Training Agent Skills

From Diagnosis to Optimization

A joint look at SkillLens and SkillOpt

Yifan Yang — Microsoft Research

SKILLLENS — FROM RAW EXPERIENCE TO SKILL CONSUMPTION

SKILLOPT — A CONTROLLABLE TEXT-SPACE OPTIMIZER FOR AGENT SKILLS

Joint work with collaborators at Fudan, SJTU, MSRA

Skills are becoming the new adaptation layer for LLM agents

Frontier models are frozen, but the procedures around them are not.

Claude Skills (Anthropic, Oct 2025) | ChatGPT Apps & GPTs | Custom skill libraries in agent stacks

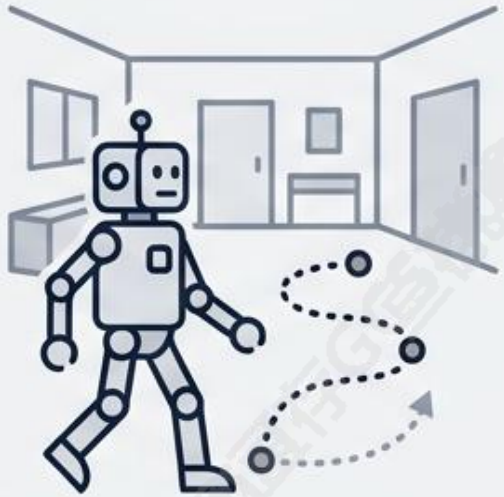
An agent skill: a portable, natural-language procedural artifact

```
# SpreadsheetBench skill

## When to use
Any task that edits or queries an .xlsx workbook.

## Procedure
1. Inspect workbook structure first (sheets, headers,
   target ranges).
2. Normalize keys / cell types before lookup.
3. Compute evaluated static values in Python; do not
   rely on Excel recalculation.
4. Re-open the saved file and check boundary rows and
   remaining blanks.

## Output format
Write the answer into the exact target range.
```

- **Compact** — typically 300 to 2,000 tokens
- **Procedural** — how to act, not the answer
- **Portable** — same text reused across tasks, models, harnesses
- **Inspectable** — a human can read it in 2 minutes

The skill lifecycle: experience → extraction → consumption

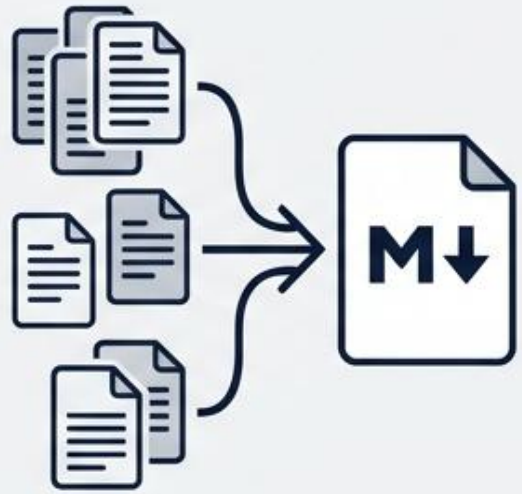
Stage 1 — Experience Generation



Target model M runs tasks; collects successes and failures.

Trajectory pool T(M,D)

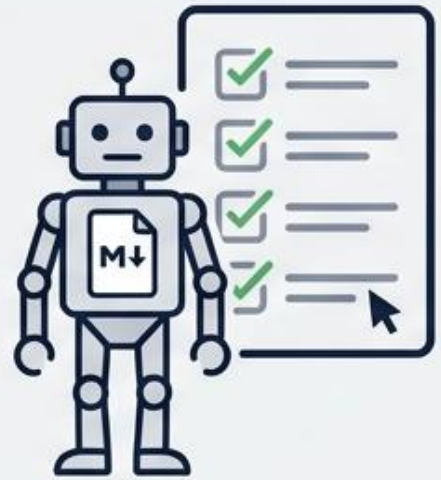
Stage 2 — Skill Extraction



Extractor model E distills the pool into a single skill S(E,M,D).

Per-trajectory analysis, then hierarchical merge

Stage 3 — Skill Consumption



Same M runs held-out tasks with the skill in its system prompt.

Delta = Perf(M with skill) minus Perf(M)

Most prior benchmarks only measure stage 3. SkillLens measures all three.

Two questions, two papers

SkillLens

Diagnosis

Do model-generated skills actually work? When do they help, when do they hurt, and what makes the difference?

- 6 targets x 5 extractors x 5 domains
- All three lifecycle stages, end-to-end
- 25% of injected skills degrade performance

SkillOpt

Prescription

If we knew what makes a skill useful, could we train one — with the discipline of a deep-learning optimizer?

- Bounded edits, validation gate, slow/meta update
- Best or tied-best on 52 of 52 evaluated cells
- +23.5 pp average gain on GPT-5.5 direct chat

Paper 1 of 2 — Diagnosis

SkillLens — Are model-generated agent skills actually any good?

"75% of model-generated skills help.
***25% hurt**. Why?"*

1
5 domains
Embodied, Productivity, Coding, Web Search, Tool Calling

2
6 target models
GPT, Gemini, Qwen at multiple scales

3
5 extractor models
same families

4
3 lifecycle stages
experience, extraction, consumption

Huang, Xu, Yang, et al., *From Raw Experience to Skill Consumption* (2026). aka.ms/SkillLens

