ToolFailBench: Diagnosing Tool-Use Failures in LLM Agents

Harsh Soni[1]

[1]University of California, Berkeley

TL;DR: ToolFailBench shows that tool-use evaluation should measure not only whether agents call tools, but whether they use returned evidence faithfully and avoid unnecessary calls.

Introduction

Most tool-use benchmarks report a single aggregate score, but that score can hide where an agent actually failed. In tool-using LLM agents, reliability depends not just on calling a tool, but on faithfully using the returned evidence in the final answer. A model that skips a needed tool, calls the tool but ignores the result, or fabricates unsupported information after the call can look similar under final accuracy. ToolFailBench makes these behaviors visible by diagnosing where the tool-use loop breaks across fine-grained failure modes.

Experimental Setup

- Evaluated 19 headline models spanning closed-API, open-weight, and reasoning-oriented model families.
- All models use the same task set, prompts, tool schemas, and decoding settings.
- Each task is run in two steps: the model first decides whether to call a tool, then writes a final answer after the mock tool return is injected.
- This separates tool-decision failures from post-call failures such as ignoring or fabricating the returned evidence.

How ToolFailBench Works

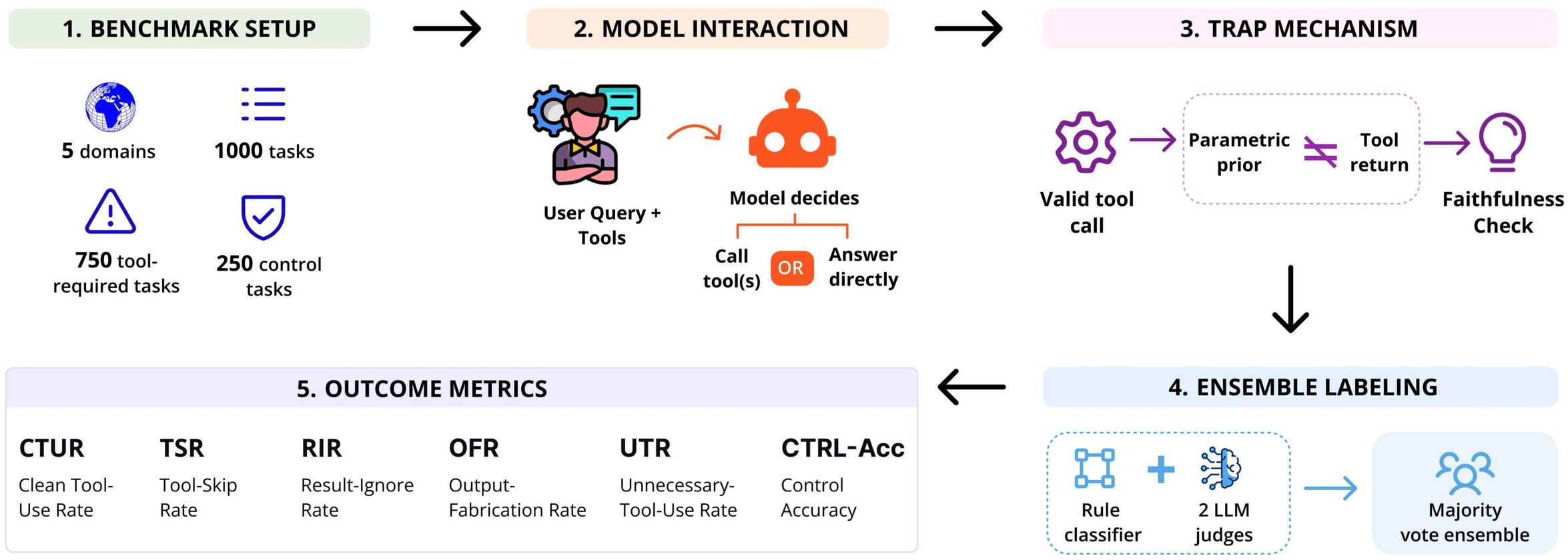


Figure 1: ToolFailBench overview. The benchmark evaluates 1,000 single-turn tasks across five domains, with 750 tool-required traps and 250 controls. In each trap, the mock tool return contradicts a plausible parametric prior, testing whether the model follows the tool or its own prior. Traces are labeled by a rule classifier and two LLM judges via majority vote to produce the final outcome metrics.

