When Tool-Backed Skill Retrieval Fails: Source-Style Collapse in Executable Capability Retrieval

Anonymous ACL submission

Abstract

Large-scale agents increasingly rely on retrieval to access external capabilities. We study this retrieval gate in structured tools and APIs, a measurable class of tool-backed executable skills that must be surfaced before an agent can plan, incorporate, or act. In this setting the retrieval layer can silently fail even when the capability corpus is fixed: on ToolRet, a retriever fine-tuned on one source-specific slice collapses on another source-specific slice of the *same* benchmark, with `FT-1100` falling to `0.7%` coverage on APIGen despite its higher lexical overlap with the gold tools. We call this failure mode *source-style collapse*. Query-side TF-IDF fingerprints flag source styles on which the fine-tuned retriever is likely to fail better than semantic or length-based proxies, giving a cheap signal for mismatch over a fixed tool corpus. We propose ToolScout, a source-aware routing method that uses this signal as a routing guard: on the mixed `4,996`-query stream, TF-IDF-based routing raises coverage from `22.3%` to `86.1%`, and across five collapsed sources `20` matched examples raise the coverage-weighted global top-1 proxy from `1.3%` to `53.9%`. The same failure and routing behaviors persist when tools are rerendered as executable skill cards, which rules out raw API-schema format as the sole cause.

1 Introduction

Agent systems are moving from short hand-crafted tool lists to large external capability pools. In this setting, retrieval decides which capabilities the model can see before reranking, planning, incorporation, or execution begins. We study the structured and executable case: We render each tool or API as an executable skill card with a name, description, input schema, and implementation. This makes tool retrieval a controlled instance of executable capability retrieval, or Tool RAG. Similar source or style shifts are well known in document retrieval, while Tool RAG changes their operational meaning: when the right tool is missed, the agent loses an executable action instead of merely working with weaker evidence (If a travel-booking API never enters the candidate set, no planner can call it later).

We show that this entry layer can fail even when the tool corpus stays fixed. On ToolRet (Shi et al., 2025), a checkpoint adapted on the `toolbench`-style `1100`-query slice remains strong on matching traffic but collapses on other source styles embedded in the same aggregated benchmark, including APIGen (`0.7%` coverage), ToolACE (`0.0%`), and UltraTool (`8.3%`); see Appendix Table 21 . Lexical overlap alone does not explain the collapse: `APIGen` has higher lexical overlap with its gold tools than the training slice. Standard retrieval adaptation usually treats shift as a change in corpus, task, or broad query distribution. Here the benchmark-wide tool registry is fixed, and the failure is induced by the source style that generated the query-tool pairs. The stress case is therefore a same-corpus source-style shift.

This mismatch matters because realistic agent deployments rarely expose a small hand-crafted function list. They operate over hundreds or thousands of heterogeneous tools and APIs drawn from multiple providers, teams, and documentation styles (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024; Li et al., 2023b; Guo et al., 2024; Shi et al., 2025). In this pipeline, candidate generation first selects a small tool pool. Downstream rerankers or planners then operate on the selected candidates. If the gold tool never enters the pool, no downstream reranker or planner can recover it. Before any downstream planning or execution happens, the system must therefore decide which small subset of tools is worth exposing to the model. On StableToolBench, benchmark-native full-catalog exposure is approximately `172k` to-

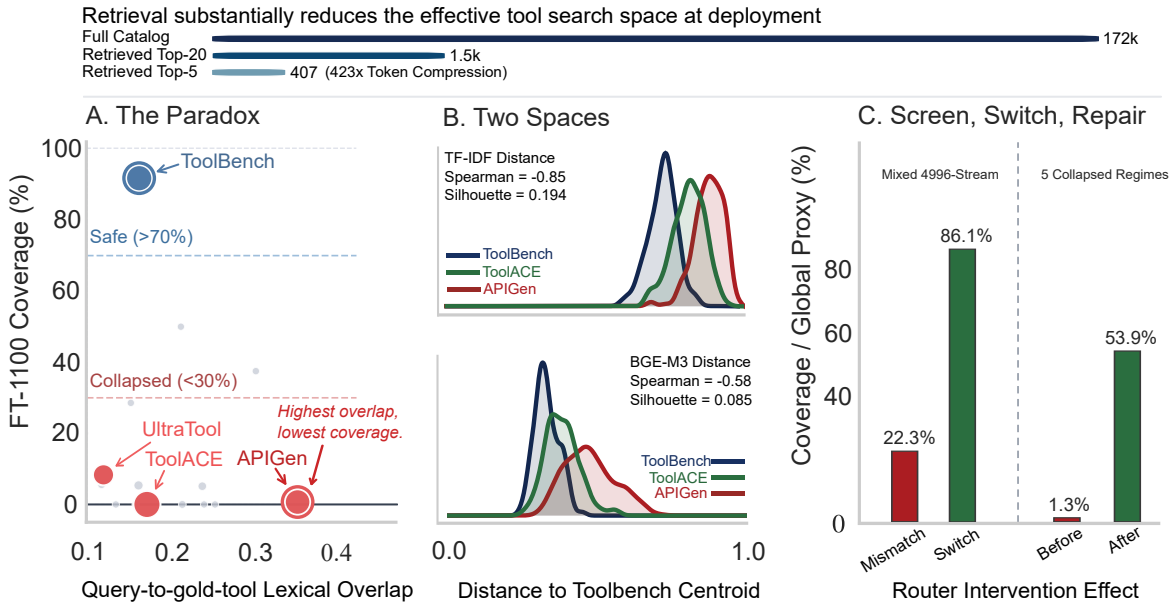

Figure 1: Tool retrieval is source-style-sensitive on a fixed benchmark corpus. **Top:** retrieval compresses benchmark-native full-catalog exposure from about `172k` tokens to about `407` at retrieved top-5. **(A)** Lexical overlap fails to predict retrievability: `APIGen` has the highest query-to-gold-tool overlap yet collapses to `0.7%` coverage under `FT-1100`. **(B)** TF-IDF distance separates source styles more clearly than `BGE-M3` semantic distance (`r=-0.85` vs. `-0.58`). **(C)** Route, switch, then repair: source-aware routing increases mixed-stream coverage by nearly fourfold, and `20` matched examples substantially improve the collapsed-source coverage-weighted global top-1 proxy.

kens, whereas dense top-5 exposure is only about `407`. Tool retrieval is therefore what makes large-catalog prompting feasible at all. Once deployment depends on that compression, retrieval failures become systemic risks instead of local inefficiencies. The broader agent ecosystem only sharpens that pressure through multi-agent orchestration layers and tool-serving standards (Wu et al., 2023; Li et al., 2023a; Hong et al., 2024; LangChain, 2026; OpenAI, 2026; Model Context Protocol, 2026).

This paper studies that problem through **ToolScout**, a source-aware routing method for the retrieval gate. We ask whether a retriever adapted to one source-defined slice of an aggregated tool benchmark remains reliable on other source-defined slices when the tool corpus is unchanged. We use the term *source-style collapse* to refer to the observed failure case in which the adapted retriever remains strong on matching-source queries but loses coverage on other source-defined slices over the same tools. We use *source style* to mean the conventions introduced by an upstream query-generation or annotation process, including query wording, verbosity, schema-reference patterns, and the way queries are paired with tools in a fixed tool corpus. In simple terms, two sources may ask for the same tool in different dialects: one writes long ToolBench-style tasks, while another issues short API-template requests. The shift first becomes visible on the query side, because a retriever adapted on one source-defined slice can collapse on another over the same tools. Further analyses show that the target tool texts preserve a correlated, though weaker, fingerprint as well, so the practical query-side signal is one view of a broader paired shift.

Recent work on Skill Retrieval Augmentation studies a complementary downstream issue, showing that agents still need to incorporate retrieved skills effectively after relevant skills have been found (Su et al., 2026). ToolScout focuses on an earlier retrieval gate. When candidate generation misses the relevant tool, later reranking or planning stages have no correct tool to select. In this setting, the exposed capability is a structured tool or API and the tool corpus is fixed. We ask whether adapting a retriever to one source style can make it unreliable under source-style shift, and whether source-aware routing can recover the lost coverage.

A skill-card rerendering experiment confirms

that raw API-schema formatting is an insufficient explanation: the same failure pattern, detector, and switch policy persist when tools are represented as executable skill cards.

ToolScout uses the query-side signal as a source-aware routing criterion. When a batch is likely to mismatch the source-specific retriever, ToolScout routes it to an aggregate-trained checkpoint and, when matched supervision becomes available, further adapts the retriever to recover the remaining coverage gap. This setting differs from reranking-focused work such as ToolRerank (Zheng et al., 2024), tool-representation work such as Tool2Vec (Moon et al., 2024), and full agent stacks such as AnyTool (Du et al., 2024) as ToolScout isolates the upstream routing layer itself under a shared retrieval and exposure protocol.

We make three linked contributions. First, using structured tools and APIs as measurable executable skill cards, we identify source-style collapse as a concrete retrieval failure mode: a dense retriever can fail on `APIGen`-style queries with *higher* lexical overlap than the source it was trained on, so simple keyword access or query length fail to explain the performance decrease. Second, we characterize the conditions for failure. Across `19` source-query configurations, query-side TF-IDF distance is the strongest lightweight detector of source styles that are likely to mismatch the fine-tuned retriever (`78.9%` leave-one-config-out accuracy), and the surrounding controls show that the signal is broader than a query-only rewrite effect. Third, we validate a source-aware routing method for this retrieval gate. On the mixed `4996`-query stream, TF-IDF-based routing raises coverage from `22.3%` under a mismatched `FT-1100` retriever to `86.1%`. Across five collapsed sources, `20` matched examples then raise the coverage-weighted global top-1 proxy from `1.3%` to `53.9%`. The routing criterion keeps source-specific retrievers on batches where they remain reliable and switches to the aggregate-trained checkpoint when source-style mismatch is likely to reduce coverage.

2 Capability Retrieval and Tool Routing

Skill and capability augmentation. Recent skill-centric work treats reusable skills as external capability units for agents (Li et al., 2026a; Jiang et al., 2026; Li, 2026). Concurrent work on Skill Retrieval Augmentation (SRA) formulates scalable agent capability access as retrieving reusable skills from a large external skill corpus, and introduces SRA-Bench for decomposed evaluation of retrieval, incorporation, and end-task execution (Su et al., 2026). These works establish the broader setting: agents must retrieve external capabilities before deciding whether and how to incorporate them. ToolScout studies an earlier reliability question at that retrieval gate. Before a retrieved skill or tool can be used correctly, the relevant capability must enter the exposed candidate set.