Study design: holding everything fixed except the extractor and target

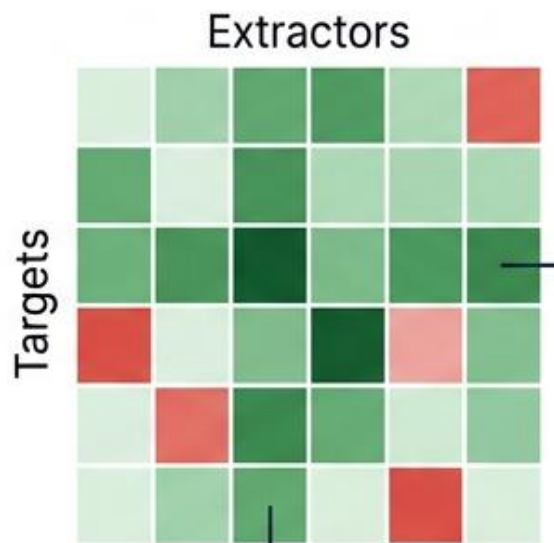
Pipeline

1. Target M runs train split, produces trajectory pool T(M,D).

2. Extractor E distills the pool into a single skill S(E,M,D).

3. Same M consumes the skill on test split, we measure Delta.

Δ-matrix



Target Evolvability (TE) — how much M benefits when others extract for it

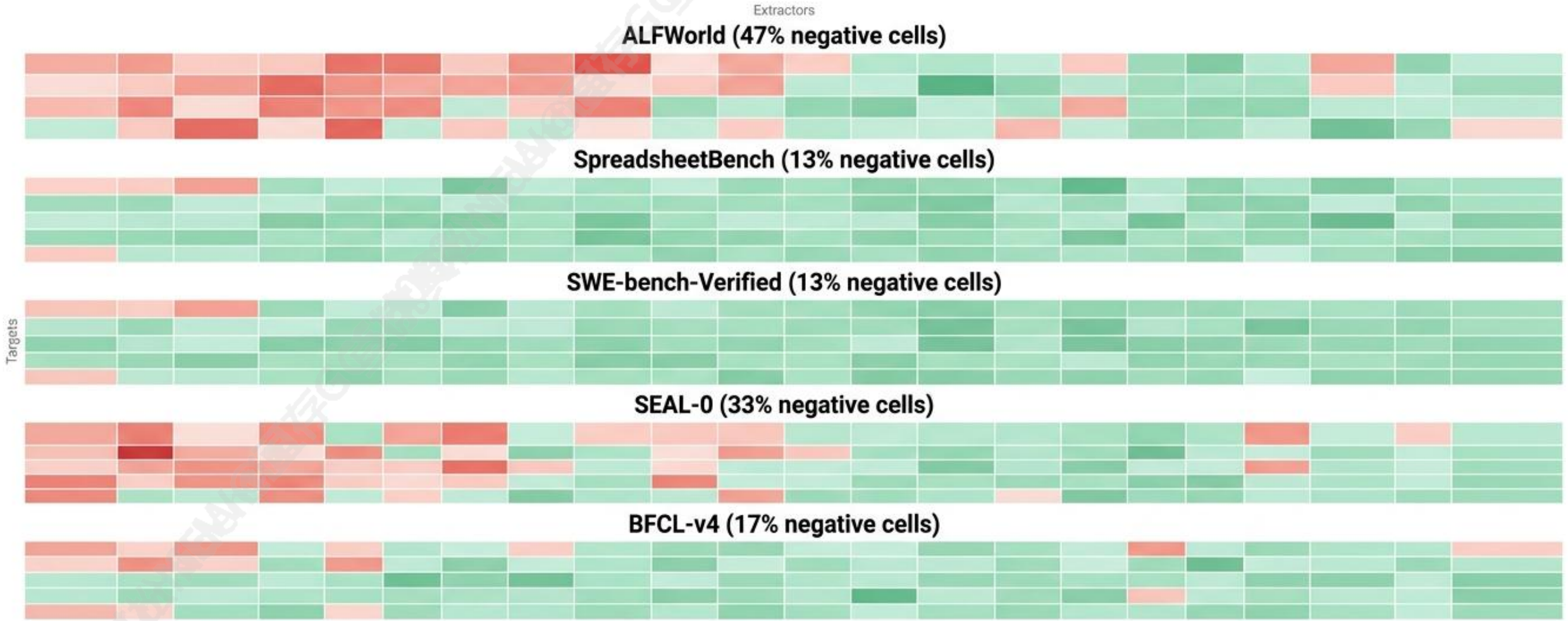
Extraction Efficacy (EE) — how good E is at extracting skills that help others

Minimal extraction framework

- ✗ No domain-specific heuristics
- ✗ No filtering rules
- ✗ No optimization tricks
- ✅ Only: per-trajectory analysis, then hierarchical merge, then skill synthesis.

Differences reflect extractor capability, not pipeline engineering.

Main result: 75% help, 25% hurt — and the risk is domain-dependent



1

On average, **model-generated skills help** — positive in 75% of all 150 cells.

