FinToolBench: Evaluating LLM Agents for Real-World Financial Tool Use

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents are increasingly used as interfaces to financial data, yet existing evaluations often score final answers while leaving tool traces weakly tested. This is risky in finance: a valid-looking call can still be unacceptable if it uses stale data, escalates user intent, or crosses market and regulatory domains. We introduce **FinToolBench**, a runnable benchmark of 760 real free-tier financial tools paired with 295 tool-required questions (166 single-tool, 129 multi-tool). Each tool is annotated with finance attributes, *i.e.*, timeliness, intent type, and regulatory domain, which support call-level compliance metrics (TMR, IMR, DMR) computed directly from execution traces. We also provide **FATR** (Finance-Aware Tool Routing), a lightweight reference baseline that retrieves candidate tools, injects finance attributes into tool cards, and records auditable traces. Across seven LLM backends, including Doubao-Seed-1.6, Claude-Sonnet-4.5, Grok-3-beta, and Gemini-3.1-Pro, no single model dominates capability and compliance jointly, separating aggressive callers from precise but conservative ones. Code and data: `https://anonymous.4open.science/r/FinToolBench-EB8B` (anonymized for review).

1 Introduction

Large Language Models (LLMs) are moving financial analysis from static question answering toward dynamic interaction with APIs, databases, and computational tools. In this setting, the tool trace is part of the answer. A response may appear grounded because it contains tool outputs, yet still be unreliable if the agent retrieved stale data, called a drifting endpoint, or used a tool from the wrong market domain (Guo et al., 2024). Evaluation must therefore assess not only whether tools are invoked and executed successfully, but also whether the resulting tool trace is acceptable under finance-specific constraints, especially as agents operate over longer horizons and tool use itself evolves (Lu et al., 2026; Jiang et al., 2025).

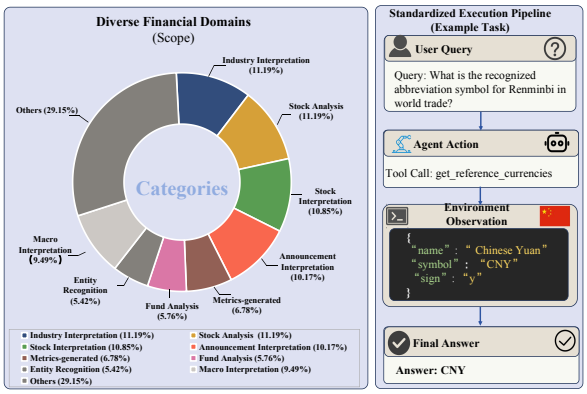

Figure 1: FinToolBench overview. Left: the scope of our benchmark across representative categories. Right: an example of the standardized execution pipeline, where an LLM agent selects a tool, observes the environment output, and produces a final answer with an auditable tool trace.

Existing benchmarks leave a gap between what is easy to measure and what is necessary to trust. General tool benchmarks emphasize API correctness and executability (Guo et al., 2024) but rarely test finance-specific acceptability. Finance benchmarks focus on knowledge- or document-centric QA and involve virtually no executable tools, relying on static datasets or a negligible number of mock interfaces. We argue that current metrics are blind to three recurring failure modes essential for financial reliability: (i) *timeliness*—a question asking for "current" exchange rates is fundamentally unanswered if the agent retrieves a daily snapshot, even if the API call is syntactically perfect; (ii) *intent restraint*—an agent must differentiate informational queries from transactional actions and never escalate to execution without explicit authorization; and (iii) *domain alignment*—the chosen

tool chain must adhere to the regulatory and market domain of the query (*e.g.*, using equity tools for a cryptocurrency inquiry is a domain hallucination).

To address these gaps, we introduce **FinToolBench**, a runnable benchmark built from real free-tier tools and tool-required questions. FinToolBench scales financial agent evaluation to 760 executable tools and 295 tool-required items (166 single-tool, 129 multi-tool). Each tool is annotated with three finance attributes, *i.e.*, timeliness, intent type, and regulatory domain, enabling us to compute call-level compliance mismatch rates (TMR, IMR, DMR) alongside standard invocation and execution metrics. We further provide **FATR** (Finance-Aware Tool Routing), a lightweight reference baseline that retrieves a small candidate set, injects finance attributes into tool cards, and stabilizes execution with caching, retries, and output compression. Figure 1 sketches the benchmark scope and the standardized execution pipeline.

In summary, this paper makes three contributions: (1) FinToolBench: a benchmark of 760 free-tier financial tools and 295 tool-required questions producing auditable tool traces under real execution. (2) Finance-aware evaluation: capability metrics plus call-level compliance mismatch rates (TMR, IMR, DMR) measuring violations of timeliness, intent restraint, and domain alignment. (3) FATR: a lightweight reference baseline and execution harness for evaluating finance-aware tool routing under a fixed stack.

2 Related Work

2.1 Tool-Using Agents and Benchmarks

Tool-augmented agents interleave reasoning with external actions to improve grounding and support up-to-date answers (Yao et al., 2022; Schick et al., 2023; Patil et al., 2024; Qin et al., 2024). Benchmarks evaluate tool selection and calling at scale (API-Bank Li et al., 2023; StableToolBench Guo et al., 2024) and long-horizon interaction in realistic environments (Liu et al., 2024; Mialon et al., 2024; Zhou et al., 2024; Drouin et al., 2024; Yao et al., 2024). Recent efforts sharpen the focus toward tool-interface competence and agentic behavior, including BFCL (Patil et al., 2025) and τ^2-bench (Barres et al., 2025), and study agents in settings where tools or capabilities evolve over time and long-horizon traces are central artifacts (Lu et al., 2026; Jiang et al., 2025; Wan et al., 2026; Yang et al., 2025).

2.2 Financial Benchmarks and Evaluation

In finance, most benchmarks emphasize domain knowledge and document-centric QA rather than executable tool use. Examples include FinanceBench (Islam et al., 2023), OpenFinData (OpenCompass, 2024), and report-focused datasets such as FinQA (Chen et al., 2021) and TAT-QA (Zhu et al., 2021). Recent works like FinEval (Guo et al., 2025b), FLAME (Guo et al., 2025a), and the Finance Agent Benchmark (Bigeard et al., 2025) broaden knowledge coverage, but none release a standardized large tool library or define call-level compliance metrics. Safety-oriented agent evaluations (Xia et al., 2025; Tur et al., 2025) probe deliberate misuse but are not finance-specific and do not operationalize domain-grounded constraints like timeliness, intent limits, and regulatory scope. FinToolBench addresses this by pairing a fully runnable tool inventory with tool-required questions and explicitly defining finance constraints at the level of each tool call via a lightweight, auditable attribute schema, enabling direct measurement of timeliness, intent, and domain mismatches from execution traces rather than relying solely on final-answer correctness or generic safety checks.

2.3 Evaluation Protocols

Because answer correctness is hard to score at scale for open-ended questions, recent work uses LLMs as judges with structured rubrics (Zheng et al., 2023; Liu et al., 2023), while noting that LLM-based scoring can be unstable across runs and sensitive to prompting (Hashemi et al., 2024; Haldar and Hockenmaier, 2025; D'Souza et al., 2025), and that comparative setups can elicit more informative judgments than independent scoring (Zhang et al., 2025). In line with these findings, we reduce variance via repeated judging and explicitly separate tool execution from correctness so that a failure to call or execute tools is not conflated with an evaluation artifact; our protocol is compatible with alternative rubric designs, since the benchmark produces complete tool traces that can be inspected and re-judged.

