Agentic Skill Router - All Experiments about retriever

实验日期: 2026-05-23 至 2026-05-27 数据: SkillRouter eval-core (arXiv:2603.22455) 裁剪匿名化语料,以及论文原始 Easy 78,361 池 范围: Claude Code paired/fresh-env 实验、Codex A-M 策略实验、J/L/M 规模扩展实验、论文 original Hard pool 对齐实验、SkillRouter 75 core × Easy × multi-skill 对齐实验

摘要

`skill-router` 的目标是把低频 skill 从宿主常驻上下文中移出,在需要时再从 disabled-skill 语料中路由回来。本报告评估多轮策略迭代中的准确率、成本和可扩展性。

主要结论如下:

1. **78K Easy × 75 core 论文对齐实验中,agentic metadata-only 路由器超过论文最强 nd 基线。** 最佳 codex/J-bounded-v2 Hit@1 = 40.0%,超 Qwen3-Emb-8B nd (30.7%) 达 +9.3pp,并高于 BM25 带 full body (34.7%)。6/6 cell 跑赢 BM25 nd,4/6 超 Qwen3-Emb-0.6B nd。详见 §9。
2. **宿主/模型选择比变体选择影响更大。** 同一 SKILL.md 模板下 Codex (GPT-5.5) 平均 36.4%,Claude Code (Opus 4.7) 平均 27.1%,差 9.3pp;三变体间最大差距仅 7.3pp。150-skill fresh-env 复跑同样如此:Codex router aggregate 为 321/360 (89.2%),Claude Code with-CLAUDE.md router aggregate 为 301/360 (83.6%);Codex native 23/24,Claude Code native 14/24。
3. **52% 的 query 构成 metadata-only 结构性天花板。** 75 query 中 39 个全 6 cell 皆 miss,其中 7 个出现跨变体跨宿主一致误选。突破需 body-on-tie。
4. **150-skill 到 1K 扩展仍较稳,但 79K Hard 显著下降。** L-agentic 150 → 1K 只掉 1 cell,但 M-bm25 在 79K Hard paper-core single 上为 14/24,J-v2 为 11/24 strict。说明 metadata-only 不能作为唯一决策源。
5. **Claude Code 下 trigger noise 是首要治理项,但不是 fresh-env 复跑后的唯一差异来源。** 早期 paired 实验中 CLAUDE.md 把 trigger 从 78% → 97.6%,accuracy 从 69% → 86.9%;新 16-variant fresh-env 复跑中 with-CLAUDE.md trigger 已达 96.9%,但仍低于 Codex router aggregate 5.6pp,剩余差异主要来自 host/model 的 metadata rerank、输出 contract 和上下文基线。

次要结论:A-router 固定 scorer 不可接受(Claude Code fresh-env 10/24,Codex 新 CLI 重跑 11/24);论文对照只能分维度比(论文未公开 Single×nd×Hard-only 格子);Codex 150-skill 新 CLI 重跑中 C-lite 与 L-agentic 达到 24/24,多数强策略落在 23/24 附近;Claude Code 最新 L-agentic restore 复跑为 22/24,说明 L 方案在 Claude 上可用但不如 Codex 稳定。

本报告只评估路由层,不评估匹配 skill 后的下游执行成功率。所有排名都是单次运行的 strict-match routing-only 分数,默认包含 ambiguous 样本。

Background

近期 agentic retrieval 方向的多篇论文都在强调同一个工程判断:相比把 retrieval 当作一次性 top-k 黑盒,更有效的方式是把检索能力包装成 agent 可控、可迭代、可检查的工具接口。本报告的实验正是沿着这个方向,把 skill routing 表达为一组可调用的 corpus/search/inspect 工具,再观察不同宿主和策略在准确率、上下文、成本上的表现。

论文	与本报告的关系
Beyond Semantic Similarity: Rethinking Retrieval for Agentic Search via Direct Corpus Interaction	将 retrieval 重新理解为 agent 与 corpus 直接交互的问题,强调 search/read/script 等工具式 corpus 操作比固定 embedding top-k 更适合多步推理。
Is Grep All You Need? How Agent Harnesses Reshape Agentic Search	说明 retrieval 结果不只取决于 ranker,还取决于 agent harness、工具输出形态和可交互程度;grep/lexical 方法在合适 harness 下仍有竞争力。
AgenticRAG: Agentic Retrieval for Enterprise Knowledge Bases	关注把企业搜索系统封装成 agent 可调用工具,让模型通过 search/find/open/summarize 等动作迭代获取证据。
Rethinking Agentic Search with Pi-Serini: Is Lexical Retrieval Sufficient?	在 agentic loop 中重新评估 BM25/lexical retrieval,突出 retrieval depth、browse/read 控制和 agent 工具循环的重要性。
SkillRouter: Skill Routing for LLM Agents at Scale	本报告的数据和对照基准来自该工作的问题设定:在大规模 skill catalog 中为 agent 找到正确 skill,而不是把所有 skill body 常驻塞进 prompt。

核心连接是:这些论文都支持"为 agent 提供更方便、更可控的检索工具可以提升效果"的方向;本报告把该方向落到 disabled-skill routing 上,比较 native、bash-based、tool-wrapped agentic retrieval 和 BM25 大规模检索几类实现。

---

1. Overview

1.1 实验目的

当已启用 skill 无法覆盖用户请求时,agent 需要从 disabled skill 中选择最合适的一个。本实验回答四个问题:

- **RQ1 准确率:** 哪种实现最可靠地选中 ground-truth skill?
- **RQ2 成本:** token、工具调用、耗时和美元成本如何权衡?
- **RQ3 机制:** 手写 bash 检索、封装成 tool 的 agentic retriever、BM25 ranker 和宿主 native skill 选择各自的边界是什么?
- **RQ4 扩展性:** 150-skill 上的策略排序能否外推到 1K synthetic、79K Hard 或 78K Easy 规模?

1.2 策略分类

本报告中所有变体可以按实现方式分成四类:

类别	变体	核心实现
Native baseline	G-native	不显式调用 router,由宿主模型基于 native skill listing 直接选择。
Bash-based retrieval	B-cc、C-lite、H-bounded、I-meta、J-bounded、J-bounded-v2、K-bounded、K-lite	agent 使用 `find` / `grep` / `sed` / `awk` 等 shell primitive 在 disabled `SKILL.md` 文件上检索;后续变体逐步加入输出预算、metadata-only 约束、frontmatter tie-break 和更稳定的关键词策略。
Tool-wrapped agentic retrieval	A-router、D-agentic、D-agentic-metadata、E-digest、L-agentic	将检索方法封装成稳定 CLI/tool primitive:一键 `skills route`、DCI `search/inspect`、`skill-corpus catalog/show`、或 `corpus search/inspect` 多步 loop。模型主要负责 query rewrite、候选比较和最终选择。
Large-scale BM25 retrieval	M-bm25	使用 BM25 ranker 做大规模 metadata shortlist,再由模型比较候选 metadata;主要服务 78K/79K 规模下的可扩展检索。

1.3 指标含义

- **Accuracy / Hit@1:** 输出 skill 与 expected skill 严格匹配;78K multi-skill 实验中按论文 any-gt Hit@1 口径。
- **Trigger:** router workflow 是否被实际调用。G-native 不适用。
- **Turns / duration / cost:** 模型调用轮次、端到端耗时和估算/上报成本。
- **ctxStart / ctxEnd / growth:** 首次和末次模型调用的上下文窗口大小,以及中间增长量。
- **Tool calls / bash errors / cache hit:** 反映实现的操作复杂度、CLI 调用可靠性和缓存经济性。
- **Failure type / miss distribution:** 用于区分选中高质量 distractor、hallucinated id、no-match 或格式错误。

1.4 数据集情况

- **24 queries / 150 skills:** 主控对比集,用于 Claude Code 与 Codex 的 16 策略横向比较。
- **1K synthetic:** 150-skill 对比语料 + 850 synthetic noise skill,只跑 Codex L-agentic,用于验证 `corpus search/inspect` 在候选数扩张时的稳定性。
- **79K Hard:** SkillRouter original Hard pool,用于检查 metadata-only 策略在高质量 distractor 下的外推边界。报告保留该部分分析,即使 HTML 主视图已弱化独立展示。
- **78K Easy / 75 core:** 与 SkillRouter 论文 nd 基线最直接可比的扩展实验,K/J-v2/M × Claude/Codex 共 6 cell。

---

2. 数据与语料

2.1 语料总览

本报告涉及三层规模递增的语料,全部来自 SkillRouter `eval-core` (arXiv:2603.22455):

层级	规模	来源	使用节	安装形态
150-skill 对比	150	19 gt + 80 targeted distractor + 51 easy noise,确定性抽样	§6-§7	完整 SKILL.md (metadata + body)
78K Easy pool	78,361	`eval_core/easy/*.jsonl.gz` 全量	§9	metadata-only (name + description,body 剥离)
79K Hard pool	79,141	`eval_core/hard/*.jsonl.gz` (Easy + 780 LLM distractor)	§8	metadata-only

三层语料共享 19 个 ground-truth skill,但 pool 大小和 distractor 构成不同:150-skill 只含 80 个 targeted distractor 和 51 个 easy noise;78K Easy 是论文原始 easy pool 全量;79K Hard 在 Easy 基础上加入 780 个 GPT-4o-mini 蒸馏出的高质量 distractor。

2.2 150-skill 对比语料

成分	数量	说明
Ground-truth skill	19	24 个 single-skill task 的 gt 实体,部分任务共享同一 gt
针对性 distractor	80	SkillRouter 用 GPT-4o-mini 生成的同域近义干扰
噪声 skill	51	从 easy pool 确定性随机抽样补齐
合计	150	150-skill 对比实验规模

首版语料曾保留 `gt-*` / `distractor-*` 目录名前缀,会泄露答案。修复后,所有目录名和 frontmatter `name` 均匿名化为 `skill-001` 到 `skill-150`,并在确定性洗牌后分配。映射只保存在 `corpus-manifest.json`,用于离线分析,不暴露给 agent。

