/arXiv.2510.04618.

A Experimental Setup and Evaluation

A.1 Benchmarks and evaluation metrics

Table 5 lists the six benchmarks and fixed data splits used in the local comparisons, following Yang et al. (2026).

Table 5: Benchmarks and data splits for the local experiments. SpreadsheetBench and LiveMathematicianBench are abbreviated as Sheet and LiveMath in the result tables.

Benchmark	Task	Train	Validation	Test
SearchQA	Question answering	400	200	1,400
SpreadsheetBench	Spreadsheet manipulation	80	40	280
OfficeQA	Document reasoning	50	24	172
DocVQA	Visual document understanding	107	53	374
LiveMath	Mathematical reasoning	35	18	124
ALFWorld	Interactive planning	39	18	134

SearchQA evaluates answers using retrieved evidence. OfficeQA provides local document tools, while DocVQA supplies a document question and visual input. For Sheet, evaluation uses the saved workbook produced by the agent. ALFWorld evaluates success in the interactive environment. We report task-level correctness or success, with exact-match accuracy for SearchQA.

For a test set of size N, let $z_i \in \{0, 1\}$ be the evaluator's outcome for example i. The reported percentage is

$$S = \frac{100}{N} \sum_{i=1}^{N} z_i. \tag{5}$$

The six-task macro-average is $\frac{1}{6} \sum_{b=1}^{6} S_b$, giving each benchmark equal weight. The GPT-5.4 extension averages its three reported tasks in the same way. A paired gain is $\Delta S = S_{\text{RECAST}} - S_{\text{Root}}$, expressed in percentage points. Scores and gains are rounded separately to two decimal places.

A.2 Data usage and evaluation procedure

Training data supply parent rollouts, offline failure labels, error-family diagnostics, and proposal evidence. Router and Expert proposal prompts use training examples, training metrics, and training diagnostics. Validation data are reserved for candidate selection: they assess local and composed-system gains, determine whether recursive expansions are accepted, and select the frozen state. The selected skills, Routers, and family assignments are fixed before Test evaluation. Test outcomes do not select candidates, thresholds, depth, or stopping time.

Router inputs consist of task inputs and deployment-visible execution trajectories. Reference answers, offline failure labels, and evaluator feedback provide supervision during optimization but are not inputs to the deployed Router. In OfficeQA, evaluator feedback is hidden from the Router. In ALFWorld, the Router receives the task and observation–action trajectory with reward and success fields removed. Family induction uses training diagnostics; assigning a new example to a family uses deployment-visible information.

A.3 Baselines and paired comparisons

Table 1 contains 24 local Root–RECAST pairs: six tasks, two Qwen Executor models, and two upstream optimizers. Each pair compares the optimized Root produced by GEPA or SkillOpt with the skill tree obtained by continuing from that Root. The target model remains the Executor throughout the paired comparison. Table 2 applies the same paired comparison to SkillOpt Roots with GPT-5.4 on SearchQA, OfficeQA, and LiveMath.

Published non-iterative and skill-optimization baselines from Yang et al. (2026) carry the reference marker in the main table; green gains use the local Root rows. The One-shot LLM Skill baseline differs from the One-shot construction ablation defined in Appendix D.3.

B Detailed RECAST Algorithm

Appendix F illustrates Expert evolution with initial and final skill texts; Appendix G gives the corresponding prompt interfaces.

B.1 Offline tree construction

Algorithm 1 organizes the operations in Section 4 into a recursive construction procedure. The initial tree contains only the optimized Root s_0. For each parent p, its scope consists of the examples that reach it under the current ancestor routing decisions; at the Root, this is the full split. Parent outputs provide both the trajectories for routing and the reference behavior for evaluating proposed Experts.

Initial Router search identifies potential parent errors with high recall subject to the precision floor. Error families are induced from routed training failures, and each supported family defines an initial Expert scope. Expert revisions use feedback from the assigned training examples; routed validation examples determine candidate acceptance. Once Experts are available, the Router candidate pool is evaluated with the active Experts using end-to-end validation utility. Reselection changes the scopes used in the next Expert refinement round. The best validated combination of routing and Expert skills is retained within the alignment budget.