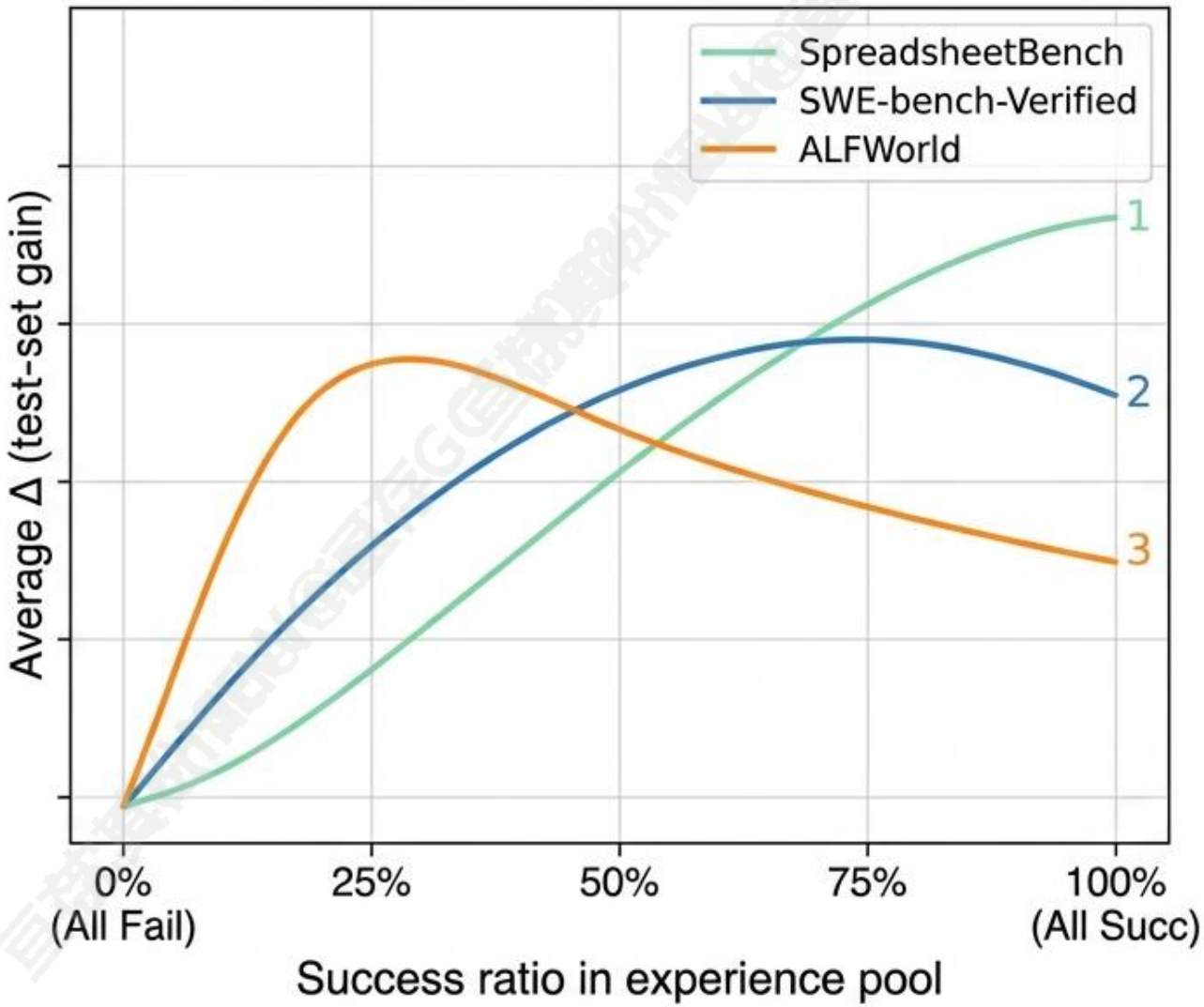
2

Negative transfer is real and **domain-dependent**. ALFWorld is fragile (47% negative); SpreadsheetBench is stable (13%).

3

Stronger executor is not the same as better extractor. Gemini-3.1-Flash-Lite tops EE on SpreadsheetBench while GPT-5.4 ranks last.

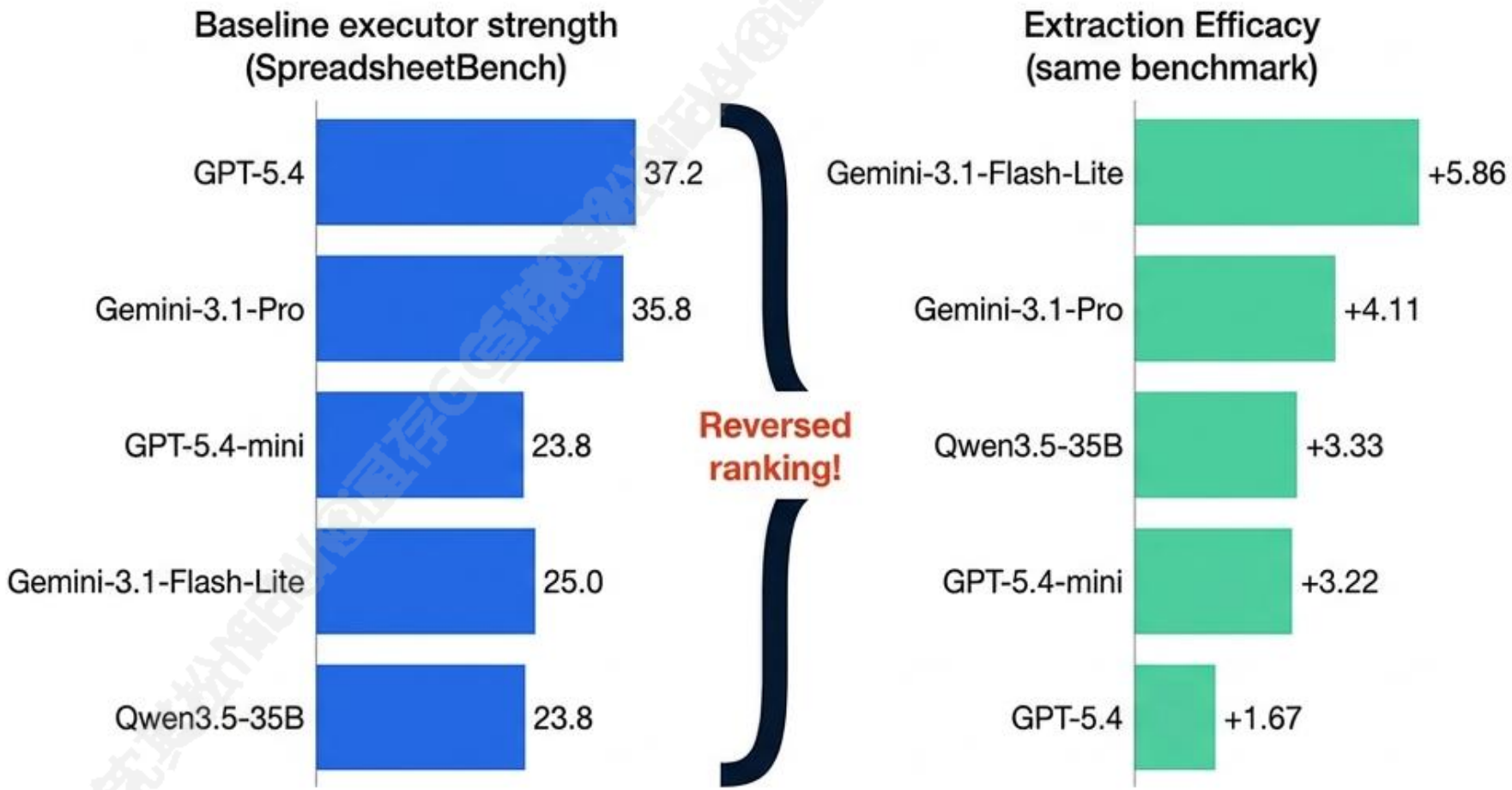
Stage 1 — What goes into the pool changes what comes out



- ✅ Successful trajectories are the foundation. — Pure-failure pools always lose.
- ✅ The optimal success/failure mix is domain-specific. — Embodied domains benefit from many failures; productivity and coding prefer mostly successes.
- ✅ Why? — Failures in ALFWorld expose dead-end states and invalid actions; in SpreadsheetBench, failed runs mostly look the same.

Skill quality is largely determined before the extractor even gets to read anything.

