The Recognition-Exposure Gap: Why Upgrading Intent Detection Can Reduce Tool Availability in LLM Agents

Yongggang Weng

Universit Teknologi Malaysia

weng@graduate.utm.my

Abstract

Tool-rich LLM agents increasingly gate tool visibility by intent-detection confidence: a narrow tool set is exposed when the agent is confident about the user's intent, and a broad fallback set is exposed otherwise. This "confidence-tiered exposure" is widely assumed to be monotone in detector quality—a better intent detector should make the correct tool *more* available. We report a counterintuitive failure that violates this assumption. In a production desktop agent (WeClaw, 119 registered tools), replacing a rule-based intent detector with an LLM-based one raised recommendation accuracy to 100% (17/17 seed queries) yet *lowered* real tool-exposure coverage to 52.9% (9/17): in 47.1% of cases the detector named the correct tool but the tool never entered the function-calling schema set, so the model could not call it on the first attempt. We name this failure the **Recognition-Exposure Gap** and trace it to a structural cause—confidence-to-tier mapping and recommended-tool-to-exposure-set construction are two *decoupled* stages, so a detector upgrade that shifts the confidence distribution can silently drop correctly recognized tools. Critically, the gap is not a harmless offline artifact: an execution-level replay that faithfully drives the production failure-escalation ladder shows the gap costs 16 failed round-trips and 8 tier escalations across the affected queries, inflates the exposed tool set by 136% (mean final-tier exposure 8.9 → 21.0), and adds ≥31.2 s of cumulative interaction latency—a "correct, but late and expensive" recovery. A zero-training co-design fix that merges recognized tools into the exposure set closes the gap completely (first-attempt reachability 52.9% → 100%, failed round-trips 16 → 0, escalations 8 → 0), and *reduces* rather than increases exposure. We position the Recognition-Exposure Gap as the first quantified tool-exposure instance of the "coupling bottleneck" between model and harness, and abstract a general failure condition—confidence-tiered exposure plus decoupled recommended-tool injection—that is detector- and framework-independent.

Keywords: tool exposure; LLM agents; function calling; intent detection; agent harness; recognition-exposure gap

1. Introduction

As LLM agents gain access to hundreds of tools, exposing all of them at once becomes prohibitively expensive: full-schema prompts inflate token cost, latency, and selection error. A common remedy is **confidence-tiered exposure**—the harness runs an intent detector over the user's request, maps the detector's confidence to a visibility tier (a narrow *recommended* set at high confidence, a broader *extended* or *full* set otherwise), and exposes only that tier's tools to the model's function-calling interface. This design is explicitly "confidence-gated" in recent systems (Franko, 2025; Singla et al., 2026) and underlies a broad "route before reasoning" family (Gan et al., 2025; Paramanayakam et al., 2024).

The design carries an implicit assumption: **exposure quality is monotone in detector quality**. If the intent detector improves—say, from keyword rules to an LLM—the correct tool should become *more* reachable, not less. This paper shows the assumption can fail silently. In WeClaw (Weng, 2026b), a production daily-use desktop agent with 119 registered tools, we upgraded the intent detector from a rule-based matcher to an LLM-based classifier. The upgrade behaved exactly as intended at the recognition stage: on a set of colloquial, ambiguous user queries, the LLM recommended the correct tool in **17 of 17** cases (100%). Yet the fraction of those queries for which the recommended tool actually appeared in the function-calling exposure set was only **52.9% (9/17)**. In the remaining **47.1%**, the agent had correctly recognized what the user needed and then failed to make the corresponding tool callable. We call this divergence the **Recognition-Exposure Gap**.

The cause is structural, not a coding mistake. Confidence-tiered exposure is implemented as two stages: **Stage A** maps detector confidence to a tier; **Stage B** constructs the concrete tool set for that tier. When the detector is upgraded, its confidence distribution shifts—in our case the LLM returns a fixed high confidence (0.9) on success, collapsing every ambiguous query into the *narrowest* tier. Stage B, however, still builds that tier's tool set from the *rule-driven* intent-to-tool mapping, which is empty for colloquial queries; the LLM's recommended tools are recorded as annotations and prompt hints but are **not merged into the exposure set**. The two stages are decoupled: a recognition improvement changes Stage A's output without propagating to Stage B's construction, so correctly recognized tools are silently dropped. Recognition accuracy, the metric everyone watches, cannot see this.

