

Tool-Description Quality Is Not One Axis: A Regime Analysis of Where It Helps and Where It Backfires

Abstract

Tool-description quality is widely treated as a broadly-applicable lever for agent tool-use, but it is not a single better/worse axis: the precision that helps an agent disambiguate within a family of confusable tools is orthogonal to, or actively harmful for, context-rich selection and for tool retrieval. We test this with a single frozen evaluation protocol — one classifier, one judge, one generator family, pre-registered thresholds — across a synthetic confusable-catalog experiment, two real production MCP-server mirrors (GitHub, AWS IAM), a synthetic internal-proxy catalog, and a pre-registered pilot of ten public Python MCP servers. The effect is real but regime-bounded, not a general law: an oracle description helps at one tested catalog density (60 tools/10 families, +34.5pp on gemma2:9b, and not collapsing on a substantially stronger model, Llama-3.3-70B) with no headroom; realizing it safely through automatic generation is a separate condition, requiring documented source. Outside these conditions the effect is null or reverses (zero headroom on two well-documented production servers; −20pp harm on one already-resolved family; harm to retrieval across three retriever types). Contributions: a falsifiable regime map of where description quality helps, harms, or does nothing; a pre-registered prevalence measurement finding the behavioral regime in 0 of 9 testable Python MCP servers — a lower bound, not a population estimate; and a localizability boundary — a pairwise LLM-judge confusability method fails under two independent framings, via two distinct failure mechanisms.

1. Introduction

This paper’s central finding is counterintuitive enough to state up front, with its scope attached: on the fixtures tested here, tool-description quality is not a single better/worse axis. The same behavioral-axis precision that helps an agent disambiguate within a family of confusable tools (§4.2.1, one density point: 60 tools / 10 families) is not what a context-rich agent already resolving from task wording needs, and is actively harmful to tool retrieval (§4.3.3, one synthetic catalog, three retriever types). Optimizing a description for one of these objectives can measurably neutralize or harm another — a real, mechanistically legible pattern on the fixtures tested, not a demonstrated general law about description quality. The rest of this paper places that finding inside a broader map (Section 4), asks how common the map’s “it matters” region is in real servers (Section 5), and tests whether that region can be found cheaply before it is needed (Section 6).

1.1 The assumption under test

Agentic systems that call external tools — via MCP servers, OpenAI-style function schemas, or similar interfaces — depend on an LLM agent correctly selecting which tool to invoke for a given task. Practitioner-facing engineering guidance treats the *quality of the tool description text* as a primary lever for improving that selection: Anthropic’s own tool-use engineering guidance states that “even small refinements to tool descriptions can yield dramatic improvements,” reporting

that precise refinements to tool descriptions helped Claude Sonnet 3.5 reach state-of-the-art on SWE-bench Verified (Anthropic Engineering, “Writing tools for agents” — publication date not independently confirmed during this paper’s preparation, cited for content only) . GitHub’s engineering team, running an offline evaluation harness against their own MCP server — the same [github/github-mcp-server](#) mirrored in RW1 (§4.3.1) — reports the same sensitivity from the other side: “tightening a description, adding or removing a tool, or combining a few similar tools can shift results a lot” (GitHub Engineering, “Measuring what matters: how offline evaluation of GitHub MCP Server works” — publication date not independently confirmed during this paper’s preparation, cited for content only) . This appears to sit in tension with RW1’s own finding of zero headroom on that exact server for gemma2:9b (§4.3.1) — we do not resolve this tension here, since GitHub’s harness details (which model(s), which metric, which task distribution) were not verified during this paper’s preparation beyond the quoted sentence; flagged as an open question rather than silently juxtaposed as if the two results agree. An empirical audit of 856 tools across 103 real MCP servers found 97.1% of tool descriptions contain at least one quality “smell” and 56% fail to state their purpose clearly, and that LLM-based description augmentation lifts task success by a median of +5.85pp — but also regresses performance in 16.67% of cases and inflates execution steps by 67.46% (Hasan et al., “Model Context Protocol (MCP) Tool Descriptions Are Smelly!,” arXiv:2602.14878, 2026) . Real tool-name collisions cause user-visible failures in production agent frameworks, independent of any synthetic fixture (e.g., a reported GitHub issue: “Duplicate tool names across MCP servers cause errors,” [openai/openai-agents-python#464](#)) . Taken together, this is real evidence that description quality matters *somewhere* and that even small edits move outcomes — but none of these sources characterize the *boundary* of where it matters, whether it generalizes across catalog scales, or whether the same fix that helps selection also helps or harms retrieval and already-correctly-resolving cases. That is the gap this paper addresses.

1.2 What this paper does instead

We hold three properties fixed across every experiment reported here: one frozen evaluation protocol (Section 3), one classifier, and null-first-class reporting — an aborted or negative result is reported as a finding, not omitted. Against that fixed protocol we ask three questions in sequence:

1. **Where does description quality change agent behavior, and where doesn’t it?** (Section 4, EXP-4 — a regime map assembled from every experiment run under the frozen protocol.)
2. **How common is the “it matters” regime in real, sampled MCP servers?** (Section 5, EXP-1 — a behavioral prevalence measurement on a pre-registered sample.)
3. **Can the regime be found cheaply, before deploying an agent against a real server?** (Section 6, EXP-3 — a pairwise confusability localizer, tested against behavioral ground truth.)

1.3 Contributions

- **A falsifiable, regime-bounded map** of where tool-description quality changes agent selection behavior. Two distinct conditions compose it, from different experiments: (i) a hand-written oracle *effect*, bracketed but not precisely located on the density axis (Section 4.2.1); (ii) realizing that effect safely through *automatic generation* additionally requires documented source code (Section 4.2.2). Outside this intersection the effect is null or negative (Section 4.3).

- **A prevalence measurement:** on a pre-registered pilot sample of 10 Python-only public MCP servers, 0 of 9 servers with a testable confusable family showed in-regime behavior, using the general behavioral regime definition (§5.1) — a lower bound, not a closed claim, on how rare the regime is among public, documented servers (Section 5).
- **A localizability boundary:** a pairwise LLM-judge confusability method — the natural cheap alternative to running the full behavioral protocol against every server — fails under two independently pre-registered framings, reaching the identical confusion matrix both times via two different degeneracy mechanisms rather than one repeated failure mode (Section 6). This extends the map from *where the regime occurs* to *where it can, and cannot, be found cheaply*.

1.4 Roadmap

Section 2 positions this work against tool-use benchmarks, MCP documentation practice, and retrieval-augmented tool selection. Section 3 fixes the evaluation protocol every experiment reuses. Section 4 presents the regime map: where the effect helps, and — as part of the same map, not a separate cautionary appendix — where it does nothing or actively harms. Section 5 presents the prevalence result. Section 6 presents the localizer result. Section 7 discusses the synthesis. Sections 8–9 cover threats to validity and the reproducibility artifact.