78K Easy 和 79K Hard 的安装使用 `sr-XXXXX` 确定性洗牌映射 (seed=20260525);frontmatter `name:` 和 `description:` 保留原始上游字符串(论文 nd 输入需要 name 信号),目录名匿名化为 `sr-XXXXX` 以避免泄露 `gt/` / `distractor/` 前缀。

2.3 查询集

24 个 query 直接使用 SkillsBench single-skill 任务的 `instruction_text`。这些 query 是完整任务描述,通常包含文件路径、输出格式和约束,不是为了本实验手写的短关键词查询。完整列表见 `queries.json`。

后续 paper-core single 对齐实验改用 SkillRouter 论文 75 core queries 中 `core_gt_ids.length == 1` 的 24 条。它与本报告初始 `queries.json` 的 24 条只有 16 条重叠;初始集合包含若干 generic file-type task,而论文 core single 集合包含 `flink-query`、`invoice-fraud-detection`、`latex-formula-extraction`、`manufacturing-*`、`paper-anonymizer`、`simpo-code-reproduction`、`xlsx-recover-data` 等任务。

需要注意两个数据层面的影响:

- `gh-repo-analytics` 的 gt 标注存在争议:gt skill 是工具中心描述,而多个 router 选择的 distractor 更贴近 query 的任务中心描述。
- `skill-105` 是通用 Excel skill,被 5 个 query 共用,会放大 spreadsheet 类描述质量对总分的影响。

---

3. 变体

3.1 按实现方式分类

类别	变体	路由方式
Native baseline	G-native	不加载 router,语料全启用,由宿主 native skill 选择机制直接匹配。
Bash-based retrieval	B-cc	agent 自由使用 shell 搜索 disabled-skill 目录,最接近 DCI-Agent-CC 的 fully free shell。
Bash-based retrieval	C-lite	仅 bash,用有界 grep / sed 管道局部读取,限制输出规模。
Bash-based retrieval	H-bounded	类似 B,但限制裸 `ls` 和 corpus 输出,减少 prompt blowup。
Bash-based retrieval	I-meta	只读 description/frontmatter metadata,禁止读取 body。
Bash-based retrieval	J-bounded	先抽 3-5 个关键词,grep 过滤 description 短列表。
Bash-based retrieval	J-bounded-v2	J 的 scale-stable 修复版,用 `find -print0

类别	变体	路由方式
Bash-based retrieval	K-bounded	J 的加强版:grep shortlist 后允许模型查看候选 frontmatter,用 metadata tie-break。
Bash-based retrieval	K-lite / K-lite fixed	降低 K 的 shell 代码和工具成本;fixed 版修正准确率回退。
Tool-wrapped agentic retrieval	A-router	调 `skill-router skills route`,由 CLI 做 metadata / dci / lexical 级联评分并直接 commit。
Tool-wrapped agentic retrieval	D-agentic / D-agentic-metadata	结构化 DCI `search/inspect` 循环;metadata-only 版禁止读取 skill body。
Tool-wrapped agentic retrieval	E-digest	`skill-corpus catalog/show` 两步 wrapper,把目录和少量候选 body/metadata 作为工具输出。
Tool-wrapped agentic retrieval	L-agentic	`corpus search/inspect` 抽象,由 CLI 承担索引、分页、候选引用和 JSON schema,模型负责迭代 query 与证据比较。
Large-scale BM25 retrieval	M-bm25	L 的 BM25 ranker 版本:metadata BM25 shortlist + Codex/Claude metadata rerank,主要面向 78K/79K 规模。

3.2 设计取舍

- **Native baseline** 用于衡量宿主内置 skill listing 本身的质量,不是 production router。
- **Bash-based retrieval** 最大优点是透明、无新增 runtime 依赖;缺点是容易受 shell glob、输出截断、路径解析、工具调用次数影响,在 79K 规模下尤其明显。
- **Tool-wrapped agentic retrieval** 把检索过程变成稳定 tool contract,减少模型手写 shell 的负担。L-agentic 的 1K 结果支持这个方向,但 79K Hard 仍显示 metadata-only 决策不足。
- **Large-scale BM25 retrieval** 在大语料上提供低成本 shortlist,但 BM25 order 只能作为候选证据,不能代替最终 rerank;M-bm25 在 150 新跑下降到 19/24 就是一个反例。

Router 变体使用 `variants/routing-only/<variant>.SKILL.md`。所有 router 变体共享相同 frontmatter;差异只在 body 工作流。

---

4. 实验设计

4.1 Routing-only 设置

每个 query 末尾追加 `STOP_TAIL`,要求 agent 只输出:

```
{"matched_skill_name":"<skill-id>"}
```

然后停止。这样可以隔离路由步骤,避免下游文件处理、表格计算、PDF/PPTX 操作等执行质量污染路由指标。

4.2 Claude Code paired 实验

Claude Code 部分使用 `routing-only-paired.mjs`:

- 9 变体 x 24 query x 2 conditions = 432 cells。
- 两个 conditions: with-CLAUDE.md 与 without-CLAUDE.md。
- 每个 `(variant, condition)` 使用隔离 `$HOME`。
- G-native 例外:语料全启用,不加载 router plugin。
- Router 变体:语料全禁用,加载 `skill-router-skills`。
- 每 cell timeout 240s。

with-CLAUDE.md 写入:

```
# Skill routing

`skill-router-skills` is a routing Skill that searches a catalog of
locally-installed disabled skills.

When no enabled Skill clearly matches the user's query, you must call
`skill-router-skills` before answering. Do not invent a Skill name or
fabricate `matched_skill_name` without a Skill/tool result.
```

在跑完整 paired 实验前,先用 64-cell gate probe 验证这段 CLAUDE.md 是否能提升 trigger,并预先设置 gate criteria:

```
PASS = trigger_rate(with) >= 95%
     AND trigger_lift >= +20pp
     AND direct_match_preservation(with) >= 90%
     AND no degradation > 10pp vs without
```

probe 结果全部通过: fallback trigger lift +62.5pp,direct-match preservation 100%。

2026-05-26 之后又补跑了 fresh-env 16 变体矩阵,用于和 Codex 新 CLI 结果做更直接的横向归因:

- Claude Code:16 变体 x 24 query x 2 conditions = 768 cells。
- Codex:16 变体 x 24 query = 384 cells,但 Codex 正式口径以 `rerun-codex-newcli-full-2026-05-26` 为准。
- 每个 `(host, variant, condition, query)` 使用新的隔离 HOME/project,避免跨 cell 状态污染。
- 这组 fresh-env 结果用于 §6.5 和 §7.6 的跨宿主归因;早期 paired 结果仍用于 trigger policy 的 A/B 结论。

4.3 Codex 16x24 新 CLI 实验

Codex 部分使用同一 24-query、同一 150-skill 语料、同一 routing-only 输出 schema。它不是 Claude Code 的附录或次级验证,而是同等地位的宿主实验,用于比较不同宿主下同一组路由策略的排序。2026-05-26 之后的主结果统一采用新 CLI 语义和 fresh-env rerun。运行环境:

- `codex exec`
- model `gpt-5.5`
- reasoning effort `high`
- 16 变体 x 24 query = 384 cells
- 每个 `(variant, query)` 使用全新的 Codex home/project 端到端执行

Codex 版 router 做宿主路径适配: `.claude/skills` 改为 `.codex/skills`,disabled 后缀统一为 `.agentic-skill-router-disabled`,并统一 skill id 前缀处理。A-router、D-agentic、L-agentic、M-bm25 等 CLI 型策略保留各自语义,只统一底层 CLI 调用约定。

4.4 规模扩展实验

规模扩展分四类,验证 150-skill 结论是否能外推到更大 skill pool:

- 150-skill: 匿名化 150-skill 对比语料。
- 1K synthetic corpus: 150-skill 对比语料 + 850 synthetic noise skill,用于低成本观察索引和 prompt 的扩展趋势。
- 79,141-candidate Hard pool: SkillRouter eval-core Hard tier (§8,变体 J-bounded-v2、M-bm25)。
- 78,361-candidate Easy pool: SkillRouter eval-core Easy tier (§9,变体 K-bounded、J-bounded-v2、M-bm25,75 core queries)。

79K original Hard pool 实验使用 SkillRouter 论文数据集原始 skill IDs 的 opaque 映射。agent 可见的候选 id 是 `sr-*`;metadata 中保留原始 name / description,因此这不是完全匿名化,但不会把 `gt/` 或 `distractor/` 目录前缀暴露为答案线索。

论文对照有两个口径:

- 论文 `nd` 指 name + description only,在 75 core queries 上报告 Easy / Hard / Avg,不拆 single-only。
- 论文 `Single Hit@1` 是 24 single queries,但属于 full skill text 主结果,不是 `nd`。

因此本报告的 `paper-core single x Hard x metadata-only` 复跑只能作为补充格子,不能和论文已发布表格做严格同分母对比。

---

5. 指标

实验追踪 8 个核心指标、5 个补充诊断指标和 1 个故障分类指标。所有指标的计算逻辑统一封装在 `experiments/dci-compare/extract-metrics.mjs`,以下表格描述每项指标的精确来源:

5.1 核心 8 指标

指标	Claude 来源	Codex 来源	说明
accuracy	`matched_skill_name` 归一化后与 expected skill 严格相等	同左	跨变体核心成功率
triggerRate	任一 `assistant.tool_use.name == "Skill"` 且 `input.skill` 匹配 `/skill-router-skills/i`	任一 `command_execution.command` 引用 `/skills/skill-router-skills/SKILL.md`	router 是否实际被调用。G-native 不适用
totalTurns / avgTurns	`result.num_turns` (Claude 官方计数)	rollout 中 `event_msg/token_count` 事件数(即内部 Responses-API 调用数)	模型轮次
ctxStart	第一个 `assistant.message.usage` 去重后的 `(input_tokens + cache_creation_input_tokens + cache_read_input_tokens)`	rollout 中**第一个** `event_msg/token_count.info.last_token_usage.input_tokens` (OpenAI 约定:input_tokens 包含 cached)	首次模型调用的 prompt 大小
ctxEnd	**最后一个** `assistant.message.usage` 去重后的 `(input + cache_create + cache_read)` (注意:不是 `result.usage` ——后者是跨轮累计而非最后一轮的窗口大小)	rollout 中**最后一个** `event_msg/token_count.info.last_token_usage.input_tokens`	最后一次模型调用的 prompt 大小
ctxGrowth	`ctxEnd - ctxStart`	同左	会话中上下文累计增长量
cost	`result.total_cost_usd` (Anthropic 真实账单,已含 cache 折扣)	`(fresh × $5 + cached × $0.5 + output × $30) / 1M`,使用 gpt-5.5 2026-05-24 list price; `fresh = cumInput - cumCached`,取 rollout 中 `totalTokenUsage`	Claude 真实计费;Codex 为 list-price 估算,适合 Codex 内部对比,不应横向与 Claude 严格对照
duration	`result.duration_ms` (Claude 原生上报全墙钟)	runner 测量的 `Date.now() - t0` (Codex 不在 stream-json 中上报 duration)	端到端墙钟时间