3 FinToolBench

FinToolBench is an execution-grounded benchmark designed to evaluate financial tool use under real execution. Its design emphasizes two principles. First, every run produces an auditable tool

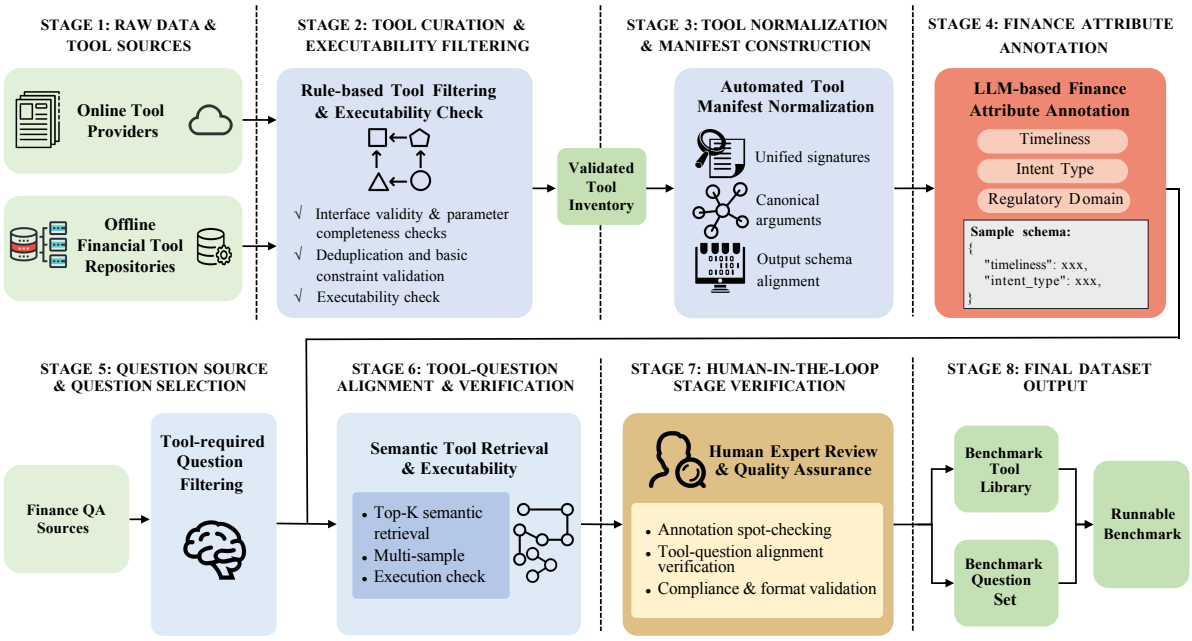

Figure 2: FinToolBench dataset construction pipeline. Stage 1 collects raw tool sources. Stage 2 performs tool curation and executability filtering to obtain a validated tool inventory. Stage 3 normalizes tools into a unified manifest with standardized signatures, canonical arguments, and aligned output schemas. Stage 4 annotates each tool with finance attributes (timeliness, intent type, regulatory domain). Stage 5 sources and selects tool-required questions. Stage 6 aligns questions with tools via semantic retrieval, multi-sample verification, and execution checks. Stage 7 adds human-in-the-loop quality assurance. Stage 8 outputs the benchmark tool library and benchmark question set as a runnable benchmark.

trace. Second, evaluation separates capability (*i.e.*, invocation and execution success) from compliance (*i.e.*, call-level timeliness, intent, and domain alignment). The benchmark measures an agent's ability to select tools from a large heterogeneous library, instantiate valid arguments, handle execution failures, and produce answers whose tool use respects finance-specific constraints.

In contrast to prior tool-use benchmarks that focus primarily on API calling accuracy, FinToolBench evaluates both capability and compliance directly from executable tool traces. Figure 2 summarizes the construction pipeline. We first build a validated tool inventory from free-tier sources, normalize heterogeneous interfaces into a unified manifest, and annotate finance attributes for each tool. We then construct a tool-required question set, align questions with candidate tools through retrieval and verification, and apply human-in-the-loop quality checks before release. This design follows lines of work that stress end-to-end tool selection, argument construction, and trace-based diagnosis under real execution.

3.1 Tool Inventory

3.1.1 Tool Sources and Executability Filtering

We construct the tool inventory from two complementary free-tier ecosystems, ensuring reproducibility without proprietary data contracts. **RapidAPI** is a large marketplace of third-party APIs providing broad coverage of real-time and web-based services under free-tier API keys; we filter raw endpoints with a rule-based pipeline that retains a tool only if it satisfies all of: (i) *interface validity* (complete parameter definitions and non-empty descriptions); (ii) *deduplication* of duplicate names and semantically identical interfaces; (iii) *rate-limit sufficiency* (at least 10/h, 100/d, 300/m); (iv) *authentication feasibility* under free-tier access; and (v) *runtime executability* via at least one successful test invocation. Endpoints with broken URLs, faulty authentication flows, or persistent failures are discarded. **AkShare** is an open-source Python library offering stable, research-oriented interfaces over a wide range of financial domains; we select interfaces using finance-related function-name cues (*e.g.*, stock, fund, bond, futures, option, index, macro, currency, crypto, rate, treasury, ETF)

and verify executability through direct invocation.

Scale. We start from 5,470 candidate interfaces (4,507 RapidAPI endpoints and 963 AkShare interfaces). After the above filtering, the final tool library contains 760 tools. Full criteria and counts are given in Appendix A.

3.1.2 Tool Normalization and Manifest Construction

To make the heterogeneous tool ecosystem amenable to retrieval, planning, and evaluation, we normalize each tool into a unified manifest schema. Each tool manifest includes: (i) a stable identifier, (ii) a short description, (iii) a machine-readable signature with canonicalized parameter names and types. Normalization reduces avoidable agent errors: date and time fields follow consistent formats, common identifiers (*e.g.*, tickers) document explicit market conventions, and output schemas are aligned across sources.

Tool traces. Every tool invocation is captured as a structured execution trace, the atomic unit of auditing, error diagnosis, and compliance evaluation. Each record preserves step index, `tool_name`, JSON arguments, raw output, and execution error, enabling reconstruction of the agent's reasoning chain and separation of model reasoning errors from system-level failures. The full schema is given in Appendix B.

3.1.3 Finance Attribute Annotation

Financial constraints are frequently implicit within user queries, rendering compliance measurement impossible based on raw execution traces alone. To bridge this gap, FinToolBench incorporates a lightweight finance attribute schema that explicitly annotates every tool in the library. The structured metadata enables both the baseline methods outlined in Section 4 and the quantitative evaluation metrics in Eq. (1) to rigorously assess operational acceptability.

As summarized in Table 1, each tool is categorized along three distinct dimensions. These annotations are generated through an LLM-based labeling pipeline utilizing a three-vote majority agreement protocol to ensure consistency. Comprehensive details regarding the labeling rubric are provided in Appendix C. By embedding these constraints directly into the tool definitions, our design decouples compliance standards from the agent under test, facilitating precise, trace-level auditing of domain mismatches.

3.2 Question Set Construction

Sources and selection. Tool-required questions are adapted from existing finance QA datasets, including FinanceBench (Islam et al., 2023) and OpenFinData (`openfindata_release`) (OpenCompass, 2024). We standardize all sources into a unified `{question, answer, category}` format and retain only questions identified by Qwen3-8B as requiring tool calls, with length capped at 500 characters. To ensure FinToolBench strictly evaluates external tool use rather than parametric memory, queries answerable via static knowledge are excluded; we keep only items that need real-time market data, specific regulatory filings, or quantitative calculations.

Tool–question alignment. For each question, we first retrieve the top-20 candidate tools using BGE-M3 dense embeddings (Chen et al., 2024), then refine via an LLM verification step with Qwen3-8B under three-sample majority voting (kept if at least two votes). To prevent dominance by high-frequency tools, single-tool questions are grouped by tool name and capped at two random samples per tool; multi-tool questions are fully retained to preserve agentic-workflow diversity.

Human-in-the-loop verification. We complement automated alignment with a stratified spot-check by domain experts, confirming the logical necessity of the aligned tools, the consistency of the attribute annotations, and compliance with execution assumptions and output formatting. Further details are in Appendix D.

3.3 Final Benchmark and Evaluation Protocol

The final benchmark comprises a unified tool library and a question set. The tool library contains 760 tools, and the question set contains 295 questions, including 166 single-tool and 129 multi-tool. Each evaluation run produces a final answer and a complete tool trace, enabling joint assessment of capability and finance compliance.

3.4 Evaluation Metrics

We evaluate each run using two groups of metrics derived from the same auditable tool trace: capability, and compliance. Capability measures whether an agent uses tools and whether tool-augmented traces execute successfully. TIR (Tool Invocation

Table 1: Finance attribute schema used in FinToolBench.

Attribute	Values	Evaluation role
`timeliness`	`realtime`, `daily`, `as_filed`, `periodic`, `static`	Penalize stale calls when timeliness is required.
`intent_type`	`informational`, `advisory`, `transactional`	Penalize escalation beyond user intent.
`regulatory_domain`	set-valued	Penalize domain-mismatched tool usage.