2. Related Work

2.1 Tool-use and function-calling benchmarks

Benchmarks for LLM tool/function-calling typically measure whether an agent calls the *correct* tool and constructs a *valid* call, usually against a fixed, hand-built catalog. Selection reliability is known to degrade as the tool menu grows — a synthetic causal-filtering study reports wrong-tool calls increasing with larger tool menus across four LLM backends (Babu and Iyer, “ToolChoiceConfusion: Causal Minimal Tool Filtering for Reliable LLM Agents,” arXiv:2606.06284, 2026) , and a large real-usage benchmark of tool-use “in the wild” finds no model, of 57 tested, exceeds 15% accuracy on tasks reflecting authentic multi-turn, mixed-intent user behavior (Yu et al., “Benchmarking LLM Tool-Use in the Wild” [WildToolBench], arXiv:2604.06185, 2026) . Neither benchmark isolates description *quality* specifically as the independent variable — which is why the regime map in Section 4 spans purpose-built synthetic catalogs at controlled scale (T18) alongside real production-server mirrors (RW1, RW2) rather than a single fixture designed for a different question.

2.2 MCP ecosystem and documentation practice

The Model Context Protocol ecosystem has produced a growing set of public servers with widely varying documentation quality (Hasan et al., 2026, above : 97.1% of a 856-tool sample shows at least one description “smell”), and a parallel set of tools (linters, description scorers, `llms.txt` conventions) that assume documentation quality is both measurable and improvable in a way that changes agent behavior. Section 4.3.4 and Section 6 directly test the measurability half of that assumption for one specific construct — can a scorer identify *which tool pairs* in a catalog are confusable — and find it fails at both the single-score and pairwise level as currently constructed, on the fixtures tested here.

2.3 Retrieval-augmented tool selection

Where a catalog is too large to place in-context, tool retrieval (lexical or embedding-based) is used to narrow the candidate set before selection. A large retrieval benchmark (7.6k tasks, 43k tools) finds even strong information-retrieval models perform poorly at this task — the best-performing model reaches only 33.83 nDCG@10 (Shi et al., “Retrieval Models Aren’t Tool-Savvy: Benchmarking Tool Retrieval for Large Language Models” [ToolRet], ACL 2025 Findings / arXiv:2503.01763) . One proposed remedy is LLM-based description *expansion*, reported to yield state-of-the-art retrieval gains on two benchmarks (Lu et al., “Tools are under-documented: Simple Document Expansion Boosts Tool Retrieval” [Tool-DE], arXiv:2510.22670, 2025) . Retrieval and direct-selection description quality are often assumed to move together on this basis — richer descriptions should help both. Section 4.3.3 shows they are anti-correlated *on the one synthetic catalog tested here*: the same behavioral-axis precision that helps direct-selection disambiguation matches poorly against the coarse, intent-level queries a retrieval index is queried with — a result that sits in tension with, rather than replicating, Tool-DE’s description-expansion gains, and that we do not claim generalizes beyond this fixture (see §8.1).

2.4 Positioning

This paper positions against the implicit claim, common across the practices surveyed above, that tool-description-quality improvement is a broadly-applicable, low-risk intervention. We do not dispute that it helps in *some* regime — Section 4.2 characterizes one such regime, bounded by the specific fixtures tested — but we show that regime is narrower and less precisely located than a flat “improve your descriptions” recommendation implies, and that the same intervention is risky outside it.

3. Methods — The Frozen Evaluation Protocol

All results in this paper are produced under a single protocol, committed once ([docs/research/frozen_protocol.md](#)) and never edited after any experiment’s results were collected. This section states it in full; individual experiment sections do not repeat it.

3.1 Governance

Every experiment’s spec — task set, gold labels, stability-screen procedure, and sidedness of the sign test — is pre-registered to its branch **before** any run starts. Any change to the judge, scorer, rubric, or calibration constants (“condition #1”) is escalated as a draft PR for human review before merge. Null and aborted results are reported as-is; results are never tuned after being observed. Judge, generator, and agent model families are structurally independent of any assistant used to author or orchestrate this research program — no shared credentials, infrastructure, or model family.

3.2 Frozen configuration

Component	Value
Judge model / seed	llama3.1:8b / 42
Generator model	qwen3:8b (always distinct from the judge)
Default agent	gemma2:9b
Trials per arm	5 (default); 3 for FRONTIER-T18 and EXP-3’s judge calls — each independently pre-registered and justified (API rate/cost limits for FRONTIER-T18; matching the codebase’s existing graded-judge convention for EXP-3)
Sign-test α	0.05, two-sided unless pre-registered one-sided
Headroom ceiling	0.85 — if Arm A (unmodified descriptions) already scores $\geq 85\%$ parse-success accuracy on the contested set, the experiment aborts as a null (no headroom), rather than proceeding to an A/B comparison
Minimum contested tasks	6

3.3 Classifier

Every trial is classified into exactly one of three outcomes, plus a separate flag:

- **SELECTED-CORRECT** — the agent selected the pre-registered gold tool.
- **SELECTED-WRONG** — the agent selected a different tool.
- **ABSTAINED-OR-HEDGED** — the agent produced output but hedged or explicitly abstained.
- **PARSE-FAILED** — the output could not be parsed into a structured selection at all.

PARSE-FAILED is always reported and never folded into the three-outcome denominator; a model or condition that appears to show “no effect” is checked against its parse-failure rate before that conclusion is accepted (Section 8.3).

3.4 Effect measurement

The unit of analysis is the **task**, not the trial. Arm A is run twice with independent seeds as a stability screen; a task whose per-run accuracy differs by more than one trial is dropped from the primary analysis and reported separately. Effect is Arm B accuracy minus Arm A accuracy on parse-success, stable, contested tasks; the sign test is computed per-task (B-beats-A vs. A-beats-B vs. tie), not per-trial.

3.5 Cross-experiment comparability

Cross-experiment effect-size comparisons (e.g., “does the T18 effect hold at a different capability tier?”) are valid only when the fixture, judge, classifier, and sign-test procedure are identical and the *only* varying factor is the one under test. The T18 gemma2:9b (+34.5pp) and FRONTIER-T18 Llama-3.3-70B (+40.8pp) figures reported in Section 4.2.3 are explicitly **not** part of a single controlled ladder — they come from different harnesses and hosts — and are never combined into a claim that the effect “grows with capability,” only that it “survives” at a higher capability tier tested independently. EXP-2, the controlled same-family capability ladder that would test this properly, was dropped from this paper’s scope (see Section 8.2).

4. EXP-4 — Regime Map

4.1 What EXP-4 consolidates

EXP-4 makes no new runs; it consolidates every experiment already run under the frozen protocol into a single map of where tool-description quality changes agent behavior and where it does not. Framing A treats this map, together with the prevalence finding in Section 5, as the paper’s primary artifact — the field’s assumption is not “false,” it is **regime-bounded**, and this section states the boundary precisely enough to be checked against a new server.