5.2 补充 5 指标

指标	说明
toolCallCount	去重后的 `tool_use` 总数(Claude)或 `command_execution` 总数(Codex)。与 `numTurns` 互补——一次 turn 可包含多个 tool 调用,反之亦然。主报告只保留总量,不再展开逐工具类型明细
bashErrorCount	`command_execution.exit_code != 0` (Codex)或 `tool_result` 含 `is_error == true` / `tool_use_result.success == false` (Claude)的 bash 调用数,用作"agent 操作受阻"的代理指标
cacheHitRatio	Claude: `cumCacheRead / (cumInput + cumCacheRead + cumCacheCreate)`;Codex: `cumCached / cumInput`。独立的 token 经济性信号
outputTextLen	最终 `result.result` / `agent_message.text` 字符长度。短输出多为合规 JSON,长输出常伴随幻觉
failureType	当 miss 时分类: `distractor` (选中 pool 中其它合法 skill)/ `hallucinated` (返回 pool 外 id 或不存在名称)/ `no_match` (显式返回 "no-match")/ `format_error` (无法解析的输出)。提供"为什么错了"而非只看"是否错"

5.3 重要修正

- **旧稿 ctx_end 计算错误**:之前用 `result.usage.cache_read + cache_create + input` 作为 ctx_end,这是 Claude 跨所有 turn 的**累计**值,会被多轮变体放大。新口径只取**最后一次** assistant message 的 usage,反映模型在最后一次调用时实际看到的窗口大小。同样 Codex 的 `turn.completed.usage.input_tokens` 也是跨内部调用累计,真正的 ctx_end 必须从 rollout 文件的 `event_msg/token_count.info.last_token_usage.input_tokens` 读取。
- **Codex stream-json 缺 duration**: `codex exec --json` 不上报 duration,runner 自行测量。
- **Claude 流式协议会复制消息**:同一 `message.id` 可能因 thinking / tool_use / text 分块多次出现。usage 按 `message.id` 去重,content 处理所有块。

---

6. Claude Code with 24 queries/150 skills

Claude Code 部分区分 with-CLAUDE.md 与 without-CLAUDE.md,因为前者显式要求在没有 enabled skill 命中时先触发 `skill-router-skills`,后者保留默认宿主行为。这个 A/B 设计用于把 retriever 本身的选择质量与 trigger noise 分开看。

以下为 with-CLAUDE.md 条件下的 strict-match 结果。D-agentic 使用 format 修复后的 rerun;因此 D 的绝对值可作为修复后结果读取,但不纳入 paired aggregate。

6.1 with-CLAUDE.md 准确率

rank	variant	corpus	accuracy	trigger	hallu	cost	duration	turns	avg ctx_end	说明
1	L-agentic v3	150	24/24 (100%)	24/24	0/24	$4.05	608s	101	35.3K	OR-only + 强制 inline 评估 top-3(无 inspect),Claude 上唯一 24/24
2	K-bounded	150	23/24 (95.8%)	24/24	0/24	$3.27	547s	102	30.7K	grep shortlist + 候选 metadata,中小 pool 低成本 Pareto 点
2	K-lite	150	23/24 (95.8%)	24/24	0/24	$3.45	569s	101	31.1K	K 的精简 prompt 版,准确率持平
2	L-agentic v3.1	150	23/24 (95.8%)	24/24	0/24	$3.47	470s	96	32.6K	v3 + `--limit 5`,从 24→23 cell,见 §6.5
2	C-lite	150	23/24 (95.8%)	24/24	0/24	$4.30	654s	156	32.7K	有界 grep / sed 局部读 body
2	B-cc	150	23/24 (95.8%)	24/24	0/24	$5.56	896s	197	34.3K	自由 shell DCI,读 body
7	J-bounded	150	22/24 (91.7%)	23/24	1/24	$3.07	374s	96	30.7K	metadata-only bounded
7	L-agentic v1	150 (原)	22/24 (91.7%)	24/24	0/24	$3.50	596s	102	32.4K	CLI corpus search/inspect,must-AND + 条件 inspect
7	L-agentic v1	150 (rerun)	22/24 (91.7%)	24/24	0/24	$3.51	494s	101	32.4K	同语料重跑,总数稳定;miss 集合与原跑不完全重合
7	D-agentic (read-body)	150	22/24 (91.7%)	23/24	1/24	$3.65	467s	100	34.1K	paired fixed rerun;读 body
7	K-bounded	1K syn	22/24 (91.7%)	24/24	0/24	$3.74	890s	116	31.2K	1K 上 23→22 掉 1 cell,shell glob 未撞 ARG_MAX
7	E-digest	150	22/24 (91.7%)	24/24	0/24	$4.11	355s	96	38.0K	catalog 拉 description 后选择
7	H-bounded	150	22/24 (91.7%)	23/24	1/24	$4.16	542s	132	32.9K	类 B,限制裸 ls 和输出
14	L-agentic v1	1K syn	21/24 (87.5%)	24/24	0/24	$3.31	499s	100	31.1K	比 150 只掉 1 cell,ctx_end 几乎不涨
14	M-bm25	150	21/24 (87.5%)	24/24	0/24	$3.62	537s	99	32.5K	BM25 shortlist + Claude metadata rerank
14	I-meta	150	21/24 (87.5%)	23/24	1/24	$4.16	404s	93	38.0K	description-only,禁止 body
17	D-agentic (metadata-only)	150	20/24 (83.3%)	24/24	0/24	$3.68	476s	100	33.8K	c113b38 改 metadata-only 后明显回退
18	G-native	150	15/24 (62.5%)	n/a	0/24	$2.98	130s	24	36.5K	不加载 router,corpus 全启用
19	A-router	150	13/24 (54.2%)	23/24	1/24	$3.90	499s	96	35.3K	固定 lexical / 级联 scorer

解读:

- B-cc / C-lite 是 Claude Code 150-skill 上的最高准确率组。
- D / E / H / J 只低 1 cell,在 n=24 下不宜解读为稳定差异。
- A-router 是唯一低于 G-native 的 router。
- G-native 15/24 表明 Claude Code native 在该语料上的元数据选择不足以处理近义 distractor。

6.2 with-CLAUDE.md 成本与上下文

variant	accuracy	cost	duration	turns	avg ctx_end
G-native	15/24	$2.98	130s	24	36.5K
J-bounded	22/24	$3.07	374s	96	30.7K
D-agentic	22/24	$3.65	467s	100	34.1K
A-router	13/24	$3.90	499s	96	35.3K
E-digest	22/24	$4.11	355s	96	38.0K

variant	accuracy	cost	duration	turns	avg ctx_end
H-bounded	22/24	$4.16	542s	132	32.9K
I-meta	21/24	$4.16	404s	93	38.0K
C-lite	23/24	$4.30	654s	156	32.7K
B-cc	23/24	$5.56	896s	197	34.3K

Pareto 角度:

- **新 Pareto 点 K-bounded**:把高准确率档的成本下沿从 J-bounded 的 $3.07 (22/24) / C-lite 的 $4.30 (23/24) 推到 $3.27 (23/24)。K-lite 同准确率但稍贵 $0.18。
- J-bounded 仍是 22/24 档的低成本选项 ($3.07);K-bounded 多对 1 cell 但贵 $0.20,需要更高准确率时选 K。
- C-lite 是读 body 档的 23/24 / $4.30,B-cc 同准确率但贵 41%、慢 64%;C-lite 仍是 paired aggregate 内"读 body"档的合理代表。
- **1K scaling 对比 K vs L**:K-bounded 1K 22/24 (掉 1 cell),L-agentic 1K 21/24 (掉 1 cell);K 仍多对 1 cell,但 K 1K duration 890s vs L 1K 499s,K 慢 78%。K shell glob 在 1K 上没撞 `ARG_MAX` (只 1000 个 dir,远低于 ~13K 的限值,见 §8.2);更大 pool (10K+) 仍未验证。**K 适合中小 pool (≤1K) 默认,L 是大 pool 的安全选项**:K 准确率略高一格但工程更脆,L 准确率扩展边界更清楚。
- **L-agentic 150 重跑方差**:两次 150 run 都为 22/24,总数稳定;但 miss 集合不完全重合(原跑 miss `shock-analysis-demand`,rerun miss `econ-detrending-correlation`),`gh-repo-analytics` 是两跑共有 miss。这印证 §10 "每 cell 只跑一次" 的局限性:边缘 ambiguous 样本在 N=1 下有跑间方差,但总分级聚合 (n=24) 仍稳定。
- D-agentic metadata-only (20/24) 明显低于 paired 中读 body 的 D-agentic (22/24),给 Claude Code 一个 body-on-tie 信号。M-bm25 / D-agentic metadata-only 的 miss 与 K / L 高度重合,说明剩余差距是 ambiguous / overloaded 样本而非这些策略本身退化。
- G-native 成本低但准确率不足,不适合作为该语料上的唯一方案。