The gap is also not a benign offline curiosity. We build an execution-level replay that faithfully drives the production failure-escalation ladder (a tool call that cannot be resolved triggers report_failure, which after a threshold escalates the tier and re-exposes tools). On the affected queries, the model's first function-call attempt fails because the recommended tool is absent; recovery happens only after the escalation ladder widens the exposure set. The measured runtime cost is concrete: **16 failed round-trips** and **8 tier escalations** across the eight gap queries, a **136%** inflation of the exposed tool set (mean final-tier exposure 8.9 → 21.0), and **≥31.2 s** of cumulative added interaction latency. The agent eventually does the right thing—but late, and at a cost. We then show a **zero-training co-design fix**: merging recognized tools into the exposure set at construction time closes the gap entirely (first-attempt reachability 52.9% → 100%, failed round-trips 16 → 0, escalations 8 → 0) and, because it avoids escalation to the broad tier, *reduces* mean exposure rather than increasing it.

Contributions. (1) We name and quantify the Recognition-Exposure Gap, a failure mode localized at the *interface between two harness modules* rather than in the model's representations, and show it occurs even when recognition is 100% correct. (2) We upgrade the offline counterfactual measurement to an execution-level trigger chain that quantifies the gap's runtime cost, and cross-validate the two methods (both yield 52.9% / 47.1%). (3) We give a zero-training co-design fix that closes the gap to 0%, and abstract a detector- and framework-independent failure condition. We position the result as the first quantified tool-exposure instance of the model–harness "coupling bottleneck" (Guo et al., 2026).

---

2. Background: Confidence-Tiered Tool Exposure

System. WeClaw (Weng, 2026b) is an open-source desktop LLM agent for daily personal use, with 119 registered tools spanning shell, file, screen, and search (the four *core* tools always exposed), plus ~23 *extended* tools (browser, clipboard, calculator, datetime, notification, app control, stock query, enterprise lookup, and others) and a long tail of domain tools.

Tiered exposure. A ToolExposureEngine maps an intent result to a visibility tier by confidence: confidence $\geq 0.8 \rightarrow$ RECOMMENDED (narrowest), $\geq 0.5 \rightarrow$ EXTENDED, else FULL. For each tier it constructs a concrete tool-name set: RECOMMENDED = core tools $\cup$ intent-to-tool mapping for the recognized intents $\cup$ a capability-info tool; EXTENDED adds the ~23 extended tools; FULL either exposes all registered tools or, under a retrieval-based fallback, a lexical Top-K union with a conservative floor. The constructed set is resolved for dependencies and rendered into function-calling schemas—the hard boundary of what the model can call.

Failure escalation. The engine maintains a per-task consecutive-failure counter. When a tool call cannot be resolved, report_failure increments it; once it reaches a threshold (2 in production), _upgrade_tier widens the tier (RECOMMENDED → EXTENDED → FULL → unbounded) and the next attempt re-exposes tools at the wider tier. This ladder is the runtime safety net that recovers from an under-exposed first attempt—at the cost of extra round-trips.

The two stages. We separate, for analysis, **Stage A** (confidence → tier) from **Stage B** (tier → tool set construction). The Recognition-Exposure Gap arises precisely because these stages are decoupled: the detector's recommended tools influence Stage A's confidence but, absent an explicit merge, do not influence Stage B's construction.

---

3. The Recognition-Exposure Gap

Let a query carry a ground-truth target tool ref (the tool a correct assistant should call). Define two booleans produced by *different* pipeline stages:

- rec_hit = the intent detector **recommended** ref (Stage A / recognition);
- exposed = ref $\in$ the **function-calling exposure set** (Stage B / construction).

The **Recognition-Exposure Gap** is the event rec_hit ∧ ¬exposed: the agent recognized the needed tool but did not make it callable. Recognition accuracy reports P(rec_hit); exposure coverage reports P(exposed); the gap is their divergence. Because standard evaluations report only the former, the gap is invisible to them.

General failure condition. The gap arises whenever a harness combines (a) **confidence-tiered exposure**, where a detector-confidence shift can move queries into a narrower tier, with (b) **decoupled recommended-tool injection**, where the detector's recommended tools are not merged into the tier's constructed set. Under (a)+(b), *any* detector upgrade that raises confidence—paradoxically, an improvement—can push ambiguous queries into the narrowest tier while the recommended tools remain outside it. The condition is detector-independent (rules, LLM, or hybrid) and framework-independent (any harness with the two-stage structure). We verify this on a production LLM detector and, via deterministic replay, isolate the mechanism.