4.2 Where it helps

4.2.1 Confusable-at-scale (T18)

On a synthetic 60-tool catalog (10 families of 6 near-neighbor tools each, gemma2:9b agent), empty descriptions leave the agent to resolve tool selection from names alone: parse-success accuracy on the contested subset is **62.9%** (110/175). Oracle descriptions that target each family’s within-family distinguishing dimension (storage medium, operation scope, permanence, notification channel, computation type) raise this to **97.4%** (190/195) — a **+34.5pp** discrimination effect (task-clustered sign test: $n+=16$, $n-=0$, ties=24, $p<0.0001$). A second, mechanistically distinct effect appears alongside it: at this catalog density, empty descriptions also destabilize call *formation* itself — parse-failure rate falls from 12.5% (25/200) to 2.5% (5/200) under oracle descriptions, contributing roughly 5–6pp of the aggregate delta separately from the discrimination effect.

This effect appears **density-gated** rather than a general property of description quality, but the paper has only two points on the density axis, not a measured curve: an earlier fixture at 16 tools / 8 clusters (T17) placed Arm A at 81.2% accuracy from names alone — headroom too high (by the ad hoc 70% target that experiment used) for the oracle arm to be run at all, so the description effect was never actually tested at that density, not tested-and-absent. The 60-tool/10-family point is positive (+34.5pp); the 16-tool/8-cluster point is untested; nothing between 16 and 60 tools has been run. The defensible claim is that density gates *whether the effect is even testable* (headroom exists or doesn’t) at these two specific fixtures — not that density is *the* gating variable in general, and not that the transition point is located more precisely than “somewhere in this untested bracket.”

4.2.2 Under-documented source with docstrings (Q3 → Q6, Guard-B)

The T18 result establishes that oracle (hand-written, ground-truth) descriptions help; it does not establish that a *generator* can produce them safely. A four-stage progression (Q3, Q4, Q5, Q6) on a 12–23-tool fixture establishes both the recovery and safety conditions; the two endpoints carry the argument (full intermediate progression — including two distinct fabrication failure modes surfaced and closed along the way — in Appendix A.7):

Condition			Recovery structural tested tasks)	(6 p con-	No-fabrication
Interface-only, (Q2a/Q2b)	no	source	12.5% (11.1pp of 88.9pp available)	0.50 (n.s.)	n/a — nothing to fabricate from

Condition	Recovery structural tested tasks)	(6 p	No-fabrication
Q5/Q6-guarded: scoped source + docstrings + target-grounded prompt (Q6 = 23-tool blanket ap- plication)	100% (6/6)	0.0313 (Q5) / 0.0312 (Q6)	PASS, 0/11 regressions on already- passing tools (Q6)

Two findings compose into the deployment answer. First, **generation from the interface alone cannot recover the T18 gap**: two independent interface-only generation strategies (per-tool, Q2a; catalog-aware, Q2b) both recover only 12.5% of the oracle gain (11.1pp of 88.9pp available), not significant ($p=0.50$). The T18-decisive distinctions live in tool *behavior* — absent from both tool names and identical parameter schemas — and no amount of interface-level prompting recovers information that is not present in the interface. Second, **source access alone is not automatically safe**: whole-file source with docstrings stripped, and per-tool scoped source with docstrings present, each open a distinct fabrication failure mode (cross-tool symbol misattribution; docstring-body vocabulary gaps, respectively). The safe *and* fully recovering configuration — scoped source, docstrings retained, target-grounded prompting that forbids comparative neighbor claims (Guard-B) — was reached only at the fourth iteration (Q5), and confirmed to cause zero regressions when applied blanket across a 23-tool catalog including three collision-prone pairs (Q6).

4.2.3 Strong-agent survival (FRONTIER-T18)

A natural objection to Section 4.2.1 is that the effect might be an artifact of gemma2:9b’s specific capability level — a stronger agent might resolve the same catalog from names alone. We re-ran the identical T18 fixture and oracle descriptions against Llama-3.3-70B (OpenRouter), a substantially stronger open-weight agent, using a 3-outcome classifier and 3 trials per task. Arm A accuracy was **59.2%** (71/120) — comparable headroom to the gemma2:9b run — and Arm B (oracle) reached **100.0%** (120/120): an effect of **+40.8pp** (sign test $n+=19$, $n-=0$, ties=21, $p<0.0001$; on the stable set excluding 5 Arm-A trial-flippers, $n+=14$, $n-=0$, $p<0.001$). All 19 Arm-A miss tasks were fully recovered by the oracle, with no oracle-resistant floor in this fixture. The effect does not collapse, and does not shrink, moving from a 9B to a 70B agent.

This is a survival claim, not a growth claim, and not a frontier claim: Llama-3.3-70B is one strong open model, one data point, not a Claude/GPT-class proprietary frontier model, and the +40.8pp figure is **not directly comparable** to T18’s +34.5pp — different harness, different classifier host (Section 3.5). The result is **inconsistent with** “the T18 effect is a gemma2:9b-specific capability artifact” as an explanation; it is one additional data point, on a different harness, not a proof that no capability tier would show the effect shrink — and it does not close the question of whether a true proprietary frontier model would behave differently.

4.3 Where it doesn’t help, or actively harms

This is not a separate cautionary appendix — under Framing A it is the other half of the same map, and the boundary in Section 4.3 is only precise because these non-regimes were tested under the identical protocol.

4.3.1 No-headroom real servers (RW1, RW2)

Two production MCP-server mirrors — GitHub ([github/github-mcp-server](#), 21 tools, 5 confusable families, real docstrings) and AWS IAM ([aws-labs/mcp](#), 29 tools, 3 confusable families, including four-way attach/detach/user/group confusability) — were tested with real, unmodified docstrings. Both resolved **100%** of tasks correctly, including every pre-registered contested task (21/21 for GitHub; 29/29 including 12 contested for AWS IAM). There is no headroom for a description fixer to recover on either server: gemma2:9b resolves these tools from task-context wording alone, and the terse, one-sentence docstrings both servers already ship are sufficient. The regime identified in Section 4.2.1–4.2.2 requires headroom that these two real, well-documented servers simply do not have.

