VADE: A VERIFIABLE AGENT DATA ENGINE FROM PROFESSIONAL SOURCES

Anonymous authors
Paper under double-blind review

ABSTRACT

Language agents have demonstrated their practical value, particularly in software engineering, where abundant public code and executable tests support training and evaluation at scale. Extending this success to other professional domains, such as law and finance, is not straightforward: many agent tasks involve open-ended analysis, making high-quality outcomes difficult to verify and, in turn, difficult to use as scalable supervision. As a result, high-quality tasks in these domains are still largely expert-authored, making them expensive to scale and limiting their use for training. To address this bottleneck, we introduce VADE, a data engine that automatically constructs verifiable analysis tasks from publicly available sources used in professional practice. VADE exploits a simple asymmetry: professional analyses are difficult to solve because the relevant evidence must first be found and interpreted, but much easier to verify once that evidence is fixed. During task synthesis, the engine first constructs a reference analysis from source evidence and asks an independent agent to reproduce its conclusions from the same evidence without seeing the reference analysis. Only after the analysis is verified is it compiled into an open-ended task with an item-level scoring contract, allowing the same task to support both evaluation and training. We apply VADE across four professional domains, with independent post-hoc expert review finding 93.5% of our constructed tasks to meet a high bar for source fidelity, answer correctness, and evaluation fairness. In finance, these tasks expose substantial remaining model errors, particularly unsupported facts and inferences, while training on verified solver trajectories improves held-out performance and yields gains that transfer to public finance benchmarks such as Finance Agent Benchmark v2. Together, these results suggest a practical path toward scalable, verifiable supervision for open-ended professional tasks.

1 INTRODUCTION

Language agents, which follow natural-language instructions to act in real or simulated environments (Su et al., 2024; Sumers et al., 2024), have become practically useful for a growing range of complex tasks. Much of this progress comes from agent data, the tasks and trajectories from which agents learn to act (Zeng et al., 2024; Prabhakar et al., 2025; Zhang et al., 2026a). Tasks with verifiable outcomes are especially valuable: they evaluate agents directly and, once solution trajectories are filtered by their outcomes, also train them. Coding agents show this most clearly (Yang et al., 2024; Wang et al., 2025): public repositories supply realistic tasks at scale, and executable tests make every outcome cheap to verify, so the same tasks serve as benchmarks (Jimenez et al., 2024) and as training environments (Pan et al., 2025; Yang et al., 2025; Jain et al., 2025).

However, other professional domains, such as law (Guha et al., 2023) and finance (Islam et al., 2023), have not yet seen the same progress, largely because their tasks are hard to verify. Many agent tasks in these domains involve open-ended analysis: the agent must find evidence scattered across the sources used in practice, interpret it under the right basis, period, and scope, and deliver a conclusion that holds under stated conditions. The sources provide evidence rather than answers, and no test can be executed against the conclusion. In a filing, for example, what drives a change in profit depends on which basis is used and how each adjustment is treated. Without an executable check, reference answers need expert review, and solver trajectories cannot be filtered at scale. As a result, tasks in these domains are still written mostly by experts. GDPval (Patwardhan et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2026), APEX (Vidgen et al., 2026; Bennett et al., 2026), Finance Agent Benchmark (Bigeard et al., 2025; Vals AI, 2026), BigFinanceBench (Wang et al., 2026), and LAB (Grupen et al., 2026) evaluate agents on realistic work tasks against expert-written criteria.

Automated task synthesis scales, but existing methods obtain verifiability by restricting the task. TaskCraft (Shi et al., 2026) uses tool-executable facts as reference answers, FinForge (Matlin et al., 2026) reduces tasks to multiple choice, and AGORA (Guo et al., 2026) limits the answer to a single number. QUEST (Xie et al., 2026) keeps research tasks open-ended, but its reference answers and rubrics are produced by the same process that writes the tasks, with no independent check. What is missing is a way to verify the reference answer of an open-ended analysis task without relying on experts or narrowing the task.

We introduce VADE, a data engine that automatically constructs verifiable analysis tasks from public sources used in professional practice (Figure 1). VADE is built on an asymmetry between solving and verifying a professional analysis: a solver must first find and interpret the relevant evidence, whereas a verifier given that evidence need only derive the conclusions from it. The engine first finds candidate analyses in the sources through ten discovery routes (Table 5) and builds a reference analysis for each from quoted evidence; code checks every quotation against the sources and recomputes every arithmetic step. The reference analysis is then verified by independent reproduction: a second agent derives its conclusions from the same evidence without seeing them, and a third agent compares the two analyses and attributes every difference either to the task, which is repaired, or to the reproduction, which is rerun. Only after the reference analysis is verified is the question written, together with an item-level *scoring contract* that specifies what a correct answer must contain and what fails it outright. Because every answer can be judged item by item against the scoring contract, the same task supports both evaluation and training.

We apply VADE across four professional domains: finance, law, medicine, and energy. In finance, it delivers 1000 tasks across 25 industry groups, and in law, 372 tasks across 12 practice areas (§3). In an independent post-hoc review of 186 finance tasks, experts judged 93.5% to meet a high bar for source fidelity, answer correctness, and evaluation fairness (§3.4). In finance, the tasks remain far from solved: six models reach pass rates of 19.4% to 66.4% (§4.2), and unsupported facts and inferences, the most common error, appear in 85.9% of failed answers (§4.3). The same tasks also serve as supervision. Training Qwen3.5-27B on verified solver trajectories from 400 tasks raises its pass rate from 6.7% to 21.3% on 75 held-out tasks, and the gains transfer to four public finance benchmarks; on Finance Agent Benchmark v2, it improves from 29.6% to 51.9% (§5).

Taken together, these results show that open-ended professional tasks need not trade verifiability for scale. Verifying each reference analysis by independent reproduction removes the need for expert authoring without narrowing the task, and the resulting tasks are hard enough to expose where current models fail and reliable enough to train on. We see VADE as a step toward agent data for professional work that grows with the available sources rather than with expert time.

2 THE VADE ENGINE

VADE turns sources used in professional practice into tasks whose reference analyses are verified before any question is written. Its design follows the asymmetry of §1: an analysis is hard to solve because its evidence must first be found and interpreted, but much easier to verify once that evidence is fixed. The engine therefore builds a reference analysis from evidence in the sources, verifies it by independent reproduction from the same evidence, and only then writes the question.

Each delivered task has three parts (Figure 2). The question states one analytical objective, an information cutoff, and the allowed tools; it is all the solver sees. The hidden reference analysis reaches its conclusion through *answer atoms*: the values and categories that a correct answer must contain. Each atom records its tolerance, the evidence it rests on, and the derivation that produces it. The scoring contract turns the reference analysis into scored items: every answer atom, and the fatal conditions that fail an answer outright. It also lists the accepted alternatives and maps every requirement of the question to the atoms that answer it.

Sections 2.1–2.4 follow the four panels of Figure 1. An orchestrator agent coordinates the flow: it dispatches each step, assesses the result, and decides whether a candidate is repaired or archived, but writes nothing that enters a task. Each step is carried out by a separate worker agent that receives

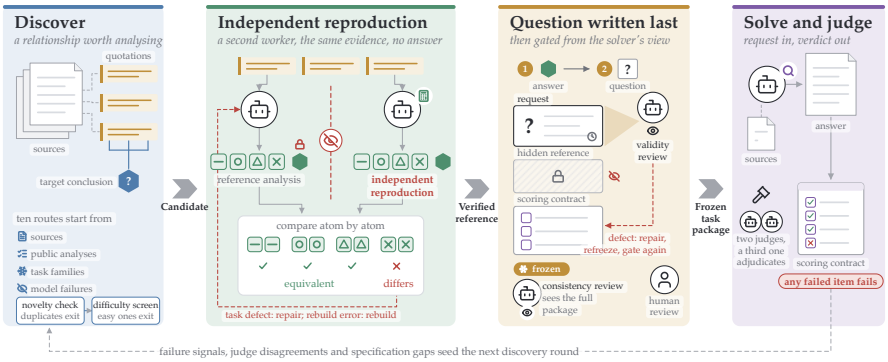

Figure 1: The VADE engine. Each panel hands one artefact to the next: a candidate found in professional sources, a reference analysis verified by independent reproduction, and the frozen task, whose question is written only after the reference analysis is verified and is all the solver receives. A defect returns to the stage that caused it (dashed red).

Question. SouthState's consolidated allowance for credit losses (ACL) rose 33% from December 2024 to June 2025. Is this representative of higher portfolio loss content, or non-diagnostic once gross-loan growth and loan-population mix are controlled for? Reconcile the acquired and non-acquired loan populations at both dates before judging.

Answer atoms (8, all required), e.g. total gross loans \$33,902,927K → \$47,267,247K (+39.42%); ACL coverage 1.3724% → 1.3139%; `representativeness_classification`: *non-diagnostic*.

Fatal conditions (7), e.g. `incorrect_judgment`: calls the increase representative of higher loss content; `unsupported_inference`: treats arithmetic growth contributions as acquisition causality.

Accepted alternatives, e.g. "purchased credit deteriorated" for PCD; amounts in millions, not thousands.

Figure 2: A task at a glance: the SouthState allowance task, with part of its scoring contract. The hidden reference analysis and a full contract are in Appendix A.4.

only the inputs the step needs; the worker that reproduces a reference analysis never sees its conclusion, and a reviewer judges each question first from the solver's view alone. No worker output is accepted until code checks that every quotation appears verbatim in the sources, that every arithmetic step recomputes to its stated result, and that every cross-reference resolves. A defect found at any step returns the candidate to the earliest step it affects; a candidate that fails a step outright, or cannot be repaired within its budget, is archived with its reason (Figure 3).

2.1 Discovery

Sources are registered in the form in which they are obtained: web pages, structured filing data, PDF documents, tables and workbooks, audio and video, and images. Each is rendered into text that can be quoted verbatim, and the rendering is frozen together with the original file. A candidate is not yet a question but an analysis the sources can support: a central object, a target conclusion, and the relationships in the sources that must hold for the conclusion to follow (for the task of Figure 2: the allowance for credit losses rose 33% while loans grew 39%). Candidates enter through ten discovery routes that start from primary sources, published analyses, earlier verified tasks, or known model failures (Appendix A.2). A candidate that passes a novelty check, which removes duplicates, goes to a worker that ties each relationship to an evidence fragment (a verbatim quotation with its location) and writes the smallest set of answer atoms needed to evaluate the candidate. A candidate whose sources do not support a unique answer exits here.

The sources support far more relationships than are worth turning into tasks, so a difficulty screen runs before any verification and without solving the candidate. From a projection of the candidate that hides every answer value, a worker without tools scores the shortest path a professional would take to the outputs, and a second worker ranks the batch (Appendix A.1). The top candidates proceed and the bottom ones are archived. The middle ones undergo difficulty evolution: a source-supported relationship that changes the conclusion is added and the result re-enters.