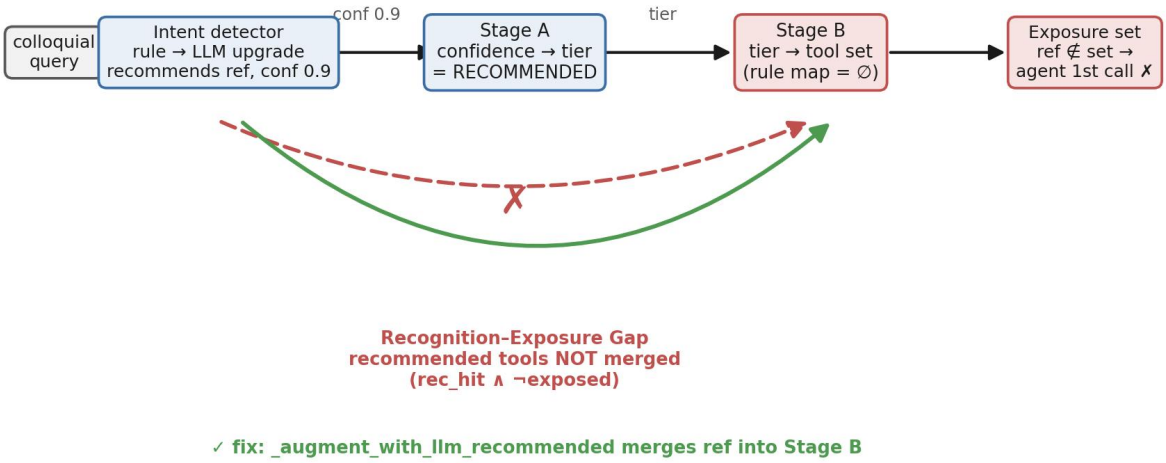


Figure 1. Mechanism of the Recognition-Exposure Gap. The intent detector's recommended tools drive Stage A's confidence (solid arrow) but are not merged into Stage B's tool-set construction (dashed, ✗), so a correctly recognized tool never enters the function-calling exposure set; the zero-training fix (green) reconnects the two decoupled stages.

Terminology. We use *tool exposure* in its **capability** sense—the set of tools a harness makes visible/callable to the model—strictly distinct from *data over-exposure* in the privacy sense (Lin et al., 2026), which concerns an agent transmitting sensitive data beyond functional necessity. The two are in fact opposite failure directions of the same interface: over-exposure risks privacy, **under-exposure** (this work) risks capability. We return to this framing in Section 6.

---

4. Method: Measuring and Closing the Gap

4.1 Offline counterfactual (round2). We reproduce the production LLM-intent path: for each query, an LLM classifier returns an intent result with recommended tools, and we check

whether ref is in the exposure set the engine actually constructs. This yields rec_hit, exposed, and thus the gap, without executing tool calls. It is the fast, first-order measurement.

4.2 Execution-level trigger chain (round3). To measure the gap's *runtime* consequence, we faithfully replay the escalation ladder. For each query we drive the real production code path (_determine_tier → _get_tool_names_for_tier → dependency resolution) round by round: if ref ∉ exposed set, we record a failed round-trip (the semantics of an unresolvable first function call) and call report_failure, which escalates the tier once the threshold is reached; we re-expose and repeat until ref becomes reachable (recovery) or the ladder tops out. To eliminate API nondeterminism and cost, round3 is a **deterministic replay**: it injects round2's measured recommendation outcome (llm_recommended_tools = [ref]) rather than re-calling the LLM, while exposure construction and escalation run through the unmodified production engine. We run two arms on the same queries: **before** (merge off, the pre-fix behavior) and **after** (merge on, the fix).

4.3 Zero-training co-design fix. The fix merges recognized tools into the exposure set at construction time: _augment_with_llm_recommended inserts llm_recommended_tools into both the FULL-fallback and non-FULL return points of Stage B, under four guards (config gate; known-tool filtering against the registry; forbidden-tool precedence; a max_merge cap of 8). No model is trained, no threshold is retuned; the fix simply connects Stage A's output to Stage B's construction.

---

5. Experiments

5.1 Setup. The seed set is 20 colloquial, ambiguous Chinese queries (e.g., "现在几点了" / "what time is it now", "帮我算个数" / "help me do a calculation", "弹个通知提醒我" / "pop me a notification"); 17 carry a single ground-truth ref tool (conservatively, the one most-central tool), and 3 are fully ambiguous with no ref (excluded from coverage/trigger statistics). The intent LLM is deepseek-v4-flash, which on success returns confidence 0.9 (collapsing queries into RECOMMENDED). Escalation threshold = 2 (production value); latency lower bound per failed round-trip = 1,950 ms (round2's measured mean intent-classification latency). Registered tools = 119.

