Evaluating Enterprise Analytics Agents: An End-to-End, Trace-Backed Methodology

Teja Venkat Kolli
tkolli@thumbtack.com
Thumbtack
San Francisco, CA, USA

Sang Su Lee
psulee@thumbtack.com
Thumbtack
San Francisco, CA, USA

Xueying Yan
xyan@thumbtack.com
Thumbtack
San Francisco, CA, USA

Jessie Chen
jechen@thumbtack.com
Thumbtack
San Francisco, CA, USA

Chi Cheng
ccheng@thumbtack.com
Thumbtack
San Francisco, CA, USA

Kartik Ravisankar
kravisankar@thumbtack.com
Thumbtack
San Francisco, CA, USA

Shishir Dash
shishirdash@thumbtack.com
Thumbtack
San Francisco, CA, USA

Vijay Anand Raghavan
vraghavan@thumbtack.com
Thumbtack
San Francisco, CA, USA

Abstract

Enterprise analytics agents are not only text-to-SQL systems. They interpret business intent and choose metric definitions. They select data sources, execute tools, inspect results, and produce natural-language answers. Those answers may influence operational, financial, or executive decisions. Grading final answers hides where these agents fail. A plausible answer can use the wrong source of truth. It can skip a required decomposition, claim causality without support, or change its table interpretation across repeated runs.

We present an end-to-end evaluation methodology for analytics agents. The methodology grades agent behavior across three families: semantic understanding, execution quality, and reliability. Grading uses question banks with human-written golden answers, repeated runs, and runtime traces. Each run first passes a run-validity check, then receives tiered, abstention-aware scores that feed a decision framework rather than a release gate.

We instantiate the methodology on a controlled internal analytics agent at a large online marketplace. The case study uses 50 analytics questions, two anonymous model configurations, and three randomized repetitions per configuration, yielding 300 traces. The higher-capability configuration reduced early refusal from 73% to 0% and increased real-data answers from 21% to 73%. However, it also exhausted the tool-round budget on 16% of runs. It overran the schema-exploration budget on 77% of traces. It changed its table interpretation on 41 of 50 questions. On finance questions with structured golden answers, source table use and escalation improved, but canonical decomposition remained weak in both configurations. These results show why trust in analytics agents requires end-to-end, trace-backed evaluation rather than SQL correctness or final answer quality alone.

CCS Concepts

• **Software and its engineering** → **Software verification and validation**; • **Computing methodologies** → *Intelligent agents*; • **Information systems** → *Decision support systems.*

Keywords

agent evaluation, analytics agents, trajectory evaluation

1 Introduction

Analytics agents are becoming natural-language interfaces to enterprise data warehouses. A user may ask why revenue changed. Another user may ask whether a daily metric is unusual or what is driving a forecast gap. To answer, an agent must interpret the business question. It must map that question to the correct metric definition, source of truth, query, and final response.

This is an analytical workflow, not a single generation task. A final answer can look reasonable while being wrong for many reasons. The agent may choose an off-source table or use the wrong date window. It may miss a business-rule filter, skip a required decomposition, or narrate unsupported causal claims. Conversely, an answer may be numerically close while failing the analytical contract that a domain expert would require.

The evaluation problem is therefore broader than SQL accuracy. SQL correctness is necessary, but it is not sufficient. Trustworthy analytics agent evaluation must inspect how the answer was produced. It must check whether the agent followed the expected method. It must also test whether repeated runs produce a stable interpretation.

We propose an end-to-end evaluation methodology for decision-grade analytics agents. We use "decision-grade" narrowly: evaluation should show whether an agent answer is reliable enough to inform a business decision. It should also show whether the agent should abstain, escalate, or route the answer to human review. We position the methodology as a measurement foundation for agent governance, not as a substitute for it.

We make three contributions:

(1) We define a workflow-level methodology for analytics-agent evaluation. It covers interpretation, source selection, SQL and tool execution, analytical method, grounding, confidence, and repeatability.

(2) We provide an analytics-specific failure taxonomy. It connects generic trajectory failures to concrete data and decision risks.
(3) We provide an empirical demonstration on a 300-trace case study (Section 5). It shows that a higher-capability configuration can answer more questions while still failing source selection, decomposition, grounding, efficiency, and cross-run consistency.

2 Why Analytics-Agent Evaluation Is Different

Standard text-to-SQL evaluation asks whether a query returns the expected answer for a natural-language question. Enterprise analytics agents face a larger task. They must reason about ambiguous business language, local metric definitions, source of truth hierarchy, table grain, and time windows. They must also reason about the decision the user is trying to make.