Rate) is the fraction of samples with non-empty tool calls. TESR (Tool Execution Success Rate) is the fraction of samples whose tool-augmented traces execute successfully. We mark a sample successful when its final tool call returns a valid parsed output without error or exception. Intermediate failures and retries are allowed. CER is the conditional execution success rate, defined as CER = TESR/TIR (0 when TIR = 0). Answer correctness is captured by SOFT SCORE and CSS. Numeric and choice tasks are scored against the gold answer with binary labels, while structured and free-text tasks are evaluated by the LLM judge (GPT-5.1) with scores in $\{1, 0.5, 0\}$, averaged across three repeats. CSS is the mean SOFT SCORE over samples with successful execution.

Compliance metrics are defined over executed tool-call traces. For each question q with trace $\tau = \{(t_k, x_k, o_k)\}_{k=1}^{m}$, we look up each tool's finance tags from metadata, $A(t) = (\tau_t(t), i(t), d(t))$ (timeliness, intent type, regulatory domains), and use an LLM judge (GPT-5.1) to assess per-call alignment in each dimension. We then mark a question as mismatched if *any* call in its trace is judged to violate the corresponding constraint:

$$\begin{aligned} TMF(q,\tau) &= \mathbf{1}\big[\exists k:\ J_T\big(q, A(t_k), \tau_k\big) = 0\big]\,, \\ IMF(q,\tau) &= \mathbf{1}\big[\exists k:\ J_I\big(q, A(t_k), \tau_k\big) = 0\big]\,, \\ DMF(q,\tau) &= \mathbf{1}\big[\exists k:\ J_D\big(q, A(t_k), \tau_k\big) = 0\big]\,. \end{aligned} \tag{1}$$

We then compute the dataset-level mismatch rates, denoted as TMR, IMR, and DMR, by averaging TMF, IMF, and DMF over all questions with at least one tool call. Full metric definitions are given in Appendix E.

4 Finance-Aware Tool Routing (FATR)

We provide FATR, a reference baseline that makes finance constraints explicit to a generic LLM planner. Rather than training a specialized policy, we reshape the context given to the planner and wrap execution with stability utilities, keeping the approach implementation-friendly and model-agnostic. FATR serves primarily as an evaluation baseline and a reference implementation. FinToolBench itself is independent of FATR: the benchmark consists of the tool inventory, question set, normalized trace schema, and metrics, and any agent that consumes the tool manifest and emits the same structured traces can be evaluated. In our experiments, FATR fixes retrieval and execution infrastructure so that the reported model differences isolate planner behavior under a common stack.

4.1 Tool Inventory & Retrieval and Tool Cards

Figure 3 summarizes the FATR pipeline end to end. Each tool t in the library $\mathcal{T}$ carries a callable signature and finance attributes $a_t = (t_t, i_t, d_t)$; the executor exposes all tools through a unified interface, validates structured arguments x_k emitted by the planner, dispatches the call, and normalizes the returned output o_k into a compact schema for caching and evaluation. Given a question q from the Question Inventory, FATR retrieves a small candidate set from Tool Inventory & Retrieval to reduce the action space. A retriever embeds q and tool metadata and selects the Top-K tools by cosine similarity (default K=20) using BGE-M3 embeddings (Chen et al., 2024). Each retrieved tool is formatted as a Tool Card containing tool name and description, together with the finance attributes a_t. The combination reduces distractors and makes prompting more stable under large catalogs.

4.2 Attribute-Aware Planning

The LLM Planner runs a ReAct loop (Yao et al., 2022): it proposes a tool call, observes tool outputs, and iterates until producing a final answer. We cap the interaction horizon at `max_steps=5` tool-augmented steps to limit latency and reduce exposure to tool drift. During planning, FATR makes three families of constraints explicit to the planner: timeliness (*e.g.*, match implied time sensitivity), intent restraint (*e.g.*, avoid transactional tools unless explicitly required, and in FinToolBench transac-

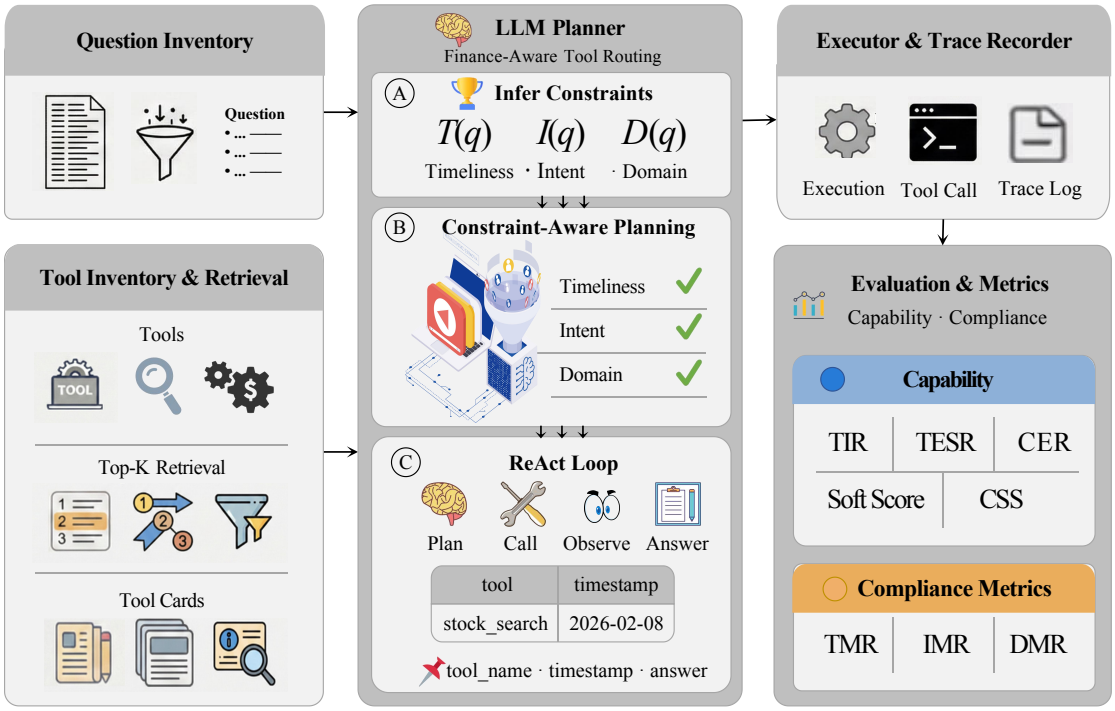

Figure 3: Overview of Finance-Aware Tool Routing (FATR). FATR takes a Question Inventory and a Tool Inventory & Retrieval module that performs Top-K retrieval and formats retrieved tools as Tool Cards. An LLM Planner runs Finance-Aware Tool Routing by (A) Infer Constraints to derive $(T(q), I(q), D(q))$ over timeliness, intent, and domain, (B) Constraint-Aware Planning, and (C) a ReAct loop. An Executor & Trace Recorder dispatches tool calls and trace logs, which are then scored by Evaluation & Metrics for capability (TIR, TESR, CER, SOFT SCORE, CSS) and compliance (TMR, IMR, DMR).

tional intent is treated as disallowed and penalized), and domain alignment (*e.g.*, ensure intersection between the tool domain and the inferred question domain). In practice, these constraints are implemented as explicit prompt rules that guide the planner's tool selection and reasoning.

Concretely, the planner first performs Infer Constraints by articulating the implied requirement sets $(T(q), I(q), D(q))$. It then performs Constraint-Aware Planning by selecting tools whose attributes are compatible with these sets and by maintaining the constraints throughout the ReAct loop. For multi-tool questions, the planner is encouraged to resolve ambiguity early (*e.g.*, determine the correct market and ticker format) before executing downstream calls whose domains must remain consistent. The planner is also instructed to surface key provenance fields in the final answer so that tool use is verifiable. Figure 4 illustrates our tool card format, which standardizes tool metadata and exposes finance-specific attributes used for both retrieval and constraint checking. While the primary goal is to evaluate models rather than to enforce policy, FATR can optionally apply conservative hard filters at inference time (*e.g.*, excluding transactional tools and removing domain-incompatible tools) to reduce unforced compliance errors. Pseudocode for the full pipeline is given in Appendix I.

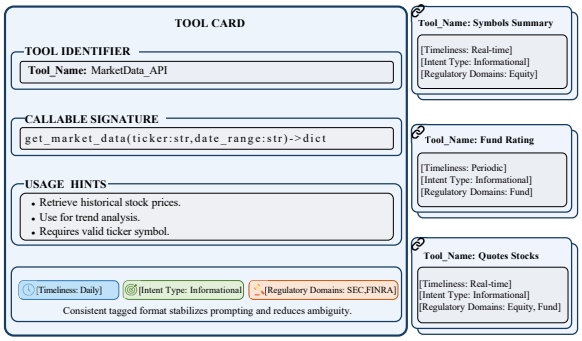

Figure 4: Tool cards for attribute injection and constraint checking.