6.3 with-CLAUDE.md vs without-CLAUDE.md

variant	acc with	acc without	trigger with	trigger without
A-router	13/24	10/24	23/24	20/24
B-cc	23/24	19/24	24/24	19/24
C-lite	23/24	16/24	24/24	16/24
D-agentic (fixed rerun)	22/24	19/24	23/24	20/24
E-digest	22/24	17/24	24/24	19/24
H-bounded	22/24	19/24	23/24	19/24
I-meta	21/24	17/24	23/24	19/24
J-bounded	22/24	18/24	23/24	19/24
paired aggregate, excl. D-agentic	**146/168**	**116/168**	**164/168**	**131/168**

G-native 两条 arm 均为 15/24。由于 G-native 不加载 `skill-router-skills`,CLAUDE.md 对它基本是 no-op;这支持一个结论:with-arm 提升主要来自 trigger 行为改变,而非单纯跨 run 随机波动。不过每 cell 仍只有一次运行,不能据此给出统计显著性结论。D-agentic 的 row 保留为修复后参考值,但因为它来自另一次 rerun,aggregate 不再混入 D-agentic。后续策略迭代(D metadata-only / K / L / M)只跑 with-CLAUDE.md 单 arm,没有同期 paired without-arm 数据。

6.3 失败构成

with-CLAUDE.md 下没有任何变体选到随机 noise skill。错误主要落在两类:

- **针对性 distractor:** G-native 9 次、A-router 10 次、其他强 router 1-2 次。
- **残余 hallucination:** router 聚合 5/192,多发生在强 keyword query 上,如 `.xlsx`、BibTeX、GitHub analytics。

这说明 SkillRouter 的 targeted distractor 设计确实构成主要难点;随机 easy noise 不是主导错误来源。

6.5 fresh-env 16-variant 复跑与 Claude 侧诊断

2026-05-26 的 fresh-env 复跑把 Claude Code 扩到与 Codex 同一组 16 个变体。该批次中每个 cell 都使用新 HOME 和新 project,因此更适合做跨宿主归因;但它不是为了替代 §6.1-§6.4 的 trigger policy A/B 实验,因为后者的价值在 paired gate 设计。

with-CLAUDE.md 主结果如下:

variant	accuracy	trigger	turns total/avg	avg ctx_end	growth	tools	bash err	cost
K-lite	22/24	23/24	97 / 4.0	30.3K	+4.0K	50	0	$4.06
E-digest	22/24	23/24	93 / 3.9	36.8K	+10.7K	46	0	$4.80
J-bounded-v2	22/24	23/24	132 / 5.5	30.5K	+4.3K	85	8	$4.83
H-bounded	22/24	23/24	130 / 5.4	32.1K	+6.2K	83	0	$5.03
M-bm25	22/24	24/24	113 / 4.7	34.6K	+8.5K	65	0	$5.36
K-lite-replicate	21/24	23/24	97 / 4.0	30.2K	+4.1K	50	0	$4.10
D-agentic	21/24	24/24	112 / 4.7	35.4K	+9.3K	64	2	$4.85
I-meta	21/24	24/24	96 / 4.0	37.7K	+11.7K	48	0	$5.17
K-bounded	20/24	23/24	98 / 4.1	29.9K	+3.6K	51	0	$3.82
C-lite	20/24	21/24	126 / 5.3	30.8K	+5.0K	81	1	$4.77
J-bounded	20/24	23/24	153 / 6.4	34.9K	+8.8K	106	0	$6.08
B-cc	20/24	23/24	177 / 7.4	33.5K	+7.7K	130	0	$6.13
D-agentic-metadata	19/24	24/24	101 / 4.2	34.5K	+8.5K	53	1	$4.80
L-agentic	19/24	24/24	110 / 4.6	33.9K	+7.4K	62	2	$4.98
G-native	14/24	n/a	24 / 1.0	34.1K	+0.0K	0	0	$2.82
A-router	10/24	24/24	101 / 4.2	35.8K	+10.1K	53	2	$4.80

该批次的 router aggregate:

condition	accuracy	trigger	cost
with-CLAUDE.md	301/360 (83.6%)	349/360 (96.9%)	$73.59
without-CLAUDE.md	297/360 (82.5%)	339/360 (94.2%)	$73.43

解读:

- Fresh-env 复跑中,CLAUDE.md 仍提升 trigger,但提升幅度小于早期 paired gate。原因是新 router descriptions 本身已经更强地提示“必须先调用 router”,without arm 也达到 94.2% trigger。因此 fresh-env 里剩余差异不再主要是 trigger,而是候选证据比较和输出 contract。
- Claude Code 16 变体的 ceiling 暂时落在 22/24,没有出现 Codex 那样的 24/24 组。K-lite 是这批 22/24 组里成本最低的点;M-bm25 也到 22/24 且没有 bash error,说明 BM25 shortlist 对 Claude 的稳定性有帮助。
- Claude Code 在 bash-based 变体上有一个局部优势:工具调用更克制。B/C/H/I/J/J-v2/K/K-lite/K-lite-replicate 9 个 bash-based 变体合计,Claude 工具调用 684 次,Codex 为 762 次。尤其自由 shell 的 B-cc,Claude 上下文增长 +7.7K,显著低于 Codex 的 +19.6K。这说明 Claude Code 的 bash harness 在部分策略上更省操作和增量上下文,但这不是整体准确率优势:同一组 bash-based 变体 Codex 为 203/216,Claude 为 188/216。
- B-cc/C-lite 在早期 paired 实验中是最高准确率,但 fresh-env 复跑分别为 20/24。差异主要不是语料变化,而是 fresh-env runner、router prompt 和 host 版本组合发生变化;这也再次说明 n=24 下单次排名容易被 2-3 个边界样本改变。
- `L-agentic` 在 full run 的 19/24 是 prompt/实现迭代中的低点。后续 `rerun-claude-lagentic-newcli-2026-05-26` 与 `rerun-claude-l-restore-2026-05-27` 均恢复到 22/24;报告中的 Claude L 结论应以 22/24 作为当前有效口径。
- `rerun-claude-bash-rerun-2026-05-26` 的 J/K 聚合显示 0/24,但该目录的 summary 缺失 expected 字段,聚合器把所有 expected 读成 null;这是统计口径错误,不纳入结论。

L/M 诊断补跑:

run	variant	corpus	accuracy	trigger	avg ctx_end	tools	bash err	cost
`rerun-claude-l-restore-2026-05-27`	L-agentic	150	22/24	23/24	33.7K	58	0	$4.57
`rerun-claude-lagentic-newcli-2026-05-26`	L-agentic	150	22/24	24/24	33.1K	57	0	$4.67
`rerun-claude-mbm25-newcli-2026-05-26`	M-bm25	150	22/24	24/24	34.6K	65	0	$5.36
`rerun-claude-1k-lm-2026-05-26`	L-agentic	1K synthetic	22/24	24/24	31.8K	57	0	$4.54
`rerun-claude-1k-lm-2026-05-26`	M-bm25	1K synthetic	21/24	24/24	31.8K	54	0	$4.32

这组补跑说明:L/M 的 Claude 侧主要问题不是 CLI 不能工作,而是同一候选证据下的最终选择更容易被 task-centric distractor 或 broad spreadsheet/economics skill 吸走。

---

7. Codex with 24 queries/150 skills

7.1 结果

Codex 没有 paired with/without arm,因此本节只报告 16x24 宿主实验结果。2026-05-26 重新执行了全量 Codex 实验,统一使用新 CLI 语义:

- 模型: `gpt-5.5`, `reasoning_effort=high`。
- 每个 `(variant, query)` 使用全新的 Codex home/project 环境端到端执行。
- disabled skill 后缀统一为 `.agentic-skill-router-disabled`。
- A/D/L/M 等 CLI 型策略仍保留各自语义:A-router 调 `skills route`;D 系调 DCI search/inspect;L/M 调 corpus search/inspect。

本表覆盖旧 Codex 9x24 与零散后续补跑结果。`cost est.` 是估算值,口径见 §5。

variant	accuracy	trigger	turns total/avg	avg ctx_end	growth	tools	bash err	cost est.
G-native	23/24	n/a	24 / 1.0	19.3K	+0.0K	0	0	$1.64
A-router	11/24	24/24	90 / 3.8	23.9K	+9.1K	48	0	$4.08
B-cc	23/24	24/24	112 / 4.7	34.5K	+19.6K	87	3	$6.00
C-lite	24/24	24/24	114 / 4.8	19.7K	+4.8K	138	0	$3.71
D-agentic	22/24	24/24	103 / 4.3	19.7K	+4.8K	73	0	$3.25
D-agentic-metadata	23/24	24/24	95 / 4.0	19.3K	+4.5K	69	0	$3.01
E-digest	19/24	24/24	84 / 3.5	20.0K	+5.1K	48	0	$2.84
H-bounded	21/24	24/24	148 / 6.2	20.6K	+5.8K	153	1	$4.68
I-meta	23/24	24/24	95 / 4.0	21.1K	+6.3K	51	0	$3.15
J-bounded	22/24	24/24	99 / 4.1	19.8K	+5.0K	77	0	$3.45
J-bounded-v2	21/24	24/24	102 / 4.3	18.1K	+3.3K	78	1	$3.48
K-bounded	23/24	23/24	125 / 5.2	18.1K	+3.2K	71	1	$3.56
K-lite	23/24	24/24	103 / 4.3	17.4K	+2.6K	54	0	$3.30
K-lite-replicate	23/24	24/24	101 / 4.2	17.4K	+2.5K	53	0	$2.96
L-agentic	24/24	24/24	92 / 3.8	17.5K	+2.7K	60	0	$2.80
L-agentic on 1K synthetic	23/24	24/24	88 / 3.7	18.0K	+3.2K	60	0	$3.13
M-bm25	19/24	24/24	102 / 4.3	18.0K	+3.1K	69	0	$3.05

Codex 与 Claude Code 的主要差异:

- Codex native G-native 在 150-skill 基准上为 23/24;Claude Code fresh-env native 为 14/24(早期 paired 为 15/24)。

- Codex router 基本全部触发,K-bounded 的 23/24 trigger 是 detector 漏识别,该 cell 仍产出合法匹配。
- C-lite 与 L-agentic 达到 24/24;L-agentic 成本最低的一档,且上下文增长只有 +2.7K。
- `L-agentic on 1K synthetic` 不是额外策略,而是同一 L-agentic 在 150-skill 对比语料 + 850 synthetic noise skills 上的扩展性检查;它只掉 1 cell,上下文增长仍在 +3.2K。
- A-router 在新 CLI 语义下没有 bash error,但仅 11/24,说明固定 scorer 本身仍不可靠。