5.2 Offline counterfactual (round2). Table 1 reports the recognition/exposure divergence. The LLM recommends the correct tool in every case, yet fewer than six in ten recommended tools are actually exposed; merging recognized tools lifts counterfactual coverage to 100%.

Table 1. Offline counterfactual measurement (round2), LLM intent detection, 17 seed queries with a ground-truth tool.

Metric	Value
LLM recommendation hit rate (rec_hit)	17/17 = **100.0%**
Real exposure coverage (exposed)	9/17 = **52.9%**
Recognition-Exposure Gap (rec_hit ∧ ¬exposed)	8/17 = **47.1%**

Metric	Value
Counterfactual coverage *with* merge fix	17/17 = **100.0%**
Mean exposure size (before → after fix)	8.3 → 9.6 tools

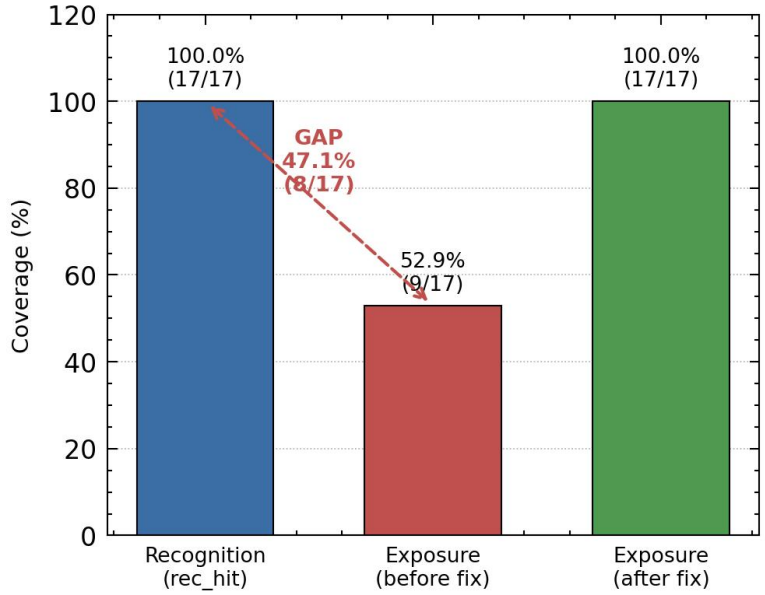


Figure 2. Recognition vs. exposure coverage (round2, 17 seed queries with a ground-truth tool). The LLM recommends the correct tool in 100% of cases, yet only 52.9% of those tools are actually exposed (gap 47.1%); merging recognized tools into the exposure set restores 100% coverage.

5.3 Execution-level trigger chain (round3). Table 2 contrasts the two arms. Before the fix, only 52.9% of queries can call the recommended tool on the first attempt; the other 47.1% must detour through the escalation ladder. All eight detouring queries do eventually recover—but each pays two failed round-trips and one tier escalation, inflating the exposed set from a mean of 8.9 (first round) to 21.0 (final tier, +136%) and accumulating a ≥31.2 s latency lower bound. After the fix, first-attempt reachability is 100%, with zero failed round-trips, zero escalations, zero added latency, and *lower* mean exposure (9.4) because no query is forced up to the broad tier.

Table 2. Execution-level trigger chain (round3), before/after arms on the same 17 seed queries (deterministic replay; escalation threshold 2).

Metric	before (merge off)	after (merge on)
First-attempt reachability	9/17 = **52.9%**	17/17 = **100.0%**
Need-detour rate	8/17 = 47.1%	0/17 = **0.0%**
Recovery after detour	8/8 = 100%	n/a
Failed round-trips (total)	**16**	**0**
Tier escalations (total)	**8**	**0**

Metric	before (merge off)	after (merge on)
Mean first-round exposure	8.9	9.4
Mean final-tier exposure	**21.0**	9.4
Extra latency lower bound (cumulative)	**31,200 ms**	**0**