5 Experiments

5.1 Benchmark Setting

The following protocol is designed so that results on FinToolBench can be reproduced and compared fairly across studies. The benchmark contains 295 tool-required questions and 760 runnable financial

tools. Each run uses a fixed tool-use limit, *i.e.*, at most 5 tool-use rounds per question. In each round, multiple tool calls may be issued, with a per-call timeout of 60 seconds and up to 2 retries. Tool execution is performed in a controlled environment with deterministic caching and full logging of tool traces. For analysis, questions are stratified by single vs. multi-tool usage and by inferred question category. We report metrics following the definitions in Section 3.4. All headline results are computed on the full 295-question benchmark under the same fixed evaluation protocol.

5.2 Baselines and Model Backends

We evaluate FinToolBench under a unified agent framework based on FATR, which integrates tool retrieval, finance-attribute injection, and stabilized execution. Unless otherwise specified, the pipeline (retriever and executor) is fixed and only the LLM planner varies across seven backends: Doubao-Seed-1.6, Qwen3-8B, GLM-4.7-Flash, Claude-Sonnet-4.5, GPT-5.4, Grok-3-beta, and Gemini-3.1-Pro (preview). All models share the same tool-call interface: the planner outputs either a final answer or a structured tool invocation specifying a tool ID and JSON arguments derived from the tool signature; outputs are returned in a normalized format, and retrieved tools are converted into function schemas with the finance tags (timeliness, intent type, regulatory domains) prepended to each tool description. The prompt emphasizes (i) using tools when timeliness is required, (ii) avoiding transactional actions, and (iii) explicitly checking that the selected tools' domains match the question. When tool outputs exceed a length threshold, an LLM-based extractor compresses responses to question-relevant fields before returning them to the planner. For answer correctness and requirement inference, we employ an LLM-as-a-judge. Implementation details, prompt templates, and compression settings are reported in Appendix H and Appendix F.

5.3 Judges and Agreement

For computing SOFT SCORE, we use GPT-5.1 as the judge and repeat each judgment three times to reduce variance, averaging the three judge scores. For compliance evaluation in Eq. (1), the current implementation uses GPT-5.1 and performs one LLM-judge decision per tool call for each mismatch dimension. We report TMR, IMR, and DMR conditioned on traces with at least one tool call; therefore these mismatch rates should be interpreted together with TIR and TESR, since conservative models can obtain low mismatch rates by attempting fewer tool-required questions. To validate the LLM-based pipeline, three finance-domain experts independently annotate a stratified sample of 60 questions and 50 traces. Inter-expert agreement is 85.3%, and expert-vs-pipeline agreement is 83.8% with F1 0.81 after second-round disagreement review. Additional robustness checks, including prompt-order sensitivity and a Qwen3-8B compliance re-judge, are reported in Appendix G.

6 Results

6.1 Main Results

Table 2 reports the main results on FinToolBench. **Tool invocation does not imply execution success.** Qwen3-8B invokes tools most often (TIR =0.8712), but its lower CER shows that aggressive tool use does not necessarily translate into reliable execution. Doubao-Seed-1.6 achieves the best end-to-end execution success (TESR =0.3254), suggesting a stronger balance between attempting tool-required questions and completing executions successfully. **Answer quality and trace compliance diverge.** Claude-Sonnet-4.5 and Gemini-3.1-Pro obtain stronger semantic scores, while Grok-3-beta is the most precise among attempted tool traces with the highest CER and low mismatch rates. GPT-5.4 shows conservative tool use with moderate conditional precision, and GLM-4.7-Flash is weaker across most capability and quality metrics. Together, these patterns show that FinToolBench separates coverage, execution reliability, answer quality, and finance-specific compliance rather than collapsing them into a single success rate. They also indicate that model ranking depends on the operational objective: a research assistant may favor higher semantic quality, whereas a monitoring workflow may prioritize low mismatch rates and auditable traces.

6.2 Finance Attribute Injection

We compare full FATR against a variant that omits finance tags from the tool cards while keeping the same retriever and executor. We perform this ablation using Doubao-Seed-1.6 as the planner. The no-injection baseline obtains TIR =0.7254, TESR =0.3356, CER =0.4626, SOFT SCORE =0.4530, CSS =0.3940, TMR =0.4110, IMR =0.6640, and DMR =0.2150. Figure 5 shows that attribute injection mainly changes selection behavior: it can

Table 2: Main results on FinToolBench. Higher is better for TIR, TESR, CER, SOFT SCORE, and CSS; lower is better for TMR, IMR, and DMR.

Model	TIR	TESR	CER ↑	SOFT SCORE ↑	CSS ↑	TMR ↓	IMR ↓	DMR ↓
Doubao-Seed-1.6	0.6508	0.3254	0.5000	0.4627	0.3958	0.3438	0.6563	0.1719
Qwen3-8B	0.8712	0.2949	0.3385	0.4040	0.4234	0.3307	0.6887	0.1673
GLM-4.7-Flash	0.4407	0.2102	0.4769	0.3791	0.2769	0.4615	0.7231	0.1769
Claude-Sonnet-4.5	0.4814	0.2407	0.5000	0.7119	0.5775	0.2746	0.6479	0.0423
GPT-5.4	0.2407	0.1322	0.5493	0.6254	0.4103	0.2535	0.5775	0.0141
Grok-3-beta	0.2576	0.1932	0.7500	0.6847	0.4298	0.1316	0.4868	0.0132
Gemini-3.1-Pro (preview)	0.1525	0.1186	0.7778	0.8220	0.5857	0.1333	0.2667	0.0444

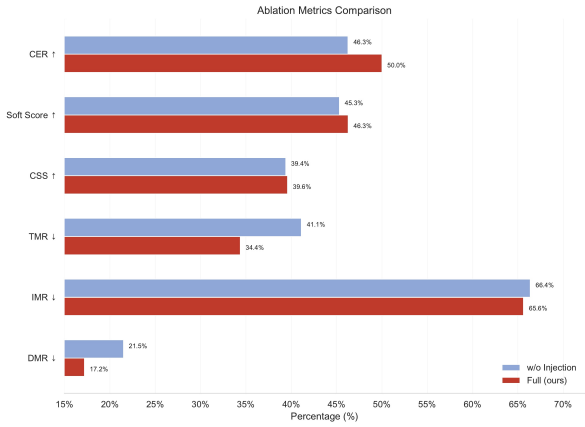

Figure 5: Attribute injection ablation in FATR. We compare full tool cards with finance tags against a variant without attribute injection. Attribute injection improves execution success conditioned on tool use (CER) and reduces mismatch rates (TMR, IMR, DMR).

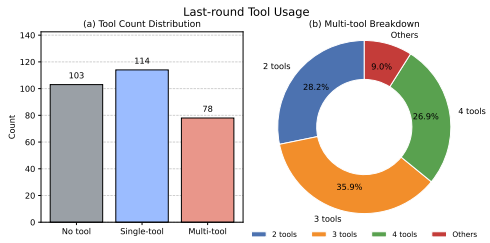

Figure 6: Last-round tool usage on FinToolBench.

reduce marginal calls, improve conditional execution reliability, and reduce mismatch rates across timeliness, intent, and domain dimensions. This supports the role of finance attributes as routing constraints rather than merely descriptive metadata. The effect is most relevant when a question underspecifies whether the answer needs current market data, historical filings, or policy-level sources, because attribute tags help rule out plausible but misaligned tools before execution. In practice, this makes the tool card closer to a financial interface contract: the planner must match not only argument types, but also the evidential role and regulatory scope of the call.

6.3 Tool Usage Distribution

Figure 6 shows that 103/114/78 of the 295 evaluation instances end with no/single/multiple final-round tool calls. Doubao-Seed-1.6 makes 923 tool calls and exercises 236 distinct tools; the gold required-tool distribution is 1/2/3/4/5/6 tools = 166/64/49/12/3/1. Thus, the 760-tool inventory functions as a realistic routing space, not as a per-tool unit-test suite. The long-tail usage pattern suggests that stronger agents should combine broad retrieval with restraint, using multi-call traces mainly when evidence aggregation is required.