7.2 错误分布

除 A-router 外,Codex 的 miss 仍集中在少数边界样本:

variant	miss
G-native	`gh-repo-analytics` → `skill-146`
B-cc	`gh-repo-analytics` → `skill-046`
C-lite	none
D-agentic	`offer-letter-generator` → `skill-072`, `shock-analysis-supply` → `skill-026`
D-agentic-metadata	`shock-analysis-demand` → `skill-026`
E-digest	`gh-repo-analytics` → `skill-046`, `protein-expression-analysis` → `skill-026`, `reserves-at-risk-calc` → `skill-026`, `shock-analysis-supply` → `skill-080`, `weighted-gdp-calc` → `skill-026`
H-bounded	`enterprise-information-search` → `skill-087`, `gh-repo-analytics` → `skill-046`, `protein-expression-analysis` → `skill-026`
I-meta	`gh-repo-analytics` → `skill-046`
J-bounded	`enterprise-information-search` → `skill-013`, `gh-repo-analytics` → `skill-046`
J-bounded-v2	`gh-repo-analytics` → `skill-046`, `shock-analysis-demand` → `skill-026`, `shock-analysis-supply` → `skill-080`
K-bounded	`gh-repo-analytics` → `skill-046`
K-lite	`gh-repo-analytics` → `skill-046`
K-lite-replicate	`gh-repo-analytics` → `skill-046`
L-agentic	none
M-bm25	`dialogue-parser` → `skill-130`, `gh-repo-analytics` → `skill-046`, `pptx-reference-formatting` → `skill-103`, `reserves-at-risk-calc` → `skill-026`, `shock-analysis-supply` → `skill-026`

A-router 的错误分布不在表中逐项展开:13 个 miss 中多数误选 `skill-140`,另有 `gh-repo-analytics` → `skill-077` 和 `shock-analysis-supply` → `skill-043`。这更像固定 route scorer 的长查询偏置,而非 runner 污染:本轮 A-router 的 bash error 为 0。

7.3 Native 差异的证据

对 `weighted-gdp-calc` 抓取真实 native 请求后,观察到:

host	endpoint	skill lines	skill section chars	avg desc chars	执行机制
Codex	`/v1/responses`	150	21,117	92	instructions 中内联 available skills
Claude Code	`/v1/messages?beta=true`	150	7,948	20	`Skill` tool + budgeted skill listing

这不是完整因果证明,因为抓包只覆盖一个代表 query 和当前 CLI / 模型版本。但它与 24-query 结果一致:同语料同 query 下,宿主呈现给模型的 skill 元数据不同,足以改变 native 选择质量。

7.4 Codex 后续策略迭代

旧版 A-J 之后曾继续补跑 D metadata-only、K、K-lite、L、M。新 CLI 全量重跑已经把这些变体统一纳入 §7.1,因此 150-skill 排名以 §7.1 为准。`L-agentic on 1K synthetic` 也已作为扩展性检查行合并进 §7.1 主表,避免把它误读成一个独立策略。

解读:

- 150-skill 上,Codex 的强策略已经接近 ceiling,但不是所有 metadata-only / metadata-first 变体都能全对:E-digest 与 M-bm25 均为 19/24,H/J-v2 为 21/24。
- L-agentic 在 150-skill 新 CLI 重跑中 24/24,在 1K synthetic 上只掉 1 cell,且上下文几乎不涨,说明 CLI 抽象比手写 shell 更适合承载索引和分页。

- K-lite 与 K-lite-replicate 均为 23/24,只错 `gh-repo-analytics`;这支持原先对该样本 ambiguous 的判断。
- M-bm25 在新 CLI 语义下为 19/24,不再支持"150 上低成本全对"的旧结论;BM25 shortlist 对 `skill-105` 相关 spreadsheet 任务和 `pptx-reference-formatting` 边界样本更敏感。

7.5 L 方案实现要点

L-agentic 将检索能力下沉到 CLI:

- `corpus search`: 返回稳定 `ref`、`shortId`、score、description 摘要和分页元信息。
- `corpus inspect`: 对少量候选返回完整 metadata,避免模型自己拼 shell 管道和解析路径。
- 模型职责从"写 grep/awk/sed 检索器"变成"生成 query terms、比较候选证据、做最终选择"。

这和纯 shell 的最大差异是扩展边界更清楚:大语料索引、缓存、分页、top-k、JSON schema、fingerprint 都由 CLI 负责,模型只消费有界结构化候选。1K 结果支持这个方向,但 79K original Hard 仍需要继续验证。

7.6 Claude Code vs Codex fresh-env 差异归因

把 §6.5 的 Claude fresh-env 结果与 §7.1 的 Codex 新 CLI 主结果并排后,差异可以拆成六层:

归因层	证据	结论
数据与 CLI 语义	两边使用同一 24 query / 150-skill 语料;L/M 都走 `corpus search/inspect`;L-agentic 的 Claude/Codex SKILL.md 主体一致,仅 host metadata 和路径不同。	差异不能主要归因于 query、gt mapping 或 L/M CLI 语义不一致。
Native skill listing	Codex native 23/24,Claude Code fresh-env native 14/24;抓包显示 Codex native skill section 21,117 chars、avg desc 92 chars,Claude Code 7,948 chars、avg desc 20 chars。	Native 差异主要来自宿主呈现给模型的 skill metadata 信息量和执行机制不同。
Trigger	Claude with-CLAUDE.md router aggregate 349/360 (96.9%),without 339/360 (94.2%);Codex router aggregate 359/360 (99.7%)。	早期 Claude 问题首先是 trigger,但 fresh-env 复跑中 trigger 已接近饱和,剩余 5.6pp aggregate gap 不是靠 CLAUDE.md alone 能解释。
模型 rerank / 候选比较	Codex L-agentic 24/24,Claude L restore 22/24;Codex C-lite 24/24,Claude C-lite 20/24;但 M-bm25 反而 Claude 22/24、Codex 19/24。	Codex 更擅长在少量 metadata 候选中做最终判别;Claude 对 BM25 shortlist 这类强排序先验更受益。
Bash harness 操作效率	bash-based 9 变体合计:Claude 188/216、684 tool calls、avg ctx growth +6.1K;Codex 203/216、762 tool calls、avg ctx growth +5.9K。B-cc 单项:Claude +7.7K growth,Codex +19.6K。	Claude 的优势主要体现在更少工具调用和部分自由 bash 场景的增量上下文控制;Codex 的优势仍是更高 accuracy、更低绝对 ctxEnd 和更低成本。
输出 contract	Claude miss 中仍出现 alias 或 router 自身,如 `citation-check`、`d3`、`xlsx`、`agentic-skill-router:agentic-skill-router-skills`;Codex 新 CLI 主跑中除旧坏跑外基本只输出合法 `skill-NNN`。	Claude 需要更强的 JSON schema/regex guard 和自动 retry;否则部分错误不是检索失败,而是最终答案 contract 漏出。
上下文与成本基线	L-agentic:Claude 22/24,ctxStart 27.1K,ctxEnd 33.7K,cost $4.57;Codex 24/24,ctxStart 14.8K,ctxEnd 17.5K,cost $2.80。Native:Claude ctx 34.1K vs Codex 19.3K。	同一 strategy 在 Claude Code 下会带来约 1.8-1.9x 上下文窗口和约 1.6x 成本,因此不能直接把 Codex 的成本/ctx 结论搬到 Claude。

典型 case:

- `gh-repo-analytics`:Claude L/M 和多种 metadata-only 变体倾向 `skill-046`,这是任务中心描述;Codex L 能选回 gt `skill-021`。该样本本身有 ambiguous 属性,但它清楚暴露了 Claude 更容易被 task-centric distractor 吸走。
- `shock-analysis-demand/supply`:gt 是通用 spreadsheet skill `skill-105`;Claude L/M 常被 economics/spreadsheet distractor (`skill-026`、`skill-080`、`skill-109`) 吸走,而 Codex L 在同类候选里能稳定选 `skill-105`。这说明 body-on-tie 或更强的"工具 substrate vs domain method"规则对 Claude 更重要。
- Native `weighted-gdp-calc`:Claude native 选 `skill-026`,Codex native 命中 `skill-105`;这与 §7.3 的 native prompt capture 一致。

因此当前最稳妥的归因是:

1. **Native gap 是 host prompt/listing 机制导致的。**
2. **Router gap 是 model rerank + output contract 导致的,不是 corpus 或 CLI 语义差异。**
3. **Claude Code 上的产品化默认策略需要更强 guardrail:**候选生成可继续用 K-lite/L/M,但最终输出必须做合法 id 校验;对 spreadsheet、GitHub analytics、PPTX 等高歧义样本要增加 body-on-tie 或二次 rerank。

---

8. 规模扩展: J-bounded

8.1 为什么扩展

150-skill 对比实验能比较变体,但不能代表真实大规模安装场景。J-bounded 在 Claude Code router 变体中、且达到 22/24 的组里成本最低,因此需要验证它在更大 skill pool 上是否仍可靠。

8.2 v1 失效机制

J-bounded v1 的模板存在两个规模问题:

1. **shell glob 撞 `ARG_MAX`:** `~/.claude/skills/*/SKILL.md.skill-router-disabled` 在约 13K paths 后超过 Linux argv 限制。
2. **`head -20` 截断正确答案:** 在 79K Hard pool 上,关键词可能匹配几十到上百个候选;gt 若排在第 20 位之后会被系统性切掉。

8.3 v2 修复

J-bounded-v2 只改 body workflow,frontmatter description 保持不变:

项	v1	v2
枚举	shell glob	`find -print0
候选上限	`head -20`	不截断;太多时收窄关键词重跑
关键词指导	distinctive keywords	narrow technical terms,避免 broad words
shortlist payload	description 文本	path-only 后再查 description