Tool and API retrieval. Tools and APIs are the structured, executable end of the capability spectrum. Toolformer (Schick et al., 2023) and Gorilla (Patil et al., 2024) established the general tool-use setting. API-Bank (Li et al., 2023b), ToolBench / ToolLLM (Qin et al., 2024), StableToolBench (Guo et al., 2024), ToolHop (Ye et al., 2025), and ToolRet (Shi et al., 2025) further sharpened evaluation around large-scale retrieval and multi-hop tool use. Their schemas and gold-tool annotations make them a controlled substrate for measuring capability retrieval failures.

Tool2Vec (Moon et al., 2024) studies tool-specific representations, while ToolRerank (Zheng et al., 2024) studies hierarchy-aware re-ranking and adaptive truncation once a candidate pool is already given. AnyTool (Du et al., 2024) and DeepAgent (Li et al., 2026b) use hierarchical retrieval inside broader agent stacks. Agent frameworks such as ReAct (Yao et al., 2023), AutoGen (Wu et al., 2023), CAMEL (Li et al., 2023a), and MetaGPT (Hong et al., 2024), together with orchestration stacks such as LangChain (LangChain, 2026), the OpenAI Agents SDK (OpenAI, 2026), and MCP tooling (Model Context Protocol, 2026), make small exposed schema sets a practical requirement before downstream planning.

Retrieval adaptation under capability shift. Our setup builds on standard retrieval practice: DPR (Karpukhin et al., 2020), ANCE (Xiong et al., 2021), ColBERT (Khattab and Zaharia, 2020), and BERT re-ranking (Nogueira and Cho, 2019) established core dense and re-ranking baselines, while BEIR (Thakur et al., 2021) made cross-domain transfer a standard concern. Modern encoders such as BGE-M3 (Chen et al., 2024) and Qwen3-Embedding (Zhang et al., 2025) pro-

vide strong off-the-shelf backbones. The closest general-retrieval literature is *domain adaptation for dense retrieval*. GPL, InPars, Promptagator, and AugTriever use pseudo-labeling or synthetic query generation to adapt retrievers to new target distributions (Wang et al., 2022; Bonifacio et al., 2022; Dai et al., 2023; Meng et al., 2022). Our few-shot repair stage is closest in spirit to that line. Our diagnostic setting keeps the capability corpus fixed while changing the source slice that generated the query-tool pairs. That change can still break a narrow dense specialist. Similar shifts can lower relevance in document retrieval, but executable capability retrieval makes the consequence harsher because a missed tool removes an action from the agent's candidate set instead of merely weakening its evidence base.

3 ToolScout Routing Policy

Given a user query `q` and tool library `T`, ToolScout returns a top-k candidate set that fits within the context-window token budget used to expose tools to the downstream model, while preserving enough coverage for downstream execution. In our main setting, this budget is operationalized by the number of retrieved tool cards and their rendered token length, rather than by monetary cost. The core system is a direct two-stage pipeline, `query` → `candidates` → `reranking`. We study staged retrieval and hierarchy only as optional extensions around the core path, and we return to them after the main collapse result.

We use *source style* to mean the conventions introduced by an upstream query-generation or annotation process, including query wording, verbosity, schema-reference patterns, and the way queries are paired with tools in a fixed tool corpus. On the query side, this appears in the distribution of user requests. On the tool side, it appears in the target tool texts that the retriever actually encodes. The main stress case is a same-corpus source-style shift: the tool registry stays fixed, while the phrasing, verbosity, and schema-reference habits of the query-tool pairs still change. When a 1100-trained checkpoint fails sharply under that shift, we refer to the resulting failure mode as *source-style collapse*.

Formally, the direct candidate generator returns

$$C(q) = \mathrm{top}_k(r(q, T)),$$

where r is the lexical or dense retriever and k is the fixed candidate budget. If a query is decomposed into sub-queries $\{q_1, \ldots, q_m\}$, staged retrieval first scores tools for each sub-query and then keeps the top-k tools under the aggregated score:

$$C_{\text{stage}}(q) = \mathrm{top}_k\left(\left\{t \in T : s(t) = \max_{j \in \{1,\ldots,m\}} r(q_j, t)\right\}\right).$$

Reranking then operates on the candidate set it actually receives.

Tools are rendered from structured metadata such as name, description, arguments, and tags, then encoded with a sentence-transformer backbone. We use `BAAI/BGE-M3` (Chen et al., 2024) as the main dense backbone; Table 1 compares it with `bge-small-en-v1.5`, and Table 19 reports additional backbone swaps. Candidate generation uses either lexical or dense retrieval. The main dense pipeline retrieves the top-`k` tools with the encoder and applies a lightweight learned reranker over retrieval, lexical, and metadata features. Appendix Table 5 reports stronger reranker variants, showing that improved reranking helps covered queries but does not remove the candidate-generation bottleneck.

Starting from an off-the-shelf sentence encoder, we fine-tune on query-tool pair supervision from ToolRet using MultipleNegativesRankingLoss (MNRL), the SentenceTransformers implementation of an in-batch contrastive objective closely related to InfoNCE (Oord et al., 2018) with retrieval-confusing hard negatives mined from the current depth-`20` dense pool. This directly targets the bottleneck identified in the paper: if candidate generation is failing upstream, the cleanest repair is to improve the retriever before adding more downstream complexity.

The resulting routing policy is source-aware and can operate with multiple retriever checkpoints. For an incoming query batch $\mathcal{Q}$ and a retriever trained on source slice $\mathcal{R}$, ToolScout computes a compatibility score

$$d_{\text{tfidf}}(\mathcal{Q}, \mathcal{R}) = 1 - \cos\big(\mu_{\text{tfidf}}(\mathcal{Q}), \mu_{\text{tfidf}}(\mathcal{R})\big),$$

where μ_{tfidf} is the batch centroid in TF-IDF space. The safe/unsafe band for d_{tfidf} is calibrated once on the natural ToolRet source splits in leave-one-config-out form and then fixed for the routing experiments. If the resulting distance falls inside that known-safe band, direct routing with the matched fine-tuned retriever is the default path; if it falls

in a clearly unsafe band, the system switches to a broader aggregate-trained checkpoint or keeps staged routing available while matched supervision is collected. Once a small amount of matched supervision becomes available, the same policy can trigger a short continuation fine-tuning step on the mismatched checkpoint. We evaluate this TF-IDF-based compatibility predictor in three settings: leave-one-configuration-out source mismatch detection, the main `1100` $\rightarrow$ `4996` mixed-source transfer test, and few-shot matched adaptation on collapsed sources.

The staged extension keeps the same retriever but changes the retrieval unit. A query can follow the identity path or be decomposed into sub-queries, each sub-query is retrieved independently, and each sub-query is retrieved independently, candidate scores are aggregated across sub-queries, and the top-k tools under the aggregated score are passed to the reranker. The reranker must then match the pool it actually sees, because merged staged pools and direct pools follow different candidate distributions. The hierarchical extension serves a different purpose. It predicts a small set of manual or automatically induced skills before tool retrieval, and we evaluate it as an efficiency mechanism around the direct dense route.

4 Experimental Setup

We evaluate on two benchmarks. **ToolRet** is the main retrieval-stress benchmark. The controlled `1100`-query normalized `web` subset serves as the main diagnostic slice; a broader `4996`-query aggregate over `19` source-query generators tests same-corpus source-style sensitivity; and a full merged ToolRet build over `44,453` tools and `7,726` queries tests whether the same pattern survives at larger scale across the `web`, `code`, and `customized` corpora. **StableToolBench**, an existing tool-learning benchmark derived from ToolBench, plays a different role in our study. As retrieval on this benchmark starts near saturation, we use it mainly for efficiency and hierarchy analyses.

The off-the-shelf comparisons include BM25, `bge-small-en-v1.5`, and `BGE-M3`. We choose these baselines to cover a lexical retriever, a lightweight English dense encoder, and a stronger general-purpose dense encoder. Candidate depth is fixed at `20` for the main dense runs. Reranker datasets use deterministic query-hash train/validation/test splits, and all reranking metrics are reported on held-out candidate-covered test queries from that protocol. Appendix A.1 reports the exact split counts and stronger reranker variants, and Appendix A.3 reports additional backbone comparisons and the mechanism-level ToolRerank comparison.

When candidate generation can fail outright, we additionally report a coverage-weighted top-1 proxy,

$$G = C_m \cdot P@1_{|\mathrm{cov}},$$

where C_m is the top-m candidate coverage for the current evaluation split, and $P@1_{|\mathrm{cov}}$ is top-1 accuracy computed only over covered queries, i.e., queries whose retrieved top-m pool contains at least one gold tool. Thus, G gives a global top-1 proxy: covered queries are evaluated by their conditional top-1 accuracy, and queries with no gold tool in the retrieved top-m pool are counted as failures.

Skill-card rendering. To separate source-style collapse from raw API-schema formatting, we also rerender each tool as an executable skill card. The card normalizes the visible representation into fields such as capability, when-to-use conditions, inputs, and output or effect, while keeping the underlying tool identity, gold labels, splits, retrievers, and routing policy unchanged. This experiment asks whether the same retrieval-gate failure persists under a more skill-like capability representation.

5 Results

5.1 Off-the-Shelf Diagnosis

Table 1 gives the main off-the-shelf two-stage reference: a direct dense retriever followed by the lightweight learned reranker. The table shows a clear difference between the two benchmarks. ToolRet has substantially lower candidate coverage, even on the controlled `1100`-query web slice, while StableToolBench starts near saturation. We therefore use ToolRet for the main source-style collapse analysis and StableToolBench primarily as a high-coverage efficiency setting. Stronger reranker variants and candidate-covered split details are reported in Appendix A.1.

5.2 Adaptation Sensitivity

Matched-source adaptation improves the global top-1 proxy. We next evaluate whether

Benchmark	Setting	Cov.	P@1	
			Sem.	Lrn.
ToolRet	bge-small / full 1100	67.7%	25.0%	50.0%
	BGE-M3 / full 1100	**80.8%**	**34.9%**	**55.8%**
StableToolBench	bge-small / full 765	97.3%	**70.1%**	**71.0%**
	BGE-M3 / full 765	**98.4%**	66.7%	68.5%

Table 1: Off-the-shelf two-stage results under a shared direct retrieval pipeline. "Sem." denotes retriever-only top-1 accuracy, and "Lrn." denotes top-1 accuracy after the lightweight learned reranker. Coverage is top-20 candidate coverage. The results show that ToolRet leaves substantially more candidate-generation headroom than StableToolBench under the same protocol. Full covered-query metrics are provided in Appendix A.1.