Algorithm 1: Offline construction of a RECAST skill tree

Input: optimized Root s_0; Train and Validation; fixed models; search limits.
Output: a frozen tree of accepted skills, Routers, and family assignments.

$T \leftarrow$ tree containing s_0 only.
Expand(p, T):
  if the depth or support requirements prohibit expansion: **return** T.
  Save the accepted state; collect parent rollouts on its Train and Validation scopes.
  Generate Router candidates using parent-error feedback from Train.
  Select an initial Router by error recall subject to the precision floor.
  Induce supported error families from routed training failures.
  if no supported routing and family proposal is available: **return** T.
  Initialize an Expert seed for each supported family.
  for each allowed alignment round:
    Assign examples to families using the current Router and visible trajectories.
    for each family, within its Expert-search budget:
      Revise the current Expert skill using paired feedback from its Train scope.
      Evaluate each revision against the parent on routed Validation examples.
      Retain revisions passing local admission and composed-system validation.
    Screen Router candidates using Equation 3.
    Reselect the Router with the active Experts using Equation 4.
    Record the best validated Router–Expert state for this expansion.
  Form the proposed expansion from the best validated state.
  if the expansion fails support, split, or system checks:
    Restore the previously accepted state and **return** T.
  Commit the accepted expansion to T.
  for each accepted Expert c eligible for further specialization:
    $T \leftarrow$ **Expand**(c, T).
  return T.
$T \leftarrow$ **Expand**(Root, T).
Freeze the accepted skills, Routers, and family assignments; **return** T.

B.2 Candidate selection and state retention

Local Expert admission. For candidate c at parent p, let $\mathcal{D}_{p\to c}$ be its routed validation subset, with $n = |\mathcal{D}_{p\to c}|$. Using binary parent and candidate outcomes $z_p(x)$ and $z_c(x)$, define

$$W = \sum_{(x,y)\in\mathcal{D}_{p\to c}} \mathbf{1}\{z_p(x) = 0,\ z_c(x) = 1\},$$
$$L = \sum_{(x,y)\in\mathcal{D}_{p\to c}} \mathbf{1}\{z_p(x) = 1,\ z_c(x) = 0\}. \tag{6}$$

Equation 2 requires enough support, sufficient net gain $W - L$, and bounded harms. Parent successes remain in the routed scope to measure regressions; the parent remains the reference during refinement.

Composed-system validation. An Expert passing local admission is provisionally installed in the tree. Its acceptance requires non-decreasing composed-system validation performance, using the utility in Equation 4. Recursive splits also require parent and child support and improvement over the unsplit parent. Failed proposals restore the accepted state.

Router selection. Precision measures the fraction of routed examples on which the parent fails; recall measures the fraction of parent errors detected. Initial selection prioritizes recall subject to the precision floor. With active Experts, Equation 3 screens candidates and Equation 4 ranks their utility, holding the rest of the tree fixed. Ties are resolved by final correctness, fewer harms, and then routing statistics.

Retained state and termination. Alignment retains the best validated combination of skills, Routers, family assignments, and structure within its round budget. Recursion is limited by depth, support, and validation gains. Nodes without an accepted expansion remain terminal. The selected tree is frozen before Test.

B.3 Frozen-tree inference

Algorithm 2 executes the frozen tree. Here Execute returns both a task output and its visible trajectory; Route uses the frozen local Router and family assignments to select a child or abstain. The current output is returned when traversal stops, so abstention retains the result already produced at that node.

For the SearchQA example in Appendix F, x contains the question and retrieved context. Expert execution receives these inputs and s_v, while the parent trajectory is supplied to the Router. Parent-relative repairs and harms are computed offline from the recorded outputs and reference outcomes.

Algorithm 2: Inference with a frozen RECAST skill tree

Input: task x, frozen tree T, fixed Executor M_E and Router M_R.
Output: the task output returned by the selected path.