Our methodology starts from four human-review dimensions:

- **Question interpretation.** Did the agent understand the metric, scope, time range, segment, and decision context?
- **SQL and data sourcing.** Did the agent use the right tables, joins, filters, grains, and calculations?
- **Insight reasoning.** Did the agent interpret the results, decompose drivers, and avoid unsupported causal claims?
- **Output quality.** Was the answer complete, readable, qualified, and useful to the user?

These four dimensions span three signal families: *semantic understanding*, *execution quality*, and *reliability*. Question interpretation and SQL and data sourcing form the agent's semantic understanding. Insight reasoning and output quality form its execution quality. Cross-run repeatability across all four dimensions is its reliability.

These dimensions imply three evaluation requirements drawn from two complementary views. The analytics view asks whether the agent selected the right metric, source, grain, and interpretation. The process view asks whether it used tools correctly, recovered from errors, avoided loops, and escalated when evidence was weak. Combining both views keeps the evaluation from hiding either domain mistakes or trajectory mistakes.

First, evaluation must be trajectory-aware. Tool-using failures are often invisible in the final answer. The agent may call the right tool with wrong arguments. It may skip a prerequisite lookup, loop over schema discovery, fail to recover, or avoid escalation despite weak evidence.

Second, evaluation must be analytics-specific. Generic trajectory checks do not know whether a table is canonical. They also do not know whether a metric has a local definition or a required filter. Analytics agent evaluation must map generic tool failures into domain failures. Examples include wrong source, wrong grain, wrong time window, missing decomposition, and unsupported guidance.

Third, evaluation must measure repeatability. A single successful run does not establish reliability. If the same question produces different tables, numbers, or conclusions across repetitions, the agent may create a false sense of stability. Repeatability is especially important for high-stakes analytics questions such as forecasting, anomaly explanation, experiment readouts, and executive reporting.

3 End-to-End Evaluation Methodology

The methodology runs in three stages. First, the evaluation builds a representative question bank with human-written golden answers and runs the agent in isolated, randomized, repeated trials, capturing runtime traces. Second, each run passes a run-validity check, then receives tiered scoring that separates primary signals from diagnostic signals and treats abstention as a first-class verdict. Third, the scores feed a decision framework calibrated against domain-expert review. Figure 1 summarizes the pipeline; the subsections that follow describe each stage.

3.1 Representative Question Bank

The evaluation begins with a question bank sampled from realistic analytics workflows. Questions should be tagged by business area, question type, source tier, ambiguity, and decision risk. This prevents one aggregate score from mixing exploratory trend questions with high-precision finance or forecast questions that require stricter reliability.

3.2 Human-Written Golden Answers

Golden answers specify the expected analytical contract. They may include required sources, prohibited sources, date windows, and filters. They may also include decomposition steps, numeric bands, expected terms, and claims the answer must not make. A golden answer is not merely a final answer. It is a compact description of the workflow that a domain expert expects. In the case study, each golden answer was drafted by the domain expert who owns that business area and independently reviewed and approved by a second analyst, tracked by an explicit draft-versus-approved status flag; we did not compute a formal inter-rater reliability statistic (Section 9).

Golden answers also bridge semantic knowledge and evaluation. A future semantic context layer may move the same knowledge into runtime guidance. That guidance can include metric definitions, source hierarchy, canonical queries, business rules, and decomposition templates. These fields define what runtime semantic guidance would need to cover.

3.3 Isolated Repeated Runs

Each question is run in an isolated session. The runner resets state between questions and randomizes question order with fixed seeds. It also prompts the agent to treat each question independently. Repetition makes stochastic behavior observable. The evaluation does not ask whether one sampled answer was good. It asks whether the agent repeatedly chooses the same interpretation.

3.4 Runtime Trace Capture

For every run, the evaluation records the raw question and configuration metadata. It also records tool calls, query text, query outputs, errors, retries, latency, tool-round counts, and final answer. These traces are the central evidence object. If a result is wrong, the trace can localize the failure. It can point to interpretation, source selection, SQL construction, tool failure, result interpretation, or final answer synthesis.