4.3.2 Harm on an already-resolved family (P2-A account_query)

On a 48-tool synthetic internal-proxy catalog built to test whether description improvement is uniformly safe across many confusable families, one family (`account_query`, 5 tools) shows the opposite of the T18/Q-series pattern. Under thin descriptions, the task’s SQL-like WHERE-clause phrasing heuristically matches `query_accounts` by name: **100%** (5/5) correct. Under both Guard-B-generated and hand-written oracle descriptions, more precise phrasing (“Executes a parameterized SQL-like WHERE clause”) introduces disambiguation noise among five semantically related tools, and accuracy **drops to 80%** (4/5) under both — a **−20pp harm**, reproducing at the same magnitude under two independent generation strategies, ruling out noise as the explanation. Two other families on the same catalog (`ticket_lifecycle`: delete/purge/archive/expire; `invoice_write`: update/upsert/patch/replace/amend) stay at 100% across all three arms — task-context phrasing already signals the dangerous distinction (“this cannot be undone”), and better descriptions add no incremental safety there either way. The one family that is *fully* recovered from a genuine failure (`order_read`, 0%→100% under both Guard-B and oracle) is a low-stakes return-shape disambiguation, not a high-stakes one. Aggregate across all 31 contested tasks (A=77.4%, Guard-B=96.8%, Oracle=93.5%) is directional but underpowered (sign test $p=0.07/0.125$); the `account_query` harm is the reproducible, per-family finding, and the aggregate number should not be read as a headline recovery rate.

4.3.3 Retrieval-readiness is anti-correlated, not neutral (F2)

If a catalog is too large to place in-context, tool selection is often mediated by a retrieval step (lexical or embedding-based lookup against description text) before the agent ever sees a candidate set. Testing the same P2-A catalog’s descriptions as retrieval documents against 93 underspecified natural-language queries, across three retriever types:

Arm	BM25 MRR	TF-IDF MRR	Embedding MRR (nomic-embed-text)
Thin	0.304	0.317	0.510
Guard-B	0.225 (−0.079)	0.204 (−0.113)	0.286 (−0.224)
Oracle	0.234	0.228	0.362

On this catalog, thin descriptions **outperform** Guard-B and oracle descriptions on every retriever tested, with the harm larger under semantic embedding retrieval than lexical retrieval. The mechanism is legible: compact verb-noun descriptions (“Get the order.”) keyword- and semantically-match coarse, intent-level queries (“get an order”) directly; precision-optimized descriptions (“Re-

turns order summary fields including status, total, item count...) match at the implementation level — exactly the level that helps within-family selection disambiguation (Section 4.2.1) and hurts coarse retrieval matching here. The property that makes a description good for one downstream use (direct selection in a fully-listed catalog) makes it worse for another (retrieval against a partial query) on this synthetic proxy — this is the central non-obvious finding connecting Sections 4.2 and 4.3, tested on one fixture and not yet checked against a real server’s retrieval behavior (§8.1).

4.3.4 Single-score judging cannot localize which pairs are confusable

Both real-server mirrors were also scored with AgentGauge’s existing `discoverability` judge, whose sub-score returns a single number per catalog. On GitHub, the score is flat at ~70/100 across every family, and does not flag either of the two families GitHub’s own maintainers consolidated to reduce confusion (0/2 overlap). On AWS IAM, the score is flat at ~68.7/100, and does not flag any of the three contested families (0/3 overlap); the accompanying name-collision heuristic instead flags verb-antonym pairs ([attach_user_policy](#)/[detach_user_policy](#)) that caused zero real confusion. A single number per catalog is structurally incapable of naming *which* pair is the problem — this is the motivating gap for the localizer tested in Section 6, not a calibration issue that more judge tuning would fix.

4.4 Description quality is multi-objective

This is the paper’s central synthesis, previewed in the Introduction and Abstract: the same behavioral-axis precision that helps within-family selection is not one general “better” direction for description quality — on the fixtures tested here, it is orthogonal or actively harmful to other things a description interacts with (what a context-rich agent needs; what retrieval rewards). Every hedge below is load-bearing, not decorative: each line names exactly which fixture the claim is scoped to.

Description precision HELPS within-family discrimination when (each tested at one fixture/point, not a swept curve):

- Catalog density is at least as high as the one positive point tested (§4.2.1), AND
- Source access includes docstrings (safe generation requires this; body-only source opens a fabrication vector), AND
- Arm A shows real headroom (the agent is not already resolving from task-context wording).

The same precision is ORTHOGONAL or ANTI-CORRELATED to, on the fixtures tested:

- what context-rich agents need (RW1, RW2, and P2-A’s high-stakes families all resolve at 100% from context alone, regardless of description quality),
- what retrieval rewards on one synthetic catalog (F2: harm across all three retriever types tested there), and
- what a single-score judge can report (localization requires a pairwise comparison, tested next in Section 6).

5. EXP-1 — Server-Population Prevalence

5.1 Question and regime definition

Section 4 establishes *where* the effect exists; it does not establish *how common* that regime is among real MCP servers. We define “in-regime” behaviorally, not by any static description-quality property (which would be circular): a confusable family on a server is in-regime iff, under the frozen protocol, (a) the server’s real shipped descriptions fail at least one contested task, **and** (b) an oracle description recovers it. “Thin descriptions” is an input property; only behavior the fix actually repairs counts as in-regime.

This is the general behavioral construct, not a re-test of Section 4.2.1’s specific density point. None of the ten sampled servers were selected or verified to actually reach the 60-tool/10-family density where T18 found an effect (most have far fewer tools; the one large-catalog server sampled, at 116 tools, failed for an unrelated reason — catalog overwhelm, malformed output under a large listing, not confident wrong-tool selection — and contributes no evidence either way about the density regime specifically). What EXP-1 measures is the prevalence of the general two-condition behavioral regime across whatever confusable families each server happens to have, at whatever scale it happens to be built at — a broader and different question than “how many real servers are built at T18’s specific tested density.”

5.2 Sampling frame

The pre-registered, ratified frame is **N=10, Python-only, doc-density-stratified**, sourced from public GitHub; 11 non-Python servers were dropped because the generic regex fallback extractor used for non-Python source proved unreliable on them (full revision history in Appendix A.6). This is a narrower claim than “public MCP servers” — it is “Python MCP servers on GitHub, N=10 pilot” — stated as such everywhere this number appears in this paper.

5.3 Result

Server	Tier	Arm A	Arm B (oracle)	Effect	Verdict
github-mcp (RW1, anchor)	anchor	100% (21/21)	—	n/a	OUT-OF-REGIME
aws-iam-mcp (RW2, anchor)	anchor	100% (29/29 incl. 12 contested)	—	n/a	OUT-OF-REGIME
lucasastorian-llmwiki	near_empty	100%	—	n/a	OUT-OF-REGIME
stefanoamorelli-sec-edgar-mcp	thin	100%	—	n/a	OUT-OF-REGIME
stickerdaniel-linked-in-mcp-server	thin	100%	—	n/a	OUT-OF-REGIME
mrexodia-ida-pro-mcp	near_empty	90%	—	n/a	OUT-OF-REGIME (above headroom ceiling)
AminForou-mcp-gsc	well_documented	75%	75%	+0pp	OUT-OF-REGIME (real functional overlap; no description fixes it)
taylorwilsdon-google_workspace_mcp	thin	0%	0%	+0pp	OUT-OF-REGIME (catalog-overwhelm failure mode, not confusability)

Server	Tier	Arm A	Arm B (oracle)	Effect	Verdict
datalayer-jupyter-mcp-server	near_empty	70%	55%	−15pp	OUT-OF-REGIME (HARM — replicates the P2-A account_query pattern on a fresh real server)
Dataojitori / blazickjp-arxiv / LycheeMem	well_documented	—	—	—	no testable confusable family (3 servers)

Headline: 0 of 9 servers with a testable confusable family show in-regime behavior (7 freshly scored + 2 RW1/RW2 anchors cited from Section 4.3.1). Per fresh-only tier: well_documented 0/1 testable, thin 0/3, near_empty 0/3 — a clean null across every tier of this pilot. Three of the ten servers had no genuinely confusable family at all (mechanical clustering found none, or the one candidate was rejected on manual review as not genuinely confusable).