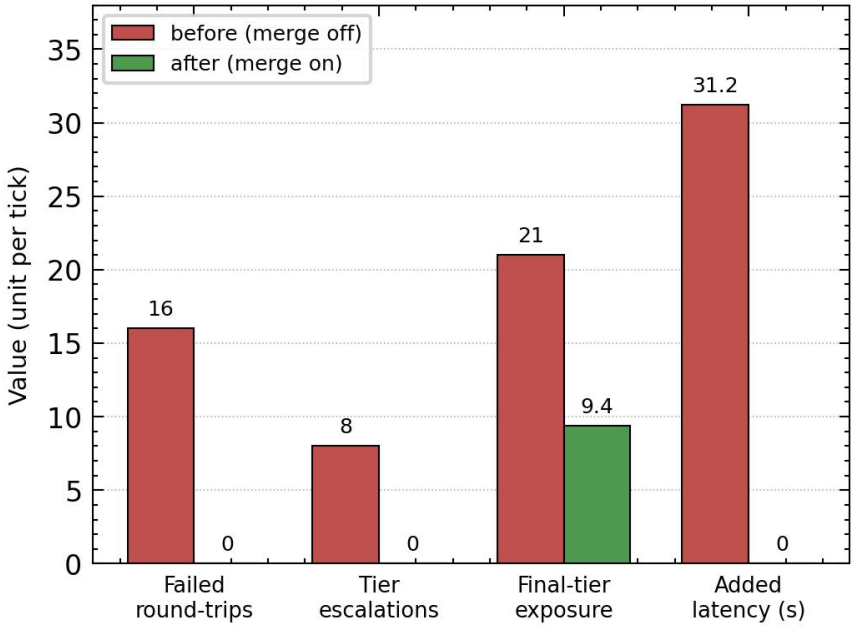


Figure 3. Execution-level runtime cost, before vs. after the fix (round3, same 17 queries). The gap costs 16 failed round-trips, 8 tier escalations, a mean final-tier exposure of 21.0 tools, and ≥31.2 s of added latency; the fix removes all of them and lowers exposure.

Table 3 shows representative trigger chains from the before arm. The pattern is uniform: two failed attempts inside the RECOMMENDED tier (the recommended tool is absent), then one escalation to EXTENDED, where the tool finally appears. Control queries whose ref is a core/always-resident tool (file, search, capability-info) reach on the first attempt in both arms, confirming the chain does not over-report.

Table 3. Representative before-arm trigger chains (rt = round-trip; tier, |exposed|, whether ref is reachable).

Query(zh/engloss)	ref	Trajectory
现在几点了 / “what time is it now”	datetime_tool	rt1[REC, 5, ✗] → rt2[REC, 5, ✗] → rt3[EXT, 31, ✓]
帮我算个数 / “help me do a calculation”	calculator	rt1[REC, 5, ✗] → rt2[REC, 5, ✗] → rt3[EXT, 31, ✓]
弹个通知提醒我 / “pop me a notification”	notify	rt1[REC, 11, ✗] → rt2[REC, 11, ✗] → rt3[EXT, 34, ✓]
查下这家公司背景 / “look up this company’s background”	enterprise_query	rt1[REC, 5, ✗] → rt2[REC, 5, ✗] → rt3[EXT, 31, ✓]