Stage 2 — A great executor can still be a poor extractor

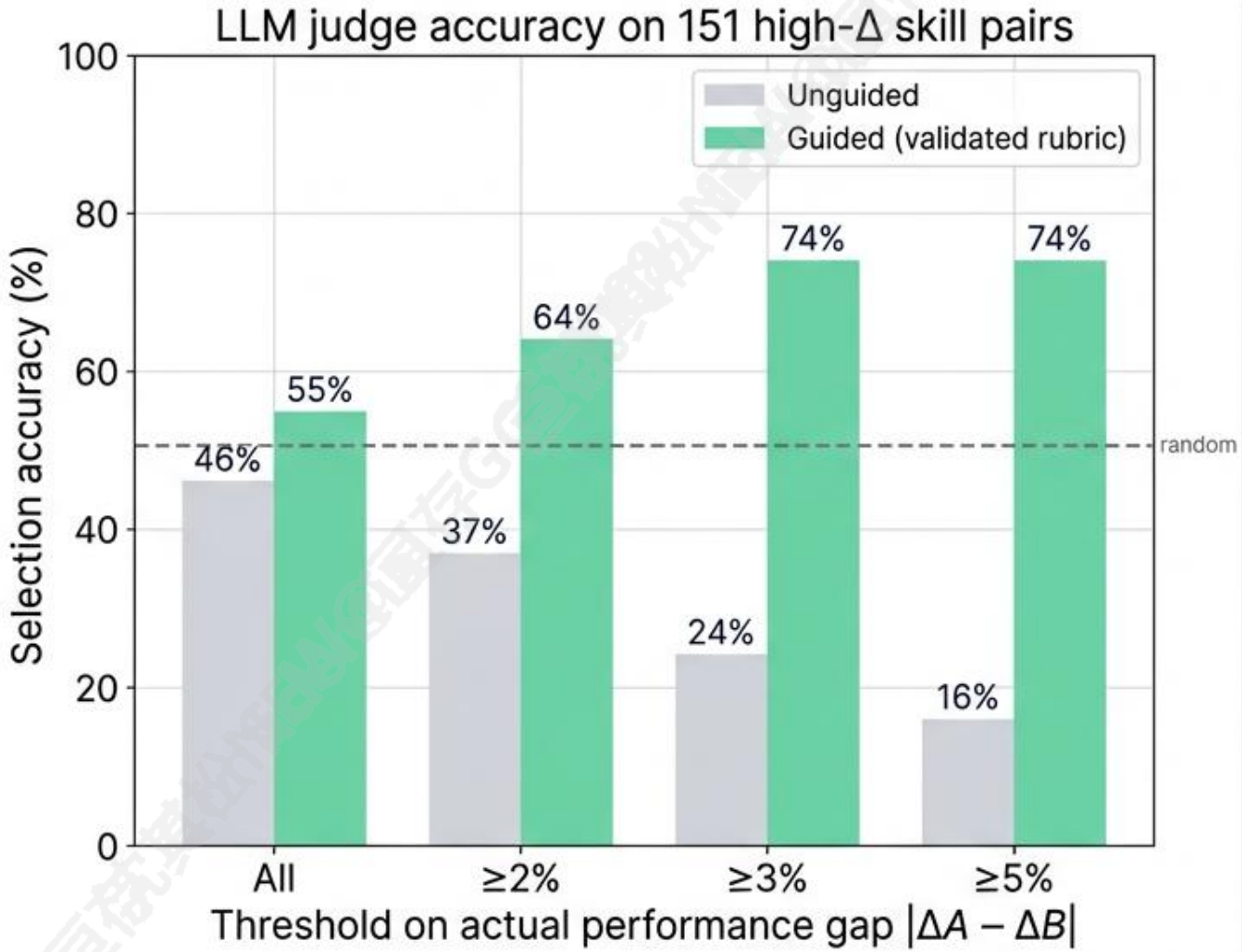


Skill extraction = compress another model's trajectories into rules that that model can follow.

It depends on perspective-taking, not raw task strength.

Implication: training-time and inference-time models can — and probably should — be different.

The skill that reads better is often the one that performs worse



Δ = each skill's real test-set gain; A and B are two skills for the same model and task — larger gap = easier to tell apart

Unguided LLM judge ≈ random. — Overall accuracy 46.4%.

It gets worse on the hardest pairs. — When the gap is at least 5 pp, the judge picks the worse skill 84% of the time.

Why this matters. — If we trust an LLM's gut on skill text, we **ship the wrong skill** more often than not.

What makes a skill good is not what makes it sound good.

Closing the loop: the meta-skill that screens for utility

Rubric discovery

1. Take all (A, B) skill pairs with large Δ gap.
2. GPT-5.4 extracts per-pair differences.
3. Iteratively merge — 7 raw candidate dimensions.
4. Test each via pairwise judge "better-rate".
5. Keep dimensions with better-rate ≥ 64%.

Validated rubric (3 dimensions)

✓

Failure Mechanism Encoding

Names why things go wrong, not just that they do.

✓

Actionable Specificity

Concrete remedies, not generic advice.

✓

High-Risk Action Blacklist

Explicit list of what not to do.

Effect on downstream Δ

Inserted as a one-paragraph prior in the extractor's system prompt — no pipeline change.

From diagnosis to prescription

SkillLens — what we learned

- Skills usually help, but 1 in 4 cells hurts.
- Pool quality matters before extraction even starts.
- Extraction is its own skill — not the same as execution.
- LLM “plausibility” of skill text is a bad predictor of utility.
- A meta-skill helps the extractor write useful skills, not pretty ones.

but...

Open problems → SkillOpt

- Even with the meta-skill, extraction is one-shot: one pool → one skill, no feedback loop.
- No way to roll back a bad edit. No way to prove an edit helps before deploying it.
- The optimizer doesn’t learn from rejected proposals.
- We need to treat the skill as a trainable state, not a one-time artifact.