2.2 Reference Analysis and Independent Reproduction

A reference analysis counts only once a second worker has reproduced it independently and every atom agrees. A worker first reopens the sources and treats nothing from discovery as established. For each evidence fragment, it checks that the quotation supports the stated relationship and that its entity, period, unit, basis, and population match those of the analysis, and it searches the sources for omissions that would change the answer. It then writes out every computation from the evidence fragments to the conclusion and produces the reference analysis, including the conditions under which its conclusion holds (in Figure 2, the atoms include the coverage ratio at each date).

The reference analysis is then verified by independent reproduction. A second worker, the reproducer, receives only the analytical objective, the evidence fragments, and a neutral description of each required output, and must derive every output from the evidence alone. Because the evidence is already fixed, the reproducer works on the easier, verification side of the asymmetry. A third worker compares the two analyses atom by atom. Agreement is not proof, so this worker also reopens the evidence and checks the premises the two analyses may share. Every difference that could change an atom is traced to its cause: a defect in the task returns the candidate to the earliest affected step for repair, while an error by the reproducer leads to a fresh reproduction. Only a candidate whose two analyses are equivalent continues. The verified analysis is then registered as an instance of a problem family, defined by its analytical objective, the roles its evidence plays, and the shape of its relationships and outputs; later candidates may extend a family only after it passes a separate review on multiple instances (Appendix A.1).

2.3 Question and Scoring Contract

The question is written last. Starting from the verified reference analysis, a worker writes the question: one high-level analytical objective, with the basis, period, and information cutoff that make its answer unique; the cutoff is set from the dates on which the sources became public. The scoring contract is written with the question, and code checks that every requirement of the question maps to atoms and every atom to a requirement. All three parts are then frozen together as one version; any later change creates a new version and keeps the old one.

The frozen task then enters the quality gate, where two reviews run in parallel. In the validity review, a reviewer sees only what the solver will see and, with access to public sources, judges whether the question is professionally meaningful, whether its answer is unique, and whether it can be answered with the allowed tools from information available before the cutoff. In the consistency review, a reviewer receives the full task and the source files and checks that the question, the reference analysis, and the scoring contract agree with one another and that reasonable alternatives are accepted. When a review fails, an adjudicator confirms each finding against the task; only confirmed defects are repaired, and the repaired task is refrozen and reviewed again. A final pass may polish only the question's wording; code then confirms that every step has passed before delivery.

2.4 Solving and Judging

The solver receives the question and the tools of a solving environment modeled on an expert-annotated benchmark in the domain, and must locate and read the sources itself. Two judges rule on every item of the scoring contract, a third settles their disagreements, and each domain decides how the item rulings combine into a verdict on the whole answer (§4).

3 Production at Scale

3.1 Preparation

The sources of the finance run are the public materials an analyst works from: periodic and current filings of listed companies and their exhibits, earnings releases and call transcripts, investor materials, regulators' orders and letters, and market and macroeconomic data. The sources of the legal run are the public materials a lawyer works from: codes, regulations, and the Federal Register; court dockets and opinions; regulators' enforcement documents and guidance; and transaction agreements

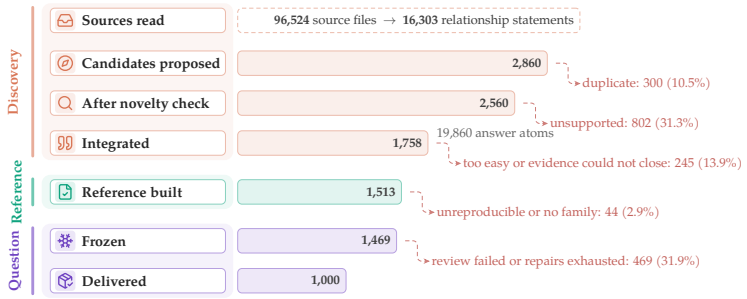

Figure 3: Production funnel of the finance run: the candidates remaining after each stage, with the archive reasons between stages and their share of the candidates entering the stage. The first row, the input, is not drawn to scale.

and securities litigation filings. We collected 96,524 finance files across 25 GICS industry groups and 53,498 legal files across 12 practice areas that we defined.

The orchestrator and the workers at every stage ran mainly on GPT-5.6 Sol; steps that require several agents also drew on other model families, including DeepSeek V4 Flash, Grok 4.6, and Claude Opus 5.

3.2 Production Run

Figure 3 shows the funnel of the finance run, from 96,524 source files to the 1000 delivered tasks; every count and exit reason comes from the run's records. On average, one task was delivered for every 2.9 candidates proposed. Attrition is concentrated before the reference analysis is built and at review. The legal run proposed 871 candidates and delivered 372 tasks.

At least 76.7% of the delivered tasks were repaired at least once, and the repaired tasks went through a median of two repair rounds. The most common cause of repair was a mismatch between the question and the scoring contract: a required item the question did not ask for, a precision stricter than the sources support, or a reasonable equivalent answer the contract did not accept.

Measured in tokens of the constructing model, the median task costs 24.4M tokens (interquartile range 17.7–34.1M). To keep the tasks verifiable, the engine spends more on verifying a task than on finding it: evidence and reference construction, independent reproduction and comparison, and review and repair together account for 34% of the total, compared with 28% for discovery, integration, and difficulty screening; the orchestrator's own judgment and checking account for 88% of the total.

3.3 The Task Sets

In all, VADE delivered 1000 tasks in finance and 372 tasks in law, together with preview sets of 85 energy and 85 medicine tasks from smaller runs. The finance set covers all 25 GICS industry groups and uses the nine task categories of Finance Agent Benchmark v2 (Vals AI, 2026); the legal set covers 12 practice areas and uses the four task types of LAB (Grupen et al., 2026). On average, a task draws on 1.6 sources and 13.1 evidence fragments, and its scoring contract has 11.9 answer atoms and 8.0 fatal conditions.

3.4 Human Review

Two reviewers with backgrounds in finance independently reviewed 186 tasks. For each task they checked four things: whether the task is realistic and close to the work finance professionals actually do, whether every cited value can be found at its stated location in the sources, whether the core conclusion is correct, and whether a reasonable alternative exists that the scoring contract does not accept. Of the 186 tasks, 93.5% needed no substantive revision, and the reviewers found substantive problems in the remaining 12. We release 100 tasks sampled from the delivered tasks, with their sources, reference analyses, and review records, as a quality showcase.

Table 1: Pass rates on VADE-Finance, VADE-Law, VADE-Energy-Preview, and VADE-Medicine-Preview (top to bottom) by model and task category. A task passes when every item of its scoring contract passes; the highest rate in each column of a block is in bold. VADE-Energy-Preview and VADE-Medicine-Preview are preview sets from smaller, experimental runs. Category abbreviations are expanded in Appendix B.

VADE-Finance

Model	Qual.	Quant.	Mkt.	Comp.	Prec.	Adj.	Earn.	Discl.	Model.	Overall
Grok 4.6	**54.1**	**76.2**	**69.4**	**65.5**	55.0	**68.1**	**65.3**	**65.7**	**63.0**	**66.4**
DeepSeek V4 Flash	24.3	46.5	44.4	51.7	**60.0**	48.9	40.6	40.1	42.5	42.9
Claude Opus 5	18.9	49.5	30.6	31.0	35.0	40.4	39.6	40.1	30.1	37.9
Qwen3.8-27B	27.0	36.6	25.0	34.5	30.0	34.0	37.6	32.8	37.0	34.1
GLM-5.2	21.6	33.7	36.1	31.0	20.0	23.4	25.7	28.5	23.3	27.7
MiniMax-M3	5.4	28.7	16.7	17.2	20.0	21.3	22.8	16.1	16.4	19.4

VADE-Law

Model	Analyze	Draft	Review	Research	Overall
Claude Opus 5	**73.6**	41.7	64.3	**76.0**	**66.0**
Grok 4.6	70.6	37.5	**68.3**	72.0	64.5
DeepSeek V4 Flash	52.9	31.8	47.5	56.0	48.6
Qwen3.8-27B	41.5	**45.8**	39.0	41.7	41.6
GLM-5.2	37.7	20.8	52.4	40.0	39.6
MiniMax-M3	20.8	25.0	22.0	25.0	22.5

VADE-Energy-Preview

Model	Reg.	Mkt.	Gen.	Rel.	Scen.	Stor.	Dem.	Overall
Grok 4.6	46.9	**100.0**	**83.3**	0.0	71.4	28.6	**71.4**	**59.8**
DeepSeek V4 Flash	46.9	16.7	58.3	**60.0**	**100.0**	0.0	28.6	45.0
Claude Opus 5	**50.0**	58.3	41.7	0.0	71.4	**40.0**	0.0	43.6
GLM-5.2	37.5	16.7	41.7	0.0	71.4	28.6	28.6	34.1
Qwen3.8-27B	31.3	16.7	**83.3**	0.0	28.6	0.0	40.0	32.1
MiniMax-M3	6.3	0.0	16.7	0.0	28.6	0.0	40.0	10.3

VADE-Medicine-Preview

Model	Reg.	Trial	Discl.	Safety	Label	Overall
Grok 4.6	**75.0**	**76.2**	**85.7**	**57.1**	**55.6**	**73.4**
Claude Opus 5	40.0	33.3	35.7	28.6	**55.6**	38.3
DeepSeek V4 Flash	32.1	47.4	35.7	0.0	**55.6**	36.4
GLM-5.2	7.1	23.8	64.3	28.6	0.0	22.8
Qwen3.8-27B	23.3	9.5	50.0	0.0	22.2	22.2
MiniMax-M3	0.0	9.5	0.0	0.0	0.0	2.5

4 The Tasks as a Benchmark

4.1 Setup

We turn the delivered tasks into a benchmark and evaluate models in each domain's solving environment. The finance environment follows the tool interface of the Finance Agent Benchmark: the solver finds its sources using web search, EDGAR search, and a document reader, and submits its answer as text. The legal environment follows the sandbox of LAB: the solver works on local materials with file tools and delivers files or text. Both environments are detailed in Appendix B.1.

Following the implementations of the Finance Agent Benchmark and LAB, we grade with language-model judges (Zheng et al., 2023; Tan et al., 2025) under a fixed procedure. Every answer is graded by two judges, GPT-5.6 Sol and Claude Opus 5, item by item under the scoring contract. In the finance environment, the judge first reads the whole answer and checks the solver's retrieval record without seeing the scoring contract, then rules on every item and fatal condition; in the legal environment, it reads the delivered files and rules on every item. In both, an answer passes only if every item passes. When the two judges disagree, their itemized findings are passed to a third judge, whose ruling is final. The full judging rules are in Appendix B.

4.2 Model Performance

On the finance tasks, overall pass rates range from 19.4% to 66.4%. Grok 4.6 leads by more than twenty points; DeepSeek V4 Flash, Claude Opus 5, and Qwen3.8-27B cluster between 34% and 43%; GLM-5.2 and MiniMax-M3 are lower. Qualitative analysis is the weakest category for five of the six models: Grok reaches only 54.1% on it, Opus 18.9%, and MiniMax 5.4%. Precedents and comparables break the overall order: DeepSeek passes 60.0% of precedent tasks, more than Grok.