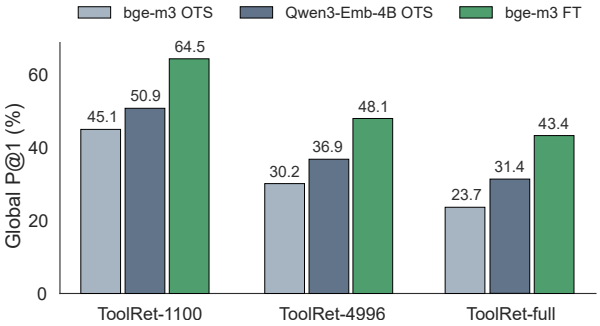

Figure 2: Direct-dense retrieval adaptation across ToolRet-1100, ToolRet-4996, and ToolRet-full. The plotted score is the coverage-weighted top-1 proxy. A substantially larger 2025 off-the-shelf backbone (Qwen3-Emb-4B) narrows the gap in every ToolRet setting while leaving sizable room for source-adapted retrieval. Exact values are listed in Appendix A.3.

adapting the retriever on matched query-tool supervision improves the coverage-weighted global top-1 proxy G, which counts uncovered queries as failed top-1 cases. Across the controlled ToolRet-1100 slice, the broader ToolRet-4996 aggregate, and the full ToolRet build, source-adapted BGE-M3 consistently improves G by about 18–20 percentage points over the off-the-shelf retriever. A larger 2025 off-the-shelf backbone, Qwen3-Emb-4B (Zhang et al., 2025), also improves G, but it remains below the smaller source-adapted BGE-M3 model across all three ToolRet settings. This means that model scale helps, while matched-source retriever adaptation still contributes a substantial additional gain under the same tool corpus. The same conclusion holds at 44,453 tools and 7,726 queries: full-corpus fine-tuning raises top-20 candidate coverage by 23.5 percentage points with only a modest increase in average candidate tokens. Appendix A.3 reports the full numeric breakdown, bootstrap results, and additional backbone comparisons. A multimodal-capable LCO-Omni run in Table 20 shows the same qualitative pattern with a frozen encoder and learned projection adapter.

StableToolBench serves as a boundary case. Nevertheless, matched-source fine-tuning is still positive: coverage rises from 96.3% to 99.1%, the learned dense top-1 score rises from 68.5% to 75.0%, and average candidate tokens decrease. The gain comes mainly from a cleaner candidate pool instead of a large uncovered-tail rescue, which is why we keep StableToolBench primarily for hierarchy and efficiency analysis (Appendix A.3).

Transfer can catastrophically fail. A checkpoint trained only on the 1100-query toolbench-style slice transfers poorly to the broader 4996-query aggregate, where depth-20 coverage collapses to 22.3%; by contrast, an aggregate-trained checkpoint still returns 87.3% coverage on the held-out 1100 test split. Appendix A.3 shows the most striking individual cases: the 1100-trained checkpoint remains strong on the embedded toolbench source (91.8% coverage) but collapses on APIGen (0.7%), ToolACE (0.0%), and UltraTool (8.3%).

Lexical overlap fails to explain the failure. Panel A of Figure 1 makes the paradox visible. Lexical cues alone fail to explain the collapse, because APIGen actually has *higher* average query-to-gold-tool lexical overlap than the toolbench slice. A lexical retrieval control in Appendix 23 sharpens that point. BM25 is competitive on high-overlap sources such as APIGen (75.0% coverage) and ToolACE (67.4%), and BM25+dense fusion usually improves further, but the gains vary by source, with dense still much stronger on UltraTool and the hybrid barely moving toollens. More importantly, in the actual collapsed FT-1100 setting, naive BM25+dense fusion still underperforms BM25 alone across all five collapsed sources (Appendix 24), so lexical fallback helps only when routed explicitly instead of mixed with a mismatched dense branch.

TF-IDF flags clearly unsafe source styles. Across all 19 source-query configurations, TF-IDF centroid distance from the training source tracks transfer collapse more strongly than

semantic-centroid, length, or multi-step proxies (query-count-weighted r=-0.85). As a leave-one-config-out compatibility routing criterion for clearly low-coverage transfer sources (FT-1100 coverage below 30%), the same TF-IDF probe reaches 78.9% accuracy and 0.88 F1, ahead of semantic-centroid distance (73.7%, 0.85), query-length gap (68.4%, 0.80), and multi-step-rate gap (47.4%, 0.58).

A matched tool-side ablation over the normalized tool texts actually encoded by the retriever shows a related but weaker source fingerprint: tool-side TF-IDF distance still tracks the same collapse at weighted r=-0.77, and remains tightly coupled to the query-side distance (weighted r=0.96; Appendix A.3). Appendix 27 and 28 then sharpen the boundary: query-only rewrites shift the fingerprint without recreating the natural collapse, while paired synthetic source styles reproduce substantial negative transfer. Together these controls point to a broader source-style shift whose footprint remains visible on both sides of the retrieval pair, even though the query-side view is the stronger operational detector.

5.3 Source-Aware Response Policy

Panel C of Figure 1 visualizes the first two stages of the response policy. On the mixed 4996-query stream, always using the mismatched FT-1100 checkpoint leaves coverage at only 22.3%. Applying the conservative TF-IDF rule $d_{\text{tfidf}} < \tau$ to route high-coverage source styles to FT-1100 and low-coverage transfer sources to the broader aggregate checkpoint raises coverage to 86.1%, essentially matching the fixed aggregate-trained retriever's 85.2%. In practice, the routing criterion preserves the narrow checkpoint on compatible traffic while routing away from severe coverage-loss cases, before aggregate-level supervision is necessarily available. Appendix Table 10 shows that naive mixed-training controls fail as a replacement: balanced and temperature-reweighted aggregate fine-tuning leave the collapsed sources unimproved while materially degrading compatible toolbench-style traffic.

The few-shot repair table reports the five natural ToolRet source slices where the FT-1100 checkpoint loses coverage most severely in the mixed-source evaluation, including reversechain; these rows test whether a small amount of matched supervision can recover each collapsed source.

Table 2 shows the next stage in that same procedure: once a small amount of matched supervision becomes available, short continuation fine-tuning can repair most of the remaining source-specific loss. Across these five collapsed sources, 20 matched examples raise the weighted MLP global proxy from 1.3% under the unusable FT-1100 checkpoint to 53.9%, close to the 59.0% aggregate-trained reference. The full shot-count sweep in Appendix A.3 shows that 5-shot repair generally fails, while 50-shot continuation approaches the aggregate ceiling on harder sources. The remaining gaps on UltraTool and especially toollens therefore point to higher source-specific sample complexity: 50 matched examples close UltraTool to its aggregate reference, while toollens remains the hardest case even at 50 shots.

	FT-1100		20-shot		FT-aggregate	
Source	Cov.	Global	Cov.	Global	Cov.	Global
APIGen	0.7%	0.3%	93.2%	74.4%	96.6%	73.5%
ToolACE	0.0%	0.0%	77.1%	46.0%	81.7%	47.2%
UltraTool	8.3%	5.7%	84.7%	48.1%	98.6%	81.9%
toollens	5.4%	1.3%	58.9%	27.1%	89.3%	47.3%
reversechain	5.1%	2.3%	89.7%	61.5%	79.5%	32.3%

Table 2: Multi-source few-shot repair from a collapsed FT-1100 checkpoint. "Global" is the coverage-weighted multilayer perceptron (MLP) top-1 proxy. The table reports the headline 20-shot setting only; the full 5/10/20/50-shot sweep is listed in Appendix A.3. The reversechain row has only 39 held-out test queries, so we treat its score cautiously. Sample complexity varies by source: the appendix sweep shows that 50-shot continuation closes the UltraTool gap and narrows toollens further.

5.4 Skill-Card Rendering Check

To test whether source-style collapse is an artifact of raw API-schema rendering, we rerendered each tool as an executable skill card while keeping the tool identities, gold labels, splits, retrievers, and routing policy unchanged. Table 3 shows that the same pattern reappears. On the 1100 slice, skill-card rendering still benefits from matched-source fine-tuning, raising coverage from 72.2% to 89.2%. The same FT-1100 checkpoint then collapses on the mixed 4996-query stream, reaching only 21.7% coverage, while the aggregate-trained checkpoint reaches 84.0%. The collapsed sources remain severely affected, with coverage staying near zero for four of the five sources and reaching only 15.4% on reversechain; see Appendix Table 10. The detector also transfers: weighted

Rendering	1100 OTS	1100 FT	FT-1100→4996	4996 OTS	4996 FT	Switch
Tool schema	80.8%	91.8%	22.3%	73.7%	85.2%	86.1%
Skill card	72.2%	89.2%	21.7%	64.5%	84.0%	85.2%

Table 3: Skill-card rendering check. Coverage is depth-20 candidate coverage. The skill-card rendering keeps the same tools, labels, splits, retrievers, and routing policy, while rewriting visible tool text into normalized executable skill cards. Collapse and the switch policy persist under the more skill-like representation. Full per-source values and the deterministic card template are in Appendix A.5.

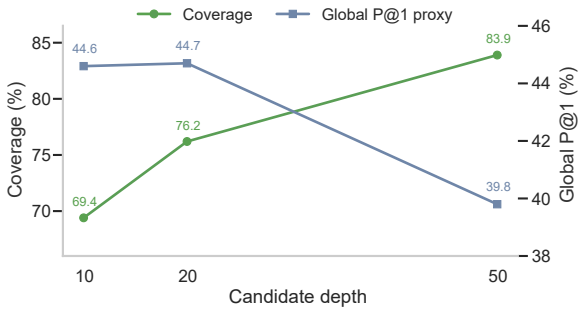

Figure 3: ToolRet depth sweep on the full 1100-query normalized web subset with BGE-M3. Coverage rises with depth, but the coverage-weighted top-1 proxy peaks around the main 20-candidate operating point and then declines. Axes use truncated ranges for readability. Exact values are listed in Appendix A.6.

TF-IDF correlation remains -0.851, ahead of semantic distance at -0.576, and the TF-IDF switch raises mixed coverage from 21.7% to 85.2%. This rules out raw schema formatting as the sole cause; the failure persists under a more skill-like executable capability representation.

5.5 Depth and Extensions

The remaining analyses support the main collapse result as targeted follow-ups. Figure 3 shows that increasing candidate depth has a cost: raising depth from 10 to 50 lifts coverage from 69.4% to 83.9%, but the global proxy peaks at depth 20 and then declines. The remaining failures are also structured: under the strongest direct dense pipeline, 152/210 uncovered queries fall into the many_tools or broad_web_aggregation buckets, which identifies the compositional multi-intent tail as the main residual failure mode.