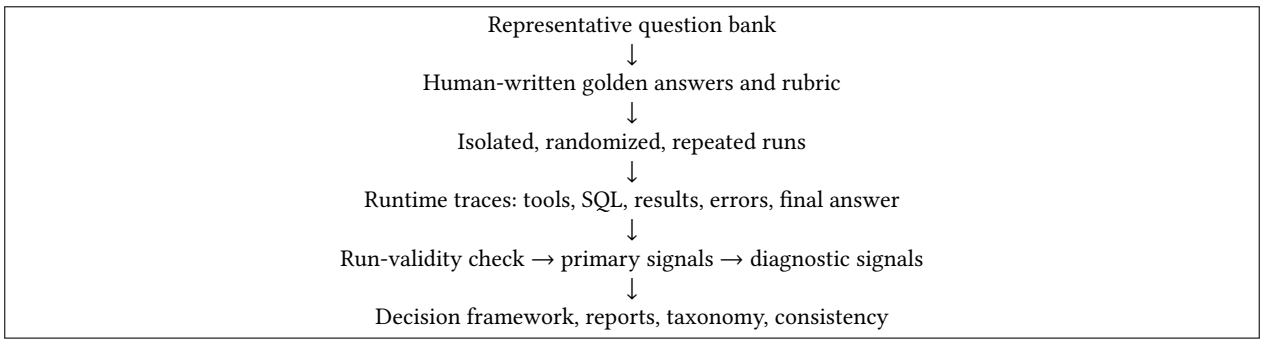

Figure 1: End-to-end evaluation pipeline for analytics agents.

3.5 Run Validity Before Scoring

Before scoring answer quality, the evaluation checks whether a run is valid enough to interpret. The case study used validation signals for hard errors and early refusal patterns. It also checked user-visible authentication mentions, real-data presence, and infrastructure authentication failures in tool outputs. This check prevents a broken execution environment from being mistaken for agent behavior.

3.6 Primary, Diagnostic, and Abstention-Aware Scoring

The methodology separates primary signals from diagnostic signals. Primary signals assess the core analytical contract for review, comparison, or intervention measurement. Diagnostic signals explain why a primary result changed. This distinction matters because analytics evals can easily accumulate many scorers; reviewers should not have to interpret every diagnostic as a release gate.

Every scorer returns one of four verdicts: *yes*, *partial*, *no*, or *unknown*. The *unknown* verdict is a first-class abstention, not a failure. Aggregates should report coverage separately from pass rate so that missing golden answers, questions outside scope, and unavailable trace fields are visible. Concretely, unknown verdicts are excluded from the denominator and reported separately as a coverage rate; among the decided runs, each verdict is scored yes = 1.0, partial = 0.5, no = 0.0, and the reported pass rate is the mean of these scores. Binary scorers that never emit *partial* reduce to yes / (yes + no).

3.7 From Scores to Decisions

Primary and diagnostic scores support engineering and review decisions. They do not define a universal release policy. Table 1 shows how this paper interprets the signals. A run-validity failure means that quality scores should not be trusted. Primary signals compare versions or interventions. Diagnostic signals identify repair work. Cross-run checks test interpretation stability. Human calibration is required before automated judges or thresholds become deployment gates.

We therefore call this a decision framework rather than a validated release policy. The method makes failures visible and actionable. Exact thresholds should be calibrated with domain-expert review before they become blocking deployment criteria.

3.8 Human Calibration

Automated scoring should be calibrated against human review before it drives blocking or approval decisions. Domain experts can grade a sample of traced runs using the four-dimension rubric, then compare automated verdicts to expert judgments. In the case study, deterministic and trace-backed signals form the main evidence; calibrated LLM judges remain a follow-up direction.

4 Analytics-Specific Failure Taxonomy

Table 2 summarizes the taxonomy used to interpret results and prioritize interventions. The taxonomy is deliberately workflow-oriented: each failure mode points to a potential repair in context, tools, workflow enforcement, or review policy.

The taxonomy operationalizes both views from Section 2. Question interpretation, semantic/source selection, analytical method, and insight reasoning capture analytics-specific correctness. SQL and tool execution, confidence and grounding, and repeatability capture process failures in tool-using agents.

The taxonomy also clarifies why semantic context alone is insufficient. Guidance tells the model what good behavior looks like. Workflow enforcement prevents the model from skipping required steps. Trajectory evaluation verifies whether those steps happened. Analytics agents need all three.

5 Case Study

We evaluated a controlled internal analytics agent at a large online marketplace. The agent answers natural-language business questions using schema lookup and query execution tools over an enterprise warehouse, then synthesizes a natural-language answer. For anonymity, we refer to the two model configurations as **Fast** and **Reasoning**.