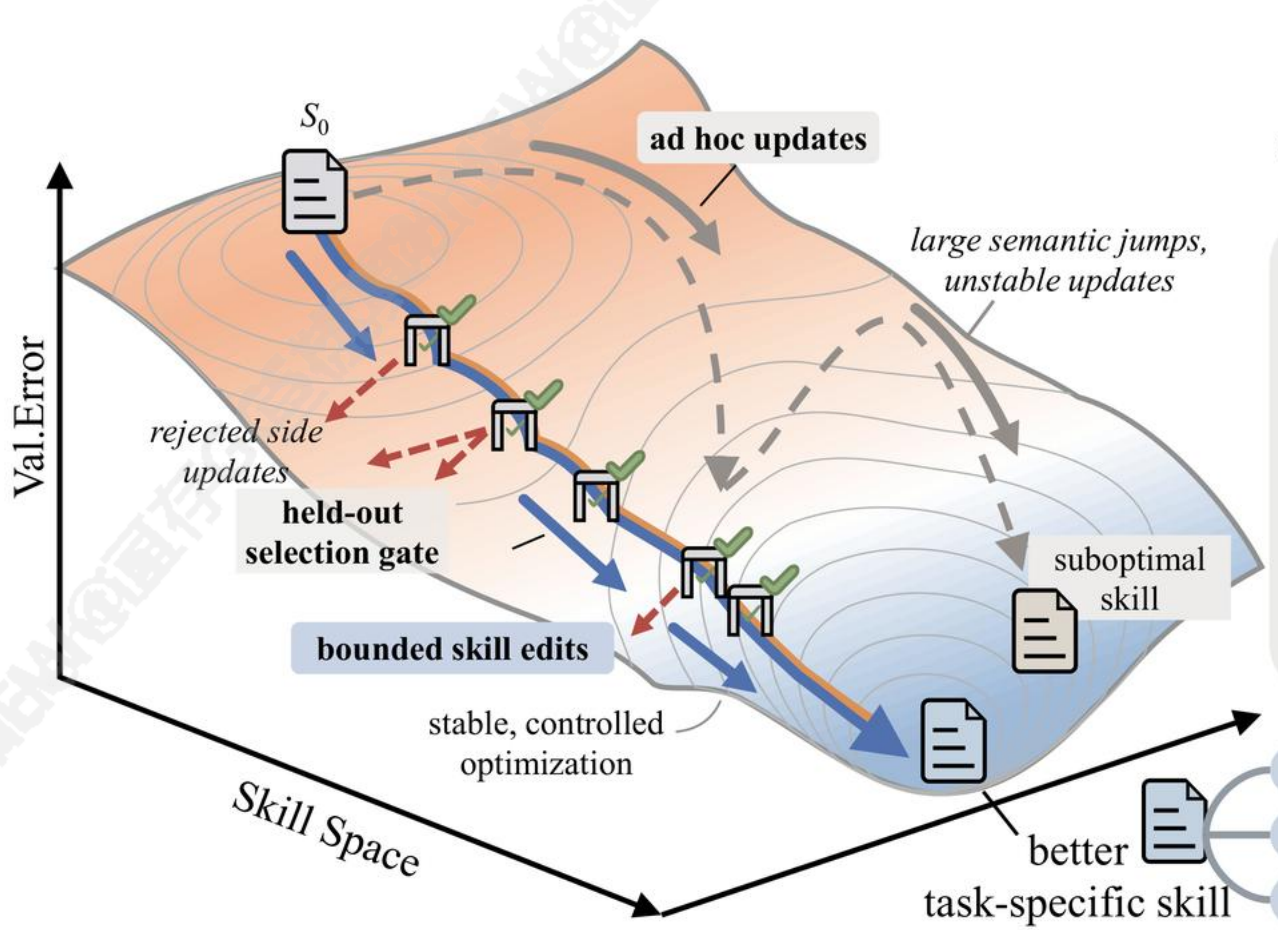
Paper 2 of 2 — Prescription

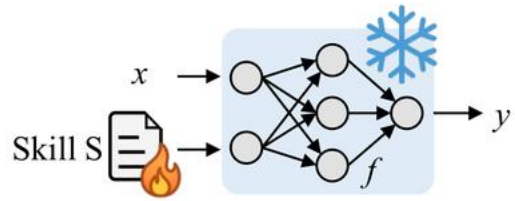
SkillOpt — the skill is the trainable external state of a frozen agent

Weight-space training	Text-space SkillOpt
Forward pass = run model on batch	Forward pass = run agent on rollout batch
Loss = scored error	Score = scored trajectory reward
Backward pass = gradient	Backward pass = optimizer LLM proposes edits
Optimizer step = update weights	Optimizer step = bounded add / delete / replace on best_skill.md
Validation gate / early stopping	Held-out gate accepts only score-improving edits
Momentum	Epoch-wise slow / meta update

Same discipline. Different state space.