Table 2: Pass rate (P) and item pass rate (R) of Qwen3.5-9B and Qwen3.5-27B before fine-tuning and after 4 and 10 epochs, on VADE-Finance-Test and on the public questions of four external finance benchmarks. The best value of each model in each column is in bold.

		VADE-Finance-Test		FAB v1		FAB v2		BigFinance-Bench		Frontier-Finance	
Model	Stage	P	R	P	R	P	R	P	R	P	R
Qwen3.5-9B	base	0.0	40.8	30.0	46.5	11.1	35.4	6.0	30.0	1.0	17.1
	epoch 4	0.0	54.8	42.0	71.5	**25.9**	**63.6**	10.0	56.1	1.0	38.6
	epoch 10	0.0	**73.1**	**54.0**	**78.5**	**25.9**	**63.6**	**18.0**	**58.7**	**3.0**	**40.8**
Qwen3.5-27B	base	6.7	70.1	58.0	79.6	29.6	61.0	14.0	41.8	0.0	24.1
	epoch 4	17.3	**85.3**	58.0	79.6	40.7	75.7	**22.0**	**64.3**	0.0	41.2
	epoch 10	**21.3**	**85.3**	**62.0**	**82.6**	**51.9**	**81.2**	**22.0**	59.2	**4.0**	**43.7**

On the legal tasks, Claude Opus 5 and Grok 4.6 are close, at 66.0% and 64.5%, and the other four models keep the order they have in finance. Opus leads on the analysis and research tasks and Grok on review. Drafting is the weakest task type for four of the six models, yet Qwen3.8-27B has the highest drafting score of any model. On the two preview sets, Grok 4.6 again leads, at 59.8% in energy and 73.4% in medicine, and MiniMax-M3 is again last.

Turn, time, and length statistics are given in Appendix B.1.

4.3 Failure Analysis

For every failed answer, the judge records which of six whole-answer checks it fails; one answer can fail several. Unsupported facts and inferences are by far the most common, present in 85.9% of failed answers. The SouthState task of Figure 2 illustrates the pattern: the allowance for credit losses rose 33% in nominal terms, and the sources determine that this rise is non-diagnostic, because gross loans grew 39% over the same period and acquisitions changed the loan mix, so coverage actually fell. All six models reach this conclusion, but each then adds a claim: three attribute the change in the allowance to the acquisitions, an inference the sources do not support and one of the task's fatal conditions (Appendix D.3). Internal inconsistencies appear in 64.8% of failed answers and unmet requirements in 57.2%, usually alongside an unsupported inference; errors of scope or period (37.6%), computation or units (32.6%), and professional concept (29.6%) are less frequent but rarely occur alone. Appendix C gives the distribution by model and task type, the classification procedure, and case studies for each kind.

5 The Tasks as Supervision

5.1 Setup

We split the finance tasks into 400 training and 75 test tasks, keeping tasks that share a source, a family, or near-duplicate content on the same side and matching the industry and category mix across sides. On the training tasks we fine-tune Qwen3.5-9B and Qwen3.5-27B on verified solver trajectories (Appendix E), and evaluate on the held-out test tasks, VADE-Finance-Test, with the solving environment and judges of §4, reporting the pass rate and the item pass rate, the share of an answer's scored items that pass. We also evaluate on the public questions of four external finance benchmarks, FAB v1 (Bigeard et al., 2025), FAB v2 (Vals AI, 2026), BigFinanceBench (Wang et al., 2026), and FrontierFinance (Zhang et al., 2026b), with a leakage filter (Appendix E).

5.2 Results

On VADE-Finance-Test, after ten epochs Qwen3.5-27B raises its pass rate from 6.7% to 21.3% and its item pass rate from 70.1% to 85.3%. Qwen3.5-9B passes no whole task, but its item pass rate rises from 40.8% to 73.1%.

Both models also improve on all four external benchmarks. The largest gains in pass rate are on FAB v2 for Qwen3.5-27B, from 29.6% to 51.9%, and on FAB v1 for Qwen3.5-9B, from 30.0% to 54.0%. The item pass rate rises more than the pass rate, and since training used only VADE tasks, the external gains suggest the tasks teach a transferable procedure rather than answers.

Table 3: Automated task construction methods compared. "Indep.": a second agent reproduces the reference from the evidence without seeing it; rejection sampling and solver panels re-solve the task but do not rebuild the reference. ✓: yes; ˜: partial; –: no.

	Deliverable	Reference from	Verification	Indep.
Execution oracle available				
SWE-Gym, SWE-smith, R2E-Gym	code patch	repository tests	execution	–
CLI-Universe	command sequence	technical materials	fail-to-pass tests	–
TaskCraft	short answer	tool execution	rejection sampling	˜
No execution oracle				
FinForge	multiple choice	LM synthesis, curated corpus	LM validation, expert spot-check	–
AGORA	single value	workplace documents	solver filtering	˜
SPARK	short answer	knowledge-graph path	graph facts	–
WebSailor-V2	short answer	entity in web-expanded graph	LM judge	–
QUEST	long-form report	synthesized rubric tree	rubric reward	–
VADE (ours)	complete analysis, item contract	professional sources	independent reproduction, cross-family review	✓

6 Related Work

Automated task construction. Methods differ in where the reference answer comes from and what checks it (Table 3). With an execution oracle, a task is verified by running it: software tasks against repository tests (Pan et al., 2025; Yang et al., 2025; Jain et al., 2025; Liang et al., 2026), terminal tasks against fail-to-pass tests (Hua et al., 2026), and tool tasks by re-executing the tool (Shi et al., 2026; Prabhakar et al., 2025). Without one, existing methods narrow the task until it can be checked, to multiple choice (Matlin et al., 2026), a single value over workplace documents (Guo et al., 2026), or an answer fixed by a knowledge graph (Park et al., 2026; Li et al., 2026), or keep a long-form deliverable but trust a reference from the same process that wrote the task (Xie et al., 2026; Cai et al., 2026). VADE keeps the deliverable a complete analysis and replaces the oracle with a reference reproduced independently from the sources.

Professional benchmarks and agentic training. Expert-authored benchmarks grade professional deliverables against expert judgment or rubrics, across occupations (Patwardhan et al., 2026; Vidgen et al., 2026; Bennett et al., 2026), in finance (Bigeard et al., 2025; Vals AI, 2026; Wang et al., 2026; Zhang et al., 2026b; Cho et al., 2026), and in law (Grupen et al., 2026); they define the task format and grading that VADE reproduces without expert authoring, and we adopt their solving environments (§4) and public questions as transfer targets (§5). Our training is the rejection-sampling recipe of STaR (Zelikman et al., 2022) as applied to agents (Zeng et al., 2024; Pan et al., 2025), with the scoring contract as the filter; neighbouring methods go on to reinforcement learning on their own tasks (Shi et al., 2026; Xie et al., 2026; Li et al., 2026; Park et al., 2026), for which the item-level contract is a reward of the same kind (Appendix G).

7 Discussion

7.1 Which Steps Need a Frontier Model

Because every step is checked downstream, removing one shifts cost rather than quality; the more useful question is how capable a model each step needs. We replayed the complete production trajectory of one finance task with the open-weight Qwen3.8-27B, a capable solver (Table 1): at each of the 81 model steps, it redid the step from a snapshot of the environment, and we rated both its output and the original on whether each could be used downstream as is (Table 4). Qwen's output was usable as is at 27 steps and unusable at 21, compared with 57 and 11 for the original run. It matches the original on well-bounded steps, such as the novelty check and numerical reconstruction, but falls short on steps that call for judgment or self-checking: it passes a task after explaining away a conflict it has noticed, and it reports checks it has not completed. These steps require more than the ability to solve the tasks (Appendix F.1).

Table 4: Outputs of the original run and of Qwen3.8-27B at the 81 model steps of one finance task: usable as is, needs fixes (the main work can be kept), or unusable (a key decision is wrong or no complete output is produced). A step counts under a kind of error if it made the error or, when its job was to check, let it pass.

	Original run		Qwen3.8-27B	
Usable as is	57		27	
Kind of error	Needs fixes	Unusable	Needs fixes	Unusable
Unstated requirement	5	9	6	8
Wrong decision	2	1	3	8
Unverified claim	1	1	7	1
Incomplete handoff	4	0	7	0
Wrong format	1	0	6	0
Factual slip	0	0	4	1
No output	0	0	0	3
Total	13	11	33	21

7.2 Findings from Production

The answer comes before the question. If the question is written first and an agent then searches the sources for its answer, the question tends to fix requirements that the evidence may not support: of 20 such attempts, 15 reached an answer that appeared to pass verification, but five of them first needed details corrected, four more than under the current engine. We later found that a task built this way can be unreliable even when its answer is reproduced independently: in one case, the question was written from the selected evidence alone, while the sources supported a better answer. This is why VADE writes the question last (Appendix F.2).

Length is not difficulty. More sub-questions do not make a task harder: across tasks with 5 to more than 20 contract items, the six models' pooled pass rate stays between 33% and 40%. Instead, models fail at judgments that change the conclusion, such as which scope, version, or definition of the data applies; difficulty evolution therefore adds such a relationship rather than more sub-questions (§2.1, Appendix F.2).

7.3 Limitations and Future Work

Model and judge bias. An automated engine carries the biases of its models into the tasks, and model judges carry their own; sources from professional practice, workers from several model families, and judging procedures taken from expert-annotated benchmarks reduce these biases but cannot remove them (Appendix B).

Dependence on the sources. The engine can only pose questions whose answers the imported sources determine: every atom must rest on evidence that can be quoted and recomputed, so analyses that need unpublished or unquotable data cannot become tasks, and domains with a richer public record yield more and harder tasks.

Cost. A task costs a median of 24.4M constructing-model tokens (§3.2). Reusing verified problem families and moving substitutable steps to cheaper models would lower it.

Reinforcement learning. Given limited compute, we only fine-tuned on verified solver trajectories; the answer atoms and scoring contracts are verifiable rewards we have not yet used.

8 Conclusion

We introduced VADE, a data engine built on the asymmetry between solving and verifying a professional analysis: each reference analysis is verified by independent reproduction before the question is written, so tasks stay open-ended without expert authoring. The resulting tasks pass expert review at a high rate, expose where current models fail, and improve a smaller model when used for training. These results show the potential of scaling verifiable agent data for professional work beyond what experts can write.

AI USE STATEMENT

Generative AI is the object of this work: the engine of §2 uses language models to construct, reproduce, review, and grade every task, as reported throughout the paper. The released tasks were therefore generated with AI assistance: their content is anchored to external sources and supervision signals, namely the frozen source documents, code checks, independent reproduction, and human review, but language models rewrote and organized that material into the tasks. Beyond the method itself, we used generative AI tools to draft and edit parts of the manuscript, to create and refine figures, and to write and refactor parts of the engine and evaluation code; they also assisted in summarizing results, which the authors verified against the raw records. We did not use them to formulate the hypotheses or to design the methodology. All AI-assisted text, code, and figures were reviewed by the authors, and the authors take responsibility for the final content, claims, and artifacts.