That tail motivates staged retrieval as a conditional extension. On the main 1100-query slice, split-and-merge retrieval raises overall coverage from 76.2% to 82.9% and improves the two main compositional buckets by about six points. Once the reranker is retrained on the merged candidate distribution, the staged global proxy rises to 49.8% with an adapted MLP and to 55.6% with a compact cross-encoder. Direct matched-source retriever fine-tuning still remains the stronger default move (64.5%), so we use staged routing as a targeted repair for the compositional tail. A compact follow-up table is reported in Appendix A.8.

Hierarchy serves as a coarse capability-layer extension. On StableToolBench, automatic taxonomies at the balanced k=5 operating point preserve 89.2% coverage while reducing candidate context by 87.4%; Appendix A.9 shows the corresponding token stakes, with benchmark-native full-catalog exposure at about 172k tokens versus about 407 for retrieved top-5 exposure. The paired benchmark-native execution check on 259 runnable queries across all six StableToolBench groups removes the quality-efficiency trade-off: under a fixed local Qwen-14B planner and a function-call-accuracy (FAC)-style judge, the official benchmark-provided exposure baseline solves 31.7% (82/259) while the automatic skill-tool exposure policy (auto_skill_tool) reaches 32.8% (85/259). This supports the broader capability-retrieval view while keeping the main reliability claim at the exact tool-backed skill retrieval gate.

6 Conclusion

This paper identifies source-style collapse as a same-corpus failure mode in tool retrieval: a retriever adapted to one source-defined slice can lose coverage on another slice over the same tool corpus. ToolScout addresses this failure with a query-side TF-IDF compatibility predictor, which routes likely mismatched batches to an aggregate-trained checkpoint and then applies short continuation fine-tuning when matched examples become available. The skill-card experiment shows that the failure is not explained solely by raw API-schema formatting. Overall, the results show that reliable tool-using agents need reliable capability retrieval before reranking, planning, or execution can be effective.

Limitations

Our results should be interpreted under four main limitations.

First, several comparisons use a shared retrieval, exposure, and downstream-selection pipeline. This design makes retrieve-then-rerank, tool-only routing, and hierarchy variants comparable at the routing layer, but it does not reproduce each system's original prompt stack, execution policy, or model choices. The results therefore support claims about the routing mechanism under a shared protocol, rather than full reproductions of the original agent stacks.

Second, the most detailed ToolRet reranking, depth, and failure-bucket analyses focus on the controlled `1100`-query normalized web subset. We repeat the core retriever-adaptation comparison on the full `7.6k`-task ToolRet build and observe the same large positive trend, but the merged `web`, `code`, and `customized` corpora provide less detailed error annotation. A positive replication on another independently generated multi-source tool corpus remains an important next test.

Third, the skill-card rendering result is limited to structured tools and APIs rendered as executable skill cards. Broader capability corpora with procedural skills, non-tool resources, and end-task skill incorporation remain outside the current study. The few-shot repair result is also scoped by supervision availability: matched examples could come from failed-query telemetry or tool-provider onboarding, but the rate at which such supervision accumulates in real deployments is left for future work.

Fourth, hierarchical routing is evaluated most strongly on StableToolBench. Our first ToolRet extension shows that manual categories remain stronger than automatic clusters, although both hierarchy variants substantially reduce candidate context. This suggests that category metadata quality matters, and broader tests on category-structured tool corpora are needed.

References

Luiz Henrique Bonifacio, Hugo Queiroz Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. InPars: Data augmentation for information retrieval using large language models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *International Conference on Learning Representations*.

Yu Du, Fangyun Wei, and Hongyang Zhang. 2024. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. *arXiv preprint arXiv:2402.04253*.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic skills – beyond tool use in LLM agents. *arXiv preprint arXiv:2602.20867*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

LangChain. 2026. LangChain documentation: Multi-agent. Accessed: 2026-04-24.

LCO-Embedding. 2026. LCO-Embedding-Omni-3B-2605. Hugging Face model card. Accessed 2026-05-15.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116. Association for Computational Linguistics.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, and 21 others. 2026a. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. 2026b. DeepAgent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*.

Xiaoxiao Li. 2026. When single-agent with skills replace multi-agent systems and when they fail. *arXiv preprint arXiv:2601.04748*.

Rui Meng, Ye Liu, Semih Yavuz, Divyansh Agarwal, Lifu Tu, Ning Yu, Jianguo Zhang, Meghana Bhat, and Yingbo Zhou. 2022. Augtriever: Unsupervised dense retrieval and domain adaptation by scalable data augmentation. *arXiv preprint arXiv:2212.08841*.

Model Context Protocol. 2026. Model Context Protocol: Introduction. Accessed: 2026-04-24.

Suhong Moon, Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Woosang Lim, Kurt Keutzer, and Amir Gholami. 2024. Efficient and scalable estimation of tool representations in vector space. *arXiv preprint arXiv:2409.02141*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

OpenAI. 2026. OpenAI agents SDK guide: Agents. Accessed: 2026-04-24.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, and Lauren Hong. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Weihang Su, Jianming Long, Qingyao Ai, Yichen Tang, Changyue Wang, Yiteng Tu, and Yiqun Liu. 2026. Skill retrieval augmentation for agentic AI. *arXiv preprint arXiv:2604.24594*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, United States. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, and Qi Zhang. 2025. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of ACL 2025*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. 2024. ToolRerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

A Appendix

This appendix is organized as follows. Appendix A.1–A.3 covers reranker checks, larger-scope ToolRet runs, and retriever adaptation details. Tables 21–34 collect source diagnostics, synthetic controls, MetaTool, skill-card rendering, and MCP-lite normalization. Appendix A.6–A.9 covers depth, staged retrieval, hierarchy, and local execution follow-ups. Readers focused only on the main collapse claim can prioritize Tables 5, 20, 25, 26, 27, 28, 31, 10, and 11.

A.1 Reranker Splits and Checks

Table 1 in the main paper gives a compact off-the-shelf dense-routing reference. Candidate coverage is computed on the full benchmark slice used for each row, while the fuller P@1, R@5, and nDCG@5 metrics are reported here on the held-out candidate-covered test split from the reranker protocol described in the experimental setup. For the main ToolRet `BGE-M3` row, this means the reranking metrics come from the `129` held-out candidate-covered test queries inside the full `1100`-query run.

The exact query-hash split counts for the main dense reranker datasets are: `635 / 126 / 129` train / val / test queries for ToolRet (`210` uncovered queries skipped) and `551 / 94 / 108` for StableToolBench (`12` uncovered queries skipped).

The learned reranker is linear, so its feature usage is easy to summarize. Table 4 reports grouped absolute coefficient mass for the two main dense runs. The StableToolBench model is dominated by retrieval-side evidence, while the ToolRet model also leans heavily on metadata-structure features. In both cases, execution-feedback features carry zero weight in the main dense setting.

Benchmark	Ret.	Meta.	Lex.	Query	Exec.
ToolRet	48.2%	44.1%	7.7%	0.0%	0.0%
StableToolBench	71.2%	14.3%	12.7%	1.8%	0.0%

Table 4: Grouped reranker coefficient mass. Ret. = retrieval scores, reciprocal-rank features, and candidate-presence indicators; Meta. = metadata-structure features; Lex. = lexical overlap; Query = query-length features; Exec. = execution-feedback features.

We also ran stronger-reranker controls to test whether the paper's diagnosis is an artifact of the linear model. Table 5 compares the main linear reranker against a two-hidden-layer MLP trained on the same feature sets and data splits, and a compact cross-encoder trained directly on the same candidate-covered reranker datasets. Both stronger rerankers improve covered-query quality, especially on ToolRet. The central interpretation stays unchanged: these gains occur only after candidate generation has already succeeded, so uncovered-query failures remain binding.

Benchmark	Reranker	P@1	R@5	nDCG@5
ToolRet	Linear	55.8%	73.0%	63.2%
ToolRet	MLP	**64.3%**	**78.6%**	**70.4%**
ToolRet	Cross-enc.	76.7%	93.1%	84.7%
StableTB	Linear	68.5%	78.8%	71.6%
StableTB	MLP	**68.5%**	**81.2%**	**73.5%**
StableTB	Cross-enc.	85.2%	89.5%	85.8%

Table 5: Stronger reranker controls on the main dense runs. Metrics are reported on the same held-out candidate-covered test splits as Table 1. Cross-encoders substantially raise the covered-query ceiling, and they still operate only after candidate generation has already succeeded.

A.2 Larger-Scope ToolRet Aggregate

To test whether the main ToolRet pattern survives beyond the `1100`-query normalized web slice, we also built a larger aggregate benchmark by merging `19` ToolRet query configurations over the same `web` tool corpus. The resulting aggregate contains `4996` deduplicated queries (`2242` single-tool and `2754` multi-step), and it is meaningfully harder upstream: under the same depth-`20` dense candidate-generation protocol, `1316` queries still fail to place any gold tool into the candidate pool, so effective candidate coverage falls to `73.7%`. Table 6 shows that this larger scope lowers absolute reranking quality, but preserves the same qualitative ordering as the main ToolRet result: stronger rerankers help substantially once a query is covered, while the dominant difficulty remains upstream candidate generation under a broader and more heterogeneous query mixture.

A.3 Task-Adapted Dense Retriever Fine-Tuning

We also tested a direct repair to the main bottleneck by fine-tuning the dense retriever itself. Starting from `BGE-M`, we train query-tool pairs with **MultipleNegativesRankingLoss** and retrieval-confusing hard negatives. This intervention is intentionally aligned with the paper's diagnosis: if candidate generation is the first-order

Reranker	P@1	R@5	nDCG@5
BM25	27.0%	43.0%	35.1%
Semantic	33.2%	57.4%	45.3%
Linear	35.2%	58.1%	47.0%
MLP	41.0%	69.5%	55.6%
Cross-encoder	**70.0%**	**89.6%**	**79.9%**

Table 6: ToolRet aggregate stress test on 4996 queries from 19 query configs. Metrics are reported on the held-out candidate-covered test split from the same reranker protocol as Table 1. The larger aggregate is substantially harder upstream (73.7% candidate coverage), and once candidates are present the same reranker ordering reappears.

failure mode, then the cleanest method variant is to improve the retriever before changing the reranker or hierarchy.