The case study used a 50-question bank balanced across five internal business areas, with 10 questions per area. Each question ran three times against each of two model configurations, yielding 300 total traces. Both configurations used the same toolset, prompt scaffold, and runner. The intended difference was the model configuration.

Finance was the one area with complete structured golden answers. The case study uses those 10 finance questions for the structured-golden scorecard. We score them across the same three randomized repetitions per configuration, producing 30 finance

Table 1: Decision framework for interpreting evaluation signals.

Eval layer	Question answered	Example signals	Decision use
Run-validity check	Was the run environment valid enough to score?	Hard errors, real-data rate, early-refusal rate, infrastructure-auth errors	Discard or rerun invalid batches before interpreting quality
Primary correctness	Did the agent satisfy the core analytical contract?	Required source, answer attempt, escalation behavior, key numeric bands	Compare model, prompt, tool, or intervention variants
Diagnostic signals	Why did the result fail or improve?	Schema overrun, execution efficiency, self-consistency, retry recovery	Choose the next repair target
Cross-run reliability	Does the same question produce the same interpretation?	Stable, value-unstable, interpretation-unstable, no-attempt labels	Route unstable patterns to human review or stricter workflow control
Human calibration	Do automated judgments match expert review?	Expert rubric grades, judge agreement, threshold review	Calibrate thresholds before they become deployment gates

Table 2: Analytics-agent failure taxonomy.

Layer	Failure mode	Typical intervention
Question interpretation	Wrong metric, scope, segment, or ambiguity handling	Clarification policy, intent parser, examples
Semantic/source selection	Wrong source of truth or metric definition	Source hierarchy, metric glossary, canonical routing
SQL and tool execution	Wrong time window, grain, join, filter, or query recovery	Schema lookup, validation checks, row-count checks
Analytical method	Missing decomposition or diagnostic workflow	Domain workflow templates, required intermediate steps
Insight reasoning	Unsupported causal or decision claim	Alternative-hypothesis checks, escalation gates
Confidence and grounding	Plausible but wrong numeric answer or contradiction	Numeric calibration, SQL-answer grounding checks
Repeatability	Cross-run table, value, or narrative instability	Repeated-run checks, deterministic planning, canonical routes

traces per configuration. The other four areas lack equivalent golden coverage. We therefore treat the finance scorecard as diagnostic evidence, not as a complete benchmark.

6 Results

The results are not presented as a model benchmark. They are evidence that end-to-end evaluation exposes failures hidden by grading final answers. The five subsections give five views of the same 300 traces: run validity, the finance scorecard, contract violations, trace-only risks, and cross-run consistency.

6.1 Run Validity and Answering Behavior

The Reasoning configuration answered far more often, as Table 3 shows. It reduced early refusal from 73% to 0% and increased real-data answers from 21% to 73%. But it was also much slower. It hit the maximum tool-round budget on 16% of runs. The dominant failure mode changed from early abandonment to over-exploration and unstable interpretation.

6.2 Finance Golden Answers

Table 4 reports selected finance scorer deltas. The Reasoning configuration improved several primary signals: it attempted questions more often, escalated more appropriately, and found required source tables more often. Yet required-table use reached only 50%, and the decomposition method remained weak for both configurations.

Table 3: Top-level run metrics across 150 traces per configuration.

Metric	Fast	Reasoning
Traces per configuration	150	150
Hard errors	1 / 150	0 / 150
Hit tool-round limit	0 / 150	24 / 150
Early refusal rate	73%	0%
Real-data rate	21%	73%
Average latency	41 s	119 s
p90 latency	86 s	234 s
Run-validity check	Fail	Pass

Better model capability improved willingness and exploration, but did not reliably produce the expected analytical method.

6.3 Correct Number, Wrong Analytical Contract

One finance question asked for a year-over-year Q1 revenue comparison. In a representative Reasoning run, the agent selected a plausible revenue table. Headline revenue numbers fell within about 0.3% of the golden answer. Monthly trends were also close.

However, the answer failed important parts of the analytical contract. It did not use the prescribed source of truth table. It did

Table 4: Selected finance scorer deltas. Pass rate is the mean verdict credit over decided runs, with unknowns excluded and reported as coverage (Section 3.6); row populations vary by scorer applicability. "Should not decline" captures cases where the agent declined despite sufficient evidence; "Decomposition method" captures whether the answer applied the expected analytical decomposition.