ETHICS STATEMENT

All sources are publicly available professional materials: company filings, earnings materials, court opinions, statutes, and regulatory documents. The engine uses no private or personal data; the names that appear in the tasks are those of public companies and of officers in their public disclosures. Released tasks are distributed with their source documents, which are public SEC filings and transcripts of public earnings calls produced from the original audio, together with source URLs and file hashes; the human review raised no copyright concerns. The two human reviewers were recruited through an online expert-sourcing platform, paid $100 per hour, and informed of the construction pipeline and the purpose of the study; the review involved no human-subjects research, since reviewers assessed model-generated tasks against public documents, and no personal data about the reviewers is released. The tasks are research artifacts for evaluating and training agents; their reference analyses are produced by models and checked by sampling, may contain errors, and are not financial, legal, or medical advice. Because both the tasks and the grading are produced by language models, they inherit those models' biases (§7.3); agents trained on them should not be deployed for real financial, legal, or medical decisions without human oversight.

REPRODUCIBILITY STATEMENT

The construction procedure is described in §2 and Appendix A, which also gives the discovery routes, a worked example, and a full scoring contract; the solving environments and the judging protocol in Appendix B; per-model and per-category results in Table 1; and the training split, trajectory filtering, and fine-tuning configuration in Appendix E. We release the 100 showcase tasks with their frozen source documents, reference analyses, scoring contracts, and human review records, together with the intermediate artifacts of their construction, namely the candidate, evidence, independent reproduction, comparison, and quality-gate records of each task, all bound to the frozen version they were produced on. We also release the solver trajectories and item-level judge verdicts behind Tables 1 and 2, the fine-tuning data selected from them, and the code of the engine, the solving environments, and the judges as supplementary material, so that every number in the paper can be traced to a record.

REFERENCES

Austin Bennett, Bertie Vidgen, Arnav Garg, and Akul Datta. Introducing APEX-Agents 1.1. https://www.mercor.com/blog/introducing-apex-agents-1-1/, 2026.

Antoine Bigeard, Rayan Krishnan, Shirley Wu, and Langston Nashold. Finance agent benchmark: Benchmarking LLMs on real-world financial research tasks. *arXiv preprint arXiv:2508.00828*, 2025. URL https://arxiv.org/abs/2508.00828.

Songcheng Cai, Zhiheng Lyu, Yuansheng Ni, Xiangchao Chen, Baichuan Zhou, Shenzhe Zhu, Yi Lu, Haozhe Wang, Chi Ruan, Benjamin Schneider, Weixu Zhang, Xiang Li, Andy Zheng, Yuyu Zhang, Ping Nie, and Wenhu Chen. SWE-QA-Pro: A representative benchmark and scalable training recipe for repository-level code understanding. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. URL https://arxiv.org/abs/2603.16124.

Eric Cho, Shawn Huang, Alice Lu, and Andy Lyu. Hedge-Bench: Benchmarking agents on hard, realistic tasks pertaining to financial reasoning. *arXiv preprint arXiv:2606.03918*, 2026. URL https://arxiv.org/abs/2606.03918.

Niko Grupen, Gabe Pereyra, and Julio Pereyra. Introducing Harvey's legal agent benchmark. https://www.harvey.ai/blog/introducing-harveys-legal-agent-benchmark, 2026.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2308.11462.

Honglin Guo, Qi Zhang, Yu Zhang, Weijie Li, Rui Zheng, Zhikai Lei, Qiyuan Peng, Zhiheng Xi, Tao Gui, and Qi Zhang. AGORA: An archive-grounded benchmark for agentic workplace document reasoning. *arXiv preprint arXiv:2606.24526*, 2026. URL https://arxiv.org/abs/2606.24526.

Zhanbo Hua, Yifan Yao, Weihao Xie, Yongchi Zhao, Minghao Liu, Ruizhi Qiu, Zhewei Huang, Zun Wang, Yiyan Ji, Yunhai Ye, Letian Zhu, Xinping Lei, Han Li, Zhiyuan Ma, Zili Wang, Zhaoxiang Zhang, and Jiaheng Liu. CLI-Universe: Towards verifiable task synthesis engine for terminal agents. *arXiv preprint arXiv:2606.22883*, 2026. URL https://arxiv.org/abs/2606.22883.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. FinanceBench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*, 2023. URL https://arxiv.org/abs/2311.11944.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.07164.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Dingchu Zhang, Xixi Wu, Xinmiao Yu, Jialong Wu, Xinyu Wang, Zile Qiao, Zhen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Zhiqin Xu, Shuai Wang, Minhao Cheng, and Jingren Zhou. WebSailor-V2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.13305.

Jiarong Liang, Zhiheng Lyu, Zijie Liu, Xiangchao Chen, Ping Nie, Kai Zou, and Wenhu Chen. SWE-Next: Scalable real-world software engineering tasks for agents. *arXiv preprint arXiv:2603.20691*, 2026. URL https://arxiv.org/abs/2603.20691.

Glenn Matlin, Akhil Theerthala, Anant Gupta, Anirudh Jaidev Mahesh, Yi Mei Ng, Rayan Castilla, and Sudheer Chava. FinForge: Semi-synthetic financial benchmark generation. *arXiv preprint arXiv:2601.06747*, 2026. URL https://arxiv.org/abs/2601.06747.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2412.21139.

Hyobin Park, Taeseop Kim, and Dong-Geol Choi. SPARK: Self-play with asymmetric reward from knowledge graphs. *arXiv preprint arXiv:2605.05546*, 2026. URL https://arxiv.org/abs/2605.05546.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.04374.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.03601.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.10055.

Yu Su, Diyi Yang, Shunyu Yao, and Tao Yu. Language agents: Foundations, prospects, and risks. In Jessy Li and Fei Liu (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pp. 17–24, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-tutorials.3. URL https://aclanthology.org/2024.emnlp-tutorials.3/.

Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=1i6ZCvflQJ. Survey Certification, Featured Certification.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.12784.

Vals AI. Finance agent v2. https://www.vals.ai/benchmarks/fabv2, 2026.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents. *arXiv preprint arXiv:2601.14242*, 2026. URL https://arxiv.org/abs/2601.14242.

Alex Wang, Georg Meinhardt, Jacob Katz, Joseph H. Kim, Pratyush K. Chaudhary, Chase Blagden, and Eric Xu. BigFinanceBench: A workflow-grounded benchmark for financial-research agents. *arXiv preprint arXiv:2606.03829*, 2026. URL https://arxiv.org/abs/2606.03829.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.16741.

Jian Xie, Tianhe Lin, Zilu Wang, Yuting Ning, Yuekun Yao, Tianci Xue, Zhehao Zhang, Zhongyang Li, Kai Zhang, Yufan Wu, Shijie Chen, Boyu Gou, Mingzhe Han, Yifei Wang, Vint Lee, Xinpeng Wei, Xiangjun Wang, Yu Su, and Huan Sun. QUEST: Training frontier deep research agents with fully synthetic tasks. *arXiv preprint arXiv:2605.24218*, 2026. URL https://arxiv.org/abs/2605.24218.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2504.21798.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2203.14465.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. URL https://arxiv.org/abs/2310.12823.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Xuefei Cao, Lawrence Keunho Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason E Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=N3dXUHY5dD.

Yuhao Zhang, Ozan Koyluoglu, Thejas Venkatesh, Richard Diehl Martinez, Vishank Bhatia, Arash Alidoust, and Ashwin Paranjape. FrontierFinance: A challenging benchmark for measuring frontier intelligence of finance agents. *arXiv preprint arXiv:2608.11683*, 2026b. URL https://arxiv.org/abs/2608.11683.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.05685.

Table of Contents in Appendix

A Task Construction **15**

A.1 Construction Procedure 15

A.2 Discovery Routes 18

A.3 Running Example: Public Storage Margin Analysis 19

A.4 Scoring Contract Example: Travelers Underwriting Decomposition 20

B Evaluation Protocol **21**

B.1 Solving Environments 22

C Failure Analysis Details **23**

C.1 How Failures Are Classified 23

C.2 By Model 23

C.3 By Task Type 23

C.4 Cases 23

C.5 Borderline Verdicts 24

D Case Studies **25**

D.1 Travelers: separating underlying underwriting improvement 25

D.2 Northrop Grumman: reconciling segment drivers to consolidated income 25

D.3 SouthState: population normalization before judgment 26

D.4 Marvell: foundry commitments without double counting (Disclosure Analysis) . . 27

D.5 CrowdStrike and Snowflake: can two retention rates be ranked? (Comparables) . . 27

D.6 EOG Resources: aligning actuals with guidance (Earnings Analysis) 27

D.7 Informatica: rebuilding a precedent-transaction set (Precedents) 27

D.8 Advance Auto Parts: event window and comparator contamination (Market Analysis) 27

D.9 Discussion 27

E Training Details **28**

F Stage Replacement and Production Findings **28**

F.1 Stage Replacement Study 28

F.2 Findings from Production 30

G Extended Related Work **30**

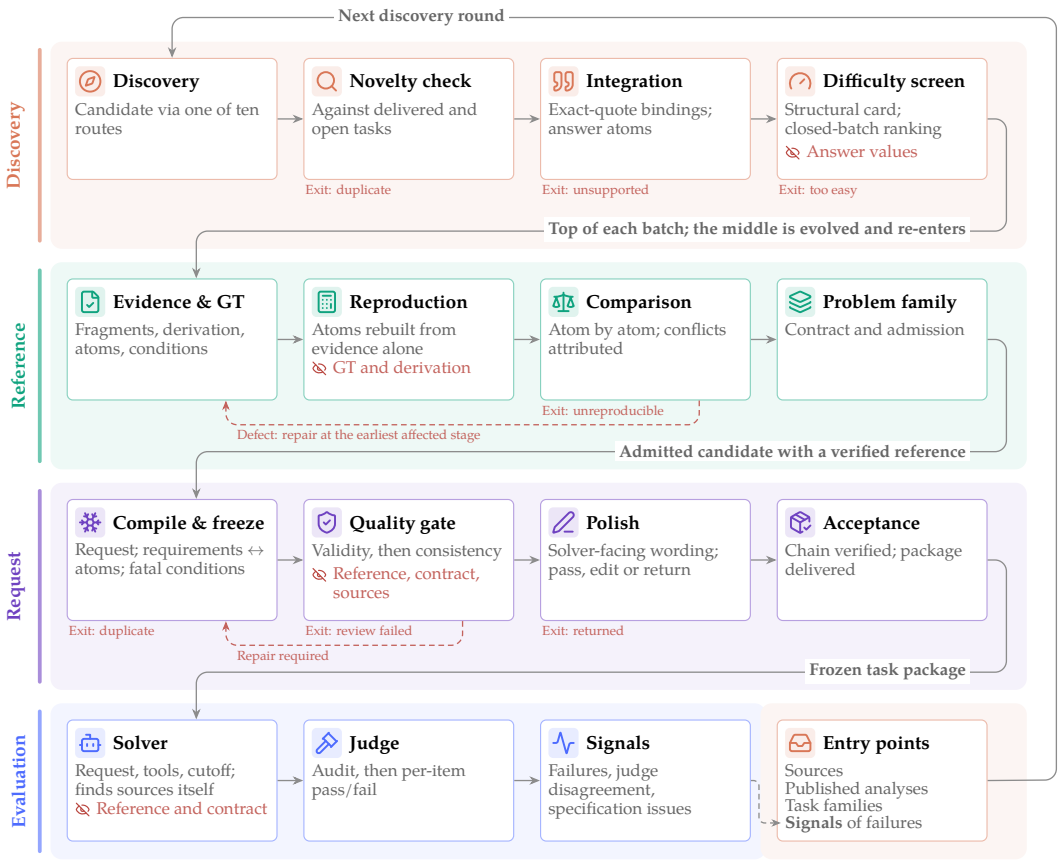

Figure 4: The complete construction flow of VADE, one box per step, with what the step produces under its title; a crossed eye marks what the step does not see (at the quality gate, the validity review). **Discovery:** a candidate enters through one of ten routes, is checked for novelty, is bound to verbatim quotations with answer atoms, and is screened for difficulty from a blind projection without answer values; the top of each batch proceeds, the middle is evolved once and re-enters as a new candidate, and the bottom is kept as reserve. **Reference:** evidence fragments and a reference analysis are built, the reproducer rebuilds every answer atom from the evidence alone, a third worker compares the two atom by atom and attributes every difference, and the verified analysis is admitted to a problem family. **Request:** the question and its scoring contract are compiled from the verified analysis and frozen, pass the validity and consistency reviews and a final wording pass, and are accepted as a task package. **Evaluation:** the solver receives only the question, the allowed tools, and the cutoff; the judge rules item by item; failures, judge disagreements, and specification issues are recorded as signals for later discovery. Every worker result is checked by code before the next step uses it; a defect returns to the earliest affected step (dashed red), and red labels mark where candidates exit.