8.4 结果

配置	host	accuracy	trigger	cost	avg cost/cell	duration	avg ctx_end
J-v1 x 150	Claude Code	22/24	23/24	$3.07	$0.128	374s	30.7K
J-v2 x 150	Claude Code	22/24	23/24	$3.95	$0.164	665s	30.8K
J-v2 x 79K Hard	Claude Code	12/24	23/24	$5.33	$0.222	1577s	36.6K
J-v2 x paper-core single Hard	Codex	11/24 strict; 14/24 alias-normalized	24/24	$12.44	$0.518	3056s	30.3K

解释:

- v2 在 150 上没有损失准确率,但更慢、更贵。
- v2 在 79K 上仍能保持 bounded payload,成本只上升约 35%,ctx_end 上升约 19%。
- 准确率从 22/24 降到 12/24 或 11/24,主要是 description-only 在大量近义 / catch-all skill 面前不足。
- Codex paper-core single run 中,J-v2 有 3 条输出了原始 skill name alias 而非 `sr-*` opaque id;strict 计 11/24,若把这些 alias 映射回 gt id 则为 14/24。这暴露了 metadata 文件物化时 `name:` 可见带来的输出格式风险。

这修正了 150-skill 结论的外推边界:J-bounded 是 Claude Code 150-skill router 变体中的低成本 Pareto 点,但不能直接等价为 80K 规模的高准确率方案。生产方向应是 metadata-first + body-on-tie,而不是永久 metadata-only。

8.5 论文 metadata-only 对照

SkillRouter 论文公开的 `nd` 指标使用 name + description only,但分母是 75 core queries,不是 single-only:

source	method	input	query / corpus	Hard Hit@1	Avg Hit@1
Paper Table 9	BM25	nd	75 core, Easy/Hard	0.0%	0.0%
Paper Table 9	Qwen3-Emb-0.6B	nd	75 core, Easy/Hard	14.7%	18.7%

source	method	input	query / corpus	Hard Hit@1	Avg Hit@1
Paper Table 9	Qwen3-Emb-8B	nd	75 core, Easy/Hard	20.0%	25.3%
Paper Table 21	Qwen3-Emb-8B x Qwen3-Rank-8B	nd	75 core, Easy/Hard	18.7%	24.0%
Paper Table 21	Qwen3-Emb-0.6B x GPT-5.4-mini	nd	75 core, Easy/Hard	29.3%	33.3%

论文 `Single Hit@1` 是另一个口径,属于 full skill text 主结果:

source	method	input	Single Hit@1
Paper Table 4 / 22	Qwen3-Emb-0.6B x Qwen3-Rank-0.6B	full	62.5%
Paper Table 4 / 22	Qwen3-Emb-8B x Qwen3-Rank-8B	full	66.7%
Paper Table 4 / 22	SR-Emb-0.6B x SR-Rank-0.6B	full	72.9%

我们的补充复跑填的是论文没有公开的 `paper-core single x Hard x metadata-only agentic router` 格子:

method	input	query / corpus	accuracy	trigger	duration	cost est.
M-bm25	metadata-only	24 paper-core single, 79,141 Hard	14/24 (58.3%)	24/24	964s	$6.822
J-bounded-v2	metadata-only	24 paper-core single, 79,141 Hard	11/24 strict; 14/24 alias-normalized	24/24	3056s	$12.436
M-bm25	metadata-only	当前 24-query subset, 79,141 Hard	11/24 (45.8%)	24/24	1016s	$6.347

这些结果说明:agentic metadata-only router 可以显著高于论文 BM25-nd top-1 的 0%,但仍明显低于论文 full-text single routing 的 62.5%-72.9% 区间。真正需要补齐的是 body-on-tie 或 full-text rerank,而不是继续把 metadata-only 做成唯一决策源。

---

9. SkillRouter 75 core × Easy 78K × multi-skill 扩展

9.1 为什么扩展

§8 的 paper-core single Hard 复跑解决了 single-skill × Hard × metadata-only 这一格,但留下两个未对齐项:

- **查询规模偏小,只覆盖 24 个 single-skill 任务**,论文 default scored eval 是 75 core queries (24 single + 51 multi),§12 #2 明确把"扩展到 75 core 并实现 multi-label Hit@1"列为后续实验。
- **只跑一个变体一个宿主一个 tier**(M-bm25/J-v2 on Codex on Hard),没有 K/J/M × Claude/Codex × Easy 的横向对比。

本节填补这两项,产物在 `experiments/skillrouter-easy/`。

9.2 实验设置

项	选择
语料	SkillRouter `eval_core/easy/*.jsonl.gz` 全量 78,361 skills
安装形态	metadata-only:每个 skill 写为 `<HOME>/.<host>/skills/sr-XXXXX/SKILL.md.agentic-skill-router-disabled`,只含 `name` + `description` frontmatter,body 完全剥离
匿名化	目录名取确定性洗牌后的 `sr-XXXXX` (seed=20260525);frontmatter `name:` 和 `description:` 保留原始上游字符串(论文 nd 输入需要 name 信号)
查询集	`relevance.json` 中 `core_gt_ids` 非空的 75 条 core queries (24 single + 51 multi,论文 Section 2 / Appendix A)
指标	Hit@1:任一 `core_gt_ids[*]` 出现在 agent 返回 top-1 即记 1,符合论文 multi-skill Hit@1 定义
输出契约	agent 单行 JSON `{"matched_skill_names":["sr-AAAAA",...,"sr-JJJJJ"]}`,有序 top-10,position 0 计 Hit@1
变体	K-bounded / J-bounded-v2 / M-bm25,fork 自 §3 同名变体,workflow 由"挑 1 个"改为"排 top-10",最大 4 个工具调用
宿主	Claude Code (claude-opus-4-7) 与 Codex (gpt-5.5,reasoning=high),各 3 变体共 6 cell
总规模	6 cell × 75 query = 450 次 agent 调用

为了让 K/J 类的 keyword grep 在 78K 文件上仍能 sub-second 完成,安装阶段额外生成 `<skillsRoot>/.flat-metadata.tsv` (78,361 行,17MB): `<sr-id>\t<name>\t<description>`,变体 SKILL.md 直接 grep 该 TSV,替代 78K 文件的 `find … | xargs grep` (后者 24s/次,4 次迭代 = 必超时)。M-bm25 在 setup 阶段预热一次 BM25 index cache。这两项工程优化不改变路由信号(name + description),只把 metadata-only 路由器在 80K 规模下的延迟从分钟级压回秒级。

9.3 主结果

variant	host	Hit@1	single (n=24)	multi (n=51)	答题率	超时	wall
J-bounded-v2	codex	**40.0% (30/75)**	9/24 (37.5%)	21/51 (41.2%)	75/75	0	2532s
M-bm25	codex	37.3% (28/75)	7/24 (29.2%)	21/51 (41.2%)	75/75	0	3444s
M-bm25	claude	34.7% (26/75)	7/24 (29.2%)	19/51 (37.3%)	75/75	0	1329s
K-bounded	codex	32.0% (24/75)	5/24 (20.8%)	19/51 (37.3%)	74/75	1	2966s
K-bounded	claude	25.3% (19/75)	4/24 (16.7%)	15/51 (29.4%)	75/75	0	1037s
J-bounded-v2	claude	21.3% (16/75)	5/24 (20.8%)	11/51 (21.6%)	75/75	0	1009s

按变体平均 (across hosts)

variant	avg Hit@1	cells
M-bm25	36.0%	2
J-bounded-v2	30.7%	2
K-bounded	28.7%	2

按宿主平均 (across variants)

host	avg Hit@1	avg wall	cells
Codex	36.4%	2981s	3
Claude Code	27.1%	1125s	3

Codex 系统高出 Claude Code 9.3pp,但 wall 时间 2.6× 长。Claude Code 三个 cell 均在 17-22 min 完成,Codex 需要 42-57 min。更慢的原因可能是 Codex reasoning 更深(GPT-5.5 high effort)或工具执行开销更大;更准可能是模型推理能力本身的差异,而非宿主 prompt 差异(§9 所有 cell 使用同一 SKILL.md 模板,与 §7.3 中 native prompt 差异的解释链不同,详见 §12 #7)。

总耗时 ~57 min,449/450 答出,1 个超时,0 错误。

9.4 与论文 Easy 基线对照

source	method	input	Easy Hit@1
Paper Table 9	BM25	nd	0.0%
Paper Table 9	Qwen3-Emb-0.6B	nd	22.7%
Paper Table 9	Qwen3-Emb-8B	nd	30.7%
Paper Table 9	BM25	**full** (含 body)	34.7%
Paper Table 9	Qwen3-Emb-0.6B	full	58.7%
Paper Table 9	Qwen3-Emb-8B	full	65.3%
Paper Table 2	SR-Emb-0.6B	full	66.7%
Paper Table 2/3	SR-Emb-0.6B × SR-Rank-0.6B	full (1.2B pipeline)	76.0% (A-Hit@1, Easy+Hard 均值;Easy-only 未单独公开)
本节	**codex / J-bounded-v2**	**nd (agentic loop)**	**40.0%**
本节	**codex / M-bm25**	nd (agentic loop)	37.3%
本节	**claude / M-bm25**	nd (agentic loop)	34.7%

直接可比的是前三行的 nd 基线。**6 个 cell 全部跑赢 BM25 nd (+21pp 起)**;4/6 cell 超过 Qwen3-Emb-0.6B nd;3/6 cell 超过最强 nd 基线 Qwen3-Emb-8B (30.7%),最佳 codex/J-v2 超出 +9.3pp。同时,最佳 cell (codex/J-v2 40.0%) 在**结构性无 body 输入**的限制下,也高于论文用 BM25 但**带 body** 的 34.7%。值得注意的是 claude/M-bm25 (34.7%) 恰好与论文 BM25-full 持平——metadata-only agentic loop 做到了 BM25 带 full body 的同等水平。这说明 agent loop 对 metadata-only 路由有可量化的提升,主要靠 LLM 的 query rewrite 和多次有界检索补偿了 BM25/embedding 在稀疏 description 上的不足(论文 Easy pool 中 18.7% 的 description 短于 10 词,本节最佳 cell 仍能稳定挑出 gt)。