Main-paper summary rows. Figure 2 in the main paper visualizes the exact ToolRet source slices that carry the main retrieval-adaptation argument. Table 7 lists the corresponding values explicitly.

Source	Retriever	Cov.	P@1	Global
ToolRet-1100	BGE-M3	80.8%	55.8%	45.1%
	Qwen3-Emb-4B	87.3%	58.3%	50.9%
	BGE-M3 FT on 1100	**91.8%**	**70.3%**	**64.5%**
ToolRet-4996	BGE-M3	73.7%	41.0%	30.2%
	Qwen3-Emb-4B	78.9%	46.8%	36.9%
	BGE-M3 FT on 4996	**85.2%**	**56.5%**	**48.1%**
ToolRet-full	BGE-M3	56.4%	42.0%	23.7%
	Qwen3-Emb-4B	72.6%	43.2%	31.4%
	BGE-M3 FT on full	**79.9%**	**54.3%**	**43.4%**

Table 7: Exact values behind the main-paper retrieval-adaptation summary figure. "Global" is the coverage-weighted top-1 proxy.

Hard-negative mining ablation. Because dense fine-tuning is standard in IR, we also asked whether the *specific* hard-negative policy matters for tool retrieval. Table 8 keeps the same BGE-M3 + MNRL training protocol on the 1100 split but swaps the negative source. Retrieval-confusing semantic negatives are strongest, BM25 negatives are close, random negatives are weaker, and same-category negatives are materially worse. This is useful method evidence: *retrieval-confusing* supervision appears to matter most for tool-query alignment.

Mixed-training controls. A natural alternative to TF-IDF-based routing is to reweight source slices during aggregate fine-tuning. Table 10 tests two such controls at candidate generation, where uncovered gold tools are unrecoverable by reranking. Simple reweighting leaves the collapsed sources unimproved relative to natural aggregate fine-tuning, and both variants materially degrade compatible toolbench-style traffic. The likely reason is that uniform balancing oversamples extremely low-support sources while undersampling high-support sources, weakening the aggregate checkpoint's natural in-distribution alignment.

Negatives	FT Cov.	Δ Cov.	FT Multi	FT Recall
Semantic hard	**88.5%**	**+16.2**	89.3%	**71.7%**
Semantic + random	**88.5%**	**+16.2**	**90.0%**	68.2%
BM25 hard	87.8%	+15.5	89.3%	69.2%
Random	84.5%	+12.2	85.7%	68.1%
Same-category	78.4%	+6.1	79.3%	58.8%

Table 8: Controlled hard-negative ablation for BGE-M3 + MNRL on the 1100 validation split. All rows start from the same off-the-shelf depth-20 base coverage of 72.3%. Retrieval-confusing semantic negatives are strongest, while same-category negatives are much weaker despite being semantically plausible distractors.

Train Source	Eval Source	Cov.	Multi	Avg Tok.
Off-the-shelf	ToolRet-1100 val	72.3%	72.9%	1104.4
FT on 1100	ToolRet-1100 val	**88.5%**	**90.0%**	**910.9**
FT on 1100	ToolRet-1100 test	**91.8%**	**92.0%**	**930.6**
Off-the-shelf	ToolRet-4996 test	65.5%	63.5%	1194.4
FT on 4996	ToolRet-4996 test	**85.2%**	**89.5%**	**836.2**
FT on 4996	Transfer to 1100 test	87.3%	88.0%	830.3

Table 9: Depth-20 candidate-generation coverage for task-adapted dense retrievers. The 1100-trained model is strongest on matched data, while the aggregate-trained model generalizes across the broader 4996-query setting.

Training mix	4996	1100 transfer	Collapsed-5	toolbench
Natural aggregate	**85.2%**	**87.3%**	**89.3%**	**87.3%**
Balanced	81.8%	74.1%	88.5%	74.1%
Temp.-reweighted	81.8%	76.6%	88.3%	76.6%

Table 10: Naive mixed-training controls for aggregate retriever fine-tuning. All rows use the same BGE-M3 + MNRL protocol and the same total number of one-per-query training examples; only the source sampling distribution changes. Coverage is reported at candidate depth 20. "Collapsed-5" is the query-count-weighted mean over APIGen, ToolACE, UltraTool, toollens, and reversechain.

Representation-geometry probe. Table 11 checks whether the mixed-training control also

changes the learned similarity geometry. The key quantity is the mean margin between the best gold-tool score and the strongest non-gold candidate score at depth 20. The collapsed-source failure of FT-1100 is visible as a large negative margin, while balanced and temperature-reweighted aggregate training leave the collapsed-source margin largely unchanged and instead reduce the toolbench margin relative to natural aggregate fine-tuning. A simple source-ID linear probe also stays similarly high for the aggregate, balanced, and temperature-reweighted checkpoints, so naive reweighting mainly weakens high-support compatible geometry.

The especially low 10.9% global score for "FT on 1100" evaluated on ToolRet-4996 is the combined effect of two cross-source failures. First, candidate generation collapses from the aggregate off-the-shelf baseline's 73.7% coverage to 22.3%. Second, the surviving covered queries are themselves harder and less aligned with the 1100-trained reranker than the covered queries on the matched slice. The covered-query MLP P@1 therefore stays non-trivial (48.7%), but once that ordering quality is multiplied by the much smaller covered set, the global proxy falls sharply.

Bootstrap significance. We also ran paired bootstrap over the held-out test queries underlying the global top-1 proxy. Table 17 shows that the direct fine-tuning gains remain large and cleanly separated from zero on both ToolRet source slices: the 1100 linear comparison improves by +19.0 points with a 95% interval of [10.8, 27.2], and the 4996 MLP comparison improves by +19.3 points with a 95% interval of [15.4, 23.1]. These numbers are close to the full-slice global proxies in the main text, with small differences because the bootstrap is computed on held-out test queries instead of the full benchmark slice.

To test whether this effect is tied too closely to BGE-M3, we repeated the same 1100-slice direct-dense evaluation with a smaller bge-small-en-v1.5 backbone. The absolute ceiling is lower, but the direction is the same: MNRL fine-tuning still raises depth-20 candidate coverage and substantially improves both linear and MLP direct dense reranking. This capacity control suggests the intervention is broader than one specific encoder family or capacity point.

Additional off-the-shelf backbone comparisons. We also checked whether the main ToolRet gap could be closed simply by swapping in different untuned embedding models. Table 19 compares the paper's off-the-shelf BGE-M3 reference against a weaker 2023 swap (e5-large-v2) and a much larger 2025 backbone (Qwen3-Emb-4B). The result is consistent across the three ToolRet source slices where we have comparable runs: stronger off-the-shelf backbones help, while a clear gap to source-adapted BGE-M3 remains. We use these rows as a backbone-replacement control, with a broader 2024/2025 retriever bake-off outside scope.

We also tested a multimodal-capable embedding backbone, LCO-Embedding-Omni-3B-2605 (LCO-Embedding, 2026). This model uses a different interface from the sentence-transformer training path in the main BGE-M3 runs, so we used its official text-embedding route and trained a shared projection adapter on top of frozen query and tool embeddings. Table 20 shows the resulting source-style sensitivity. The 1100-adapted projection reaches high matched-source coverage (86.7% raw, 88.0% skill-card) and then drops to about 58% on the mixed 4996 stream. Aggregate adaptation returns to 83.3% on the mixed stream and about 88% on the five collapsed sources. The TF-IDF signal keeps the same direction as the main BGE-M3 result: weighted correlations with 1100-adapted coverage are -0.712 for raw tool text and -0.682 for skill-card text, stronger than semantic distance in both cases.

The source-style collapse result is also analyzable. The 1100 and 4996 test splits differ less in raw query length than in *source-style mix*: the main 1100 slice is longer and more consistently multi-step (43.3 tokens, 94.9% multi-step), whereas the aggregate test split is shorter on average (36.2 tokens, 53.0% multi-step) but mixes 19 different source-query configurations. Table 21 shows that the 1100-trained checkpoint remains strong on the embedded toolbench slice inside the aggregate, then collapses on several dissimilar source-query configurations. The aggregate-trained checkpoint is consistently positive across the main high-count configs. This supports the updated interpretation in the main text: the observed transfer failure is visible first on the query side and reflects a broader source-style shift across the retrieval pair.

We also reran the lightweight post-hoc source-distance probe across *all* 19 aggregate source-

Retriever	toolbench Cov.	toolbench Margin	Collapsed-5 Cov.	Collapsed-5 Margin	Probe F1
Off-the-shelf	73.4%	-0.033	71.9%	-0.026	82.2%
FT-1100	**91.8%**	**+0.018**	2.5%	-0.371	76.6%
FT-aggregate	87.3%	-0.016	**89.3%**	**+0.003**	**91.3%**
Balanced aggregate	74.1%	-0.044	88.5%	-0.009	90.9%
Temp.-reweighted aggregate	76.6%	-0.039	88.3%	-0.002	91.0%

Table 11: Representation-geometry probe for the mixed-training control. Margins are the mean best-gold score minus strongest non-gold score at candidate depth 20; higher is better. Probe F1 is a cross-validated source-style linear classifier over query embeddings for the six displayed sources.

Source	FT-1100	5-shot	10-shot	20-shot	50-shot	FT-aggregate
APIGen	0.7%	0.7%	—	93.2%	—	96.6%
ToolACE	0.0%	0.0%	68.0%	77.1%	86.3%	81.7%
UltraTool	8.3%	8.3%	81.9%	84.7%	98.6%	98.6%
toollens	5.4%	5.4%	28.6%	58.9%	87.5%	89.3%
reversechain	5.1%	5.1%	84.6%	89.7%	84.6%	79.5%

Table 12: Coverage sweep for lightweight matched-source repair from a collapsed FT-1100 checkpoint. The main paper reports the headline 20-shot setting; this table gives the full 5/10/20/50-shot progression. Dashes for APIGen mark shot counts skipped in the initial pilot.