5.4 The seed-bug episode

A first trial run passed a single fixed seed (`seed=42`) to every one of five nominal repeated trials per task, instead of the codebase’s established `seed=42+trial_idx` convention — meaning the first run sampled zero real trial-to-trial variance. That run reported two servers as in-regime (`mrexodia-ida-pro-mcp` +50pp; `datalayer-jupyter-mcp-server` +25pp). Fixing the seed and re-running **reversed both findings**: ida-pro’s real Arm A accuracy is 90% (correctly aborts, no headroom), and jupyter’s Arm B is a genuine harm (−15pp, not a +25pp recovery). Both apparent in-regime results were seed artifacts, caught before being reported as findings — a credibility asset for the pipeline’s own error-detection, reported here as such.

This asset has a mirror-image limitation that is *more* important than the credit above, and is given its own standalone treatment, not folded into this paragraph: **the note opening §8** states why a bug that suppressed a real in-regime signal, rather than manufacturing a false one, would not have been caught the same way.

5.5 Scope

Public servers skew documented relative to the internal/custom long tail; this prevalence figure is a **lower bound** on how common the regime is, not a closed population estimate. We do not claim the regime “does not occur” — we claim it is rare in this sampled population of public, source-available Python MCP servers as of the frame’s fixed date, and that the under-documented internal segment this bounds is, by construction, unsampleable from public data.

6. EXP-3 — Confusability Localization

6.1 Motivation

Section 4.3.4 established that the existing single-score **discoverability** judge cannot say *which* tool pair in a catalog is confusable — a structural limit of returning one number per catalog. If the regime Section 4 characterizes is to be found cheaply, before running the full behavioral protocol against every server, a localizer is needed that outputs a per-pair signal. This section builds and tests the natural next attempt: ask the same frozen judge a pairwise question directly.

6.2 Method

For each candidate tool pair (A, B), one judge call (llama3.1:8b, frozen) asks whether a task intended for A could plausibly select B instead, and vice versa. Ground truth is a pre-registered, 24-pair fixture (4 CONFUSED / 20 NOT_CONFUSED) built entirely from already-collected behavioral trial data — 6 fresh servers from Section 5’s EXP-1 scoring (14 pairs), plus **10** pairs from the RW1/RW2 anchors (5 GitHub + 5 AWS IAM), including several pairs the *old* Levenshtein-based heuristic had already false-positived (a deliberately adversarial inclusion). 3 trials per pair (seed 42+trial_idx), majority vote. The pre-committed bar for “a real positive method”: precision ≥ 0.50 **and** recall ≥ 0.50 .

6.3 Binary result

Confusion matrix: TP=4, FP=20, FN=0, TN=0. **Precision = 0.167, recall = 1.00** — recall clears the bar; precision does not. All **24 of 24** pairs were verdicted CONFUSABLE, including all **ten** sampled pairs from the two 100%-accuracy anchor servers (GitHub, AWS IAM) and both pairs where the old heuristic already false-positived. The judge catches every real behavioral confusion (perfect recall) by flagging almost everything — a direct yes/no confusability question elicits a near-uniform “yes” from this judge on this fixture, not a discriminating per-pair signal.

6.4 Graded-confidence retry

One further, pre-registered, time-boxed retry (ratified before running, per the “is this a framing artifact” question) replaced the binary question with a 0–10 confidence score, using the same aggregation and ground truth, threshold ≥ 5.0 (scale midpoint, fixed before any result was seen). Result: **every one of the 24 pairs’** mean score landed in a **5.00–5.67 band** — identical confusion matrix to the binary run (precision 0.167, recall 1.00). This is a *different* degenerate failure mode from the binary run’s uniform-yes: rather than defaulting to one categorical answer, the judge anchors near the numeric midpoint of its own scoring guide regardless of input — genuinely confused pairs and 100%-resolved anchor pairs score in the same narrow band.

6.5 Conclusion

The negative is robust to framing, not an artifact of question phrasing: two structurally different question formats (categorical yes/no; graded 0–10 confidence) produce the same confusion matrix via two different degeneracy mechanisms. **Pairwise judging, binary or graded, with this frozen judge, is not a usable per-pair confusability signal on this fixture.** Per pre-registration, this is a hard stop — no third variant was attempted. We note the scope of what this rules out: the experiment validates the *judging* step only, given a candidate pair; it does not test an automatic candidate-*generation* mechanism (which pairs to even consider from a full catalog), and two of the four real confusions in the ground truth cross mechanical name-prefix family boundaries that a family-scoped candidate generator would not have proposed in the first place.

This is itself a boundary finding, not a second defeat stacked on Section 4’s negatives. Section 4 established that the helps-regime is real and characterizable at the catalog level (density, source access, headroom). This section establishes a further, independent boundary: even where the regime is real, it is not currently *findable* below the cost of running the full behavioral protocol, using the one localization approach natural to try with an existing frozen judge. That is new information the map did not have before this experiment — it narrows where a cheap pre-

deployment check can substitute for the protocol, which is a distinct question from whether the regime itself exists.

7. Discussion

7.1 Synthesis

Read together, Sections 4–6 answer the paper’s title question precisely rather than broadly. Tool-description quality *does* change agent selection behavior — but the evidence for this comes from separate findings that compose, not one single joint condition: an oracle-effect bounded to one tested catalog density (§4.2.1), a generation-safety condition that requires documented source (§4.2.2), and a prevalence measurement, on the general behavioral regime construct (§5.1), that a pre-registered pilot sample of real public Python servers places at 0 of 9 tested (Section 5). Separately, even where the regime does occur, it has not been shown findable cheaply: a pairwise LLM-judge localizer fails to identify which specific tool pairs are confusable on a new, unscored server (Section 6). Outside the helps-conditions, the same intervention that helps direct selection in the tested fixtures can be neutral, harmful to selection on at least one already-resolved family, or harmful to retrieval on the one synthetic catalog tested — three independently-measured non-regimes, not one, and not (yet) shown to generalize beyond the fixtures each was measured on.