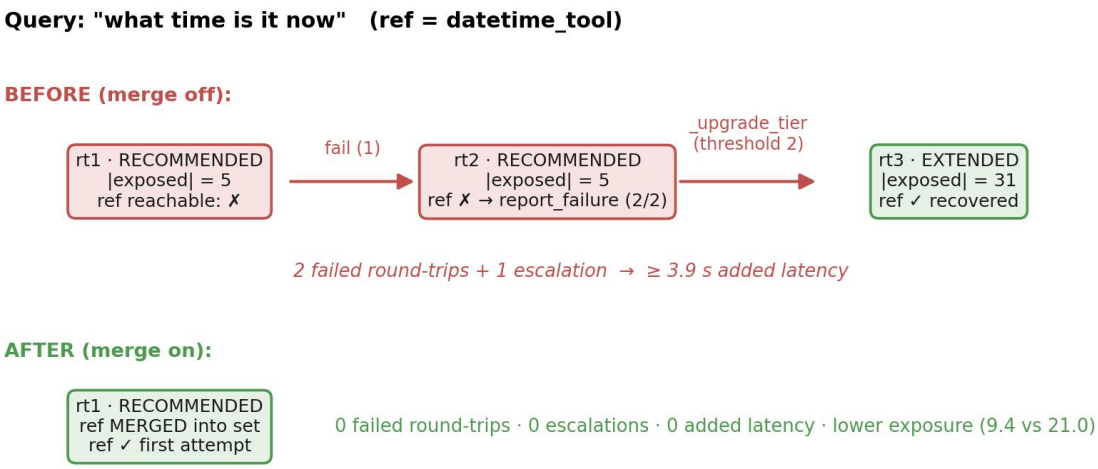
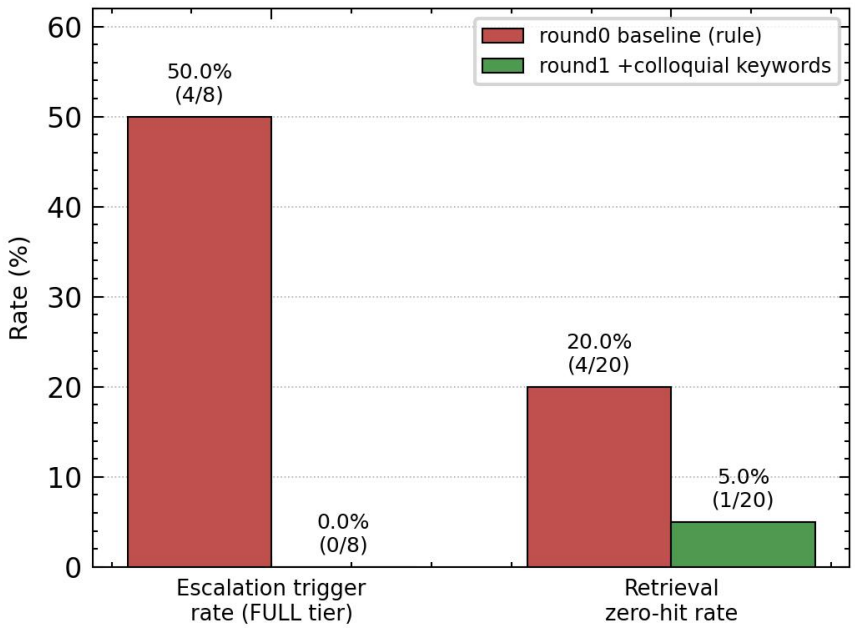


Figure 4. Representative before-arm trigger chain for "what time is it now" (ref = datetime_tool): two failed round-trips inside the RECOMMENDED tier (|exposed| = 5, ref absent), then one escalation to EXTENDED (|exposed| = 31) where the tool finally becomes reachable; after the fix the tool is reachable on the first attempt.

5.4 Floor ablation (round0/round1). A separate ablation (Figure 5) isolates the extended-tool floor of the FULL tier. Removing the floor causes **50%** of ambiguous queries to miss a high-frequency tool on the first attempt in the FULL tier (e.g., datetime, calculator, browser); back-filling colloquial keywords into tool descriptions drops the escalation trigger rate from **50% → 0%** and retrieval zero-hits from 4/20 → 1/20. This confirms the gap's mechanism is about *which tools enter the constructed set*, and that the failure condition is not specific to the LLM detector—any confidence distribution that lands ambiguous queries in a narrow tier without the recommended tools reproduces it.



† round1 keywords mirror eval query wording → 0% may overfit; hold-out validation required before dropping floor.

Figure 5 (supplementary). FULL-tier floor ablation (round0 vs. round1, rule-based intent). Removing the extended-tool floor makes 50% of ambiguous FULL-tier queries escalate (4/8) and leaves a 20% lexical-retrieval zero-hit rate (4/20); back-filling colloquial keywords into tool descriptions drops both to 0% (0/8) and 5% (1/20). † Because the back-filled keywords mirror the evaluation query wording, the 0% escalation may overfit; a hold-out check is required before dropping the floor.

5.5 Cross-validation. The two methods agree exactly: round3's first-attempt reachability (52.9% before / 100% after) matches round2's real exposure coverage (52.9%) and gap (47.1%), and the after-arm matches round2's post-fix 100% / 0%. The offline counterfactual and the execution-level replay are independent measurements of the same quantity, which strengthens confidence that the gap is a real property of the pipeline rather than an artifact of either probe.