Scorer	Fast	Reasoning	Delta
Escalation discipline	29%	93%	+64 pp
Required table used	0%	50%	+50 pp
Should not decline	65%	100%	+35 pp
Revenue YoY band	62%	83%	+21 pp
Decomposition method	3%	10%	+7 pp
Execution efficiency	74.2%	6.7%	-67.5 pp
Prohibited table avoided	100%	100%	0 pp
Retry recovery	100%	100%	0 pp

not compute the required projects and revenue per project decomposition. It also omitted expected driver checks. A final answer evaluator might reward the answer because the headline number was close. An end-to-end evaluator flags it as incomplete because it skipped the method required to explain why revenue moved. This divergence is not rare. Contract checks require structured golden answers, so the divergence rate is measurable on the finance subset rather than on all 300 traces. Across the 30 Reasoning finance traces, a final-answer grader sees a completed answer on every trace, yet trace-backed scoring fails at least one contract check on 27 of 30 traces (90%): a non-canonical source-of-truth table on 15 of 30 and a skipped decomposition on 27 of 30. Final-answer-only grading would therefore still rank Reasoning above Fast, but it would report the upgrade as an unqualified win and hide these contract failures.

6.4 Trace-Only Risks

Trace-only diagnostics expose risks that do not require a complete golden answer. Table 5 reports five such signals across 150 traces per configuration. The Reasoning configuration produced more implausible numeric claims and more hard contradictions than the Fast configuration. Fast looks safer at first glance, but its lower numeric-claim rate is partly an artifact of answering fewer questions with numerics. Silence is not reliability.

Table 5: Trace-only diagnostic signals across 150 traces per configuration. Counts are over 150 traces; percentages are out of 150.

Diagnostic	Fast	Reasoning
Schema lookup calls (total)	115	961
Schema lookups per trace (avg)	0.77	6.4
Schema overrun rate	1%	77%
Implausible numeric claims	3 (2.0%)	16 (10.7%)
Hard SQL-answer contradictions	8 (5.3%)	11 (7.3%)

6.5 Cross-Run Consistency

Table 6: Cross-run determinism across 50 questions.

Model	Stable	Value	Interpretation	Partial	No attempt
Fast	0	19	21	4	6
Reasoning	1	8	41	0	0

Table 6 shows the cross-run determinism breakdown. The strongest reliability signal was interpretation instability. The Reasoning configuration attempted every question. But on 41 of 50 questions, it changed the table set or query interpretation across repetitions. This is a decision-grade reliability risk. A user may receive a coherent answer today and a different coherent answer tomorrow. The answer may not show that the underlying source of truth changed.

7 From Evaluation to Intervention

The case study supports three lessons that connect the methodology back to intervention strategy. Each lesson points to a specific class of intervention. Section 7.1 names the governance primitives those interventions require before any answer can be trusted for a business decision.

Model upgrades are not enough. The Reasoning configuration answered more questions and produced more grounded outputs. It still had source of truth errors, weak decomposition, schema overrun, contradictions, and interpretation drift. Model capability changed the failure surface; it did not remove the need for workflow control.

Analytics agents need semantic context plus enforcement. The observed failures point to interventions such as metric glossaries, source hierarchy, canonical table routing, domain decomposition templates, row-count checks, and post-query validation. But guidance must be paired with enforcement. The agent must do more than access definitions or canonical patterns. The workflow must make lookup, validation, and escalation observable and testable.

Evaluation should guide intervention measurement. Broad interventions such as a semantic context layer should be decomposed where possible. Smaller tests can target table metadata, metric definitions, source hierarchy, query patterns, and workflow scaffolds. When decomposition is not possible, per-layer scoring can still show what changed. A semantic intervention might improve source selection without improving answer completeness. That is still actionable.

7.1 Governance for Business Decisions

In this paper, governance means the evaluation primitives needed before an analytics-agent answer can be trusted for business decisions. The methodology provides five such primitives, each grounded in a Section 3 mechanism:

(1) **Traceability** via runtime trace capture.
(2) **Abstention and escalation** via abstention-aware scoring.
(3) **Primary vs diagnostic roles** via tiered scoring.
(4) **Run stability** via isolated repeated runs and cross-run consistency analysis.

(5) **Version comparison** via the trace-backed contract.

These primitives make governance possible; they do not constitute a deployed governance program.

8 Related Work

Agent evaluation increasingly recognizes that final answers are insufficient. ReAct shows the value of interleaving reasoning and acting, making trajectories central to agent behavior [6]. GAIA and tau-bench evaluate tool use and multi-step agent reliability beyond simple question answering [3, 5]. Tau-bench is especially relevant because it evaluates repeated task success and consistency across trials in tool-using domains.