A Task Construction

A.1 Construction Procedure

Figure 4 shows the complete construction flow; the paragraphs below follow it step by step. Before the next step uses a worker's result, code checks it against the files the worker was given: every quotation must appear verbatim in its source, every arithmetic expression must recompute to its stated result, and every reference must resolve; a result that fails is redone. Whether a result is good enough to hand on is decided by the orchestrator.

Candidate discovery. The orchestrator sets the scope for a set of sources: which sources to use, their boundaries, the parts to exclude, and the excerpts it has already checked. A worker then proposes one to three candidates along an assigned route. Each candidate states the research question a professional would actually ask, the decision it informs, the one judgment its answer turns on, and

every relationship needed to reach the conclusion. Among the directions rejected at this step are those whose difficulty comes only from stacking unrelated sub-questions, those that depend on non-public data, and those that can end only in an unsupported causal or predictive claim.

Novelty check. Code lists the closest delivered tasks and open candidates by text similarity; the orchestrator compares each of them with the candidate on the central analysis object, the conclusion target, the necessary relationships, and the deliverable, and decides whether the candidate is a duplicate. Changing only the company or the period while the analysis stays the same does not make a candidate new. Duplicates exit.

Integration. The orchestrator first confirms that the conclusion target has one determinate reading: one measure, one period, one population, one unit. A worker reopens the sources, binds every relationship to a verbatim quotation and its location, and writes the smallest set of answer atoms needed to evaluate the candidate. Each atom has an expected value or category, keeps reported and computed values apart, and names the quotations it rests on; its label says what is measured, never what was found. A candidate whose sources do not support a unique answer exits; one that lacks a load-bearing public source waits until the source is supplied.

Difficulty screen. Code projects the candidate onto a blind view: a neutral objective, the bound excerpts, and the form of each output (label, type, and unit), with every answer value removed, and checks that no answer value appears outside the excerpts. From this view alone, a worker without tools estimates the shortest professional path to the outputs and scores it from 0 to 4 on eight structural dimensions: relation depth, cross-source reconciliation, entity or population identity, temporal or version reconciliation, conditional branching, professional judgment, state transitions, and post-action verification. It also raises two flags: a material ambiguity returns the candidate to integration to fix the scope, and a task that can be bypassed as a whole, with its conclusion stated outright in an excerpt or forced by units or sign, is kept as basic reserve. The remaining candidates wait until a batch is full. A second model then ranks the batch by comparing the candidates with one another and gives a reason for each adjacent pair; the first and last candidates of the previous batch serve as anchors that keep the batches on one scale. The top of the batch proceeds to reference construction and is budgeted first, and the bottom is kept as basic reserve. The middle undergoes one round of difficulty evolution: a relationship from the authorized sources that changes the conclusion is added, for example a temporal or version reconciliation, a population reconciliation, a hierarchy of source authority, or a conditional branch, and the result passes again through the novelty check, integration, and a later batch as a new candidate. Both the scoring criteria and the ranking criteria were refined from downstream solver results.

Evidence and reference analysis. A worker reopens the sources and treats nothing from the candidate stage as established. It checks, binding by binding, that the quotation supports the stated relationship and that entity, period, unit, basis, and population agree, and it searches the sources for omissions that would change the answer, a ranking, a sign, an attribution, or a classification. That a quotation exists and that it supports the claim are separate decisions. The worker then extracts the evidence fragments, completes the domain's semantic review (in finance, the reporting entity and the accounting basis, among others), writes out every derivation step from the fragments to the conclusion, and produces the reference analysis: a core conclusion, answer atoms with tolerances, accepted alternatives, and the conditions under which the conclusion holds. Every atom rests on at least one evidence fragment and one derivation step. When it finds a defect, the worker names the earliest step that must change: integration if a binding or the relationship itself is wrong, or this step if only its own work must be redone.

Independent reproduction. The reproducer receives only the analytical objective, the evidence fragments, and a neutral description of each required output; it does not see the reference analysis, the derivation, the scoring contract, or the question. It derives every output from the evidence alone and records the calculation or classification and the fragments it used. An output that the evidence does not determine uniquely is reported as a gap rather than answered.

Comparison. A third worker compares the reference analysis with the reproduction atom by atom and records each atom as an exact match, a match within tolerance, a difference, or not reproduced; tolerances are never widened. Every difference keeps both sides' content, the dimensions it touches (entity, period, population, basis, unit, calculation, or conclusion), and its effect (none, load-bearing,

or unresolved). Agreement is not proof: the worker reopens the evidence and checks the premises the two results may share, namely the entity and period, the source and its version, the load-bearing calculation or classification, and the boundary of the inference. Every load-bearing difference is attributed. A defect in the task returns the candidate to the earliest affected step, and only this attribution counts as a repair; an error of the reproducer leads to a fresh reproduction; an execution failure reruns only what failed; and a difference that cannot be attributed goes to independent adjudication. The two analyses are equivalent only when every atom is reproduced, no load-bearing or unresolved difference remains, and the shared premises hold. The same conflict may be repaired at most twice; a candidate that still does not converge exits.

Problem family. A worker selects a problem family that is already reusable, or defines a narrow boundary for this analysis: its objective, positive boundary, explicit exclusions, evidence roles, relationship shape, and output contract. It then checks the analysis against each of these six parts separately and admits it only if all six pass. A family defined from one instance cannot claim to be reusable or be used to extend to other instances; reusability requires a separate review with two distinct positive examples, one near-negative case, a boundary draft, and an independent verification. A candidate whose boundary cannot be settled exits.

Compilation and freezing. From the verified analysis, a worker writes the question the solver will see: one high-level analytical objective, with the basis, period, and information cutoff that make its answer unique. The question does not list the hidden calculation steps, the order of the sources, or intermediate conclusions, and it does not state hidden answers. The scoring contract is written with it. Every requirement of the question maps to answer atoms; every atom is either scored or explicitly unscored, and no load-bearing atom may go unscored; every load-bearing atom has a stated fatal condition; every accepted alternative of the reference analysis is marked as accepted or as not applicable to this question; and helpful but unrequired derivations are recorded as unscored. Code checks that these links hold in both directions and that each requirement is anchored in the exact wording of the question; once they do, the question, the reference analysis, and the scoring contract are frozen together as one version. A second novelty check then compares the frozen question with the delivered tasks, and a duplicate exits.

Quality gate. Two reviews run in parallel. The validity review sees only what the solver will see (the question, the cutoff, and the allowed tools) and uses public sources to judge whether the entity, period, unit, version, and cutoff are unambiguous; whether the requested measures are compatible in definition, unit, and basis; whether the question separates association, official findings, and causal claims; whether every load-bearing input can be obtained with the allowed tools before the cutoff; whether another supported definition or version would give a materially different answer; and whether the question asks for one high-level result. The consistency review receives the frozen task and the source files and independently re-derives every load-bearing number and conclusion. It checks that the question and the reference analysis refer to the same object, that every scored atom is supported by evidence, that reasonable alternatives are accepted, that no item requires something the question does not ask for, that the tolerances are reasonable, and that the conclusion does not go beyond the evidence. If either review fails, an adjudicator reads every review and confirms each finding against the task; only confirmed defects are repaired. A repair returns to compilation or to the evidence step, after which the task is refrozen and reviewed again; a task may be repaired at most five times before it exits.

Polish. A worker sees the question as the solver sees it, the full scoring contract, and the allowed tools, but not the reference analysis, the sources, or earlier reviews. It checks that the question states the cutoff, what identifies each source, the entity and scope, the deliverable, and the answer form and boundary conditions that the scoring requires, and that it neither gives away an answer nor recites the items. It passes the question; edits its wording only, which leaves the scoring contract and the reference analysis unchanged so that the gate's result stands; or returns it for repair, for example when the problem lies in the scoring contract.

Acceptance. Code confirms that the chain is complete: the reference analysis is built, the comparison is equivalent, the problem family is admitted, and the quality gate and the polish have passed, all for the current version. The task package is then delivered; solving and judging come after delivery.

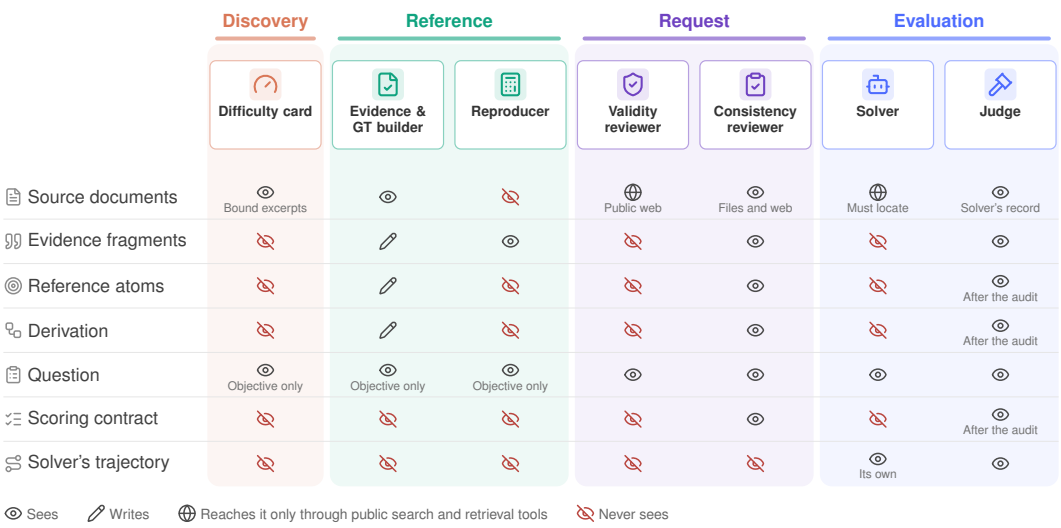

Figure 5: What each isolated role of Figure 4 is allowed to see. The difficulty card scores the relationship from the bound excerpts, without answer values and before any evidence fragment is built; the reproducer rebuilds the atoms from the evidence fragments alone; the validity reviewer and the solver work from the request and the public web; the consistency reviewer sees the whole package and the bound source files; the judge sees the solver's record and evidence first, and the reference and scoring contract only after it has filed a preliminary audit. The question exists only after the reference analysis is verified, so the earlier roles see just the analytical objective.