Why bounded updates? Skill training is gradient descent in text space





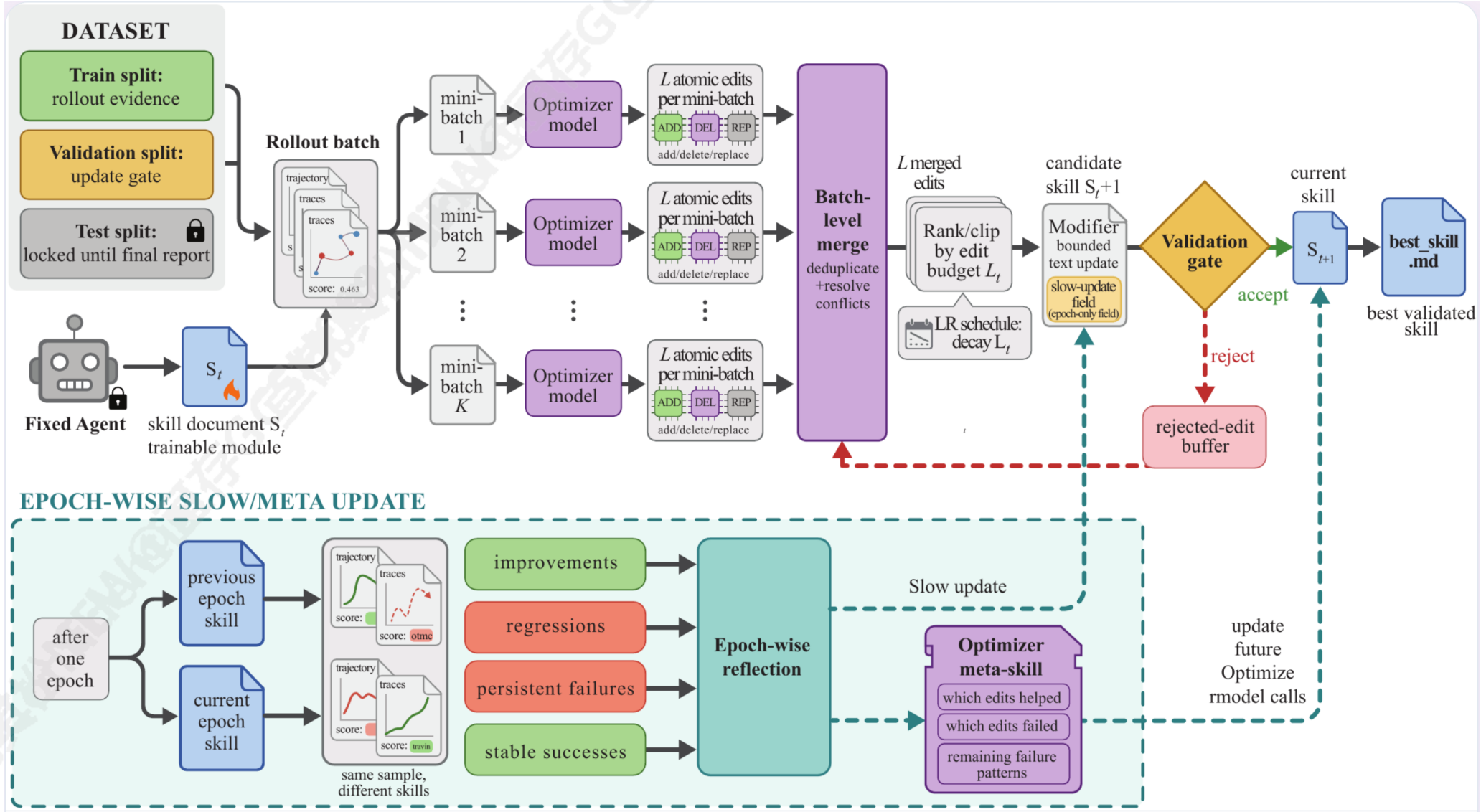
Text-space optimization analogy

parameter	→	skill document
gradient direction	→	trajectory-derived edit direction
learning rate	→	edit budget
validation check	→	held-out selection gate
stable training setting	→	batch / minibatch/ schedule / gate

- better task-specific ability
- cross-model /harness generalization
- low-cost transfer and iteration

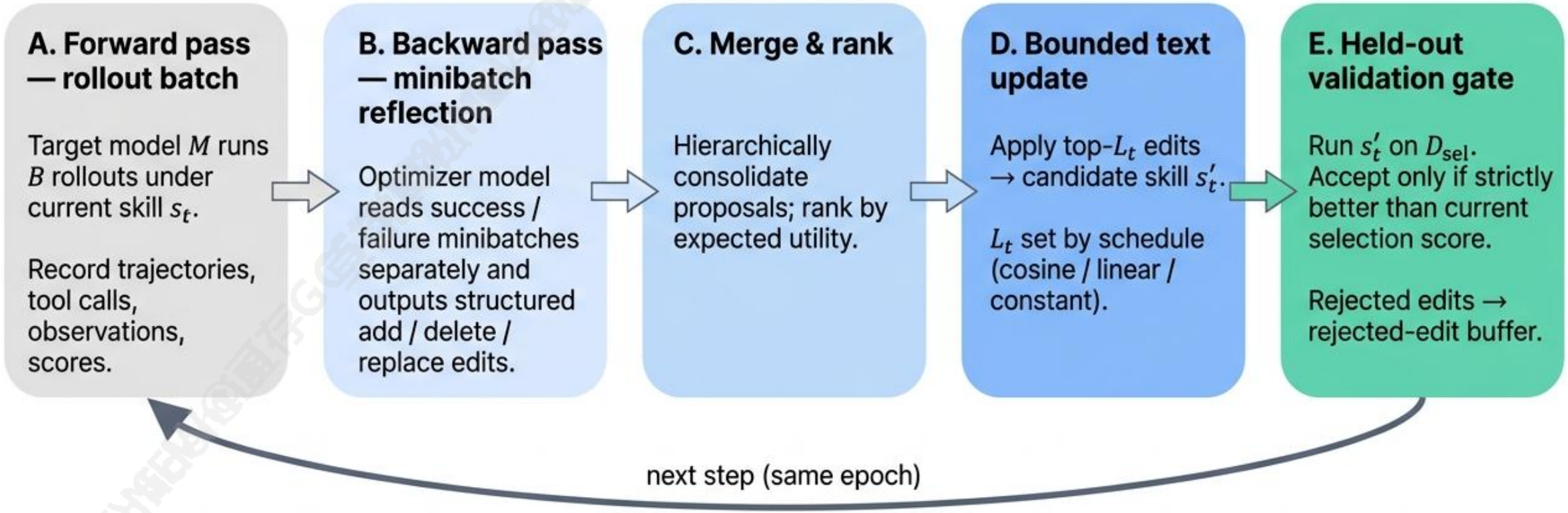
Bounded, validation-gated edits descend the validation-error surface step by step; ad hoc rewrites jump and stall at a sub-optimal skill.

A training loop for natural-language skills.



SkillOpt pipeline from the paper. The frozen target model executes with the current skill; the optimizer model proposes bounded edits; held-out validation decides whether the candidate becomes the new current skill.

SkillOpt pipeline — one optimization step end-to-end



end-of-epoch slow / meta update

Compare current vs previous-epoch skill on shared tasks; write longitudinal guidance into a protected region of the skill.

Four design choices that turn skill-editing into training

Bounded learning rate L_t

Why it matters: Unbounded rewrites erase useful rules.

What we found: Any moderate edit budget (L_t = 2–8) beats unbounded rewriting; "without lr" drops Spreadsheet 77.5 → 75.7, LiveMath 61.3 → 57.3.

Held-out validation gate

Why it matters: Plausible diagnoses can still hurt the target.

What we found: Gate keeps only 1–4 of dozens of proposed edits per epoch — OfficeQA's +39 pp came from a single accepted edit.

Rejected-edit buffer

Why it matters: Failed proposals are negative gradients.

What we found: Removing the buffer drops Spreadsheet 77.5 → 72.9, LiveMath 61.3 → 58.9.