---

6. Related Work

Closest neighbors (we differentiate directly). Five works are rhetorically or architecturally adjacent and must be addressed head-on. (i) The **Knowing-Doing Gap** (Cheng et al., 2026, arXiv:2605.14038) is the most similar in shape—"mismatches persist even at high meta-cognitive confidence"—and also splits tool use into cognition/execution stages, calling for "architectural modifications to bridge cognition-action representations." It differs from ours on six axes: the fault location (model hidden-state representation geometry vs. an interface between two *harness software modules*); the failure timing (the model *sees* the tool yet does not call it vs. the tool *never enters* the visible set, i.e., our failure is strictly upstream); the diagnosis (white-box linear probing vs. black-box pipeline coverage audit); the fix (auxiliary loss / model change, requiring training vs. zero-training remapping); the trigger (an intrinsic property across four models vs. *detector-upgrade-triggered*); and the metric (mismatch rate vs. exposure-coverage gap). (ii) **"How Many Tools Should an LLM Agent See?"** (Repantis et al., 2026a, arXiv:2605.24660) states "show too few and the correct tool may not appear" and learns per-query shortlist depth by RL; it attributes under-exposure to *shortlist depth (K too small)*, whereas we show that **even with sufficient K and 100% correct recognition**, the construction stage silently drops tools—a failure depth-tuning cannot fix in principle. (iii) **The 99% Success Paradox** (Repantis et al., 2026b, arXiv:2605.18857) shows "selectivity vanishes even with perfect selectors," but is an *evaluation-methodology critique* (report a chance-corrected metric); ours is a *systems causal diagnosis* plus a verifiable fix (gap → 0%). (iv) **ITR / Dynamic System Instructions and Tool Exposure** (Franko, 2025, arXiv:2602.17046; note: arXiv prefix 2602 but published 1 Dec 2025) is the architectural prototype of confidence-gated exposure and reports only positive gains (context −95%, routing +32%); it never audits the "recognized vs. actually exposed" coverage gap. We position our work as a *failure-mode audit of ITR-class confidence-gated exposure*. (v) **AgentWeave: Routing Before Reasoning** (Singla et al., 2026, arXiv:2608.23078) is closest architecturally—"candidate-space construction can materially affect a fixed model's function-calling behavior"—and reports a *positive* result (deterministic routing beats all-tools/random/semantic baselines); ours is the *negative* counterpart (a detector upgrade silently

breaks construction). Its finding that semantic top-8 = 0/48 in fact corroborates ours: semantic ranking alone does not construct a usable exposure set.

The named-"gap" family. Several gaps have been named for tool use, all on the *model or content side*: the adoption gap (tools available but unused; Fan et al., 2026, arXiv:2608.03327), the knowing-doing gap (Cheng et al., 2026), the prerequisite gap (retrieved but missing dependencies, arXiv:2604.05333), query–skill misalignment (arXiv:2609.01642), and knowledge-retrieval dissociation (arXiv:2606.12451). None is caused by *two decoupled modules inside the harness*. Ours is the only gap localized at the **pipeline interface** and the only one shown to occur **even when recognition is 100% correct**.

Disciplinary anchor. Guo et al. (2026, arXiv:2606.20683) survey agent system and harness design around the question of "where does the bottleneck reside—in the foundation model, in the execution harness, or in the coupling between them?" and list model–harness co-evolution as an open challenge. We position the Recognition-Exposure Gap as the **first quantified tool-exposure instance of that coupling bottleneck**: the model is not wrong, the harness modules are each individually correct, and the failure lives entirely in their coupling.

Methodological allies and the counterintuitive framing. PluginEval (arXiv:2608.08700) separates generation from verification and attributes errors stage-by-stage; our separation of recognition from exposure follows the same logic and supports our experimental design's validity. RubricRefine (arXiv:2605.09730) and PAEF (arXiv:2605.01604) both stress failures that "run to completion without raising errors" / evade standard metrics—echoing our "silent reduction." The mainstream consensus that *fewer tools is better* (Paramanayakam et al., 2024, arXiv:2411.15399; Gan et al., 2025, RAG-MCP) and that *ranking does not decide how many to select* (Feng et al., 2026, arXiv:2607.27083) frames our contribution: narrowing is right, but narrowing **silently past the point of correctness** is a distinct hazard. On the low tool-invocation rates reported for computer-use agents (36.3% in OSWorld-MCP; Jia et al., 2025, arXiv:2510.24563), usually attributed to model capability/willingness, our result offers an alternative partial explanation: some fraction may stem from exposure under-coverage.

---

7. Discussion and Limitations

Why it matters. The gap inverts a natural engineering instinct. Teams upgrade intent detectors expecting strictly better tool access; under confidence-tiered exposure with decoupled injection, the upgrade can *reduce* availability while every recognition metric improves. Because the failure is silent (tasks still recover via the escalation ladder), it can persist in production while dashboards stay green—much like the "silent" collapses reported for agent memory (Weng, 2026c) and evaluation (Pandey, 2026). The fix is cheap and structural: connect the two stages.