$v \leftarrow \mathrm{Root}(T)$.
$(\hat{y}, \tau) \leftarrow \mathrm{Execute}(M_E, x, s_v)$.
while v has accepted children:
  $c \leftarrow \mathrm{Route}(M_R, r_v, x, \tau, \mathrm{children}(v))$.
  if $c = \mathrm{abstain}$: **return** $\hat{y}$.
  $v \leftarrow c$.
  $(\hat{y}, \tau) \leftarrow \mathrm{Execute}(M_E, x, s_v)$.
return $\hat{y}$.

The traversal activates one path. At maximum deployment depth d, that path contains the Root and at most d Expert nodes; unselected siblings remain inactive. Appendix E.2 accounts for the model calls used to execute and route these paths.

C Implementation Details and Hyperparameters

C.1 Model roles and generation settings

Table 6 reports the Qwen implementation. The target model executes tasks and proposes Expert skills; gpt-oss-120b executes Routers and proposes Router revisions. Both retain fixed weights. The table separates generation settings for task execution and text proposals.

Table 6: Model and execution settings for the Qwen models.

Setting	Value
Models	
Executor	Qwen3.5-4B or Qwen3.6-35B-A3B
Expert proposer	Same as Executor
Router executor / proposer	gpt-oss-120b
Generation	
Task execution in Expert search	Temperature 0
Expert / Router text proposals	Temperature 0.7
Frozen Expert / Router execution	Temperature 0
Proposals per search round	2 per Router or Expert
Execution limits	
SearchQA / DocVQA / LiveMath	12,000 task output tokens
Sheet / OfficeQA / ALFWorld	16,384 task output tokens
Sheet interaction limit	30 turns
OfficeQA interaction limit	24 tool turns
ALFWorld interaction limit	50 environment steps

C.2 Routing and admission thresholds

Table 7(a) gives the settings for all 24 Qwen task–model–Root conditions. The two Router precision entries specify the Root and local precision floors, while ϵ sets the recall tolerance in Equation 3. The Expert-gate columns report Parent/Child pairs for minimum routed validation support $n_{\min}$, minimum absolute gain $g_{\min}$, gain rate ρ_{gain}, and loss allowance floor $\ell_{\min}$.

For an Expert with routed validation support n, the required net gain is $\max\{g_{\min}, \lceil \rho_{\text{gain}} n \rceil\}$ and the allowed loss count is $\max\{\ell_{\min}, \lfloor \rho_{\text{loss}} n \rfloor\}$, as in Equation 2. The support n is the size of the Expert's assigned subset. In particular, $\ell_{\min} = 0$ still permits losses when the proportional term is positive. The reported loss-rate setting is $\rho_{\text{loss}} = 0.10$.

Configuration rationale. Configurations account for residual error patterns across tasks and models and the available training and validation data. Training data provide evidence for error-family induction and skill proposals, while validation data guide candidate selection and growth. The support, routing, and admission thresholds in the table jointly constrain specialization search. Candidates must also pass composed-system validation, and validation performance determines the retained tree structure.

C.3 Search budgets and execution limits

Table 7(b) lists Root-Router, Local-Router, Expert, and alignment round limits in that order $(R/L/E/A)$. Alignment rounds alternate Expert refinement and Router reselection; Expert and Router search budgets govern their respective candidate-generation stages. All 24 profiles cap outer cycles at one and maximum tree depth at three. The minimum Train support entries specify Parent/Child requirements for recursive expansion. These limits define the permitted search; validation determines which branches are retained.

Output-token and interaction limits appear in Table 6. Appendix A.1 describes the task outputs and evaluation, and Appendix A.2 specifies the deployment-visible Router inputs.

Table 7: Core admission thresholds and search budgets for the 24 Qwen conditions.