Epoch-wise slow / meta update

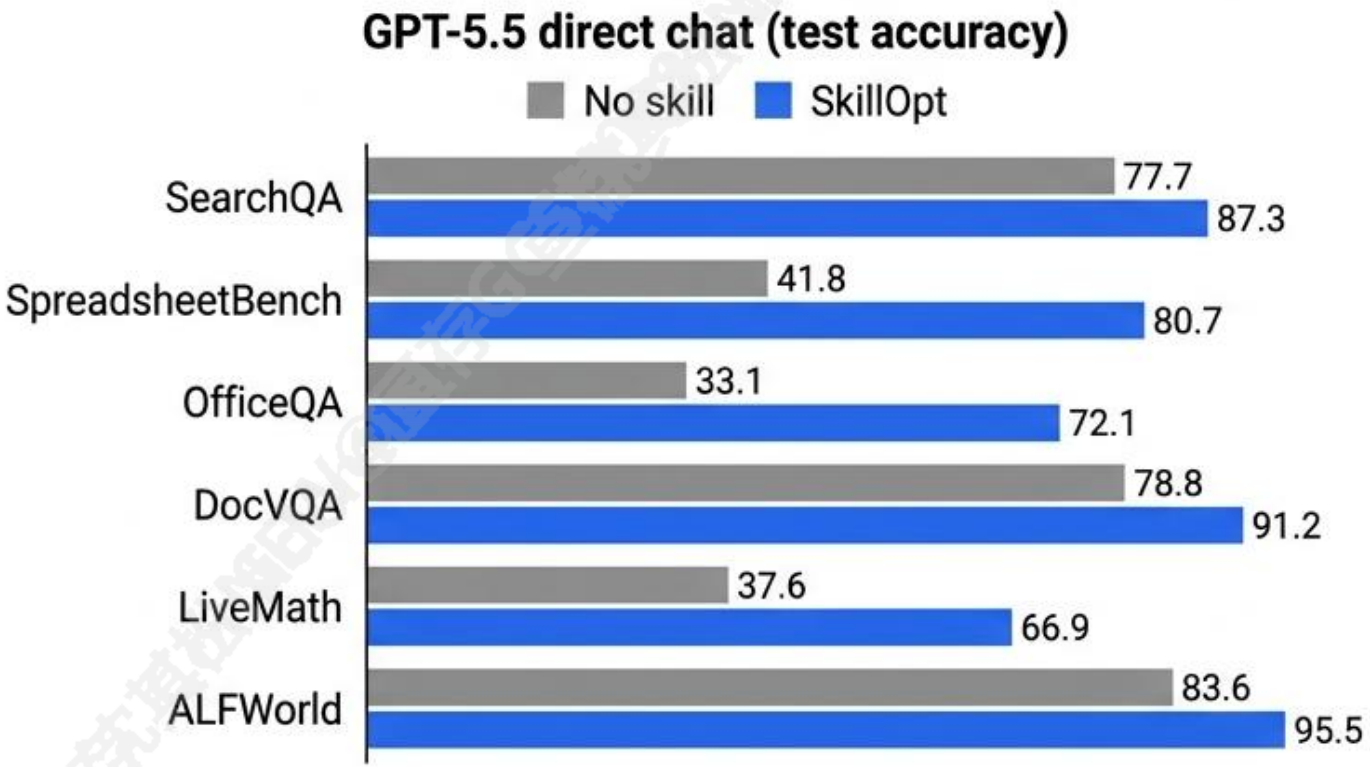
Why it matters: Some patterns only emerge across epochs.

What we found: Removing both meta + slow drops Spreadsheet 77.5 → 55.0 — the single largest degradation in any ablation.

Best or tied-best on **52 / 52** evaluated cells

52 / 52 cells

6 benchmarks × 7 target models × 3 execution modes — SkillOpt is best or tied on every single one.



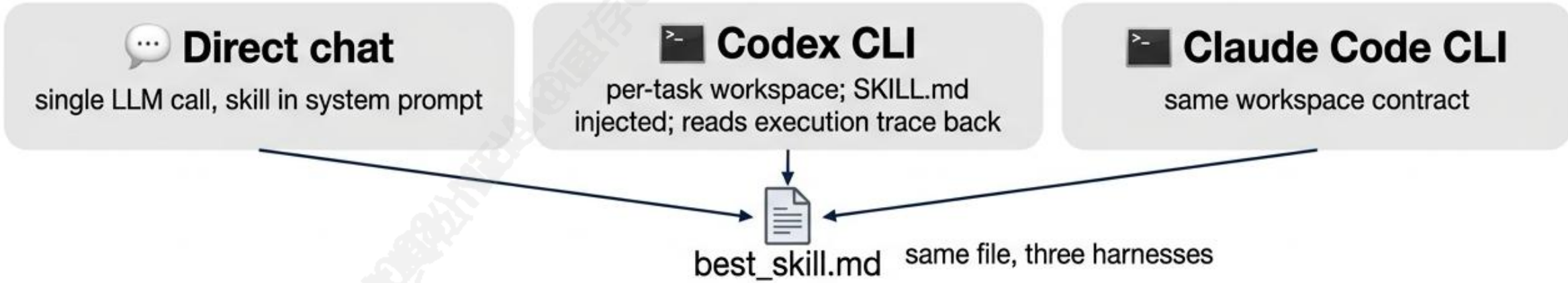
Average — 58.8 → 82.3 (+23.5 pp over no skill)
Best per-cell baseline avg = 76.9 (+5.4 pp over an oracle baseline)

1. **Procedural benchmarks benefit most.**
   Spreadsheet 41.8 → 80.7 (+38.9 pp),
   OfficeQA 33.1 → 72.1 (+39.0 pp),
   LiveMath 37.6 → 66.9 (+29.3 pp).

2. **Helps small models even more (relative).**
   Qwen3.5-4B Spreadsheet 9.3 → 23.9 (×2.6),
   GPT-5.4-nano ALFWorld 34.3 → 69.4 (×2.0).

3. **Gain over no skill by harness (GPT-5.5).**
   Direct chat +23.5 pp, Codex +24.8 pp,
   Claude Code +19.1 pp.