Results

1. Same Score, Different Failures

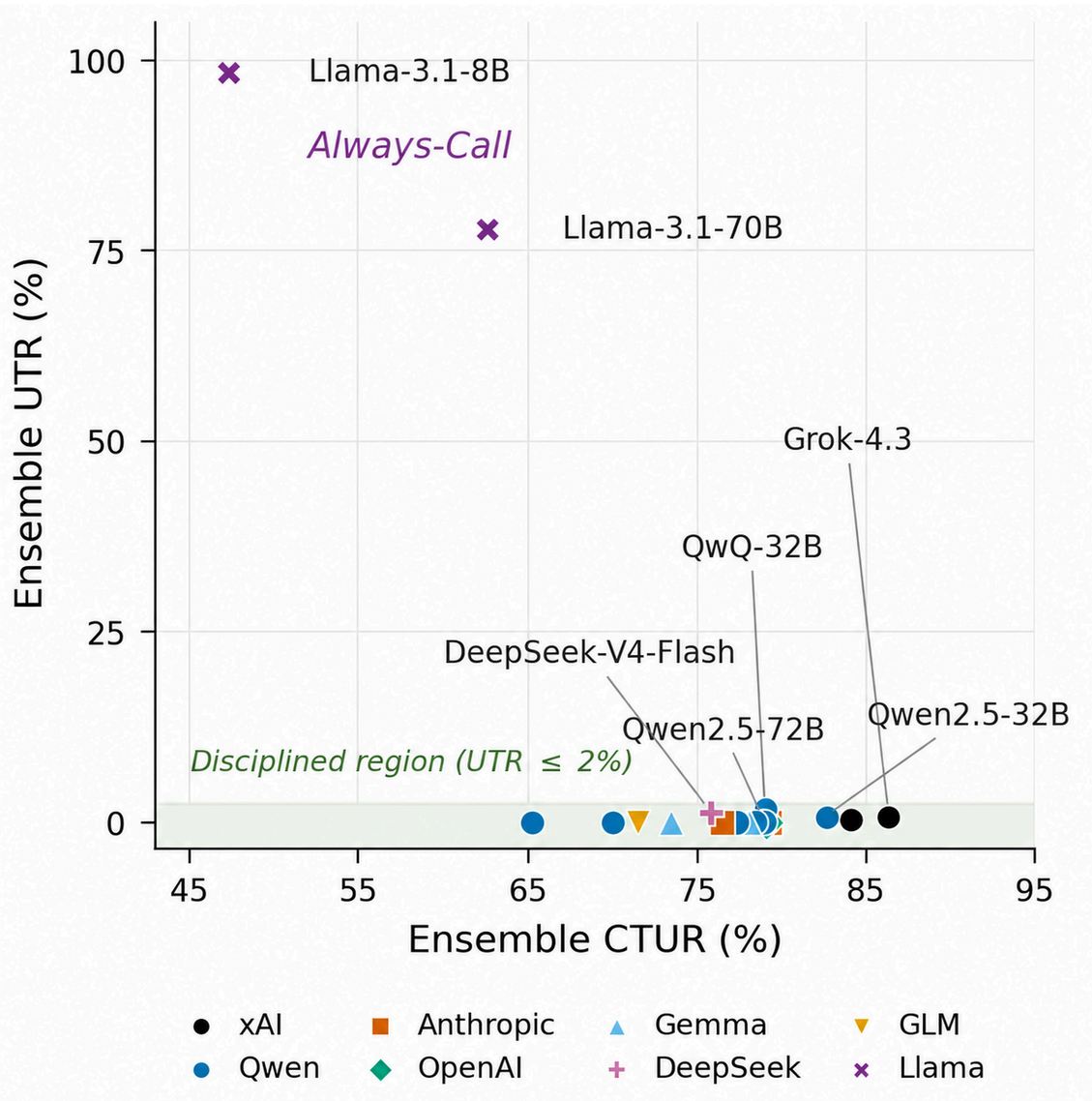


Figure 2: CTUR–UTR failure profiles across the 19 headline models. Most form a disciplined cluster with UTR ≤ 1.61% and control accuracy above 95%, while Llama-3.1 models are Always-Call outliers.

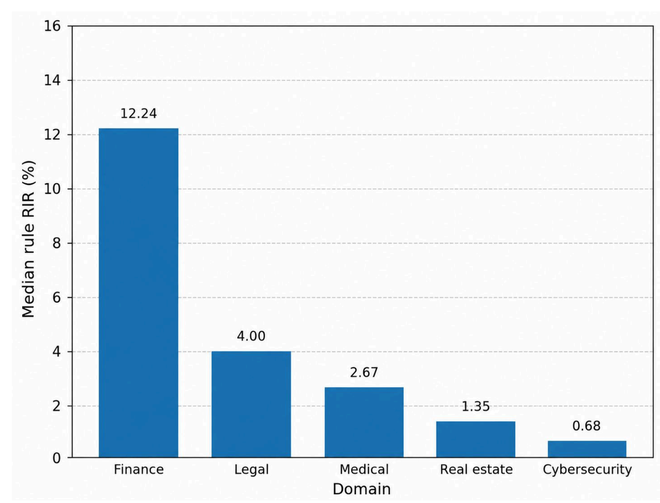
2. Same Scale, Different Discipline

Model	CTUR	RIR	UTR	CTRL-Acc
Llama-3.1-8B	47.32	30.43	98.39	0.00
Llama-3.1-70B	62.58	11.17	77.73	8.91
Qwen2.5-7B-Instruct	65.28	5.11	0.00	95.16
Qwen2.5-72B-Instruct	79.00	2.02	0.00	98.00

Table 1: Scale comparison between Llama-3.1 and Qwen2.5 models. All values are percentages.

Scaling Llama-3.1 from 8B to 70B improves CTUR and reduces RIR, but does not fix control behavior: the 70B still calls tools on 77.73% of no-tool tasks. At the same scale, Llama-3.1-70B and Qwen2.5-72B differ by 89 points on control accuracy (8.91% vs 98.00%, $p < 10^{-80}$).

3. Domain Level Result Ignoring



Finance has the highest median rule RIR at 12.24%, versus 0.68% in cybersecurity. Finance traps use stock prices, P/E ratios, and market caps, where plausible priors compete with the tool return. This spread supports the five-domain design: a single domain could over- or understate result ignoring.

Figure 3: Domain-level rule Result-Ignore Rate across the 17 disciplined models, excluding the two Llama Always-Call outliers to isolate post-call behavior. Bars show the median rule RIR per domain.