Solving, judging, and signals. The solver receives only the question, the allowed tools, and the cutoff, and finds and reads the sources itself; the judge rules item by item under the scoring contract (§4, Appendix B). Each evaluation records three kinds of signal: the items on which solvers fail, the items on which judges disagree, and genuine defects that a judge finds in the scoring contract or the reference analysis. The first two are kept as candidate directions for later discovery and are verified against evidence before use; they were not fed back automatically in this run. A defect that a judge finds returns the task to the earliest affected step.

Returns and archiving. A defect returns the task to the earliest step it affects, and the steps after it are redone on a new version; reviews, solver runs, and judgments of an earlier version are not carried over. An execution failure, such as a provider error, a timeout, or output that does not match the required form, reruns only that step and does not count as a repair. Candidates exit for one of the following reasons, and every exit is kept on record: duplicate, novelty undecided, unsupported by the sources, basic reserve, not reproducible, family undecided, review failed, repair limit reached, or cost above four times the batch median.

What each role sees. Figure 5 shows what each isolated role is allowed to see. These are the roles whose separation the construction relies on: the difficulty worker judges a candidate without its answer values, the reproducer rebuilds the answer without the reference analysis, the validity review judges the question from the solver's position, and the solver never sees the reference analysis or the scoring contract. The other roles are not separated in this way. The orchestrator reads every result but writes nothing into the task package; the comparison worker sees both analyses and the evidence; and the polish worker sees the question, the scoring contract, and the allowed tools.

A.2 Discovery Routes

A route tells the discovery worker where to start looking; it is a preference, not a promised output. The worker tries the assigned route first and may switch to another route before a candidate forms only when the sources require it; once a candidate forms, the route it actually came from is recorded with it. The two failure-first routes use what is known about model failures to choose where to look, but every relationship they propose must be closed from the sources like that of any other candidate. Table 5 lists the ten routes.

Table 5: The ten discovery routes, grouped by where they start.

Route	Starting point and rule
Source-first	
Raw source relationships	Extract analytical relationships directly from primary sources.
Source-supported extension	Use a verified simple relationship as a search scope; keep only external extensions that add a load-bearing relation.
Analysis-first	
Complete-analysis reverse authoring	Start from a public analysis with conclusion, method, facts, and sources; verify against primary sources.
Verified-relationship reverse authoring	Build a higher-level task on a verified data or evidence relationship.
Real problem, then answer	Start from a real professional objective; compile no question until evidence and reference analysis close.
Family-first	
Mature-family expansion	Add a new entity, period, event, or state to a validated narrow family.
Necessary-relation accumulation	Create a child task only when every inherited relation stays necessary and new evidence adds a conclusion-sensitive dependency.
Taxonomy-gap discovery	Prioritize underrepresented categories without relabeling ordinary tasks to fill quotas.
Failure-first	
Historical error-structure transfer	Use known model failure structures to find new source-grounded instances, without copying old questions.
Model-boundary hypothesis	A frontier model names relations missing from its own prospective solution path; every hypothesis must be closed from sources.

A.3 Running Example: Public Storage Margin Analysis

Figure 6 follows one task, PS-MARGIN, through the first three rows of Figure 4. The task comes from an earlier production batch and is not part of the benchmark described in §3.3; we use it to illustrate the method.

Question. "Using Public Storage's public disclosures available through July 29, 2026, assess whether its year-over-year Same Store Facilities gross margin after indirect costs movement in Q2 2026 differed from the movement for the six months ended June 30, 2026, and quantify the difference. Confine the analysis to the disclosed Same Store Facilities population and exclude separately reported non-same-store facility populations. Support the judgment by reconciling the disclosed revenue and direct and indirect operating costs to reported same-store NOI and by comparing the revenue and cost relationships for both periods. Explain the differing margin movements with a bounded cost-to-revenue arithmetic attribution, preserving the disclosure's rounding limits and avoiding causal claims."

Source data. The evidence comes from the Same Store Operating Performance table of Public Storage's Q2 2026 earnings supplement, which covers 2,755 facilities and 192.1M net rentable square feet, about 83% of the U.S. consolidated portfolio (Table 6). All four columns refer to the same facility population.

Table 6: PS-MARGIN source data (thousands of USD). Indirect costs shown as positive amounts; the source presents them as deductions from Direct NOI.

Item	**Q2 2026**	**Q2 2025**	**H1 2026**	**H1 2025**
Total revenue	1,006,549	1,012,439	2,007,382	2,013,460
Property taxes	111,362	105,186	217,485	212,411
On-site managers	33,713	34,325	68,075	68,907
Repairs	20,824	20,172	43,431	44,182
Utilities	11,888	11,506	27,344	27,469
Marketing	22,029	20,732	44,625	44,382
Other direct	27,855	26,437	55,999	53,946
Direct costs	227,671	218,358	456,959	451,297
Indirect costs	32,505	30,748	64,650	62,133
NOI	746,373	763,333	1,485,773	1,500,030
Disclosed margin	**74.2%**	**75.4%**	**74.0%**	**74.5%**

Reference analysis. The reference analysis has nine answer atoms: the same-store scope; the reported changes in revenue and in direct and indirect costs for the quarter and for the six months; the reconciliation of the six direct-cost components, and of revenue less costs to NOI, in all four columns; the quarterly and the six-month margin movements; their difference; and a bounded arithmetic attribution of that difference. On the issuer's one-decimal margins, the margin fell 1.2 points in Q2 and 0.5 points over the six months, so the Q2 movement was 0.7 points worse. Recomputed from the thousand-dollar amounts, the additional Q2 erosion is about 0.76 points, of which about 0.70 comes from the direct-cost burden and about 0.06 from the indirect-cost burden. The scoring contract accepts the amount-based figure when it is labeled as a cross-check, and either rounding of the two burdens as long as they add up to the amount-based gap. The six-month period includes Q2, so the two periods are not independent, and the attribution is arithmetic: an answer that states the decline in percent rather than percentage points, or that explains it by operational causes, fails.

Independent reproduction. The reproducer received only the analytical objective, the evidence fragments, and a neutral description of each output. Its first reproduction recovered the margins and their difference but omitted the bounded attribution, which the comparison recorded as a load-bearing difference. In the next reproduction all nine atoms were reproduced, and the comparison, after reopening the evidence to check the premises the two analyses share, found them equivalent.

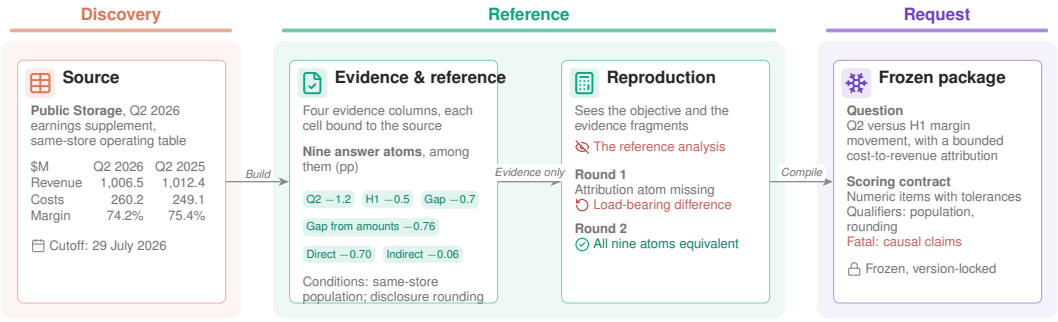

Figure 6: The PS-MARGIN running example through the first three rows of Figure 4. One earnings table becomes four bound evidence columns and a reference analysis of nine answer atoms, five of which are shown; the one-decimal gap of −0.7 points and the gap of −0.76 points recomputed from the thousand-dollar amounts are both part of it. The reproducer, which never sees the reference analysis, missed the attribution atom in its first reproduction and reproduced all nine atoms in the next; the frozen package exposes only the question and the tools to the solver.

A.4 Scoring Contract Example: Travelers Underwriting Decomposition

This section reproduces the complete scoring contract of the Travelers case study (§D.1), an Adjustments task. The question reads: "Using only public information available through July 17, 2026, assess how much of The Travelers Companies, Inc.'s consolidated underwriting improvement for the three months ended June 30, 2026 versus June 30, 2025 reflects improved current-accident-year performance rather than changes in catastrophe losses and favorable prior-year reserve development. Support your judgment by presenting the signed reported-to-underlying combined-ratio bridge for both quarters, decomposing the year-over-year reported improvement and quantifying the share retained in the underlying movement. Use the consolidated three-month pre-tax underwriting-income reconciliation as a non-interchangeable cross-check, likewise showing each quarter's signed bridge, the year-over-year decomposition, and the retained share. Conclude with a bounded historical classification of the underlying direction, retained share, and dominant driver; do not infer persistence, causality, reserve adequacy, solvency, or future performance."

Table 7 maps each of the question's four requirements to the answer atoms it makes scorable and lists the five accepted alternatives; Table 8 gives the nine atoms, and Table 9 the eight fatal conditions. All atoms are fatal, so omitting or materially misstating any one fails the answer, and no atom is left unscored.

Table 7: Travelers question requirements and the answer atoms each makes scorable (top), and the accepted alternatives with the atoms they apply to (bottom). Each requirement is anchored in the question's own wording; all five alternatives are accepted for scoring.

Requirement (anchor in the question)	Answer atoms
Assessment: "assess how much of … consolidated underwriting improvement … reflects improved current-accident-year performance"	earnings_quality_judgment
Ratio support: "presenting the signed reported-to-underlying combined-ratio bridge for both quarters …"	ratio_bridge_2026, ratio_bridge_2025, ratio_change_decomp., ratio_retained_share
Pre-tax cross-check: "use the consolidated three-month pre-tax underwriting-income reconciliation as a non-interchangeable cross-check …"	pretax_bridge_2026, pretax_bridge_2025, pretax_change_decomp., pretax_retained_share
Conclusion boundary: "conclude with a bounded historical classification …; do not infer persistence, causality …"	earnings_quality_judgment
Accepted alternative	**Applies to**
Equivalent signed forms of the bridges, e.g. $83.6 + 5.4 - 4.9 = 84.1$	ratio_bridge_2026, ratio_bridge_2025
Retained ratio share as about 9%, or 8.96% when marked as calculation precision	ratio_retained_share
Pre-tax retained share as about 6.1% or approximately 6%	pretax_retained_share
"Limited", "minority", or "modest" in place of "small", with the 9.0% and 6.1% calculations	earnings_quality_judgment
Catastrophe and favorable prior-year development supplying about 91% of the ratio improvement and 94% of the pre-tax improvement, with bases kept apart	both decompositions, both retained shares, earnings_quality_judgment

Table 8: Travelers answer atoms; each is fatal if missing or materially incorrect.