Eval Source	Retriever	Lin. P@1	MLP P@1	MLP Global
ToolRet-1100	Off-the-shelf	55.8%	64.3%	51.9%
	FT on 1100	**70.3%**	**69.7%**	**63.9%**
	FT on 4996	54.8%	55.5%	48.5%
ToolRet-4996	Off-the-shelf	35.2%	41.0%	30.2%
	FT on 1100	36.9%	48.7%	10.9%
	FT on 4996	**56.3%**	**56.5%**	**48.1%**

Table 13: Direct dense reranker pipelines under task-adapted dense retrievers. "MLP Global" is the coverage-weighted top-1 proxy using the appropriate candidate-generation coverage for each source. The 1100-trained checkpoint is strongest on the main ToolRet slice, while the aggregate-trained checkpoint remains stable on the broader 4996-query aggregate.

query configurations, using TF-IDF centroid distance from the toolbench_1100 query distribution as a coarse measure of query-side shift. On the full set, Pearson correlation with FT-1100 coverage is -0.61; the same comparison is weaker for BGE-M3 sentence-embedding centroid distance (-0.31), query-length gap (-0.22), multi-step-rate gap (-0.29), and the config's own off-the-shelf base coverage (0.44). The gap between -0.61 and the stronger high-count slices is explained by the aggregate long tail: several source-query configs contribute only 3–8 held-out queries, so their empirical coverage values are highly discrete and high-variance. We therefore also report two additional slices in the artifact: the query-count-weighted TF-IDF correlation remains strong at -0.85, while the same weighted

Retriever	Cov.	Multi	Avg Tok.	Lin. Global	MLP P@1	MLP Global
Off-the-shelf	56.4%	62.8%	1334.7	20.8%	42.0%	23.7%
FT on full	**79.9%**	**86.6%**	1454.5	**42.5%**	**54.3%**	**43.4%**

Table 14: Full ToolRet scale test on the merged 44,453-tool / 7,726-query build spanning web, code, and customized corpora. The same task-adapted dense-retrieval conclusion survives at full scale: candidate coverage rises by +23.5 points and the direct dense MLP global proxy rises from 23.7% to 43.4%, while average candidate tokens increase by only about 9%.

correlation is only -0.58 for sentence-embedding centroids; over the 9 configs with at least 10 held-out queries, TF-IDF again stays stronger (-0.84 vs. -0.67). We report the full per-config probe in the supplemental artifact and treat it as descriptive. It supports the same qualitative conclusion as the main text: brittle transfer is better explained by source-style shift than by length, keyword overlap, or a simple semantic-centroid proxy alone.

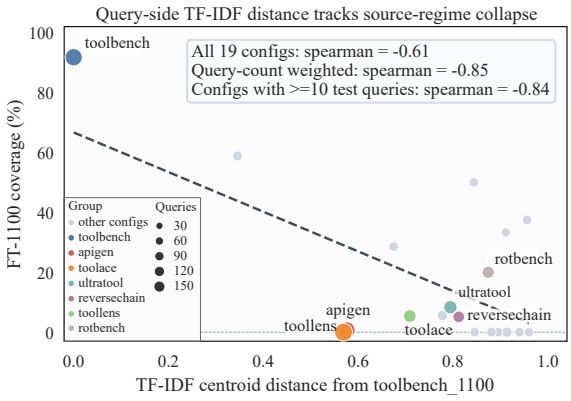

Figure 4: Full 19-config source-distance probe. Each point is one aggregate source-query configuration; point size reflects held-out query count. Across all configs the unweighted correlation is r=-0.61; weighting by query count gives r=-0.85; and restricting to configs with at least 10 held-out queries gives r=-0.84.

Table 25 gives the exact leave-one-config-out compatibility check behind the main-paper sum-

Retriever	Cov.	Avg Tok.	Lin. P@1	MLP P@1	MLP Global
Off-the-shelf	96.3%	1508.0	68.5%	68.5%	66.0%
FT on StableToolBench	**99.1%**	**1304.5**	**75.0%**	**75.0%**	**74.3%**

Table 15: StableToolBench direct dense reranker pipeline under the same `BGE-M3 + MNRL` retriever fine-tuning protocol. Because the benchmark already starts at very high depth-`20` coverage, the gain is smaller than on ToolRet in coverage terms (`96.3%` to `99.1%`), but the learned reranker still improves from `68.5%` to `75.0%` P@1 and from `66.0%` to `74.3%` on the coverage-weighted global proxy, while average candidate tokens decrease.

Eval Source	Retriever	CE P@1	CE Global
ToolRet-1100	Off-the-shelf	**76.7%**	61.9%
ToolRet-1100	FT on 1100	73.1%	**67.1%**
ToolRet-4996	Off-the-shelf	**70.0%**	51.6%
ToolRet-4996	FT on 4996	65.0%	**55.4%**

Table 16: Compact cross-encoder view of the same retriever fine-tuning intervention. Candidate-covered cross-encoder P@1 is already high before retriever adaptation, so the fine-tuned checkpoints leave that covered-query margin similar; the global proxy still improves because matched-source fine-tuning raises candidate coverage substantially before reranking.

Comparison	Baseline	Candidate	Δ	95% CI
`1100` direct FT (Linear)	45.6%	64.6%	+19.0	[10.8, 27.2]
`4996` direct FT (MLP)	30.6%	49.9%	+19.3	[15.4, 23.1]

Table 17: Paired bootstrap significance for the main direct dense fine-tuning gains. Means are computed over the held-out test queries underlying the global top-1 proxy; both improvements have $p < 0.001$.

Backbone	Setting	Cov.	Lin. P@1	MLP P@1	MLP Global
`bge-small`	Off-the-shelf	67.7%	50.0%	60.5%	38.1%
	FT on 1100	**86.1%**	**68.3%**	**69.7%**	**57.6%**
`BGE-M3`	Off-the-shelf	80.8%	55.8%	64.3%	51.9%
	FT on 1100	**91.8%**	**70.3%**	**69.7%**	**63.9%**

Table 18: Cross-backbone validation of the direct retriever fine-tuning effect on the `1100` ToolRet slice. Both `bge-small` and `BGE-M3` improve substantially under the same MNRL fine-tuning protocol, even though the larger backbone remains stronger in absolute terms.

mary sentence. Exact leave-one-config-out coverage regression is noticeably noisier: the weighted linear predictor built from TF-IDF centroid distance attains `0.17` mean absolute error, slightly better than the `BGE-M3` centroid-distance baseline (`0.18`), query-length gap (`0.21`), and multi-step-rate gap (`0.27`) but still too coarse for precise downstream coverage forecasting. The checker rests on only `19` source-level samples, so the exact accuracy intervals remain wide; we interpret source distance as a deployment-time *compatibility routing criterion*.

To test whether this detector mainly sees the query side of the shift, we repeated the same analysis on the *normalized tool texts actually encoded by the retriever*. Table 26 shows that tool-side TF-IDF distance remains informative but is weaker than the query-side probe: its weighted correlation with `FT-1100` coverage is `-0.77`, with `73.7%` leave-one-config-out accuracy and `0.85` F1 for clearly low-coverage transfer sources. The query-side and tool-side distances are themselves tightly coupled (query-count-weighted `r=0.96`), and both are much larger cross-source than under matched-source split-half baselines (`0.15` for toolbench queries and `0.15` for the corresponding tool texts). Combining the two signals by averaging or taking a max leaves the detector unchanged, so the paper keeps the simpler query-side routing criterion in the main text. Mechanistically, this is broader *source-style shift* across both sides of the retrieval pair.

A.4 Additional Boundary Experiments

We include three additional experiments to clarify when the ToolRet source-style effect does and does not appear. These experiments are not intended as new primary benchmarks. They test whether the failure can be reproduced by query-only rewriting, by paired synthetic source generation over the same tool corpus, or by task-family partitions in a second tool corpus.

Query-only rewrite control. We first test whether rewriting the query surface alone can reproduce the large source-style collapse observed on natural ToolRet sources. Using a local `Qwen2.5-14B-Instruct` rewriter, we converted the same `100` held-out `toolbench`-style queries into five target styles patterned after collapsed ToolRet sources. Table 27 shows that the rewrite changes the query distribution: in TF-IDF space, it closes `37.8-142.0%` of the natural target-source distance gap from `toolbench_1100`, and it also

Source	Backbone	Cov.	Avg Tok.	MLP P@1	MLP Global
ToolRet-1100	BGE-M3	**80.8%**	—	**64.3%**	**51.9%**
	Qwen3-Emb-4B	87.3%	**924.6**	58.3%	50.9%
	e5-large-v2	73.4%	1141.6	56.5%	41.5%
ToolRet-4996	BGE-M3	73.7%	—	41.0%	30.2%
	Qwen3-Emb-4B	**78.9%**	**905.7**	**46.8%**	**36.9%**
ToolRet-full	BGE-M3	**56.4%**	**1334.7**	**42.0%**	**23.7%**
	Qwen3-Emb-4B	72.6%	**1097.3**	43.2%	31.4%
	e5-large-v2	52.0%	1337.1	38.7%	20.2%

Table 19: Additional off-the-shelf backbone comparisons on ToolRet under the same direct dense MLP reranking pipeline. The BGE-M3 rows reuse the paper's off-the-shelf reference coverage together with the appendix MLP reranker results from Table 13; the Qwen3-Emb-4B and e5-large-v2 rows come from corresponding held-out direct-dense runs on the same sources. The comparison tests whether changing the untuned encoder family closes the gap; a full retriever bake-off is outside the scope of this paper.

Rendering	Adapter training	1100 test	4996 test	Collapsed-5	TF-IDF switch
Tool schema	1100	86.7%	57.8%	55.1%	83.6%
Tool schema	4996	85.4%	83.3%	88.5%	—
Skill card	1100	88.0%	58.7%	57.4%	84.0%
Skill card	4996	83.5%	83.3%	88.1%	—

Table 20: LCO-Omni projection-adapter validation. Values are depth-20 candidate coverage using official last-token text embeddings from LCO-Embedding-Omni-3B-2605. The backbone is frozen and a shared linear projection is trained over query/tool embeddings. The pattern matches the main story: matched-source adaptation is strong, transfer to the mixed stream drops sharply, and the TF-IDF switch restores aggregate-level coverage.

Config	Qs.	Base	FT-1100	FT-4996
toolbench	158	73.4%	**91.8%**	87.3%
APIGen	146	82.9%	0.7%	**96.6%**
ToolACE	175	79.4%	0.0%	**81.7%**
UltraTool	72	69.4%	8.3%	**98.6%**
toollens	56	21.4%	5.4%	**89.3%**
reversechain	39	74.4%	5.1%	**79.5%**

Table 21: Per-config candidate coverage on the 4996 aggregate test split. The 1100-trained checkpoint remains strong on the embedded toolbench source, while the aggregate-trained checkpoint improves coverage across heterogeneous source-query generators.