但同样需要注意:最佳 cell (40.0%) 与论文 full-body 强基线(Qwen3-Emb-8B full 65.3%,SR-pipeline 74-76%)仍有 25-36pp 差距,这部分差距结构上无法通过 metadata-only 路由器补回,与 §8.5 结论一致。

9.5 与 §8 paper-core single Hard 复跑的关系

§8.5 的 paper-core single Hard 复跑给出 J-v2 strict 11/24 (alias-normalized 14/24)、M-bm25 14/24。本节是同变体家族在 Easy + 75 core 上的横向扩展,得到三个新信息:

1. **multi-skill 子集表现略高于 single-skill 子集**(M-bm25/codex single 7/24 = 29.2% vs multi 21/51 = 41.2%)。这是 metric 定义带来的:multi-skill query 有多个 gt,任一进 top-1 都计 hit,分子分母比 single 更松。论文 Table 4 也观察到类似 R@10 远高于 single Hit@1 的现象。
2. **宿主差异在 75 query 上比 24 query 上更稳定。** §8 只能在 Codex 上跑(成本/时间限制),本节首次给出两个宿主同语料同 variant 的并排对比,Codex 平均高出 Claude Code 9.3pp。但 §7.3 的 native prompt capture 解释(skill listing 信息密度)不能直接搬来:§9 用的是 router SKILL.md 而非 native listing,差异更可能来自模型本身(GPT-5.5 vs Opus 4.7 的 metadata rerank 能力),需要进一步抓包确认(见 §12 #7)。
3. **K-bounded 在 78K 下的"shell glob 撞 ARG_MAX"问题被工程层面解决。** §8.2 列出 v1 在 13K 即崩,本节通过装语料时同时生成 `.flat-metadata.tsv`(K/J 直接 grep 该单文件),把"枚举 78K 文件"从命令层抽走,K-bounded 在 78K 上可以正常完成 4 步 workflow,Codex 取 24/75 = 32.0%,Claude Code 取 19/75 = 25.3%。这不是 SKILL.md prompt 的胜利,只是说明 K-bounded 路由器在"语料形态可配合时"仍可用。

9.6 工程优化记录

跑 6 cell × 75 query 过程中修了 5 个问题,均与 prompt / agent 行为无关,与"把 metadata-only 路由器跑到 80K 规模"直接相关。摘要(详细见 `experiments/skillrouter-easy/IMPLEMENTATION_PLAN.md`):

1. **auth 缺失**:tmp HOME 没有 `.credentials.json` / `auth.json`,首次启动报"Not logged in"。修复:setup 阶段把真实 HOME 的对应 auth 文件复制到 tmp HOME。
2. **K-bounded shell glob 撞 ARG_MAX**:见 §8.2,这里通过 §9.2 的 flat-TSV 索引绕开。
3. **78K `find | xargs grep` 单次 24s,agent 迭代 4 次必超时**:同上,grep 17MB TSV 文件后 <100ms,提速 ~250×。
4. **M-bm25 首查冷启动 17s**:setup 阶段预热 BM25 index cache,并将 `AGENTIC_SKILL_ROUTER_CORPUS_CACHE_TTL_MS` 设为 24h。
5. **Codex shell session 在长 agent loop 中 stdin 关闭、卡在 retry**:配合 #3 把工具调用数压到 4 个以内后基本消失,全 75 query 只剩 1 个超时(`codex/K-bounded` 的 `xlsx-recover-data`,520s 后被 600s 上限 kill)。

9.7 错误分布分析

75 个 query 按 6 cell 命中情况分为三层:

命中类型	query 数	占比	说明
全 6 cell 命中	12	16.0%	稳定 easy 样本
部分 cell 命中 (mixed)	24	32.0%	变体/宿主分化区
全 6 cell 皆 miss	39	52.0%	metadata-only 天花板

全 miss 的 39 个 query 是 metadata-only 路由器的结构性天花板:无论变体和宿主如何组合都无法在 top-1 命中 gt。按 tier 分,14/39 是 single-skill (58.3% 的 single query miss),25/39 是 multi-skill (49.0% 的 multi query miss)。其中 7 个 query (18%) 出现**全 6 cell 一致误选同一个 wrong skill** 的强共识现象,例如 `earthquake-plate-calculation` 全部选择 `sr-26212`、`quantum-numerical-simulation` 全部选择 `sr-68239`。这 7 个共识 miss 说明 metadata description 中存在比 gt skill 更高相关度的 distractor,可能是标注争议或 gt description 不够具体。

24 个 mixed query 中,15 个至少被两个宿主各命中 1 次,6 个仅被 Codex 命中,3 个仅被 Claude Code 命中。Codex-only 命中数 (6) 是 Claude-only (3) 的 2 倍,进一步支持 Codex 在 metadata rerank 上的系统性优势。从 mixed 分布看:5 个 query 被 5/6 cell 命中(差 1 cell 的边界样本),6 个 query 仅被 1/6 cell 命中(幸运一跳)。

这些分布意味着:想要 Easy 78K metadata-only 的 Hit@1 从 40% 推到 50%+,需要攻克 39 个全-miss query 中的至少 8 个,单纯调优变体参数只能在 24 个 mixed query 上再捞 1-3 个。body-on-tie 或 full-text rerank 是突破 52% 天花板的必要方向。

每个 (query, cell) 的完整工具调用链见 `experiments/skillrouter-easy/runs/traces-report.html`(可折叠浏览)和 `traces-compact.json`(结构化数据)。

---

10. 关键失败案例

10.1 `gh-repo-analytics`

8 个 Claude router 变体在两条 arm 下均未命中。gt `skill-021` 的 description 是工具中心:gh CLI 用于操作 repo / issue / PR。多个变体选择的 `skill-046` 是任务中心:追踪和可视化 GitHub 贡献、PR、issue resolved over time。query 要求写 December community pulse,统计 PR、issue、top contributor。按 description 语义, `skill-046` 更贴近 query。该样本应标注为 ambiguous,不宜用来单独否定 router 设计。

主表采用 strict gt,且默认包含该 ambiguous 样本。敏感性上,如果仅从 Claude Code router 表中剔除这一条,所有 router 的分母都会变为 23;B-cc / C-lite 将变为 23/23,D / E / H / J 变为 22/23,I-meta 变为 21/23,A-router 变为 13/23。因此 1-cell 排名差异应按"含争议样本的 strict score"解读,不应过度放大。

10.2 `shock-analysis-supply`

gt 为通用 Excel skill `skill-105`。with-CLAUDE.md 强制路由后,部分 metadata-only 变体被更专精的 economics / timeseries skill 吸引;without-arm 中一些零工具直接输出 `skill-105` 反而碰巧命中。读 body 的 B-cc / C-lite 能恢复正确选择。这是 body-on-tie 的典型适用场景。

10.3 A-router

A-router 直接把长 query 交给固定 scorer。真实 query 中大量步骤说明、路径和格式约束会稀释高信号关键词。相比之下,J / B / C / H 都让 LLM 先做 query rewrite 或 keyword extraction。A-router 后续应改为返回候选证据并交给 LLM rerank,而不是由 CLI 直接 commit。

---

11. 局限性

1. **每 cell 只跑一次。** 24 query 下 1 cell 即 4.2pp,23/24 与 22/24 不能视为统计显著差异。
2. **CLAUDE.md 不是 system prompt。** 它是 user-message context block,效果强但仍不能等同于强制 `tool_choice`。
3. **150-skill 对比实验规模有限。** scaling 实验显示 metadata-only 准确率会随规模明显下降。
4. **只测路由,不测执行。** 下游 skill body 是否能完成任务未纳入本报告指标。
5. **Codex cost 为估算。** Codex 表中的 `cost est.` 不是同一计费口径下的真实账单值,适合比较量级,不适合做精确财务结论。
6. **数据存在 ambiguous / overloaded 样本。** `gh-repo-analytics` 和 Excel 相关 query 会影响总体排名。
7. **宿主版本会漂移。** Claude Code / Codex 的 native skill 展示策略可能随版本变化。
8. **语言覆盖不足。** 24 个 query 均为英文任务描述,未测中文或混合语言请求。
9. **论文对照口径不完全一致。** 论文未公开 `Single x nd x Hard-only` 指标;本报告的 paper-core single Hard 复跑(§8.5)和 75 core x Easy x multi-skill 复跑(§9)都是补充实验,不是论文 BM25 / embedding pipeline 的复现。最佳 cell 与论文 nd 基线可严格对照,与论文 full-body 强基线只能作为上限参考。
10. **§9 multi-skill 指标用论文 any-gt Hit@1 定义,不是 strict set match。** multi-skill query 有 2-7 个 gt,任一进 top-1 即记 hit,因此分数会系统性高于 single-skill。若改用 strict-set 或 nDCG@K 评分,排序可能改变。R@10 / FC@10 / nDCG@10 等指标的原始 top-10 输出都保存在 `runs/<host>-<variant>/<query>.jsonl`,可离线重算,本报告未一并跑。
11. **§9 只跑 Easy tier,未跑 Hard tier。** Hard 在 Easy 78,361 基础上加 780 个 LLM 蒸馏 distractor;论文 Table 9 显示不同方法 Hard 比 Easy 降幅差异较大(BM25 无降幅,Emb-8B 降 10.7pp),因此不能对 agentic router 做简单线性外推。本节最佳 40.0% 是 Easy 数字,Hard 表现需要实跑才能确定。Hard 跑是 §12 的 follow-up。

---

12. 建议

短期工程建议:

- **Claude Code 默认:** fresh-env 复跑下 K-lite 是 22/24 组里成本最低的点,L-agentic/M-bm25 在 restore/newcli 诊断中也稳定到 22/24。实际产品路径应优先做合法 id guard + retry,再在 K-lite/L-agentic/M-bm25 之间按依赖和成本选择;失败或低置信时升级到 body-on-tie。