7.2 A diagnostic practitioners can use today: the Two-Condition Regime Test

Section 5.1’s regime definition is not only an analysis tool for this paper — the same two-step procedure is directly usable, unchanged, as a pre-investment check on a given MCP server:

The Two-Condition Regime Test

1. **Fail** — Does the agent fail at least one contested task under the server’s real, currently-shipped descriptions?
 2. **Recover** — If it fails, does a hand-written, ground-truth (oracle) description recover it?
- (1) is NO → the agent already resolves the task from context; description tooling has nothing to fix here.
 - (1) is YES but (2) is also NO → the failure is not description-shaped (e.g., genuine functional overlap between tools, or a catalog-overwhelm failure mode); description tooling will not help either.
 - (1) and (2) are both YES → this server’s family is inside the regime this paper’s Section 4.2 findings speak to.

This is exactly the check used to produce every OUT-OF-REGIME/IN-REGIME verdict in Section 5’s table — the check *this paper used*, not a separately validated general instrument. It has been exercised on ten Python MCP servers plus two anchors (Section 5); nothing about the check’s own reliability has been tested on a larger or non-Python population, and naming it is meant to make it adoptable, not to claim it has been validated beyond what this pilot supports. Applying it does not require re-running this paper’s full protocol at T18 scale — steps (1) and

(2) are exactly the frozen-protocol headroom gate and oracle A/B (§3), run at whatever scale the server in question actually has.

The practical implication for MCP server authors and tool-catalog builders is not “descriptions don’t matter” — it is “check whether you are in the regime before investing in description tooling,” where “the regime” is checked behaviorally via the test above, rather than assumed from catalog size alone. A server whose agent already resolves tasks from context gains nothing from more precise descriptions and risks the `account_query`-style harm pattern (Section 4.3.2) if it applies them blanket. A server that *is* in the regime should generate descriptions from documented source with a target-grounded, non-comparative prompt (Section 4.2.2’s Q5/Q6 configuration), not from the interface alone.

7.3 What would change this picture

A larger or non-Python-verified EXP-1 sample (Section 5.5); a true proprietary-frontier-model replication of Section 4.2.3; or a localizer signal that does not route through this frozen judge’s pairwise-question failure mode (Section 6.5) — for example, a signal derived from name-embedding distance plus schema overlap rather than an LLM judgment call.

8. Threats to Validity / Limitations

Full list: [docs/paper/threats_to_validity.md](#). Summarized here by category; every item there traces to a specific finding already reported in Sections 4–6, not a hedge added at write-up time. **The note opening §8 is read first** — it is the single most important threat to a paper whose headline results are nulls, and it is stated standalone, not as a clause inside another paragraph.

The false-negative asymmetry — the epistemic bound on this paper’s null claims (read this one first)

Every null and boundary claim in this paper — EXP-1’s 0-of-9 headline, EXP-3’s localizer-fails headline — is bounded by one asymmetry in what this pipeline’s error-detection can catch, and this section states that bound precisely. Section 5.4 reports two false positives from a seed-configuration bug, caught and reversed before publication *because* they were surprising enough to trigger a recheck (a server unexpectedly showing in-regime behavior got a second look). That same mechanism has a mirror-image blind spot: a bug that instead silently *suppressed* a real in-regime signal — turning a true positive into a false null — would produce a result indistinguishable from a correctly-measured null: “this server is not in-regime” looks the same whether it’s true or an artifact, and nothing about a null result prompts the same “that’s surprising, let’s recheck” response that caught the two false positives. We have not found evidence of such a bug; we also have no mechanism that would necessarily surface one, which is precisely the point — this is the bound this paper’s nulls should be read against, not a suggestion that they are unreliable. Any reader treating EXP-1 or EXP-3’s null as more secure than “we looked and did not find a positive, using a pipeline whose false-positive-catching capacity is demonstrated and whose false-negative-catching capacity is not” should recalibrate against this paragraph specifically.

8.1 Sampling / generalizability

N=10, Python-only, public-GitHub pilot. The strength claimed is *convergence* — the EXP-1 null, the RW1/RW2 anchors, the P2-A `account_query` harm, and the doc-density-rarity pattern across tiers all point the same direction independently — not N=10 in isolation. Non-Python mechanical extraction was dropped as a capability limit of the method, not a data-availability inconvenience. Tier labels (`well_documented`/`thin`/`near_empty`) are relative ranks within the sampled pool, not absolute bands.

8.2 Agent / model scope

One default agent (gemma2:9b) across the regime map, Q-series, RW1/RW2, and P2-A. The controlled capability ladder that would test agent-capability sensitivity directly (EXP-2) was dropped from this paper’s scope — ratified, justified by the regime already being uncommon in the sampled population (a ladder over a rare regime has limited external relevance). FRONTIER-T18 is one model, one data point, not apples-to-apples with the gemma2:9b harness, and not a proprietary-frontier-model result.

8.3 Measurement / judge validity

Two seed-bug false positives were caught and reversed before reporting (Section 5.4; asymmetry discussed prominently in §8’s opening note, not here). EXP-3’s negative is robust across two framings, not a single-question artifact, but validates the judging step only, not candidate-pair generation. Trial counts deviate from the 5-per-arm default for FRONTIER-T18 and EXP-3 (3 each), independently pre-registered and justified.

8.4 Harm / asymmetric risk

Blanket description-fixing is not universally safe: P2-A’s `account_query` family shows reproducible –20pp harm under two independent generation strategies. Do-no-harm testing (Q6) covers only the case where honest descriptions remain semantically distinct across sibling tools; total description collapse to identical phrasing is explicitly untested.

8.5 Reproducibility-artifact note

The FRONTIER-T18 result (Section 4.2.3) required, during this paper’s preparation, recovering a raw result file that existed only as a gitignored local artifact and independently re-deriving its numbers before committing it as a hashed fixture (Section 9.2). This is now resolved for the reported number; the harness code that produced it is now merged to `main` as well (Section 9.2).

9. Reproducibility Artifact

9.1 Claim

Every figure in Sections 4–6 traces to a committed file on this paper-writing branch, verified against a specific commit reachable from `HEAD` (full trace: [docs/paper/evidence_table.md](#)). Most fixture files carry SHA-256[:12] hashes recorded at pre-registration time, before their experiment’s results were known (protocol appendix: [docs/research/frozen_protocol.md](#)). **The FRONTIER-T18**

fixtures are the stated exception — see §9.2 immediately below: their hashes were computed post-hoc, during this paper’s preparation, not at pre-registration time, because the original pre-registered artifact had to be recovered from a gitignored local file rather than read from a tracked one.