Atom ID	Expected display	Tol.	Unit
ratio_bridge_2026	$83.6\% - (-5.4\,\text{pp}) - 4.9\,\text{pp} = 84.1\%$	± 0.1	%
ratio_bridge_2025	$90.3\% - (-2.9\,\text{pp}) - 8.5\,\text{pp} = 84.7\%$	± 0.1	%
ratio_change_decomp.	6.7 pp improvement: 3.6 pp catastrophe, 2.5 pp PYD, 0.6 pp underlying	—	pp
ratio_retained_share	≈9.0% (0.6 of 6.7 pp)	± 0.2	%
pretax_bridge_2026	\$1,738M − \$578M + \$518M = \$1,678M	± 1	\$M
pretax_bridge_2025	\$1,022M − \$315M + \$927M = \$1,634M	± 1	\$M
pretax_change_decomp.	\$716M improvement: \$409M catastrophe, \$263M PYD, \$44M underlying	—	\$M
pretax_retained_share	≈6.1% (\$44M of \$716M)	± 0.1	%
earnings_quality_judgment	Underlying improved; retained share small; reported improvement predominantly catastrophe-and-PYD-driven.	—	—

B Evaluation Protocol

Every answer is graded independently by two judges, GPT-5.6 Sol and Claude Opus 5, each following the procedure of its domain and ruling item by item. When their verdicts on an answer differ, the two sets of item rulings are passed to a third judge, whose ruling is final. The rubric pass rate is computed from the final item rulings.

Task categories. The category abbreviations of Table 1 are as follows. Finance: Qual./Quant. general qualitative/quantitative analysis, Mkt. market analysis, Comp. comparables, Prec. precedents, Adj. adjustments, Earn. earnings analysis, Discl. disclosure analysis, Model. financial modeling. Energy: Reg. regulation and infrastructure, Mkt. markets and interchange, Gen. generation and capacity, Rel. reliability events, Scen. scenarios and mitigation, Stor. storage, Dem. demand and

Table 9: Fatal error conditions for the Travelers scoring contract. Any triggered condition produces a **Fail** verdict regardless of other atom results.

Condition ID	Description
fec_ratio_bridge	Either quarterly combined-ratio bridge is omitted, uses an incorrect component or sign, or does not reproduce the underlying ratio.
fec_ratio_decomposition	Reported combined-ratio change is not correctly decomposed into catastrophe, PYD, and underlying components, or the retained share is omitted or materially incorrect.
fec_pretax_bridge	Either quarterly pre-tax underwriting-income bridge is omitted, uses an incorrect adjustment sign or amount, or fails to reproduce underlying pre-tax underwriting income.
fec_pretax_decomposition	Reported pre-tax underwriting-income change is not correctly decomposed into catastrophe, PYD, and underlying components, or the retained share is omitted or materially incorrect.
fec_quality_judgment	Answer omits the three-part quality classification or materially misclassifies the underlying direction, retained share, or dominant driver.
fec_scope_population	Analysis uses a segment, year-to-date, audited-status assumption, later disclosure, or other population inconsistent with consolidated three-month Q2 2026 vs. Q2 2025 disclosures.
fec_measurement_basis	The ratio and pre-tax-dollar retained shares are treated as interchangeable or combined without preserving their measurement bases.
fec_historical_boundary	Conclusion asserts persistence, causality, reserve adequacy or redundancy, solvency, or future performance beyond the historical evidence.

forecasting. Medicine: Reg. regulatory decisions, Trial trial registration, Discl. disclosure and versions, Safety safety and benefit-risk, Label product labeling. Law uses the four task types of LAB: analysis, drafting, review, and research.

Finance. The judge works in two steps. First, before seeing the scoring contract, it reads the whole answer, checks the solver's retrieval record, and notes preliminary findings. Second, it rules on every item and every fatal condition of the scoring contract, quoting the answer and giving the evidence and the reason for each ruling. It then runs six checks on the whole answer—professional concepts, scope and period, facts and inference, internal consistency, computation and units, and completion of the task requirements—and resolves each of its preliminary findings. To fail an item or a check, the judge must identify the statement in the answer that fails it. An answer passes only if every item passes, no fatal condition is triggered, and all six checks pass; an item the judge cannot decide does not count as passed.

Law. The judge has no tools. It reads the files or text that the answer delivers and rules on the items of the scoring contract one at a time, following the per-criterion procedure of LAB. An answer passes only if every item passes, and an answer that delivers nothing fails.

B.1 Solving Environments

Finance. The environment follows the tool interface of the Finance Agent Benchmark: web search, EDGAR search, a reader for web pages and documents, a calculator, price history, and FFIEC bank call reports. Every source tool returns only material published before the task's information cutoff, and the solver delivers its answer through a submission tool. A task allows at most 120 tool calls and one hour.

Law. The environment follows the sandbox of LAB: the task's materials are local, the solver works with seven file tools in a container without network access, and it delivers files or text. A task allows at most 200 turns and one hour, with at most ten minutes per model request and two minutes per shell command.

Table 10 summarizes the finance runs of the six models.

Table 10: Finance solving runs of the six models: turns per run (median and interquartile range), wall-clock minutes per run (median), answer length in characters (median), and mean turns of passed and failed answers.

Model	Turns	Minutes	Answer length	Turns, pass / fail
Grok 4.6	30 (20–43)	4.7	8,752	30.4 / 39.5
DeepSeek V4 Flash	25 (17–32)	5.1	7,614	24.2 / 28.1
Claude Opus 5	20 (13–27)	2.5	9,927	18.7 / 23.8
Qwen3.8-27B	22 (14–31)	9.2	7,807	21.0 / 25.4
GLM-5.2	16 (11–24)	2.9	7,674	17.0 / 20.6
MiniMax-M3	22 (15–34)	2.5	7,570	21.1 / 30.6

C Failure Analysis Details

C.1 How Failures Are Classified

Besides checking each item of the scoring contract, the judge runs six checks over the whole answer, and an answer fails if any check fails:

- **Fact and inference**: a statement contradicts the sources, or a conclusion exceeds the evidence;
- **Internal consistency**: numbers, directions, or statements in the answer contradict one another;
- **Task requirements**: an output the task asks for is missing or incomplete;
- **Scope or period**: the wrong population, period, statistical scope, or basis of comparison is used;
- **Computation or units**: an operation, unit, sign, or order of magnitude is wrong;
- **Professional concept**: an accounting, financial, or regulatory concept is misused.

To fail a check, the judge must identify the statement in the answer, the evidence or task requirement it conflicts with, and the error that results. An answer can trigger several checks, so the shares below sum to more than 100%. All shares are of failed answers, taken over the failed answers of the six models on all finance tasks.

C.2 By Model

Table 11: Share of each model's failed answers that trigger each whole-answer check (%). An answer can trigger several checks.

Model	Fact and inference	Internal consistency	Task requirements	Scope or period	Computation or units	Professional concept
Grok 4.6	75.0	47.3	47.8	25.0	28.3	19.0
DeepSeek V4 Flash	85.0	60.7	54.5	34.6	28.0	26.2
Claude Opus 5	83.8	60.4	47.6	34.5	29.3	21.1
Qwen3.8-27B	88.3	67.6	55.3	36.2	33.8	32.7
GLM-5.2	86.3	68.6	65.1	40.9	35.4	30.7
MiniMax-M3	90.1	72.7	65.0	45.6	36.8	39.4
All	85.9	64.8	57.2	37.6	32.6	29.6

For every model, errors of fact and inference are the most common. Weaker models trigger more checks per failed answer: 2.4 on average for Grok 4.6 and 3.5 for MiniMax-M3 (Table 11).

C.3 By Task Type

Errors of fact and inference lead in every task type (Table 12). Scope or period errors peak in precedents (53.3%) and qualitative analysis (48.2%), errors of computation or units in financial modeling (41.7%), and unmet task requirements in precedents (69.3%) and qualitative analysis (67.1%).

C.4 Cases

Each of the following six cases corresponds to one check. In each, most of the answer is right, and the error lies at a point that changes the conclusion.

Table 12: Share of the failed answers in each task type that trigger each whole-answer check (%), over the six models.

Task type	Fact and inference	Internal consistency	Task requirements	Scope or period	Computation or units	Professional concept
Qualitative analysis	79.9	52.4	67.1	48.2	22.0	30.5
Quantitative analysis	83.0	69.4	55.2	31.9	33.8	33.4
Market analysis	92.6	57.0	44.4	35.6	25.2	24.4
Comparables	87.4	61.1	54.7	37.9	33.7	36.8
Precedents	85.3	66.7	69.3	53.3	40.0	24.0
Adjustments	89.1	73.1	48.7	27.6	36.5	24.4
Earnings analysis	84.2	63.0	57.1	31.9	36.2	25.1
Disclosure analysis	90.1	64.1	59.3	44.2	27.4	31.0
Financial modeling	81.7	70.5	58.6	35.6	41.7	32.4

Fact and inference: a recorded adjustment called final (EOG Resources, MiniMax-M3). The model correctly rebuilt the rollforward of asset retirement obligations: a net increase of $37 million in 2026 and $24 million in 2025, a difference of $13 million; it also correctly placed the $15 million from Encino within the $19 million of estimate revisions. It then wrote that the adjustment "was finalized as a measurement-period adjustment." Note 12 of the filing still describes the purchase price allocation as preliminary, and an adjustment that has been recorded is not thereby final.

Internal consistency: a correct table with the opposite explanation (Boeing, DeepSeek V4 Flash). The model's table gives GAAP operating earnings of $604 million and core operating earnings of $294 million for the first half of 2026, both correct. Explaining the FAS/CAS adjustment, it wrote that "excluding it raises operating earnings" and gave $604 + 310 = 294$; the issuer's reconciliation is $604 - 310 = 294$. The table and the explanation cannot both hold.

Task requirements: eligibility conditions without product matching (CMS Part D, GLM-5.2). The task asks when an immediate substitution qualifies. The model introduced five conditions with "only if all of the following are satisfied" but omitted the requirement that the substitute match the original product in dosage form, route of administration, and strength, which the CMS document states on page 47. Without it, the set of acceptable substitutes changes.

Scope or period: a half year in place of the quarter the task names (EOG Resources, Grok 4.6). The task asks to compare actual production in the second quarter of 2026 with guidance for the third quarter and the full year. The model's data include the quarter, the half year, and the guidance, but for the full-year comparison it used the half year. For natural gas liquids, the quarter was 346.8 thousand barrels per day, the half year 339.5, and the midpoint of full-year guidance 342: the quarter lies 4.8 above the midpoint and the half year 2.5 below it, so the change of basis reverses the direction.