(a) Router selection and Expert admission

Task	Model	Root	$\tau_{\text{root}}/\tau_{\text{local}}$	ϵ	$n_{\min}$	$g_{\min}$	ρ_{gain}	$\ell_{\min}$
SearchQA	4B	GEPA	0.5/0.5	0.05	4/4	1/1	0/0	1/1
SearchQA	4B	SkillOpt	0.5/0.5	0.05	4/4	1/1	0/0	1/1
SearchQA	35B	GEPA	0.5/0.5	0.05	4/4	2/1	0.05/0.05	1/1
SearchQA	35B	SkillOpt	0.5/0.5	0.05	4/4	2/1	0.05/0.05	1/1
Sheet	4B	GEPA	0.45/0.4	0.1	2/1	1/1	0.05/0.05	2/1
Sheet	4B	SkillOpt	0.45/0.4	0.1	2/1	1/1	0.05/0.05	2/1
Sheet	35B	GEPA	0.45/0.4	0.1	2/1	1/1	0.05/0.05	2/1
Sheet	35B	SkillOpt	0.45/0.4	0.1	2/1	1/1	0.05/0.05	1/1
OfficeQA	4B	GEPA	0.55/0.55	0.1	2/1	1/1	0.05/0.05	1/1
OfficeQA	4B	SkillOpt	0.55/0.55	0.1	1/1	1/1	0.05/0.05	0/1
OfficeQA	35B	GEPA	0.75/0.6	0.1	2/1	1/1	0.05/0.05	1/1
OfficeQA	35B	SkillOpt	0.65/0.5	0.1	2/1	1/1	0.05/0.05	1/1
DocVQA	4B	GEPA	0.5/0.5	0.1	1/1	0/1	0/0.05	0/1
DocVQA	4B	SkillOpt	0.5/0.75	0.1	1/1	1/1	0/0.05	0/1
DocVQA	35B	GEPA	0.25/0.5	0.1	1/1	0/1	0/0.05	0/1
DocVQA	35B	SkillOpt	0.5/0.5	0.1	1/1	1/1	0/0.05	0/1
LiveMath	4B	GEPA	0.55/0.5	0.1	1/1	1/1	0.05/0.05	0/1
LiveMath	4B	SkillOpt	0.45/0.5	0.1	1/1	1/1	0.05/0.05	0/1
LiveMath	35B	GEPA	0.3/0.5	0.1	1/1	1/1	0.05/0.05	2/1
LiveMath	35B	SkillOpt	0.3/0.5	0.1	1/1	1/1	0.05/0.05	2/1
ALFWorld	4B	GEPA	0.25/0.3	0.2	1/1	1/1	0.05/0.05	1/1
ALFWorld	4B	SkillOpt	0.25/0.3	0.2	1/1	1/1	0.05/0.05	1/1
ALFWorld	35B	GEPA	0.25/0.3	0.2	1/1	1/1	0.05/0.05	1/1
ALFWorld	35B	SkillOpt	0.25/0.3	0.2	1/1	1/1	0.05/0.05	1/1

(b) Search budgets and recursive expansion

Task	Model	Root	Max. rounds ($R/L/E/A$)	Max. depth	Min. Train (P/C)
SearchQA	4B	GEPA	4/3/8/3	3	16/8
SearchQA	4B	SkillOpt	4/3/8/3	3	4/4
SearchQA	35B	GEPA	4/3/8/3	3	8/8
SearchQA	35B	SkillOpt	4/3/8/3	3	7/7
Sheet	4B	GEPA	4/3/4/2	3	5/1
Sheet	4B	SkillOpt	4/3/4/2	3	5/1
Sheet	35B	GEPA	4/3/4/2	3	5/3
Sheet	35B	SkillOpt	4/3/4/2	3	5/3
OfficeQA	4B	GEPA	4/3/4/3	3	5/1
OfficeQA	4B	SkillOpt	4/3/4/3	3	5/1
OfficeQA	35B	GEPA	4/3/4/3	3	5/1
OfficeQA	35B	SkillOpt	4/3/4/3	3	5/1
DocVQA	4B	GEPA	4/4/6/5	3	1/1
DocVQA	4B	SkillOpt	4/4/6/5	3	1/2
DocVQA	35B	GEPA	4/4/6/5	3	1/1
DocVQA	35B	SkillOpt	4/4/6/5	3	1/1
LiveMath	4B	GEPA	4/3/3/2	3	3/2
LiveMath	4B	SkillOpt	4/3/3/2	3	3/2
LiveMath	35B	GEPA	4/3/3/2	3	3/2
LiveMath	35B	SkillOpt	4/3/3/2	3	3/2
ALFWorld	4B	GEPA	3/3/2/1	3	3/2
ALFWorld	4B	SkillOpt	3/3/2/1	3	3/2
ALFWorld	35B	GEPA	3/3/2/1	3	3/2
ALFWorld	35B	SkillOpt	3/3/2/1	3	3/2