Text-to-SQL work such as Spider, BIRD, and Spider 2.0 evaluates natural language to database-query capability [1, 4, 7]. These benchmarks are essential, but enterprise analytics agents add requirements beyond SQL execution. The agent must choose among competing source of truth tables, respect local business definitions, and explain results according to domain practice.

LLM-as-a-judge work, including MT-Bench and pairwise arena benchmarks, studies when model judges agree with humans [8]. Self-refinement methods show how LLMs can critique and improve their own outputs [2]. Our approach is compatible with judge-based scoring, but the reported evidence relies primarily on deterministic and trace-backed signals. LLM judges are most useful for nuanced interpretation and insight quality after calibration against domain-expert grades.

Enterprise analytics evaluation differs from public benchmarks. Governed warehouses, local metric definitions, and source-of-truth hierarchies make analytics-agent correctness domain-specific in ways that public text-to-SQL or trajectory benchmarks cannot capture. The methodology in this paper adds three contributions not covered by the prior work above. First, a workflow-level evaluation ties trajectory checks to analytical contracts. Second, an analytics-specific failure taxonomy maps generic tool failures to domain repair targets. Third, a 300-trace case study demonstrates the methodology and exposes failure modes that final-answer evaluation hides.

9 Limitations and Future Work

This study has several limitations. First, the data and agent are internal to one company. Second, the case study compares two model configurations, but it is not a full model benchmark. Third, the case study did not evaluate a deployed semantic layer. The evidence shows where semantic context and canonical routing are needed but does not show that they already improved the agent. Fourth, cross-run consistency is estimated from three repetitions per question; three samples bound but do not precisely estimate run-to-run stability, so the interpretation-instability counts should be read as lower bounds rather than exact rates. Fifth, latency and execution-efficiency figures are specific to this data warehouse and execution environment; they reflect the deployment, not intrinsic model properties, and should not be read as portable model benchmarks. Sixth, the finance golden answers were authored internally without a reported inter-rater reliability check, and the two model configurations are anonymized as “Fast” and “Reasoning,” which limits how far a reader can attribute the observed deltas to specific model families. Authoring and reviewing a structured golden answer took on the order of a few hours of a domain-familiar data scientist per question, dominated by encoding the analytical contract and the second-analyst review; reducing this cost through calibrated automation of the more mechanical checks is future work.

Future work has three priorities. First, calibrate the decision framework against domain-expert review so that primary-signal thresholds become trustworthy gates. Second, instrument a semantic context layer (metric glossary, source hierarchy, canonical routing) and measure each component against the same trace-backed contract. Third, apply the methodology to a second analytics agent on a different domain to test how the failure taxonomy generalizes.

Reproducibility. The methodology, scorers, and decision framework are independently reproducible. Practitioners can apply them to their own analytics agents, question banks, and golden answers. The specific case study is not directly reproducible. The question bank, the agent system, and the data warehouse are internal to one organization and cannot be shared. The portable contribution is the methodology, not the specific rates.

10 Conclusion

Analytics agents need evaluation at the level of the analytical workflow, not only the final answer. The methodology combines question banks, human golden answers, repeated runs, and runtime traces. Each run passes a run-validity check before tiered, primary and diagnostic scoring. It also adds abstention-aware scoring, a decision framework, and cross-run consistency analysis.

The case study shows why that end-to-end view matters. A higher-capability configuration answered more questions and used real data more often. It still produced source of truth errors, weak decomposition, schema overrun, contradictions, and interpretation drift. These are exactly the failures that final answer evaluation hides. Trace-backed, end-to-end evaluation gives teams the measurement surface needed to improve analytics agents without mistaking plausibility for reliability.

References

[1] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as A Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, 42330–42357.

[2] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, 46534–46594.

[3] Gregoire Mialon, Clementine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. GAIA: A Benchmark for General AI Assistants. arXiv preprint arXiv:2311.12983. arXiv:2311.12983 [cs.AI] https://arxiv.org/abs/2311.12983

[4] The Spider 2.0 Team. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. arXiv preprint arXiv:2411.07763. arXiv:2411.07763 [cs.CL] https://arxiv.org/abs/2411.07763

[5] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv preprint arXiv:2406.12045. arXiv:2406.12045 [cs.AI] https://arxiv.org/abs/2406.12045

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, Kigali, Rwanda. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[7] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. doi:10.18653/v1/D18-1425

[8] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., New Orleans, LA, USA, 46595–46623.