6.4 Category-Level Diagnosis

Appendix J further breaks down capability and compliance by category, showing that aggregate scores can hide failures specific to value extraction, macro interpretation, and other tool scopes. This diagnosis helps distinguish tool-selection failures from rigid output-format or answer-scoring mismatches. This is important in finance, where superficially similar questions may require different data freshness, tool domains, and compliance constraints.

7 Conclusion

We present FinToolBench, a runnable benchmark for financial tool-use agents with 760 free-tier tools and 295 tool-required questions. Its evaluation separates capability metrics from compliance metrics over timeliness, intent, and domain constraints. Across seven LLM backends, no model dominates both sides: aggressive callers improve coverage but introduce noisy traces, while conservative models produce cleaner but sparser tool use. We also provide FATR as a reference baseline and release the tool manifest, question set, and evaluation scripts to support reproducible comparison. Future work may extend FinToolBench to paid real-time data feeds, richer market coverage, and stronger policy-constrained agents, while preserving the trace-level auditability that financial tool use requires.

Limitations

FinToolBench covers 760 free-tier tools and 295 tool-required questions, not paid terminals, brokerage or order-routing systems, or all markets and jurisdictions. Its metrics are benchmark checks, not professional compliance review. API, data, and model drift, together with residual LLM-judge sensitivity, mean results should be read as a dated protocol snapshot; FATR is diagnostic, not deployable.

Ethical Considerations

FinToolBench is a research benchmark, not financial advice or authorization for regulated automation. It uses read-only or sandboxed tools, disallows transactions, excludes personal data, credentials, paid-feed content, and non-public financial information, and any production use would require license compliance, privacy protection, human oversight, security validation, and review.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik R Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Antoine Bigeard, Langston Nashold, Rayan Krishnan, and Shirley Wu. 2025. Finance agent benchmark: Benchmarking llms on real-world financial research tasks. *arXiv preprint arXiv:2508.00828*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 4(5).

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, and 1 others. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*.

Jennifer D'Souza, Hamed Babaei Giglou, and Quentin Münch. 2025. Yescieval: Robust llm-as-a-judge for scientific question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13749–13783.

Jiayu Guo, Yu Guo, Martha Li, and Songtao Tan. 2025a. Flame: Financial large-language model assessment and metrics evaluation. *arXiv preprint arXiv:2501.06211*.

Xin Guo, Haotian Xia, Zhaowei Liu, Hanyang Cao, Zhi Yang, Zhiqiang Liu, Sizhe Wang, Jinyi Niu, Chuqi Wang, Yanhui Wang, and 1 others. 2025b. Fineval: A chinese financial domain knowledge evaluation benchmark for large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6258–6292.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156.

Rajarshi Haldar and Julia Hockenmaier. 2025. Rating roulette: Self-inconsistency in llm-as-a-judge frameworks. *arXiv preprint arXiv:2510.27106*.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. 2024. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13806–13834.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Yankai Jiang, Wenjie Lou, Lilong Wang, Zhenyu Tang, Shiyang Feng, Jiaxuan Lu, Haoran Sun, Yaning Pan, Shuang Gu, Haoyang Su, and 1 others. 2025. Scp: Accelerating discovery with a global web of autonomous scientific agents. *arXiv preprint arXiv:2512.24189*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3102–3116.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pages 52989–53046.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522.

Jiaxuan Lu, Ziyu Kong, Yemin Wang, Rong Fu, Haiyuan Wan, Cheng Yang, Wenjie Lou, Haoran Sun, Lilong Wang, Yankai Jiang, and 1 others. 2026. Beyond static tools: Test-time tool evolution for scientific reasoning. *arXiv preprint arXiv:2601.07641*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pages 9025–9049.

OpenCompass. 2024. OpenFinData: An open financial evaluation dataset. GitHub repository.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina Stańczak, and Siva Reddy. 2025. Safearena: Evaluating the safety of autonomous web agents. *arXiv preprint arXiv:2503.04957*.

Haiyuan Wan, Chen Yang, Junchi Yu, Meiqi Tu, Jiaxuan Lu, Di Yu, Jianbao Cao, Ben Gao, Jiaqing Xie, Aoran Wang, and 1 others. 2026. Deep research arena: The first exam of llms' research abilities via seminar-grounded tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33341–33349.

Hongfei Xia, Hongru Wang, Zeming Liu, Qian Yu, Yuhang Guo, and Haifeng Wang. 2025. Safetoolbench: Pioneering a prospective benchmark to evaluating tool utilization safety in llms. *arXiv preprint arXiv:2509.07315*.

Cheng Yang, Jiaxuan Lu, Haiyuan Wan, Junchi Yu, and Feiwei Qin. 2025. From what to why: A multi-agent system for evidence-based chemical reaction condition reasoning. *arXiv preprint arXiv:2509.23768*.

S Yao, J Zhao, D Yu, N Du, I Shafran, K Narasimhan, and YR Cao. 2022. Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Qiyuan Zhang, Yufei Wang, Yuxin Jiang, Liangyou Li, Chuhan Wu, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, Fuyuan Lyu, and 1 others. 2025. Crowd comparative reasoning: Unlocking comprehensive evaluations for llm-as-a-judge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5059–5074.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 3277–3287.

A Tool Curation Criteria

This section spells out the criteria used to build the FinToolBench tool inventory. The pipeline retains only tools that are executable under free-tier constraints.

A.1 RapidAPI Endpoints

Our RapidAPI pool is initialized from finance-related tools collected from the ToolBench paper, and then filtered with the two-stage pipeline below.

First-stage (rule-based) filters. Because RapidAPI listings vary in how authentication, billing, and rate limits are described, we programmatically crawl and parse endpoint pages to extract these fields, and then apply the deterministic criteria in Table 3. An endpoint is excluded if it fails any row.

Table 3: First-stage RapidAPI filter criteria.

Criterion	Rule
Description	Missing or empty → excluded.
Deduplication	Duplicate names: keep first only.
Authentication	Exclude if `Authorization` or multi-step token (beyond single API key) required.
Existence	Remove if endpoint does not resolve or returns persistent errors.
Bank-card	Exclude if free-tier requires card binding.
Rate limits	Require $\geq$10/h, $\geq$100/d, $\geq$300/m.
Map	Use an LLM to align endpoint parameter names with the tool signature and then manually spot-check the mappings.

Second-stage (mapping and executability). Each endpoint that passes the first stage is mapped into our normalized schema. We align its parameter names to our tool signature with an LLM and then manually spot-check the mappings. We then test each mapped endpoint with at least one successful invocation (valid request and parsed response). Endpoints that fail consistently (timeouts, validation errors, or empty responses) are dropped.

Outcome. After the two-stage filtering process, we obtain 261 executable RapidAPI endpoints. All retained endpoints have at least one documented required or optional parameter. Endpoints that are parameter-free or rely solely on implicit path/query conventions are excluded.

A.2 AkShare Interfaces

AkShare functions are selected and then validated for executability.

Finance-domain filter. Function names are matched against a fixed set of finance-related keywords. A function is retained if its name (or module path) contains any of the following tokens:

- `stock`, `fund`, `bond`, `futures`, `option`, `index`, `macro`, `currency`, `fx`, `crypto`, `rate`, `treasury`, `etf`
- `finance`, `bank`, `insurance`, `security`, `derivative`, `swap`, `gold`, `commodity`, `interest`, `libor`, `shibor`, `exchange`, `margin`

Executability. Each candidate is invoked with minimal valid arguments, using defaults or small example values where possible. Interfaces that raise import errors, signature errors, or runtime errors under a timeout are discarded.

Outcome. After finance-domain filtering and executability validation, we retain 499 AkShare interfaces in the final tool inventory.

A.3 Combined Inventory

After executability filtering, the final tool library contains 760 tools. All tools in FinToolBench are normalized into a single manifest schema, including tool name, description, and parameters. This unified manifest enables auditable tool traces and call-level compliance metrics (TMR, IMR, DMR), where each run records the invoked tools together with their arguments and execution outcomes.

B Tool Trace Schema

Each tool invocation is captured as a structured execution trace. The schema (Table 4) records the chronological context through a sequential `step` index, the specific `tool_name`, the JSON-formatted `parameters` generated by the model, and both the raw `output` and any execution `error`, so as to differentiate model reasoning errors from system-level failures such as API rejections.

C Finance Attribute Schema and Labeling

Each tool is annotated with finance attributes that make timeliness, intent, and domain constraints explicit. These attributes support both tool cards (for planning) and compliance evaluation (TMR, IMR, DMR).