4B and 35B denote Qwen3.5-4B and Qwen3.6-35B-A3B. Paired entries in the four Expert-gate columns are Parent/Child settings; $R/L/E/A$ denotes Root-Router, Local-Router, Expert, and alignment rounds. All 24 profiles cap outer cycles at one. $\rho_{\text{loss}} = 0.10$. The loss allowance is $\max\{\ell_{\min}, \lfloor \rho_{\text{loss}} n \rfloor\}$.

D SUPPLEMENTARY ANALYSES

D.1 SINGLE-SKILL SEARCH STAGNATION

Let v_t be a candidate's validation score at optimizer step t and $b_t = \max_{j \le t} v_j$ the retained best-so-far score. For an observation endpoint T, define the first step attaining the final best as $t^* = \min\{t : b_t = b_T\}$. The reported *steps after best* is $T - t^*$. This statistic counts the observed steps following the last improvement in the best score, while the candidate curve records variation during search.

Figure 5 shows the candidate and best-so-far curves for SkillOpt and GEPA on SearchQA/Qwen3.5-4B. Table 8 additionally reports LiveMath with both Qwen models. The SearchQA validation set contains 200 examples and the observation window ends at step 150; the LiveMath validation set contains 18 examples and the window ends at step 80.

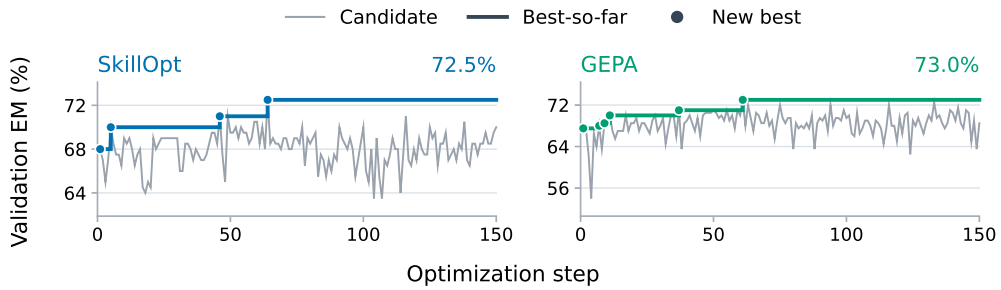

Figure 5: **Candidate variation persists while the retained best stagnates.** Gray lines show individual candidate validation scores; colored lines show best-so-far scores, and dots mark new best scores, including initialization. Each panel uses its own vertical range to preserve within-method detail. Table 8 reports the final best scores and their observed stagnation lengths.

Table 8: **Stagnation over aligned observation windows.** T is the common endpoint within each task–model group. *Steps after best* equals T minus the first best step. Reported continuation runs retain the same best scores through T.

Method	Best val. (%)	First best step	Steps after best
SearchQA / Qwen3.5–4B ($n = 200$; $T = 150$)			
SkillOpt	72.50	64	86
GEPA	73.00	61	89
TextGrad	75.00	61	89
Trace2Skill	74.50	41	109
LiveMath / Qwen3.5–4B ($n = 18$; $T = 80$)			
SkillOpt	83.33	2	78
GEPA	88.89	11	69
TextGrad	100.00	14	66
Trace2Skill	55.56	17	63
LiveMath / Qwen3.6–35B-A3B ($n = 18$; $T = 80$)			
SkillOpt	88.89	4	76
GEPA	88.89	49	31
TextGrad	50.00	18	62
Trace2Skill	88.89	61	19