Limitations. (1) *Sample size*: the 20 seed queries are a prototype; the mechanism is confirmed on 17 with ground truth, but a 500-query public tool-selection artifact (seeded from 648 intent-verified library entries) is planned to widen the estimate. (2) *Single framework*: we demonstrate the gap on WeClaw and abstract a framework-independent

condition, but a reproduction on ≥1 open-source agent framework with the same two-stage structure remains future work. (3) *Deterministic replay*: round3 injects round2's measured recommendation outcome rather than re-calling the LLM (zero cost, fully reproducible), so it does not re-verify the LLM hit rate itself; the per-round-trip latency is a *lower bound* (excludes tool execution and second-pass generation). (4) *Single detector*: multi-model replication (does the gap persist across different LLM intent detectors?) is planned. None of these limitations affects the core structural claim, which is established by the deterministic production code path and cross-validated by two independent probes.

---

8. Conclusion

We identified, named, and quantified the **Recognition-Exposure Gap**: in confidence-tiered LLM agents, upgrading the intent detector can silently reduce tool availability because confidence-to-tier mapping and recommended-tool-to-exposure-set construction are decoupled. On a production desktop agent, a detector with 100% recommendation accuracy exposed only 52.9% of the recognized tools (gap 47.1%); an execution-level replay showed the gap costs 16 failed round-trips, 8 escalations, 136% exposure inflation, and ≥31.2 s latency, all while recovering "correctly but late." A zero-training co-design fix closes the gap to 0% and reduces exposure. The gap is the first quantified tool-exposure instance of the model–harness coupling bottleneck, and its failure condition—confidence-tiered exposure plus decoupled recommended-tool injection—is detector- and framework-independent. We hope the naming and the audit method prompt broader coverage-vs-recognition reporting for tool-exposure systems.

References

Note: APA 7th; arXiv preprints cited by number and year. Metadata cross-validated against the 2026-09-10 global survey (84 arXiv IDs verified). See references.bib.

- Cheng, Y., et al. (2026). *Model-adaptive tool necessity reveals the knowing-doing gap in LLM tool use*. arXiv:2605.14038.
- Fan, S., et al. (2026). *Screenshots or tools? Eliciting tool use and managing multimodal context in hybrid GUI-MCP computer-use agents*. arXiv:2608.03327.
- Feng, Y., et al. (2026). *Scores are not decisions: Cost-aware stopping for tool acquisition in LLM agents*. arXiv:2607.27083.
- Franko, U. (2025). *Dynamic system instructions and tool exposure for efficient agentic LLMs*. arXiv:2602.17046. [arXiv prefix 2602; published 1 Dec 2025.]
- Gan, T., et al. (2025). *RAG-MCP: Mitigating prompt bloat in LLM tool selection via retrieval-augmented generation*. arXiv:2505.03275.

- Guo, J., Hao, Z., & Wang, C. (2026). *From question answering to task completion: A survey on agent system and harness design*. arXiv:2606.20683.
- Jia, H., et al. (2025). *OSWorld-MCP: Benchmarking MCP tool invocation in computer-use agents*. arXiv:2510.24563.
- Lin, Y., …, & Zheng, Z. (2026). *AgentRaft: Automated detection of data over-exposure in LLM agents*. arXiv:2603.07557.
- Pandey, M. (2026). *Evaluating agentic AI in the wild: Failure modes, drift patterns, and a production evaluation framework*. arXiv:2605.01604.
- Paramanayakam, V., et al. (2024). *Less is more: Optimizing function calling for LLM execution on edge devices*. arXiv:2411.15399.
- Repantis, V., et al. (2026a). *How many tools should an LLM agent see? A chance-corrected answer*. arXiv:2605.24660.
- Repantis, V., et al. (2026b). *The 99% success paradox: When near-perfect retrieval equals random selection*. arXiv:2605.18857.
- Singla, S., et al. (2026). *AgentWeave: Routing before reasoning for efficient function calling in tool-rich language models*. arXiv:2608.23078.
- Weng. (2026b). *WeClaw: An open-source desktop LLM agent* [Software].
- Weng. (2026c). *EBEAC: Event-driven zero-LLM-cost experience capture for agent memory*.
- *[PluginEval]* (2026). *PluginEval: A diagnostic benchmark for fine-grained error attribution in function calling*. arXiv:2608.08700.
- *[RubricRefine]* (2026). *RubricRefine: Improving tool-use agent reliability with training-free pre-execution refinement*. arXiv:2605.09730.
- *[Prerequisite gap]* (2026). arXiv:2604.05333. · *[Query–skill misalignment]* (2026). arXiv:2609.01642. · *[Knowledge-retrieval dissociation / ToolSense]* (2026). arXiv:2606.12451.