Table 4: Normalized tool-trace fields.

Field	Description
`step`	The sequential order of the call within the multi-turn process.
`tool_name`	The identifier of the specific tool invoked (*e.g.*, `symbols_sec_filings`).
`parameters`	The JSON-formatted arguments generated by the model.
`output`	The tool response, including data or structured error messages.
`error`	Overall execution status, which captures null or specific system-level failures.

C.1 Attributes of the Tools

- **Timeliness** (`timeliness`): one of `realtime`, `daily`, `as_filed`, `periodic`, or `static`.
  - `realtime`: intra-day, low latency, such as ticks and order book.
  - `daily`: updated once per trading day or batch, such as closing prices and NAV.
  - `as_filed`: event-driven, when a regulated entity files, such as filings and announcements.
  - `periodic`: fixed calendar schedule, such as quarterly reports and GDP.
  - `static`: rarely changes, such as identifiers and listing dates.
- **Intent type** (`intent_type`): one of `informational`, `advisory`, or `transactional`.
  - `informational`: read-only data access and factual retrieval without recommendations or actions.
  - `advisory`: analysis or recommendation-oriented outputs that go beyond pure retrieval.
  - `transactional`: action-triggering operations such as order placement, transfer, or account-changing behavior.
- **Regulatory domain** (`regulatory_domain`):
  Subset of {equity, bond, fund, forex, derivatives, macro,
  economic_policy, sentiment_trading, esg, crypto}. Multiple values are allowed per tool.

C.2 Labeling Protocol

Annotations are produced by an LLM (Qwen3-8B) given the tool name and description. Each attribute is labeled independently with three samples, and the final label is the majority vote. This protocol keeps labeling separate from the agent under test and makes the compliance layer auditable.

D Question Set Construction

The question set is built so that every retained question requires tool use. Questions answerable by static memorization or general reasoning are excluded.

D.1 Sources

Questions are drawn from the FinanceBench release and the OpenFinData release (dataset identifier `openfindata_release`). Only questions that require tool calls to answer are retained, such as current prices, time series, or structured data.

D.2 Filters

- **Standardization:** Convert all sources into a unified `{question, answer, category}` format.
- **Maximum length:** 500 characters to keep prompts within a fixed budget.
- **Candidate retrieval:** For each question, retrieve Top-K tools (K=20) using BGE-M3 embeddings.
- **LLM tool selection:** Given the Top-K tools, Qwen3-8B selects the most suitable tool(s). We sample three times and keep tools with at least two votes.
- **Deduplication:** For single-tool questions, keep at most two questions per tool to reduce skew. Multi-tool questions are not deduplicated by tool set.

D.3 Outcome

The final question set contains 295 questions, spanning both simple and compositional tool use: 166 single-tool questions and 129 multi-tool questions, classified by the number of aligned required tools.

D.4 Human Verification

We conduct human-in-the-loop quality assurance as an artifact validation step rather than as an experiment on human subjects. We use stratified sampling to select 60 questions and 50 execution traces, covering single-tool and multi-tool items as well as the three compliance dimensions. Three domain experts in financial data analysis independently check benchmark items against a fixed rubric: whether the question truly requires tool use, whether the

aligned tool or tool sequence is logically necessary, whether the annotated timeliness, intent type, and regulatory domain are plausible, and whether the expected answer format is compatible with the available tool outputs. Reviewers do not provide personal data, are not evaluated as study participants, and only inspect benchmark artifacts. Disagreements are resolved in a second-round review with the benchmark maintainers, and unresolved items are removed from the release. The resulting inter-expert agreement is 85.3%. Against the automatic pipeline, expert agreement is 83.8% with F1 0.81. Dimension-level results are shown in Table 5.

Table 5: Human validation of question-side attributes and trace-level compliance.

Validation target	Agreement	Secondary metric
Question timeliness	85.0%	Jaccard 0.82
Question intent	93.3%	Jaccard 0.91
Question domain	80.0%	Jaccard 0.76
Trace TMR	86.0%	F1 0.83
Trace IMR	94.0%	F1 0.91
Trace DMR	82.0%	F1 0.78

E Metric Definitions

All metrics are defined over a fixed question set and a fixed evaluation protocol (timeout, retries, caching). We report capability (TIR, TESR, CER) and compliance (TMR, IMR, DMR) to separate whether an agent can run tools from whether its tool choices satisfy finance constraints. Below, N denotes the number of questions.

E.1 Capability

- **Tool Invocation Rate (TIR).** Fraction of questions whose execution invokes at least one tool call:

$$\mathrm{TIR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\,|\mathrm{tool_calls}_i| > 0\,\right].$$

- **Tool Execution Success Rate (TESR).** Fraction of questions whose tool execution succeeds. We mark a question successful when its final tool call returns a valid parsed output without error or exception; intermediate failures and retries are allowed:

$$\mathrm{TESR} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\,\text{final tool call succeeds on question } i\,\right]$$

- **Conditional Execution Rate (CER).** Success rate among questions that invoked at least one tool call:

$$\mathrm{CER} = \frac{\mathrm{TESR}}{\mathrm{TIR}}, \quad \mathrm{CER} = 0 \text{ when } \mathrm{TIR} = 0.$$

- **Soft Score (SOFT SCORE).** We partition questions into three types based on the gold-answer format: (i) numeric/choice questions, identified by the presence of a `numeric` or `choices` field in the gold answer; (ii) structured-analysis questions, identified by the presence of a `criterium` field in the gold answer; (iii) `other` questions, all remaining non-numeric/choice/`criterium` cases. All questions are judged by LLM (GPT-5.1). Numeric/choice questions are scored in $\{1, 0\}$, while structured-analysis and `other` questions are scored in $\{1, 0.5, 0\}$. Scores are averaged over three judging repeats. When a gold answer provides a machine-readable numeric value or option label, a deterministic exact-match or tolerance-match audit can be added as a complementary check; the current headline metric uses the LLM score because several questions require explanatory or structured answers.

$$\mathrm{SoftScore} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{r=1}^{3} s_{i,r}.$$

- **Conditional Soft Score (CSS).** Mean SOFT SCORE over questions with successful execution:

$$\mathrm{CSS} = \frac{\sum_{i=1}^{N} e_i \left(\frac{1}{3}\sum_{r=1}^{3} s_{i,r}\right)}{\sum_{i=1}^{N} e_i},$$

$$\mathrm{CSS} = 0 \quad \text{when } \sum_{i=1}^{N} e_i = 0.$$

where $e_i = \mathbf{1}[\text{final tool call succeeds on question } i]$.

E.2 Compliance Mismatch Rates

For each question q, an LLM(GPT-5.1) judge assesses whether the tools used in the execution are aligned with the question's finance constraints in timeliness, intent type, and regulatory domain. Let the executed tool-use trace be $\tau = \{(t_k, x_k, o_k)\}_{k=1}^{m}$, where t_k is the tool name selected at step k, x_k is the tool input (arguments), and o_k is the tool output. Tool attributes are

not part of the trace; instead, for any tool t we look up its finance tags from the tool metadata: $A(t) = (\tau_t(t), i(t), d(t))$, corresponding to timeliness $\tau_t(t)$, intent type $i(t)$, and regulatory domains $d(t)$.

Define judge-level call alignment indicators: $J_T(q, t_k, x_k, o_k, \tau_t(t_k)) \in \{0, 1\}$, $J_I(q, t_k, x_k, o_k, i(t_k)) \in \{0, 1\}$, and $J_D(q, t_k, x_k, o_k, d(t_k)) \in \{0, 1\}$, where 1 means matched and 0 means mismatched. We then define mismatch-at-least-once indicators at the question level:

$$\begin{aligned} \mathrm{TMF}(q, \tau) &= \mathbf{1}[\exists k : J_T(q, t_k, x_k, o_k, \tau_t(t_k)) = 0], \\ \mathrm{IMF}(q, \tau) &= \mathbf{1}[\exists k : J_I(q, t_k, x_k, o_k, i(t_k)) = 0], \\ \mathrm{DMF}(q, \tau) &= \mathbf{1}[\exists k : J_D(q, t_k, x_k, o_k, d(t_k)) = 0]. \end{aligned}$$

TMR, IMR, and DMR are computed over questions with at least one executed tool call, and measure the fraction of traces that contain at least one mismatch in the corresponding dimension. Lower is better. Let

$$\mathcal{Q}_{\text{tool}} = \{\, q_i \mid i \in \{1, \ldots, N\}, |\text{tool_calls}_i| > 0 \,\}$$

denote the set of questions with at least one executed tool call. We then define

$$\begin{aligned} \mathrm{TMR} &= \frac{1}{|\mathcal{Q}_{\text{tool}}|} \sum_{q_i \in \mathcal{Q}_{\text{tool}}} \mathrm{TMF}(q_i, \tau_i), \\ \mathrm{IMR} &= \frac{1}{|\mathcal{Q}_{\text{tool}}|} \sum_{q_i \in \mathcal{Q}_{\text{tool}}} \mathrm{IMF}(q_i, \tau_i), \\ \mathrm{DMR} &= \frac{1}{|\mathcal{Q}_{\text{tool}}|} \sum_{q_i \in \mathcal{Q}_{\text{tool}}} \mathrm{DMF}(q_i, \tau_i). \end{aligned}$$

Because these rates are conditional on tool invocation, they diagnose the quality of attempted tool traces but do not by themselves reward coverage. For this reason, the main table reports them jointly with TIR and TESR so that coverage and attempted-trace quality are visible together.