One artifact, three harnesses — and it transfers



Cross-model (GPT-5.4 → GPT-5.4-mini)	SpreadsheetBench	**+9.4 pp**
Cross-model (GPT-5.4 → GPT-5.4-nano)	SpreadsheetBench	+3.0 pp
Cross-harness		
Cross-harness (Codex → Claude Code)	SpreadsheetBench (GPT-5.5)	**+59.7 pp**
Cross-harness (Claude Code → Codex)	LiveMath (GPT-5.5)	+12.8 pp
Cross-benchmark		
Cross-benchmark (OlympiadBench → Omni-MATH)	GPT-5.4	+3.7 pp

"Optimize the skill once. Re-use it across model scales, execution environments, and nearby benchmarks. No weight updates."

Ablations confirm the four-knob story

Setting	SearchQA	Spreadsheet	LiveMath
Default (lr = 4, gate, buffer, slow/meta)	**87.1**	**77.5**	**61.3**
Without learning-rate bound (uncontrolled rewriting)	84.6 ↓	75.7 ↓	57.3 ↓
Without rejected-edit buffer	85.5 ↓	72.9 ↓	58.9 ↓
Without meta-skill AND slow update	86.3 ↓	**55.0** ↓	59.7 ↓

Sweeps over learning rate, schedule, batch size, and mini-batch size all stay within ±2 pp of the default. The four design choices above are what carries the gain.

- ✅ Bounded LR matters — uncontrolled rewrites lose 2–4 pp across the board.
- ✅ Rejected buffer is a stabilizer — its main effect is on procedural benchmarks.
- ✅ Slow / meta update is crucial on long-horizon procedural tasks — losing it costs –22 pp on Spreadsheet alone.

Hyperparameter analysis for the text optimizer.

(a) Training set size

Setting	SearchQA	Spreadsheet	LiveMath
1 example	81.0	47.5	59.1
20% train	84.1	69.0	65.9
40% train	**86.1**	73.5	64.8
80% train	**86.1**	77.6	67.0
100% train	84.1	**78.0**	**70.5**

(b) Mini-batchsize

Setting	SearchQA	Spreadsheet	LiveMath
1	85.9	75.4	60.5
2	86.3	77.1	54.8
4	86.9	75.4	**64.5**
8	**87.1**	77.5	61.3
16	87.0	**77.9**	61.3
32	86.9	77.5	58.9

(c) Batchsize

Setting	SearchQA	Spreadsheet	LiveMath
8	85.1	76.8	58.1
24	86.4	77.1	**62.9**
40	87.1	**77.5**	61.3
56	86.5	76.8	56.5
full epoch	**87.2**	75.0	53.2

(d) Learning rate

Setting	SearchQA	Spreadsheet	LiveMath
lr=1	85.5	77.5	62.1
lr=2	86.7	77.5	60.5
lr=4	86.5	**78.2**	56.5
lr=8	**87.0**	73.6	**66.9**
lr=16	86.8	**78.2**	65.3

(e) Learning-rate scheduler

Setting	SearchQA	Spreadsheet	LiveMath
constant	**87.3**	**80.7**	62.1
cosine	87.1	77.5	61.3
linear	87.2	72.9	**62.9**

(f) Slow-update samples

Setting	SearchQA	Spreadsheet	LiveMath
5	86.8	76.4	64.5
10	86.4	74.3	**65.3**
20	**87.1**	**77.5**	61.3
40	86.9	75.4	54.8

Table 2 Hyperparameter analysis for the text optimizer. Each panel changes one scalar or scheduling factor from the default setting unless noted. Panel (a) fixes the split to 4:1:5 train/selection/test; the 1-example, 20%, 40%, and 80% rows use subsets of the training partition, and the 100% row reuses the completed 4:1:5 split-ratio run. Panel (b) sweeps the reflection mini-batchsize B_m; panel (c) sweeps the rollout batchsize B.

The learned skills are small, procedural, and readable

< 2,000 tokens
final skill fits on one screen

1–4 accepted edits
the validation gate keeps very few

0.6 M – 46 M tokens / pt
measurable training cost, zero added at inference

SearchQA

"Infer the expected answer type from clue wording, then choose the shortest canonical entity supported by co-occurring distinctive evidence."

SpreadsheetBench

"Inspect workbook structure and formulas, then write evaluated static values across the full requested target range instead of relying on Excel recalculation."

OfficeQA

"Treat oracle parsed pages as primary evidence, lock table/date/unit context, and output exactly the requested rounded value without extra labels."

DocVQA

"For tables, forms, charts, and legends, first bind the question to the exact visual row/header/field, then copy only the aligned answer span."

LiveMath

"In strongest-statement MCQs, rank choices by theorem strength and prefer a justified stronger-result option over true but weaker corollaries."

ALFWorld

"Keep a horizon-aware visited/frontier ledger, diversify search after repeated same-type failures, and avoid revisiting the destination until holding the target."

These read like rules a thoughtful practitioner would write after a day with the benchmark — only they were learned automatically, edit-by-edit, on held-out data.

SkillLens

SkillOpt

Two papers, one story — diagnosis and prescription

SkillLens finding (diagnosis)	SkillOpt mechanism (prescription)
25% of one-shot model-generated skills hurt performance.	Held-out validation gate — strictly score-improving edits only.
Plausibility of skill text does not predict utility.	Propose-and-test loop — every candidate measured on the selection split before it can ship.
Pool quality and extraction quality vary with model and domain.	Rollout batches + reflection minibatches + rejected-edit buffer — keep generating evidence, learn from negatives.
A **meta**-skill helps extractors write useful skills, not pretty ones.	Epoch-wise slow / meta update — durable longitudinal lessons in a protected region.

Same target. Same artifact. Different position on the maturity curve.

Outlook: where the trainable skill goes next

1 Skill libraries, not single skills.

Move from one best_skill.md per domain to libraries with retrieval, selection, composition. Interference and reuse become first-class.

2 Reward-free / preference-driven gates.

For open-ended tasks without a scalar grader, replace the strict score gate with pairwise preference judges. The validated rubric becomes the gate itself.

3 Optimizer-side meta-skills shared across domains.

The training-time meta-skill can be reused across benchmarks — a meta-curriculum for skill optimization.

4 Self-distillation back into weights.

Use the optimized text artifact as a teacher signal to update the target model itself — text-space training as a stepping stone to weight-space training.

SkillLens — aka.ms/SkillLens

SkillOpt — arXiv

Q & A

Thank you — happy to take questions.