9.2 FRONTIER-T18 data/code split (state plainly, not silently)

The result data underlying Section 4.2.3 is committed and independently re-derivable: `evals/fixtures/frontier_t18_step2_result.json` (per-task, sha256[:12]=3ca4a25dbd25) and `evals/fixtures/frontier_t18_step2_raw_calls.json` (all 240 raw per-call LLM responses, sha256[:12]=93fb0d77262) — re-counting SELECTED-CORRECT directly from the raw calls reproduces 71/120 (Arm A) and 120/120 (Arm B) exactly. The **harness code** that produced these files (`agentgauge/frontier.py`, `scripts/run_frontier_t18.py`) is now committed to `main` alongside the data — a from-scratch re-run of this experiment is possible directly from this repository, with no further merge required.

9.3 Governance

All research-program branches route through draft, human-reviewed PRs; any change touching the judge, scorer, rubric, or calibration constants is escalated before merge (Section 3.1). No PR in this research program has auto-merge; a human merges every one.

10. Conclusion

Tool-description quality changes agent tool-selection behavior in a real, measurable, and mechanistically legible way — but only inside a narrow, checkable set of conditions: a catalog at least as dense as the one point where an effect was tested and found (§4.2.1), and generation with access to documented source. That effect does not collapse under a substantial capability increase in the one additional agent tested (Section 4.2.3) — a finding *inconsistent with* “this is a gemma2:9b-specific capability artifact,” though one data point on a different harness does not prove the effect is capability-independent in general. Outside these conditions: the same behavioral regime construct (§5.1) appears in 0 of 9 servers with a testable confusable family in a pre-registered pilot sample of real, public, documented Python MCP servers (Section 5); the same kind of intervention harms selection on at least one already-resolved family tested, while two other already-resolved families in the same experiment show no harm (Section 4.3.2); it harms retrieval across every retriever type tested on the one synthetic catalog used for that test (Section 4.3.3); and the regime cannot currently be located cheaply by asking an LLM judge pairwise whether two tools are confusable (Section 6) — itself a boundary finding, not a second defeat (§6.5). One threat bounds all of these null and boundary claims at once and is not a footnote: a bug that silently suppressed a real in-regime signal would look identical to a correctly-measured null, and this paper’s demonstrated error-detection catches false positives, not false negatives (§8, opening note). With that caveat stated plainly, the field’s assumption that description-quality improvement is a broadly beneficial, low-risk lever reads, on this evidence, as over-general rather than wrong: this paper’s contribution is a first pass at the boundary that makes “broadly” precise enough to check and re-test.

Appendix

- **A.1** Full cross-experiment regime table: [docs/research/exp4_regime_map.md](#) §Cross-Experiment Regime Table.
- **A.2** Frozen judge/generator prompts (Guard-B target-grounded prompt; pairwise binary/graded confusability prompts): [agentgauge/fixer.py](#) (`_DESC_GENERATOR_GUARD_B_PROMPT`), [docs/research/exp3_pre_registration.md](#) §2, §7.
- **A.3** Fixture hash manifest: generate per [docs/research/frozen_protocol.md](#) Appendix; the two FRONTIER-T18 hashes added during this paper’s preparation are recorded in Section 9.2 above and in [docs/paper/evidence_table.md](#) §1.3.
- **A.4** Per-server EXP-1 table: Section 5.3 above (reproduced from [STATUS.md](#) EXP-1 section).
- **A.5** Bibliography — every citation below was independently fetched and checked against its claimed content during this paper’s preparation (title, authors, and the specific figure/quote cited all confirmed against the primary source, not copied from a prior internal desk-research doc without re-verification). None were invented; where a candidate citation from prior internal research ([reports/frontier_phase1_research.md](#), itself gitignored/uncommitted and self-flagged as “arXiv IDs not independently re-resolved”) could not be verified to support its claimed content, it was dropped rather than included on trust — see the exclusions list below.

Cited in this paper:

1. Anthropic Engineering, “Writing tools for agents” (publication date not independently confirmed during this paper’s preparation — content verified, date not). <https://www.anthropic.com/engineering/writing-tools-for-agents> — verified: contains the quoted claim that small tool-description refinements yielded SOTA SWE-bench Verified results for Claude Sonnet 3.5.
2. GitHub Engineering, “Measuring what matters: how offline evaluation of GitHub MCP Server works” (publication date not independently confirmed during this paper’s preparation — content verified, date not). <https://github.blog/ai-and-ml/generative-ai/measuring-what-matters-how-offline-evaluation-of-github-mcp-server-works/> — verified: contains the quoted claim that small description edits “shift results a lot.”
Second-pass audit note: this is a claim about the *same* GitHub MCP server mirrored in RW1 (§4.3.1), which found zero headroom for gemma2:9b on that server. The tension is flagged explicitly in §1.1 body text, not silently juxtaposed; not resolved (harness/model/metric details of GitHub’s own eval were not verified during this paper’s preparation).
3. Hasan, M. M., Li, H., Rajbahadur, G. K., Adams, B., and Hassan, A. E., “Model Context Protocol (MCP) Tool Descriptions Are Smelly! Towards Improving AI Agent Efficiency with Augmented MCP Tool Descriptions,” arXiv:2602.14878, 2026. Verified via arxiv.org/abs/2602.14878 — 856 tools / 103 servers, 97.1% ≥ 1 “smell,” 56% unclear purpose, +5.85pp median task-success gain from augmentation, 16.67% regression rate, +67.46% execution-step inflation, all confirmed against the abstract.
4. Lu, X., Huang, H., Meng, R., Jin, Y., Zeng, W., and Shen, X., “Tools are under-documented: Simple Document Expansion Boosts Tool Retrieval” (Tool-DE), arXiv:2510.22670, 2025 (arXiv ID year prefix “25” = 2025 — corrected in second-pass audit; originally

- misabeled 2026). Verified via arxiv.org/abs/2510.22670 — Tool-Embed/Tool-Rank, SOTA retrieval gains from LLM-driven document expansion, confirmed against the abstract.
5. Shi, Z., Wang, Y., Yan, L., Ren, P., Wang, S., Yin, D., and Ren, Z., “Retrieval Models Aren’t Tool-Savvy: Benchmarking Tool Retrieval for Large Language Models” (ToolRet), ACL 2025 Findings (aclanthology.org/2025.findings-acl.1258) / arXiv:2503.01763. Verified via web search cross-referencing the ACL Anthology and arXiv listings — 7.6k tasks / 43k tools, best model (NV-embed-v1) 33.83 nDCG@10, both figures confirmed.
 6. Yu, P., Liu, W., Yang, Y., Li, J., Zhang, Z., Feng, X., and Zhang, F., “Benchmarking LLM Tool-Use in the Wild” (WildToolBench), arXiv:2604.06185, 2026. Verified via arxiv.org/abs/2604.06185v1 — confirmed: “no model achieves an accuracy of more than 15%” across 57 evaluated LLMs. **Not cited: a specific “Grok-4 24.07% wrong-name error” figure** from the prior internal desk-research doc — this could not be confirmed from the abstract and full-text confirmation was not obtained; excluded rather than asserted on trust.
 7. Babu, R. S., and Iyer, L. G., “ToolChoiceConfusion: Causal Minimal Tool Filtering for Reliable LLM Agents,” arXiv:2606.06284, 2026. Verified via arxiv.org/abs/2606.06284v1 — confirmed: Causal Minimal Tool Filtering (CMTF), 102 tasks / 100 tools / four LLM backends, reliability-vs-tool-menu-size problem statement. **Not cited: the prior internal doc’s characterization of this paper’s evaluation as “fully synthetic” with an explicit disclaimer of “no empirical measurement of how frequently this confusion regime occurs in practice”** — this specific framing/quote could not be confirmed from the fetched abstract; excluded rather than asserted on trust.
 8. GitHub issue, “Duplicate tool names across MCP servers cause errors,” github.com/openai/openai-agents-python#464 (closed). Verified via `gh api` — title and closed state confirmed.
 9. GitHub issue, “Feedback for dynamic tool selection,” github.com/github/github-mcp-server#275 (closed). Verified via `gh api` — title and closed state confirmed; referenced only in this appendix as supporting context, not cited in body prose in this draft.