D.2 CANDIDATE REPAIRS AND COMPLEMENTARITY

The analysis in Section 3.2 evaluates $K = 39$ distinct SkillOpt candidates on the same $N = 200$ SearchQA validation examples with Qwen3.5-4B. Let $z_{ki}^{(j)}$ be candidate k's outcome on example i in

repetition j. With J_{ki} evaluations, its averaged per-example score is

$$q_{ki} = \frac{1}{J_{ki}} \sum_{j=1}^{J_{ki}} z_{ki}^{(j)}. \tag{7}$$

The reference is the optimized SkillOpt Root in Figure 1, with 72.50% validation accuracy. Using these means and the Root's binary outcomes z_{0i}, the repair and harm counts in Figure 2(a) are

$$W_k = \sum_{i:z_{0i}=0} q_{ki}, \qquad L_k = \sum_{i:z_{0i}=1} (1 - q_{ki}). \tag{8}$$

These mean counts satisfy $S_k - S_0 = 100(W_k - L_k)/N$, where $S_k = 100 \sum_i q_{ki}/N$ and $S_0 = 100 \sum_i z_{0i}/N = 72.50$. A candidate can therefore repair errors ($W_k > 0$) without a global gain ($W_k - L_k \leq 0$), as observed for the reported candidate pool.

The candidate oracle selects the highest averaged candidate score separately for each example:

$$S_{\text{oracle}} = \frac{100}{N} \sum_i \max_{1 \leq k \leq K} q_{ki}. \tag{9}$$

The oracle reaches 82.50%, exceeding the Root by 10 percentage points in Figure 2(b). This gap measures the gains available from per-example candidate selection using reference outcomes. Deployed routing uses the visible evidence described in Appendix A.2.

D.3 Router–Expert coevolution ablations

Table 9 defines the four conditions whose results appear in Table 3. They start from the same SkillOpt Roots within each task–model setting. Full coevolution alternates Expert refinement and Router reselection; the other conditions restrict one or both update directions.

Table 9: Update rules for the coevolution ablation. Expert utility guides Router selection, and the selected Router defines subsequent Expert scopes.

Condition	Router updates	Expert updates
One-shot	Fixed seed Router	One construction round
Router-frozen	Fixed initial Router	Continued Expert evolution
Expert-frozen	Router updates	Fixed Expert pool
Full coevolution	Reselection with current Experts	Refinement on updated routed scopes

D.4 Effect of deployment depth

The SearchQA/Qwen3.5-4B analysis in Section 5.4.1 limits the deployment depth of a SkillOpt-initialized tree. Depth 0 returns the SkillOpt Root output; depth 1 permits a flat Expert layer; depths 2 and 3 enable further residual branches. At the depth limit, execution returns the current node's output.

Table 10: Tree-depth analysis on SearchQA with Qwen3.5-4B and a SkillOpt Root. Scores are exact match. The Test column is reported for mechanism analysis only.

Maximum depth	Val.	Test	Δ Test	Net examples
0 (SkillOpt Root)	72.50	71.50	—	0
1 (Flat MoE)	74.50	73.50	+2.00	+28
2 (Tree)	78.50	76.21	+2.71	+38
3 (Tree)	80.00	77.64	+1.43	+20

Table 10 gives the values behind Figure 4. The Δ Test and Net examples columns compare each depth with the preceding one on the 1,400-example Test set. The full tree gains 86 correct predictions over the Root and 58 over the flat pool.

E Optimization and Inference Costs

E.1 Optimization cost accounting

Table 4 compares completing upstream single-skill optimization with an upstream prefix followed by RECAST. Each total counts optimization and validation calls. If C_{prefix} denotes the cost of generating the starting skill and C_{tree} the subsequent RECAST optimization and validation cost, then

$$C_{\text{RECAST}} = C_{\text{prefix}} + C_{\text{tree}}, \qquad \text{Reduction}(\%) = 100\left(1 - \frac{C_{\text{RECAST}}}{C_{\text{baseline}}}\right). \tag{10}$$

Here C_{baseline} is the completed upstream workflow's cost. The displayed unit is one million tokens. The Test columns report the terminal performance of the two workflows, with the score gain calculated as their difference in percentage points.

Across the six reported comparisons, token reductions range from 11.0% to 52.8%, while Test gains range from 6.14 to 27.42 points. For example, the SearchQA/4B SkillOpt comparison uses 25.285 million tokens for the completed baseline and 14.303 million for the prefix-plus-RECAST workflow, corresponding to a 43.4% reduction. The associated Test scores are 71.50% and 77.64%. These totals describe the actual consumption of each workflow.