F Prompt Templates

This section documents the prompts for (i) intent mismatch evaluation, (ii) planner system instructions, and (iii) RapidAPI parameter alignment.

F.1 Intent-Match Evaluator (IMR)

We use an LLM (GPT-5.1) judge to decide whether a tool call's `intent_type` is compatible with the question requirement. The judge considers only intent (informational, advisory, transactional) and ignores execution success.

Intent-match evaluator prompt (JSON-only output)

```
You are a financial tool compliance
evaluator. Determine whether the
"question requirement" matches the
"tool intent".
Consider only tool intent. Do not
consider whether the tool call
succeeds.
Tool intent must be one of
informational / advisory /
transactional.

Question: {question}
Tool name: {tool_name}
Tool intent (intent_type):
{intent_type}
Tool description: {description}

Output JSON only: {"match":
"yes|no", "reason": "within 20
words"}
```

F.2 Planner System

The planner is explicitly instructed to use the `financial_tags` field inside each tool description when choosing tools. This makes timeliness, domain, and intent cues salient at selection time.

Planner system

```
Subject to satisfying the question,
first infer the question's required
finance tags: (a) required
timeliness, (b) required
regulatory_domains, and (c) allowed
intent_type.
Then select tools whose
financial_tags best match these
requirements: require domain
overlap, prefer matching timeliness,
and choose the lowest-risk intent
(informational > advisory; avoid
transactional unless explicitly
requested).
If multiple tools match, choose the
one whose signature/description best
fits the needed inputs/outputs.
```

F.3 RapidAPI Parameter Alignment

RapidAPI documentation often uses parameter names that differ from those used in request examples. To reduce argument instantiation errors, we generate normalized Python wrappers whose function parameters exactly match the keys used in the request schema, and we verify mappings with manual spot-checking (Table 3, Map rule). For security, the API key is shown as a placeholder.

Parameter-aligned Python code generation prompt

```
You are a professional Python tool
function generator.  Rewrite the
following raw API call code into a
standard reusable function.  Strictly
follow the rules below.

1.  Function name:  {func_name}
2.  Function parameters:
  - All input parameters:
{param_str}
  - Additional fixed parameter:
rapidapi_key:  str = "<RAPIDAPI_KEY>"
3.  The function must:
  - Use requests to send the HTTP
request
  - Use X-RapidAPI-Key and
X-RapidAPI-Host in headers
  - Return response.json() if
possible, otherwise return
response.text
  - Do not use print; always return
  - Include a full docstring
describing the function and every
parameter
  - Output pure Python code only
  - Parameter names must exactly
match the keys in the parameter
dictionary (case sensitive)

Description:  {desc}
Parameter spec:  {parameters}
URL: {url}
X-RapidAPI-Host:  {host}
```

G Additional Robustness Analyses

G.1 Retrieval Depth

We evaluate how often the gold required tool set is covered by the retriever as the number of retrieved candidates changes. Table 6 shows that retrieval improves sharply up to K=20 and then saturates, motivating the default K=20 used in the main experiments.

Table 6: Retrieval-depth ablation. Tool Hit@K measures whether the gold required tool set is covered by the retrieved candidate pool.

K	1	5	10	20	30
Tool Hit@K	61.2%	75.1%	83.0%	88.1%	88.3%

G.2 Judge Stability and Prompt Order

For answer scoring, the three-repeat GPT-5.1 judge produces a representative SOFT SCORE standard deviation of 0.02 and a 95% confidence interval of approximately ±0.03. For compliance prompt order, we compare the original IMR prompt, which asks for the binary decision before the rationale, against a reversed-order variant on 50 sampled IMR cases. The binary labels agree on 94.0% of samples; the overall IMR differs by 1.0 percentage point (18.0% vs. 19.0%), with a 6.0% sample-level flip rate.

G.3 Alternative Compliance Judge

To test whether compliance conclusions depend on GPT-5.1 alone, we rerun requirement inference with Qwen3-8B on the same traces. For Doubao-Seed-1.6, Qwen3-8B gives TMR/IMR/DMR = 0.3346/0.6848/0.1673, close to GPT-5.1's 0.3438/0.6563/0.1719. This does not remove all judge noise, but it suggests that the broad compliance pattern is not an artifact of a single judge model.

H Commands and Reproduction Checklist

The following enables independent reproduction of the benchmark and evaluation pipeline so that results can be compared fairly across studies.

H.1 Environment and Models

RapidAPI keys are obtained following the official documentation. AkShare usage follows its documentation. The retriever is BGE-M3, and the output compressor uses Qwen3-8B. Planner backends are Doubao-Seed-1.6, Qwen3-8B, GLM-4.7-Flash, Claude-Sonnet-4.5, GPT-5.4, Grok-3-beta, and Gemini-3.1-Pro (preview). The judge is GPT-5.1 with three repeats for SOFT SCORE and one decision per tool call for requirement metrics in the current implementation. The released code and configuration specify the evaluation stack, including prompts, decoding settings, timeout, retry count, maximum tool steps, cache setting, and prompt-template hash. Trace logs can be replayed or rejudged under later model versions.

H.2 Artifact Documentation and Licensing

The release contains four artifact families: (i) the normalized tool manifest, (ii) the 295-question benchmark set, (iii) the evaluator and FATR reference implementation, and (iv) cached execution traces and schema snapshots used for replay. For each tool, the manifest records source ecosystem (RapidAPI or AkShare), normalized signature, finance attributes, schema extraction date, last successful execution date, and whether a cached output is available. RapidAPI endpoints are invoked under the endpoint providers' free-tier terms through user-supplied credentials; we do not redistribute

Algorithm 1 FATR pipeline

Require: Question q, tool library $\mathcal{T}$, retriever $\mathcal{R}$, planner LLM π, Top-K (default K=20)
Ensure: Final answer $\hat{y}$, tool trace τ
$\mathcal{C} \leftarrow \mathcal{R}(q, \mathcal{T}, K)$ ▷ Retrieve Top-K tools by similarity
Build tool cards: for each $t \in \mathcal{C}$, format $\mathrm{Card}(t)$ with signature and finance attributes
$\tau \leftarrow \emptyset$, $h \leftarrow \emptyset$ ▷ History of tool calls and outputs
repeat
  Planner π proposes next action given $(q, \{\mathrm{Card}(t)\}_{t\in\mathcal{C}}, h)$: tool t and arguments x (or final answer $\hat{y}$)
  if action is tool call (t, x) **then**
    Validate $t \in \mathcal{C}$
    Execute: $o \leftarrow \mathsf{Exec}(t, x)$ with timeout, optionally compress o
    Append (t, x, o) to τ, update h with observation
  else
    $\hat{y} \leftarrow$ planner output, **break**
until planner emits final answer or max steps
return $\hat{y}$, τ

API keys or paid-feed outputs. AkShare interfaces are documented with the upstream package version and license metadata. FinanceBench, OpenFinData, BGE-M3, RapidAPI, and AkShare are cited as upstream artifacts, and derivative benchmark files are distributed only under terms compatible with the source artifacts. The intended use of the release is research evaluation of financial tool-use agents, trace auditing, and reproducibility studies. It is not intended for live trading, client advice, transaction execution, or production compliance certification.

H.3 PII and Sensitive-Content Audit

Before release, we scan questions, tool manifests, prompts, and cached traces for personal identifiers, credentials, API keys, account numbers, emails, phone numbers, and non-public company information. Secret-like strings are replaced with placeholders, and any item requiring user-specific account data is excluded. The benchmark uses public or free-tier financial information and does not include private user portfolios, bank records, or proprietary market data. Because live API outputs can change, users who regenerate traces are instructed to repeat the same audit before redistribution.