Checked and excluded (candidate citations from prior internal desk research that did NOT verify — listed so the exclusion is visible, not silent):

- “arXiv:2603.20313, retrieval quality fundamentally bounded by the informativeness of tool descriptions” — fetched; the paper at this ID (“Semantic Tool Discovery for Large Language Models: A Vector-Based Approach to MCP Tool Selection”) does not support this claim per its abstract. Not used anywhere in this paper.
 - Any secondary-source selection-accuracy-vs-tool-count percentages (e.g., “>90% at 5-7 tools → ~13% at 100+ tools”) from [docs/research/phase1-buyer-and-landscape.md](https://docs.research.phase1-buyer-and-landscape.md) — that document’s own “Confidence & gaps” section already flags these as “secondary-source paraphrase, not verified against a primary benchmark.” Not used in this paper for the same reason its original author already gave.
- **A.6 EXP-1 sampling-frame revision history** (moved here from §5.2 body — the finding that survives in §5.2 is unaffected; this is provenance, not evidence). The pre-registered frame was rebuilt five times over the course of the experiment, each rebuild escalated and ratified before proceeding, never silently:

- **v1** — star-stratified GitHub-topic pool.
 - **v2** — doc-density-stratified, N=23.
 - **v3–v4** — three, then a fourth, confirmed Python-AST-extractor bug found and fixed while preparing the trial batch.
 - **v5 (final, ratified)** — a systematic audit found that the generic regex fallback used for every non-Python server pulls in systemic noise (template literals, parameter names, category labels, unrelated example data) — a capability limit of blind text-proximity matching, not a fixable parsing bug. All 11 non-Python servers were dropped, yielding the final N=10, Python-only frame reported in §5.2. Full commit-level trail: [STATUS.md EXP-1](#) section, “Frame history” paragraph; ratification commit 538affe; [docs/paper/evidence_table.md](#) §2.
- **A.7** Full Q3→Q6 source-aware generation progression (moved here from §4.2.2 body — both prose findings in §4.2.2 are unaffected; this is intermediate detail, not a third finding). The two-row table in §4.2.2 shows only the endpoints (interface-only failure; Q5/Q6 safe-and-recovering). The full four-stage progression, showing *how* Q5/Q6 was reached and the two distinct fabrication failure modes it closes:

Condition	Recovery (6 structural con- tested tasks)	p	No-fabrication
Q3 F-DOC (whole-file source + docstrings)	83.3% (5/6)	0.0625 (marginal)	PASS
Q3 F-BODY (whole-file, docstrings stripped)	83.3% (invalidated)	—	FAIL — cross-tool source misattribution
Q4-DOC-scoped (per-tool source + docstrings)	100% (6/6)	0.0313	FAIL — docstring-body vocabulary gap
Q4-BODY-scoped	100% (6/6)	0.0313	PASS
Q5-guarded (per-tool source + docstrings + target-grounded prompt)	100% (6/6)	0.0313	PASS
Q6-guarded (23-tool blanket application)	100% (6/6)	0.0312	PASS, 0/11 regressions on already-passing tools

Reading order: Q3 (whole-file) shows docstrings are safe there but only marginally recovering, and body-only is unsafe (cross-tool misattribution). Q4 (scoped) fixes the cross-tool issue but *inverts* the safety finding — scoped docstrings now fabricate via a docstring-body vocabulary gap, while scoped body-only is safe but discards the docstring signal. Q5 (guarded prompt) resolves the inversion: scoped + docstrings + a target-grounded, non-comparative prompt is safe AND 100% recovering. Q6 confirms this holds blanket across a larger, 23-tool catalog with zero regressions. Full narrative: [docs/research/exp4_regime_map.md](#) §Regime 2; [STATUS.md](#) Q3/Q4/Q5/Q6 sections; [docs/paper/evidence_table.md](#) §1.2.

References

- Anthropic Engineering. Writing tools for agents. <https://www.anthropic.com/engineering/writing-tools-for-agents>, 2026. Publication date not independently confirmed by this paper’s authors; content verified against the primary source.
- Ramesh S. Babu and Lakshmi G. Iyer. ToolChoiceConfusion: Causal minimal tool filtering for reliable LLM agents, 2026.
- GitHub Engineering. Measuring what matters: How offline evaluation of GitHub MCP server works. <https://github.blog/ai-and-ml/generative-ai/measuring-what-matters-how-offline-evaluation-of-github-mcp-server-works/>, 2026. Publication date not independently confirmed by this paper’s authors; content verified against the primary source.
- github/github-mcp-server contributors. Feedback for dynamic tool selection. <https://github.com/github/github-mcp-server/issues/275>. GitHub issue #275 (closed); publication date not independently confirmed by this paper’s authors; cited only as supporting context, not in body prose.
- Mohammad Masudur Hasan, Hao Li, Gopi Krishnan Rajbahadur, Bram Adams, and Ahmed E. Hassan. Model context protocol (MCP) tool descriptions are smelly! towards improving AI agent efficiency with augmented MCP tool descriptions, 2026.
- Xinyi Lu, Haocheng Huang, Rui Meng, Yifan Jin, Wei Zeng, and Xinyu Shen. Tools are under-documented: Simple document expansion boosts tool retrieval, 2025. Referred to in text as Tool-DE.
- openai/openai-agents-python contributors. Duplicate tool names across MCP servers cause errors. <https://github.com/openai/openai-agents-python/issues/464>. GitHub issue #464 (closed); publication date not independently confirmed by this paper’s authors.
- Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Suzan Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren’t tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. Referred to in text as ToolRet; also available as arXiv:2503.01763.
- Peng Yu, Wei Liu, Yue Yang, Jing Li, Zihan Zhang, Xin Feng, and Fan Zhang. Benchmarking LLM tool-use in the wild, 2026. Referred to in text as WildToolBench.