- **Codex 默认:** 新 CLI 重跑下 L-agentic 是当前首选:150-skill 为 24/24,上下文增长低,且 1K synthetic 只掉 1 cell。C-lite 也为 24/24,但工具调用更多;M-bm25 新跑降至 19/24,不再作为默认路径。
- **所有宿主:** 对 `matched_skill_name` 做 regex 校验和存在性校验;若不是合法已安装 disabled skill,自动 retry 或走 rescue fallback。
- **A-router:** 不再作为默认路径;改造为 candidate generator + LLM reranker。
- **大规模方向:** metadata-first 可保留为快路径,但 79K Hard 结果显示必须增加 body-on-tie / full-text rerank,否则近义自然 pool skill 会稳定吸走流量。
- **跨宿主解释:** 不要用 Codex 24/24 的 L-agentic 结果直接推断 Claude Code 也会 24/24;Claude 的 native listing、上下文基线、最终 id contract 都不同,需要单独 gate。

后续实验建议:

1. 对边界 query 做 N=3 或 N=5 重复,给出置信区间。
2. ~~扩展到 SkillRouter 75 core queries,实现 multi-label Hit@1。~~ ✅ 已完成,见 §9(K/J-v2/M × Claude/Codex × Easy 78K × 75 core,最佳 codex/J-v2 40.0%)。下一步是把同一矩阵跑到 Hard 79,141 tier,与论文 Avg = (Easy+Hard)/2 直接可比。
3. 实现 metadata-first + body-on-tie 变体,与 J-v2、L-agentic、M-bm25 对比。预期能补 §9 与论文 full-body 强基线之间 25-36pp 的差距。
4. 增加端到端执行验证,至少覆盖 Excel、PDF、PPTX、GitHub analytics 四类。
5. 将 `gh-repo-analytics` 标注为 ambiguous 或重标 gt,避免把数据争议解释为 retriever 失败。
6. 从 §9 已保存的 `runs/<host>-<variant>/<query>.jsonl` top-10 输出里离线算 R@10 / FC@10 / nDCG@10 / MRR@10,与论文 Table 4 multi-skill 指标对照。原始数据已就位,无需新跑 agent。
7. 调查 §9 中 Codex 系统高出 Claude Code ~9pp 的原因。§7.3 给出"native skill listing 信息密度"的解释,但 §9 用的是同一 SKILL.md 模板,所以差异来源更可能是模型差异(GPT-5.5 vs Opus 4.7)而非 host listing,需要 §7.3 那样的抓包确认。

---

13. 总结

本报告从 150-skill 对比实验出发,经 1K synthetic → 79K Hard → 78K Easy 75-core 四轮规模递增,系统评估了 metadata-only agentic router 的能力边界。核心 takeaway 四条:

1. **Agentic metadata-only 路由器在 78K Easy pool 上能稳定跑赢传统 BM25/embedding nd 基线。** 最佳 cell (codex/J-v2) 40.0% Hit@1 超过论文最强 nd 基线 Qwen3-Emb-8B (30.7%) 达 +9.3pp,并高于 BM25 带 full body 的 34.7%。这证明 LLM-driven query rewrite + 多步有界检索对稀疏 metadata 有实质性信息提升。
2. **52% 的 query 构成 metadata-only 结构性天花板。** 全 6 cell 皆 miss 的 39/75 query 中,7 个出现跨变体跨宿主一致误选,说明 description 信号本身不足以区分 gt 和高质量 distractor。突破 40% 需要 body-on-tie 或 full-text rerank,这也与 §8 Hard 池结论一致。
3. **宿主/模型选择比变体选择影响更大。** 同一 SKILL.md 模板下,Codex (GPT-5.5) 系统高出 Claude Code (Opus 4.7) 9.3pp (36.4% vs 27.1%);而三个变体间最大差距仅 7.3pp (M-bm25 36.0% vs K-bounded 28.7%)。生产部署应先选对模型,再调 prompt。
4. **Claude Code 与 Codex 的 150-skill 差异不是单一策略问题。** Fresh-env 复跑显示 Codex router aggregate 321/360,Claude Code with-CLAUDE.md 301/360;native 差距更大(23/24 vs 14/24)。归因上,native gap 主要来自 host skill listing 信息密度,router gap 主要来自模型 rerank、输出 id contract 和上下文/成本基线差异。Claude Code 需要额外的合法 id guard、retry 和 body-on-tie 才能接近 Codex 的稳定性。

附录 A. 文件索引

内容	路径
All-experiments visualization (HTML)	`experiments/dci-compare/runs/report-all-experiments.html`
All-experiments renderer	`experiments/dci-compare/render-all-experiments.mjs`
Claude paired driver	`experiments/dci-compare/routing-only-paired.mjs`
Claude paired raw summary	`experiments/dci-compare/runs/routing-only-9x24-claudemd/summary.json`
Claude paired HTML report	`experiments/dci-compare/runs/routing-only-9x24-claudemd/report.html`
Claude/Codex fresh-env 16-variant raw run	`experiments/dci-compare/runs/rerun-150-full-2026-05-26/`
Claude L-agentic restore rerun	`experiments/dci-compare/runs/rerun-claude-l-restore-2026-05-27/aggregates.json`

内容	路径
Claude L/M 1K synthetic rerun	`experiments/dci-compare/runs/rerun-claude-1k-lm-2026-05-26/aggregates.json`
Claude newvariants driver	`experiments/dci-compare/routing-only-newvariants.mjs`
Claude newvariants render	`experiments/dci-compare/render-newvariants.mjs`
Claude D-meta/K/K-lite/L/M 150	`experiments/dci-compare/runs/claude-routing-only-newvariants-150-20260525/summary.json`
Claude K-bounded 1K	`experiments/dci-compare/runs/claude-routing-only-newvariants-k-1k-20260525/summary.json`
Claude L-agentic 1K	`experiments/dci-compare/runs/claude-routing-only-newvariants-1k-20260525/summary.json`
Claude L-agentic 150 rerun	`experiments/dci-compare/runs/claude-routing-only-newvariants-l-150-rerun-20260525/summary.json`
Claude L-agentic v2 150	`experiments/dci-compare/runs/claude-routing-only-newvariants-l-v2-150-20260525/summary.json`
Claude L-agentic v3 150	`experiments/dci-compare/runs/claude-routing-only-newvariants-l-v3-150-20260525/summary.json`
Claude L-agentic v3.1 150	`experiments/dci-compare/runs/claude-routing-only-newvariants-l-v3.1-150-20260525/summary.json`
L-agentic v1 / v2 / v3 SKILL.md 备份	`experiments/dci-compare/variants/routing-only/L-agentic-v{1,2,3}.SKILL.md.backup`
Codex 9x24 summary	`experiments/dci-compare/runs/codex-routing-only-9x24/summary.json`
Codex new-CLI 16x24 rerun	`experiments/dci-compare/runs/rerun-codex-newcli-full-2026-05-26/aggregates.json`
Codex D-agentic metadata-only	`experiments/dci-compare/runs/codex-routing-only-d-agentic-metadata-24-20260524/summary.json`
Codex K-bounded	`experiments/dci-compare/runs/codex-routing-only-k-24-20260524/summary.json`
Codex K-lite fixed	`experiments/dci-compare/runs/codex-routing-only-k-lite-fix-24-20260524/summary.json`
Codex L-agentic 150	`experiments/dci-compare/runs/codex-routing-only-l-agentic-24-20260525-v2/summary.json`
Codex L-agentic 1K new-CLI rerun	`experiments/dci-compare/runs/rerun-codex-newcli-l-agentic-1k-2026-05-26/aggregates.json`
Codex M-bm25 150	`experiments/dci-compare/runs/codex-routing-only-m-bm25-index-fix-24-20260525/summary.json`
Codex M-bm25 original Hard current 24	`experiments/dci-compare/runs/codex-routing-only-m-bm25-skillrouter-hard-24-20260525/summary.json`
Codex M-bm25 paper-core single Hard	`experiments/dci-compare/runs/codex-routing-only-paper-single-hard-m-bm25-24-20260525/summary.json`
Codex J-v2 paper-core single Hard	`experiments/dci-compare/runs/codex-routing-only-paper-single-hard-j-v2-24-20260525/summary.json`
Native prompt capture	`experiments/dci-compare/runs/native-prompt-capture/analysis.md`
Query set	`experiments/dci-compare/queries.json`
Corpus manifest	`experiments/dci-compare/corpus-manifest.json`
Router variants	`experiments/dci-compare/variants/routing-only/*.SKILL.md`
Codex router variants	`experiments/dci-compare/variants/routing-only-codex/*.SKILL.md`
Scaling notes	`experiments/scaling-jbounded/EXPERIMENT_NOTES.md`
J-bounded-v2	`experiments/scaling-jbounded/variants/J-bounded-v2.SKILL.md`
150-scale v2 report	`experiments/scaling-jbounded/runs/sweep24-v2-150-cmd/report.md`
79K-scale v2 report	`experiments/scaling-jbounded/runs/sweep24-v2-full-cmd/report.md`
§9 README + protocol	`experiments/skillrouter-easy/README.md`
§9 implementation plan	`experiments/skillrouter-easy/IMPLEMENTATION_PLAN.md`
§9 install script	`experiments/skillrouter-easy/install-easy-pool.mjs`
§9 variants (Claude/Codex × K/J-v2/M)	`experiments/skillrouter-easy/variants/{claude,codex}/{K-bounded,J-bounded-v2,M-bm25}.SKILL.md`
§9 runner + scorer	`experiments/skillrouter-easy/run.mjs`
§9 report renderer	`experiments/skillrouter-easy/render-report.mjs`
§9 aggregated report	`experiments/skillrouter-easy/runs/report.md`
§9 cross-cell summary	`experiments/skillrouter-easy/runs/all-summary.json`
§9 per-cell summary (metrics + per-query top-10)	`experiments/skillrouter-easy/runs/<host>-<variant>/summary.json`
§9 per-query execution traces (HTML)	`experiments/skillrouter-easy/runs/traces-report.html`

内容	路径
§9 per-query execution traces (JSON)	`experiments/skillrouter-easy/runs/traces-compact.json`
§9 full run log	`experiments/skillrouter-easy/runs/full-run.log`
§9 install manifests	`experiments/skillrouter-easy/runs/install-<host>-<variant>/{manifest.json,queries.json,install.log}`