Config	Avg Tok.	Multi-step	Lex. Overlap	FT-1100 Cov.
toolbench	43.3	94.9%	0.195	91.8%
APIGen	18.0	24.0%	0.427	0.7%
ToolACE	58.3	18.3%	0.196	0.0%
UltraTool	37.3	75.0%	0.157	8.3%

Table 22: Simple query-side feature probe for the main source-query generators discussed in the paper. "Lex. Overlap" is the average fraction of query content tokens that also appear in the corresponding gold-tool names, descriptions, or tags. The collapse of the 1100-trained retriever fails to track query length or simple keyword overlap alone: APIGen has the highest overlap here, yet still collapses to near-zero coverage.

moves in BGE-M3 semantic space. The downstream coverage change is much smaller than the natural ToolRet collapse. The largest drop is the ToolACE rewrite, where FT-1100 coverage changes from 92.0% to 83.0%; the other target styles leave FT-1100 coverage between 89.0% and 93.0%. This suggests that query-only rewriting can partially move queries toward target-source styles, but the large natural collapse also depends on changes in the paired tool representations.

Paired synthetic-source transfer. We next test whether transfer loss appears when both queries and tool renderings are generated under paired source styles over the same ToolRet tool corpus. Table 28 reports two variants. The first uses the same local Qwen2.5-14B-Instruct generator with two prompts over the same candidate tool packs. The second keeps Qwen2.5-14B-Instruct for the natural task-description style and uses a separate MirrorAPI checkpoint for the compact API-template style. Both variants produce stronger transfer loss than query-only rewriting: fine-tuning improves matched-source coverage and loses coverage cross-source, with the cross-generator variant also falling below the off-the-shelf retriever in the mismatched direction.

Source	Qs.	BM25	Dense	BM25+Dense
toolbench	1099	51.0%	60.1%	**64.9%**
APIGen	1000	75.0%	77.9%	**82.1%**
ToolACE	1000	67.4%	72.7%	**77.8%**
UltraTool	500	41.8%	**67.4%**	62.4%
toollens	314	20.4%	21.3%	**21.7%**
reversechain	200	46.5%	61.0%	**63.5%**
collapsed_5	3014	59.4%	67.4%	**69.9%**

Table 23: Lexical BM25, off-the-shelf dense BGE-M3, and BM25+dense reciprocal-rank fusion on key ToolRet source-query configurations under the same top-20 candidate budget. BM25 is competitive on high-overlap sources such as APIGen, and fusion usually improves further, but the gains remain source-dependent and leave hard cases. We use this table as a lexical control for the source-style transfer analysis.

Source	Qs.	BM25	FT-1100	BM25+FT
toolbench	1099	51.0%	**94.7%**	91.3%
APIGen	1000	**75.0%**	0.7%	68.3%
ToolACE	1000	**67.4%**	0.0%	63.8%
UltraTool	500	**41.8%**	6.2%	36.2%
toollens	314	**20.4%**	2.5%	15.0%
reversechain	200	**46.5%**	1.5%	42.0%
collapsed_5	3014	**59.4%**	1.6%	54.2%

Table 24: Lexical BM25 versus the actually collapsed FT-1100 dense checkpoint, plus reciprocal-rank fusion between the two, under the same top-20 candidate budget. In the real mismatched setting, naive lexical+dense fusion underperforms BM25 alone across all five collapsed sources, even though the same FT-1100 checkpoint remains excellent on the matched toolbench slice. This is why the paper routes away from an unsafe dense specialist instead of blending it with a lexical branch.

MetaTool task-family comparison. As a second-corpus comparison, we convert MetaTool into the same retrieval format and use its task-family labels as a coarse partition. This experiment asks whether arbitrary task-family differences are enough to reproduce ToolRet-style collapse. MetaTool's public task-family labels describe shared-process task families rather than independent query-generation sources, and the subtask1_similar and subtask3_reliable families share 100% of their queries. Table 29 shows that family-specific fine-tuning does not produce ToolRet-style collapse: off-the-shelf BGE-M3 already reaches 79.0% coverage overall, and all reported cross-family cells remain at 88.5% coverage or higher. This supports a bounded interpretation of the main result: the large ToolRet collapse is tied to independent source-generation effects over a shared tool corpus, rather than to any task partition within a tool benchmark.

Proxy	LOO Acc.	95% Acc. CI	LOO F1
TF-IDF centroid distance	**78.9%**	[54.4, 93.9]	**0.88**
BGE-M3 centroid distance	73.7%	[48.8, 90.9]	0.85
Query-length gap	68.4%	[43.4, 87.4]	0.80
Multi-step-rate gap	47.4%	[24.4, 71.1]	0.58

Table 25: Leave-one-config-out compatibility check for unsafe transfer sources, defined as FT-1100 coverage below 30%. F1 is reported for the positive unsafe class.

Feature	Weighted r	LOO Acc.	LOO F1
Query-side TF-IDF	**-0.86**	**78.9%**	**0.88**
Tool-side TF-IDF	-0.77	73.7%	0.85
Average(query, tool)	-0.83	**78.9%**	**0.88**
Max(query, tool)	-0.86	**78.9%**	**0.88**

Table 26: Query-side versus tool-side source fingerprints on ToolRet-4996. Tool-side distance is computed over the *normalized tool texts actually encoded by the retriever* (name, description, arguments, examples) for each config's gold tools. The tool-side signal remains informative and tightly correlated with the query-side one, while the query-side detector stays strongest.

A.5 Skill-Card Rendering Check

The main text reports a compact skill-card rendering check. This appendix gives the deterministic rendering template and the full coverage details. The check keeps tool identities, gold labels, train/test splits, retriever backbones, fine-tuning protocol, and routing policy fixed; only the visible tool text is rerendered. The template is:

Table 31 gives the main summary values. The original tool-schema row is copied from the main ToolRet experiments for comparison, while the skill-card row is retrained and evaluated on the rerendered corpora. Matched-source fine-tuning remains useful, the FT-1100 transfer collapse remains severe, the aggregate checkpoint remains stable, and the same TF-IDF switch policy remains effective.

Table 32 shows the collapsed sources under skill-card rendering. The same sources remain severely affected by the FT-1100 specialist, while aggregate training recovers high coverage.

The detector result also persists. With skill-card rendering, weighted TF-IDF distance correlates with FT-1100 coverage at r=-0.851, while

Target	TF-IDF gap	Sem. gap	Off-the-shelf cov.	FT-1100 cov.
APIGen	71.1%	30.7%	75.0 → 77.0	92.0 → 90.0
ToolACE	142.0%	87.2%	75.0 → 66.0	92.0 → 83.0
UltraTool	40.1%	10.3%	75.0 → 76.0	92.0 → 93.0
rotbench	52.1%	32.4%	75.0 → 72.0	92.0 → 89.0
reversechain	37.8%	23.0%	75.0 → 76.0	92.0 → 89.0

Table 27: Synthetic query-only rewrite control on 100 held-out toolbench-style queries. Gap columns measure how much of the natural target-source centroid-distance gap from toolbench_1100 is closed by rewriting, under the same TF-IDF and BGE-M3 semantic views used in the main source-style probe. Coverage columns report original-query → rewritten-query top-20 coverage under off-the-shelf BGE-M3 and the source-specific FT-1100 retriever. Values above 100% indicate that the rewrite moves beyond the natural target source in that representation space.

	Qwen-only		Qwen/MirrorAPI	
Train / Eval	Natural	API	Natural	API
Off-the-shelf BGE-M3	72.1%	56.3%	72.1%	39.1%
FT natural	**89.5%**	60.6%	**88.4%**	42.0%
FT API-template	55.8%	**84.5%**	43.0%	**82.6%**

Table 28: Paired synthetic-source transfer over a shared ToolRet tool corpus. Values report top-20 coverage. The Qwen-only variant uses the same Qwen2.5-14B-Instruct generator for both styles; the Qwen/MirrorAPI variant keeps the natural style fixed and uses MirrorAPI for the API-template style. Both settings show transfer loss under a shared tool corpus, with stronger loss in the cross-generator setting.

Train ↓ / Eval →	Task1	Similar	Scenario
task1	96.3%	91.0%	92.0%
similar	100.0%	91.7%	96.7%
scenario	93.8%	88.5%	100.0%
Off-the-shelf BGE-M3	82.7%	78.8%	74.3%

Table 29: MetaTool task-family comparison. Rows fine-tune BGE-M3 on one single-tool task family and evaluate top-20 coverage on another. MetaTool is useful as a second tool corpus, but its task-family labels are shared-process task labels. The absence of collapse here supports a bounded interpretation: ToolRet's source-style collapse is tied to independent generation sources more than arbitrary task-family differences inside a mostly shared generation process.

Field	Template content
Skill	{tool_name}
Capability	This skill enables the agent to {description}.
When to use	Use this skill when the user request matches {tags/categories/description}.
Inputs	{arguments or schema fields}
Output/effect	{available output or API-effect description}
Metadata	{tags and categories}

Table 30: Deterministic skill-card rendering template. The rendering is template-based and changes the visible capability representation while keeping labels, queries, and supervision fixed.

weighted semantic distance reaches r=-0.576. A one-threshold TF-IDF switch raises mixed 4996 coverage from 21.7% under FT-1100 to 85.2%; adding a base fallback gives 86.0%. This is a representation control for tool-backed executable skills; broader skill retrieval remains outside the scope of this experiment.

Finally, we tested a stricter schema-normalization counterfactual motivated by emerging tool standards such as MCP. Starting from the exact tool texts consumed by the retriever, we rebuilt both ToolRet-1100 and ToolRet-4996 under an *MCP-lite* rendering that keeps only name, a cleaned description, and an arguments-style field with fixed ordering, while removing source-specific metadata and boilerplate. Table 33 shows that this paired retrain/eval ablation leaves the failure mode in place. The narrow-checkpoint transfer collapse remains severe (22.3% → 25.1%), while off-the-shelf retrieval becomes noticeably weaker in absolute terms. We read this as evidence for a structural tradeoff: stricter standardization reduces some source-specific variation, yet it also discards surface cues that a default retriever exploits.

A complementary representation-level check helps explain why the aggregate-trained checkpoint remains a stable fallback under the same normalization. Using the paired original/MCP-lite corpora above, we sampled 200 gold tools from each collapsed source (1,000 tools total) and measured cosine similarity between the tool embeddings produced by the same retriever on the original versus MCP-lite text. Table 34 shows a large and directionally consistent gap: the aggregate-trained checkpoint is much more stable than FT-1100 across all five sources. The APIGen row is the most extreme because its source format is already close to the MCP-lite template, but the mean gap remains substantial even after excluding APIGen.

Rendering	1100 OTS	1100 FT	FT-1100→4996	4996 OTS	4996 FT	Switch
Tool schema	80.8%	91.8%	22.3%	73.7%	85.2%	86.1%
Skill card	72.2%	89.2%	21.7%	64.5%	84.0%	85.2%

Table 31: Core skill-card rendering check. Values are depth-20 candidate coverage. Rerendering tools as executable skill cards lowers the off-the-shelf baseline but preserves the main transfer-collapse and switch-policy pattern.