E.2 Frozen inference cost

Table 11 reports mean input plus output tokens per example over 1,400 SearchQA Test examples. The frozen Root column measures execution with the SkillOpt Root alone. Within RECAST, Router tokens include routing and Expert-assignment calls using gpt-oss-120b, while Executor tokens include the Root and all executed Experts. Consequently,

$$C_{\text{total}} = C_{\text{Router}} + C_{\text{Executor}}, \qquad r_{\text{total}} = \frac{C_{\text{total}}}{C_{\text{Root}}}. \tag{11}$$

The Executor/Root ratio isolates Executor-token usage; Total/Root includes both model roles. Root execution is already included in the RECAST Executor column and is therefore counted once in the total.

Table 11: Frozen inference cost on 1,400 SearchQA Test examples with three Executor models and SkillOpt Roots. Token counts are mean input plus output tokens per example, rounded to integers. Router includes all routing and Expert-assignment calls using gpt-oss-120b; Executor includes the Root and all Expert executions. Total is Router plus Executor usage. The two ratios compare Executor and Total tokens, respectively, with the frozen Root.

		RECAST			Ratio to Root	
Executor model	Frozen Root	Router	Executor	Total	Executor	Total
Qwen3.5-4B	1,824	7,341	5,872	13,213	3.22×	7.24×
Qwen3.6-35B-A3B	4,268	5,168	9,691	14,859	2.27×	3.48×
GPT-5.4	2,260	9,336	2,506	11,842	1.11×	5.24×

Executor-token ratios are $3.22\times$, $2.27\times$, and $1.11\times$ for Qwen3.5-4B, Qwen3.6-35B-A3B, and GPT-5.4, respectively. Including routing, the corresponding total ratios are $7.24\times$, $3.48\times$, and $5.24\times$. The single-path execution in Algorithm 2 avoids activating unselected branches; the selected path still incurs Router calls and sequential Expert execution. These inference measurements complement the optimization costs in Appendix E.1.

F SKILL EVOLUTION EXAMPLE

Figure 6 shows the initial and final retained skills of a SearchQA Expert evolved with Qwen3.6-35B-A3B in a tree initialized from a GEPA Root. The Expert specializes in wrong-entity and unsupported-answer errors. Its initial skill is a generic question-answering seed; the optimized Root supplies the reference behavior for paired evaluation. The evolved skill adds query-target parsing, evidence checks, and answer-span refinement. Both skills operate on the question and retrieved context.

Initial Expert Skill (Complete Text)

Solve the SearchQA item directly from the supplied context. Identify the exact requested target and return the shortest answer span that fills it. Ignore unsupported associations and explanations. End with exactly one <answer>...</answer>.

Evolved Expert Skill (Complete Text)

Identify the specific semantic role requested by the question and verify it against the context. Do not assume the most prominent entity in the text is the answer; instead, parse the question's syntactic and logical constraints to determine exactly what is being sought.

1. **Parse the Query Target**: Determine if the question asks for an entity (Person, Place, Thing), a specific attribute (Name, Title, Color, Fraction), or a relationship.
   - If the question is a fragment (e.g., "Money put in a collection plate"), identify the specific term defined by that fragment (e.g., "offering", not "charity").
   - If the question asks for a completion (e.g., "Not worth a..."), identify the specific idiom or phrase ending (e.g., "red cent", not "picayune").
   - If the question asks for a common link between two entities (e.g., "Edward Kennedy, John Kerry"), identify the shared attribute (e.g., "Massachusetts", not one of the names).
   - If the question asks for a specific name used by a group (e.g., "called him this"), identify the relational title (e.g., "Grandpa", not the given name "Zeb").
2. **Validate Against Context**:
   - Ensure the candidate answer is explicitly supported by the provided text snippets.
   - Check for distractors: The context may contain the correct entity but in a different context (e.g., a nearby town vs. the specific city described). Use specific clues (dates, distances, exact phrasing) to disambiguate.
   - If the context provides a specific definition or quote that matches the question's phrasing, prioritize that span.
3. **Refine the Answer Span**:
   - Return the minimal, exact span from the context that satisfies the query.
   - Avoid verbose explanations or full sentences.
   - If the question asks for a fraction, return the fraction (e.g., "1/4"), not the associated entity (e.g., "Venus") unless the question explicitly asks for the entity.
   - If the question asks for the person who performed an action, ensure the subject is correctly identified (e.g., the Vice President announcing the election, not the President-elect).
4. **Output**:
   - Provide the reasoning briefly if necessary to justify the selection, then conclude with the final answer.
   - End with exactly one `<answer>...</answer>` tag containing the minimal correct span.