I Algorithm — FATR Pipeline

The algorithm summarizes FATR: retrieval, tool-card formatting, ReAct-style planning with finance constraints, and stabilized execution with timeout, retries, cache, and optional output compression. When compression is enabled, Qwen3-8B extracts question-relevant fields from long tool responses.

J Category-Level Diagnosis

To further localize failure modes beyond aggregate averages, we report metrics broken down by question category. Figure 7 presents category-level performance for Doubao-Seed-1.6. The heatmap highlights substantial heterogeneity across categories. Categories with low TIR cap end-to-end execution success by limiting tool coverage. In contrast, categories with high TIR but low TESR indicate difficulties in argument construction, disambiguation, or endpoint instability even after the agent commits to tool use.

K Case Studies

This section presents representative execution traces with condensed tool calls, final answers, correctness labels, and trace-level analysis.

K.1 Case Study 1: Finance Attribute Injection Changes Tooling Behavior

This case examines whether finance-attribute injection changes tool selection when the first attempted interface is partially incompatible with the benchmark wrapper. We compare the no-attribute baseline against FATR under the same retrieval pool and execution budget.

K.2 Case Study 2: Finance Attributes Reduce Redundant Tooling

This case focuses on trace efficiency: the question can be answered with one fund-specific informational endpoint, so extra calls mostly reflect planner uncertainty.

K.3 Case Study 3: Finance Attributes Do Not Guarantee Numerical Fidelity

This case isolates a harder failure mode: the required numeric target is not directly exposed by the available tools, so the planner must choose between abstaining and using a proxy.

Metrics by Category

	TIR	TESR	CER ↑	Soft Score ↑	CSS ↑	TMR ↓	IMR ↓	DMR ↓
announcement interpretation	63.33%	10.00%	15.79%	40.00%	50.00%	36.84%	68.42%	5.26%
business compliance	60.00%	10.00%	16.67%	23.33%	0.00%	33.33%	100.00%	16.67%
data inspection	83.33%	50.00%	60.00%	50.00%	66.67%	20.00%	60.00%	0.00%
domain-relevant	71.43%	64.29%	90.00%	13.10%	16.67%	20.00%	50.00%	20.00%
entity disambiguation	58.33%	50.00%	85.71%	66.67%	66.67%	28.57%	42.86%	14.29%
entity recognition	50.00%	18.75%	37.50%	81.25%	83.33%	25.00%	75.00%	37.50%
financial facts	54.55%	18.18%	33.33%	81.82%	50.00%	50.00%	66.67%	16.67%
fund analysis	76.47%	58.82%	76.92%	24.51%	13.33%	15.38%	38.46%	7.69%
industry interpretation	72.73%	24.24%	33.33%	35.86%	64.58%	50.00%	87.50%	41.67%
intent understanding	0.00%	0.00%		100.00%				
macro interpretation	85.71%	32.14%	37.50%	41.67%	40.74%	50.00%	91.67%	20.83%
metric calculation	0.00%	0.00%		100.00%				
metrics-generated	80.00%	70.00%	87.50%	12.50%	17.86%	12.50%	18.75%	0.00%
novel-generated	66.67%	66.67%	100.00%	38.89%	58.33%	50.00%	66.67%	16.67%
sector analysis	33.33%	0.00%	0.00%	88.89%		0.00%	100.00%	0.00%
security compliance	25.00%	0.00%	0.00%	39.58%		50.00%	100.00%	50.00%
stock analysis	60.61%	45.45%	75.00%	62.12%	43.33%	0.00%	35.00%	0.00%
stock interpretation	78.12%	21.88%	28.00%	44.79%	40.48%	60.00%	84.00%	24.00%
value extraction	0.00%	0.00%		100.00%				

Figure 7: Metrics by category for Doubao-Seed-1.6 on FinToolBench. Rows are question categories and columns are evaluation metrics.

Case Study 1 (`financebench_9`): injection recovers tool execution but not task framing

Question. Does AMEX have an improving operating margin profile as of 2022? If operating margin is not a useful metric for a company like this, then state that and explain why.

Ground truth. "Performance is not measured through operating margin."

Baseline (no finance attributes).
Tool trace.
Step 1: `symbols_get_fundamentals` → error (unexpected keyword argument `fields`).
Final answer. No final answer is produced.
Correctness. Incorrect or unscored.

FATR (with injected finance attributes).
Tool trace.
Step 1: `symbols_get_fundamentals` → empty.
Step 2: `symbols_get_sector_metrics` → single-period margin.
Step 3: `symbols_get_fundamentals` (`limit=3`) → empty.
Step 4: `symbols_financials_metrics` → FY22 operating income grows slower than revenue.
Final answer. AMEX's operating margin did not improve in 2022; revenue outpaced operating income.
Correctness. Incorrect.

Analysis. Finance attributes change tooling behavior. The injected run recovers from early API-interface failures and produces a tool-backed narrative, while the baseline fails to complete a trace. However, correctness still depends on aligning with the dataset's intended criterion. Here, the tool-backed answer discusses margin trends instead of adopting the ground-truth stance that operating margin is not the right performance metric.

Takeaway. Attribute injection improves execution continuity, but does not by itself guarantee semantic alignment with the ground-truth criterion.

Figure 8: Single-box, end-to-end comparison for `financebench_9`. The baseline fails due to an interface mismatch. FATR recovers a valid tool trace, but the final answer remains incorrect because it does not follow the dataset's intended reasoning criterion.

Case Study 2 (`openfindata_release_155`): injection prunes redundant tools while preserving correctness

Question. As a financial analyst, evaluate the downside risk profile of the Tianhong Yu'e Bao Money Market Fund using the downside risk data provided below.

Ground truth. Tianhong Yu'ebao has a low downside risk and is among the better performers in its category.

Baseline (no finance attributes).
Tool trace.
`fund_individual_analysis_xq` → volatility metrics inconsistent with money funds.
`fund_money_fund_info_em` → timeout or serialization error.
`rate_interbank` → execution error.
`fund_overview_em` → confirms low downside risk.
Final answer. Tianhong Yu'e Bao Money Market Fund exhibits extremely low downside risk, significantly below the peer average, making it suitable for conservative investors.
Correctness. Correct.

FATR (with injected finance attributes).
Tool trace.
`fund_overview_em` → confirms near-zero downside risk.
Final answer. Over the past year and across multiple short-term horizons, Tianhong Yu'e Bao Money Market Fund exhibits near-zero downside risk, significantly outperforming the peer average.
Correctness. Correct.

Analysis. Injection improves trace quality by pushing the planner toward a fund-specific, informational tool with appropriate scope. The baseline eventually reaches a suitable tool, but only after several failed or ill-suited calls. In this qualitatively straightforward item, both runs are correct, so the benefit mainly appears as a cleaner and more stable execution trace.

Takeaway. Both runs are correct, but the injected run reaches the answer with a shorter and cleaner trace.

Figure 9: Single-box comparison for `openfindata_release_155`. Injected finance attributes reduce redundant and incompatible tool calls while preserving correctness.

Case Study 3 (`financebench_33`): injection changes strategy but remains incorrect

Question. What is Nike's three-year average cost of goods sold as a percentage of revenue from FY2016 to FY2018?

Ground truth. 55.1%.

Baseline (no finance attributes).
Tool trace.
Income-statement tools → historical data locked or unavailable.
`stock_get_cost_of_revenue` → single-point value without year context.
Final answer. The requested three-year average cannot be calculated with the available data.
Correctness. Incorrect.

FATR (with injected finance attributes).
Tool trace.
`symbols_get_sector_metrics` → gross margin retrieved.
Final answer. Using gross margin (38.6%), the implied average COGS ratio is approximately 61.4%.
Correctness. Incorrect.

Analysis. Injection pushes the agent away from unavailable historical statements toward a proxy metric. This avoids dead ends, but it does not ensure numerical fidelity. Even with a cleaner trace, the agent can still produce an incorrect estimate when the required quantity is not directly supported by the available tools.

Takeaway. Finance attributes can redirect the planner to a more plausible tool path, but they are not a substitute for numeric-grounding checks.

Figure 10: Single-box comparison for `financebench_33`. Injected finance attributes change the tool strategy, but the resulting proxy-based estimate remains incorrect.