Source	Queries	OTS	FT-1100	FT-aggregate
APIGen	146	82.9%	2.1%	96.6%
ToolACE	175	75.4%	0.6%	85.7%
UltraTool	72	66.7%	1.4%	98.6%
toollens	56	23.2%	0.0%	87.5%
reversechain	39	66.7%	15.4%	82.0%

Table 32: Collapsed-source coverage under skill-card rendering. The natural collapsed sources remain collapsed under FT-1100, supporting the interpretation that raw schema formatting alone fails to explain the failure.

A.6 Depth Sweep Details

Figure 3 in the main paper compresses the candidate-depth tradeoff into coverage and global top-1. Table 35 gives the full covered-query and global metrics for the same sweep.

A.7 Qualitative Case Studies

Table 36 grounds the two main qualitative claims in concrete queries. The first row shows the same-corpus source-style shift behind the APIGen collapse: the gold tool names are almost literally present in the query, yet the phrasing follows a compact API-template style instead of the longer task-description supervision seen on the main toolbench-style slice. The second row shows the compositional tail that motivates staged retrieval: a single Yosemite request bundles weather lookup, trail discovery, and recommendation-style aggregation, and the conservative splitter rewrites it into two much cleaner retrieval problems before merged-pool reranking.

The staged pipeline reacts differently. Replacing the base retriever with the aggregate-trained checkpoint inside the conservative Qwen-14B staged pipeline improves the larger aggregate only marginally (33.2% to 33.5% under the staged MLP) and degrades the held-out 1100 staged evaluation (58.9% to 22.2%). These rows already use rerankers retrained on the corresponding staged merged-pool datasets, so the weak transfer is deeper than an unfitted second stage. This reinforces a separate point already visible in the main text: staged routing imposes its own merged-pool reranking problem, and a stronger retriever alone leaves the second-stage adaptation issue unsolved.

Setting	Original	MCP-lite	Δ
1100 off-the-shelf	80.8%	68.4%	-12.4
1100 FT (matched)	91.8%	90.5%	-1.3
FT-1100 → 4996	22.3%	25.1%	+2.8
4996 off-the-shelf	73.7%	66.8%	-6.9
4996 FT (matched)	85.2%	84.3%	-0.9
FT-4996 → 1100	87.3%	87.3%	+0.0

Table 33: Paired retrain/eval under a stricter MCP-lite tool rendering. Both ToolRet-1100 and ToolRet-4996 are rebuilt with unified name + description + arguments fields and the retrievers are retrained on the corresponding normalized corpora before evaluation. Schema normalization leaves the narrow-checkpoint collapse in place and lowers off-the-shelf coverage substantially. All values are retriever coverage, before reranked global top-1.

Source	FT-1100	FT-aggregate	Δ
APIGen	0.337	0.924	+0.587
ToolACE	0.341	0.621	+0.279
UltraTool	0.454	0.915	+0.461
toollens	0.470	0.747	+0.277
reversechain	0.451	0.705	+0.254
Mean (all 5)	0.411	0.782	+0.372
Mean (excl. APIGen)	0.429	0.747	+0.318

Table 34: Representation-level invariance under MCP-lite normalization on collapsed-source gold tools. Each row reports the mean cosine similarity between the embedding of a tool's original text and the embedding of its MCP-lite text under the same retriever checkpoint, using 200 sampled gold tools per source (1,000 tools total). The aggregate-trained checkpoint produces more stable tool representations than FT-1100 in every source. The especially high APIGen value reflects that its source format is already close to the MCP-lite template.

A.8 Split-and-Merge Follow-Up Metrics

The main text reports the split-and-merge probe first as a coverage intervention and then as a staged-retrieval baseline with adapted rerankers. Here we report the same story in compact form. All rows below use the same semantic + semantic_rerank retrieval setup and the same fixed 40-candidate budget as the main split-and-merge probe; only the reranker changes.

We keep the broader staged-routing family compact here. Learned decomposition variants

Depth	Cov.	P@1		R@5		nDCG@5
		Covered	Global	Covered	Global	Covered
10	69.4%	**64.3%**	44.6%	**90.7%**	**62.9%**	**77.1%**
20	76.2%	58.7%	**44.7%**	78.7%	60.0%	67.6%
50	**83.9%**	47.4%	39.8%	64.8%	54.4%	55.2%

Table 35: Exact ToolRet depth-sweep values behind Figure 3. Increasing depth raises coverage, but covered-query quality decays enough that the global top-1 proxy peaks at the main `20`-candidate operating point.

and adaptive first-stage triggers were directionally consistent with the main split-and-merge result, and staged retrieval remains a conditional extension around the main collapse result.

A.9 Hierarchy Efficiency and Benchmark-Native Execution

Automatic taxonomies are induced with **MiniBatchKMeans** (Pedregosa et al., 2011) over tool embeddings. In the main StableToolBench result, we use `46` clusters to match the benchmark's `46` manual categories, then summarize each cluster with its nearest representative tools and use those clusters as the first-stage skill space.

The benchmark-native local-execution check keeps only the two most policy-relevant hierarchy conditions: the official benchmark-provided exposure baseline and `auto_skill_tool`. Under exact validation against the local StableToolBench tool store, this yields the complete shared executable subset across *all six* StableToolBench groups: `74` queries in `G1_instruction`, `65` in `G1_category`, `57` in `G1_tool`, `26` in `G2_instruction`, `16` in `G2_category`, and `21` in `G3_instruction`, for a total of `259`. Table 40 reports the resulting benchmark-native execution comparison under the official StableToolBench execution code, a fixed local `Qwen-14B` planner, and the same local function-call-accuracy-style judge used throughout this appendix. On that larger paired set, `auto_skill_tool` slightly exceeds the official exposure baseline overall (`32.8%` vs. `31.7%`) while also improving token cost per solved task (`17.7k` vs. `18.3k`). This is the only hierarchy-side execution result we keep in the appendix because it is the one that directly supports the main-text efficiency claim.

Phenomenon	Query Form	Gold Tools	Why It Matters
Same-corpus source-style shift	`APIGen`: "Please provide the latest songs from the artist Beyoncé and also list the countries in ascending order."	`all_songs_from_artist`; `getallcountry`	The query has obvious lexical access to the tool schemas, yet it is phrased as a compact API-style command instead of a longer end-user task description. This is the kind of high-overlap cross-source case where the `1100`-trained retriever still collapses.
Compositional tail / staged repair	Yosemite query: weather forecast + trail recommendations + scenic-view filtering in one request	`VALUE_SERP_Google_Search`; `VALUE_SERP_Google_Place_Details`; `VALUE_SERP_Google_Video`	Single-shot retrieval must compress heterogeneous intents into one embedding. The conservative splitter rewrites this into "Provide the weather forecast for Yosemite ..." and "Suggest popular hiking trails in Yosemite ...", which produces two cleaner retrieval problems before the candidates are merged.

Table 36: Two concrete cases behind the main findings. The first illustrates why same-corpus source-style shift goes beyond missing keywords. The second illustrates why staged retrieval helps the compositional tail even when direct dense retrieval is already strong on average.

Setting	Cov.	P@1	R@5	Global top-1
Single + original reranker	76.2%	45.6%	77.5%	34.7%
Split + original reranker	82.9%	39.2%	77.3%	32.5%
Split + adapted linear	82.9%	46.8%	78.8%	38.8%
Split + adapted MLP	**82.9%**	**55.7%**	**83.6%**	**46.2%**
Split + adapted cross-encoder	82.9%	67.1%	93.1%	55.6%

Table 37: Staged-retrieval follow-up on ToolRet under the same candidate-generation setup as the main split-and-merge probe. The merged pool hurts top-1 quality under the unchanged reranker, but adapted rerankers recover and then surpass the single-query global proxy. A compact cross-encoder pushes the same staged pool further upward, showing that the staged candidate generator remains competitive under stronger rerankers as well.

top-k skills	Manual skill → tool				Automatic skill → tool			
	P@1	R@5	Cov.	Cand. Red.	P@1	R@5	Cov.	Cand. Red.
1	34.9%	46.4%	49.8%	96.5%	42.6%	57.1%	65.0%	97.2%
3	47.1%	63.0%	67.8%	89.8%	53.7%	71.8%	80.5%	92.3%
5	51.8%	69.5%	77.3%	83.5%	**58.2%**	**76.9%**	**89.2%**	**87.4%**
7	55.3%	73.9%	84.4%	77.6%	60.4%	79.2%	92.3%	82.5%
10	59.6%	79.0%	90.3%	69.6%	62.1%	82.6%	95.4%	75.2%
Tool-only ref.	**66.1%**	**87.2%**	**100.0%**	0.0%	*accuracy anchor*			

Table 38: Main StableToolBench hierarchical-routing Pareto table. Tool-only retrieval is the accuracy anchor; the hierarchy rows trade quality for candidate-context reduction.

Routing Mode	Cand. Tok.	Final Tok.	Cov.	P@1	R@5	Cand. Red.
Full Catalog Injection	172,426	172,426	100.0%	–	–	0.0%
Tool-Only Retrieval	172,426	407	100.0%	66.1%	87.2%	0.0%
Manual Skill → Tool (k=5)	28390	410	77.3%	51.8%	69.5%	83.5%
Automatic Skill → Tool (k=5)	**21785**	411	89.2%	58.2%	76.9%	**87.4%**

Table 39: Routing-stage token efficiency on StableToolBench. Full catalog versus tool-only explains final-prompt compression, while tool-only versus hierarchy explains candidate-pool compression.

Group	Count	Official	Auto	Δ Auto
G1_instruction	74	35.1%	31.1%	-4.1
G1_category	65	33.8%	46.2%	+12.3
G1_tool	57	31.6%	29.8%	-1.8
G2_instruction	26	26.9%	26.9%	+0.0
G2_category	16	25.0%	6.2%	-18.8
G3_instruction	21	23.8%	33.3%	+9.5
All 6 groups	**259**	**31.7%**	**32.8%**	**+1.1**

Table 40: Benchmark-native local execution on the 259-query paired common subset shared by official_api_list and auto_skill_tool across all six StableToolBench groups. The two methods use the same local Qwen-14B planner and the same local function-call-accuracy-style judge. Average end-to-end token counts stay nearly identical (5,807 vs. 5,822) because planner/execution overhead dominates the full trace length, but token cost per solved task still favors the automatic hierarchy (17.7k vs. 18.3k).