Figure 6: Initial and final retained skills of the same Expert (P05). The evolved skill was generated by revising the displayed initial skill and was selected in the frozen tree. Both texts are reproduced in full, with Markdown rendered for readability.

G PROMPT TEMPLATES

G.1 ROUTER EXECUTION INTERFACE

Figure 7 gives the input and output interface used by the Router in the SearchQA run illustrated in Appendix F. The system message is the selected Router skill. The user message contains four deployment-visible fields: the question, retrieved context excerpts, Root trajectory, and extracted Root answer. Placeholder values below stand for the corresponding example-specific text. The Router returns a binary error judgment and, when routing, a concise reason used in family assignment.

SearchQA Router Execution Prompt

System message:

```
{selected_router_skill}
```

User message:

Judge this Root attempt:

```
{
  "question": "{question}",
  "context_excerpts": [
    {"source_id": "{source_id}",
     "source_rank": {source_rank},
     "text": "{retrieved_text}"}
  ],
  "root_trajectory": "{root_response}",
  "root_final_answer": "{root_answer}"
}
```

Return exactly:

```
{"root_is_wrong": true_or_false,
 "error_reason": "specific reason if wrong, otherwise null"}
```

Valid response when keeping the Root answer:

```
{"root_is_wrong":false,"error_reason":null}
```

Valid response when routing:

```
{"root_is_wrong":true,"error_reason":"{reason}"}
```

Figure 7: Router prompt interface reconstructed from the recorded request. The context array repeats its source object for each supplied excerpt. The two valid-response examples make the boolean and null types explicit.

The system text carries the evolving decision rules; the user interface stays the same across Router candidates. Reference outcomes are used for offline scoring and are absent from these four deployment input fields.

Expert execution interface. For the same SearchQA example, the Expert system message contains its selected skill and the task and answer-format instructions. Its user message contains only `Context` and `Question`. The parent answer, parent trajectory, and Router diagnosis are not passed to the Expert.

G.2 EXPERT SKILL REVISION PROMPT

Figure 8 gives a system-message template for Expert revision under the data-use protocol in Appendix A.2. The user message supplies the current candidate skill, family definition, paired training metrics, and training diagnostics, as listed in Table 12. Validation evaluates the resulting candidates for selection and acceptance.

Expert Skill Revision: System Message Template

You evolve one reusable SearchQA Expert Skill.

The Skill will be executed by the configured SearchQA task model on cases that a Router assigns to one residual family. Revise the current Skill to repair the recurring error mechanism while preserving successful behavior.

Use the supplied training examples, paired training metrics, and training diagnostics as offline feedback. Infer reusable corrective procedures from this feedback. The deployed Skill receives only the question and context and must end with exactly one minimal answer in <answer>...</answer>.

Return a complete, immediately deployable Skill with concrete procedures and the required answer format.

Return only the complete revised Skill inside:
<EXPERT_SKILL>...</EXPERT_SKILL>

Figure 8: Template for proposing Expert revisions from training feedback. The returned skill is enclosed in `<EXPERT_SKILL>` tags for extraction and subsequent evaluation.

Table 12: Inputs to the Expert-revision template. Example-level feedback and paired metrics come from the training split.

User-message field	Content
`family_id`, `family_concept`	Expert identifier and broad correction mechanism
`current_skill`	Complete candidate text to revise
`train_metrics`	Paired performance on the assigned training scope
`train_diagnostics`	Training examples and diagnostic feedback
`optimization_intent`	Objectives guiding repair and preservation