Computation or units: basis points inflated a hundredfold (Visa, GLM-5.2). The model correctly took net revenue of $40,000 and $35,926 million for fiscal 2025 and 2024 and total payments volume of $13,894 and $12,988 billion. With units aligned, net revenue is 28.79 basis points of payments volume, against 27.66 a year earlier, a change of $+1.13$ basis points. The answer gives levels of about 2,878.94 and 2,766.09 basis points and a change of $+1{,}128.49$: the levels are about 100 times too large and the change about 1,000 times.

Professional concept: receivables called a contract asset (Best Buy, Claude Opus 5). The model correctly summed four contract liabilities to $1,333 million and correctly noted that the $287 million of restricted cash is not broken down for the membership program. It then called the $528 million of receivables, net "a contract asset, not a liability." An unconditional right to payment is presented as a receivable, whereas a contract asset is a right that still depends on something other than the passage of time; the error changes the class of the asset.

C.5 Borderline Verdicts

Under the strict standard, some failed answers contain errors of little consequence. For NVIDIA, $22{,}563/26{,}044 = 86.634\%$, which rounds to 86.63%, but Grok 4.6 wrote 86.64% in its additional analysis. For Curtiss-Wright, the task asks for one decimal place, and MiniMax-M3 wrote 108%

instead of 108.0%. For Boeing, DeepSeek V4 Flash stated that every page carries the preliminary-data header, which the cover page does not. These verdicts stand. To gauge how common such cases are, we screened for failed answers that pass every item, fail exactly one whole-answer check, and give a reason involving rounding, decimal places, pages, or formatting; they make up 2.8% of failed answers. The screen only estimates the extent; no verdict was changed.

D CASE STUDIES

Three cases show what the tasks demand (source data, scoring contracts, and model outcomes below). *Travelers*: how much of a 6.7-point combined-ratio improvement is underlying? Removing catastrophe losses and reserve releases leaves 0.6 points, about 9%, and the solver must build this bridge in both ratio points and pre-tax dollars without treating the two retained shares as interchangeable. *Northrop Grumman*: why did operating income fall 23% on 5% sales growth? Four segment changes, an intersegment elimination, a pension adjustment, and an absent divestiture gain must be layered without treating any one layer as exhaustive—and the strongest model overall fails this task while a weaker one passes. *SouthState*: does a 33% rise in the loan-loss allowance mean worse loans? Loans grew 39%, mostly through acquisitions, and coverage fell; the correct judgment is that the increase is non-diagnostic.

For each case, we describe the analyst's question, what the source determines, how the scoring contract encodes it, and how models performed.

D.1 TRAVELERS: SEPARATING UNDERLYING UNDERWRITING IMPROVEMENT

The task asks how much of Travelers' consolidated underwriting improvement in Q2 2026 versus Q2 2025 reflects improved current-accident-year performance rather than lower catastrophe losses and favorable prior-year reserve development (PYD). The task tests whether the solver keeps two measures apart: it must construct two independent bridges—one in combined-ratio points, one in pre-tax dollars—and compute the retained share in each, without treating the two metrics as interchangeable or merging their bases.

The source determines that the reported combined ratio improved 6.7 percentage points, from 90.3% to 83.6%. Removing catastrophe impact (8.5 to 4.9 pp) and favorable PYD (-2.9 to -5.4 pp) yields underlying combined ratios of 84.7% and 84.1%, an improvement of only 0.6 pp—approximately 9% of the reported change. A parallel pre-tax bridge shows reported underwriting income rising \$716M, of which underlying income contributes only \$44M (approximately 6.1%). Both bridges lead to the same qualitative conclusion: the underlying direction improved, but the retained share was small; the reported improvement was predominantly catastrophe-and-PYD-driven.

Table 13: Travelers Q2 improvement decomposition.

Metric	**Catastrophe**	**PYD**	**Underlying**	**Reported**
Combined ratio (pp)	3.6	2.5	0.6	6.7
Pre-tax (\$M)	409	263	44	716
Retained share	—	—	$\sim$9% / $\sim$6%	100%

The scoring contract has nine answer atoms, covering both quarterly ratio bridges, the ratio and pre-tax change decompositions, both retained shares, and an earnings-quality judgment. Its eight fatal conditions guard against scope errors (using a segment-level or year-to-date population), measure conflation (treating the 9% ratio share and 6.1% dollar share as the same number), and unsupported inference (asserting reserve adequacy or future persistence from one quarter's result). Five of the six models pass; GLM-5.2 fails.

D.2 NORTHROP GRUMMAN: RECONCILING SEGMENT DRIVERS TO CONSOLIDATED INCOME

The task asks why Northrop Grumman's Q2 2026 consolidated operating income declined 23% (\$329M) despite 5% sales growth. The task tests whether the solver can reconcile layers of different scope: it must correctly aggregate four segment changes, account for intersegment eliminations, layer in FAS/CAS pension adjustments and an absent prior-year divestiture gain, and reconcile the

resulting $348M subtotal with the actual $329M decline through a favorable corporate residual—without treating any single layer as exhaustive.

Table 14: Northrop Grumman Q2 operating-income bridge ($M, YoY change).

Component	Δ ($M)
Defense Systems	−97
Space Systems	−44
Mission Systems	+60
Aeronautics Systems	+41
Four-segment subtotal	−40
Intersegment eliminations	−21
Total segment OI change	−61
FAS/CAS adjustment	−56
Prior-year divestiture gain (absent)	−231
Named-driver subtotal	−348
Other corporate (favorable)	+19
Consolidated OI change	−329

The scoring contract has 14 answer atoms, covering consolidated operating income and margins in both quarters, the dollar and percentage decline, aggregate segment operating income, the segment bridge with the Defense and Space declines, the lower FAS/CAS adjustment, the absent $231M divestiture gain, and an explanation of the decline despite sales growth. Its seven fatal conditions include basing the conclusion on another issuer or period; omitting or misstating the absent gain, the FAS/CAS change, or the segment totals; and claiming that the named program EAC amounts exhaust the segment changes. Claude Opus 5, Qwen3.8-27B, and GLM-5.2 pass; Grok 4.6, DeepSeek V4 Flash, and MiniMax-M3 fail. Grok 4.6, the strongest model overall, fails where GLM-5.2 passes.

D.3 SouthState: population normalization before judgment

The task asks whether the nominal 33% increase in SouthState's consolidated allowance for credit losses (ACL) from December 2024 to June 2025 is representative of higher portfolio loss content, or whether it is non-diagnostic after controlling for gross-loan growth and loan-population mix. The task tests whether the solver draws only the conclusion the evidence supports: it must build a quantitative bridge before judging, not treat the nominal increase as self-evidently meaningful.

Table 15: SouthState loan populations and ACL coverage.

Population	Dec 2024 ($K)	Jun 2025 ($K)	Growth contrib.
Acquired PCD	862,155	3,409,186	19.06%
Acquired non-PCD	3,635,782	12,492,553	66.27%
Non-acquired	29,404,990	31,365,508	14.67%
Total gross loans	33,902,927	47,267,247	100%
ACL coverage	1.3724%	1.3139%	—

Total gross loans grew 39.42%, driven almost entirely by acquired portfolios (85.33% of growth). The ACL increased 33.48% in nominal terms, but ACL-to-loan coverage *decreased* from 1.3724% to 1.3139% (−5.85 basis points). The scoring contract classifies the correct answer as "non-diagnostic": the nominal ACL increase does not by itself establish higher portfolio loss content because loans grew faster than the allowance and the acquired-loan mix changed materially.

The contract has eight answer atoms, requiring the solver to present population balances at both dates, reconcile them to total gross loans, compute population shares and changes, build a gross-loan growth bridge, and state the coverage movement before rendering a judgment. Its seven fatal conditions guard against omitting required quantitative support, calling the increase "representative," and drawing unsupported inferences (treating arithmetic growth contributions as acquisition causality or asserting reserve adequacy). None of the six models passes. All reach the non-diagnostic conclusion, but each adds an error: three attribute the change in the allowance to the acquisitions

beyond what the sources support, and the others miscalculate the allowance needed to hold coverage flat, misstate the allowance recorded for acquired loans, or miss the required precision.

The five cases below give each objective, the trap that makes it hard, and how the six models fared.

D.4 MARVELL: FOUNDRY COMMITMENTS WITHOUT DOUBLE COUNTING (DISCLOSURE ANALYSIS)

A fabless chip company depends on foundry capacity, and an analyst needs its future purchase obligations by year. The commitment table totals $2,185.5M, but the notes state that at least $482.5M of capacity-reservation purchases are already included in it, while $23.1M of fees and deposits and $204.7M of capital expenditure are disclosed separately without the same mapping. The task is a set-membership problem: which amounts are inside the table, which are outside, and which cannot be placed. None of the six models passes it.

D.5 CROWDSTRIKE AND SNOWFLAKE: CAN TWO RETENTION RATES BE RANKED? (COMPARABLES)

Software analysts routinely place net retention rates side by side. CrowdStrike measures retention on annualized contract value for a fixed cohort of subscription customers; Snowflake measures it on consumption-based product revenue over a two-year customer window that includes reseller end-customers. The correct answer is that the two figures cannot be ranked without a common basis—the task tests judgment about whether a comparison is meaningful, not arithmetic. The error is reading the metric's name rather than its definition. Only DeepSeek V4 Flash passes it.

D.6 EOG RESOURCES: ALIGNING ACTUALS WITH GUIDANCE (EARNINGS ANALYSIS)

An analyst updating a quarterly model must put actual production, realized prices, unit costs, and capital spending on the same basis as forward guidance. Production must be restricted to the geographies the guidance covers; absolute realized prices cannot be subtracted from differentials quoted relative to WTI; and some rows should be computed while others must be left as not comparable. No model passes it.

D.7 INFORMATICA: REBUILDING A PRECEDENT-TRANSACTION SET (PRECEDENTS)

An M&A analyst reconstructs the disclosed precedent set behind a fairness opinion: 28 dated transactions, with the summary rows at the bottom of the table excluded, the minimum, maximum, and median EV/NTM revenue multiples recomputed, and the offer premium tied to the correct reference date. Precedent multiples and share-price premiums are different reference points and must not be merged. Grok 4.6 and DeepSeek V4 Flash pass it.

D.8 ADVANCE AUTO PARTS: EVENT WINDOW AND COMPARATOR CONTAMINATION (MARKET ANALYSIS)

After an earnings release, the stock rises. The analyst must first place the release relative to the trading session (the filing was accepted before the open, so the prior close is the baseline), then compare the move against peers, a sector index, and the market—while noticing that one peer had its own event inside the comparison window. The error is comparing against a contaminated reference set. Only Qwen3.8-27B passes it.

D.9 DISCUSSION

Performance on these cases does not follow the overall ranking of Table 1. Grok 4.6, the strongest model overall, fails the Northrop Grumman task that GLM-5.2 passes, and only Qwen3.8-27B passes the Advance Auto Parts task. The failed answers are classified by check in Appendix C.

E TRAINING DETAILS

Training data. Training data are selected from successful solving trajectories on the training tasks. A trajectory keeps the reasoning, the tool calls, and the tool results; only the assistant's reasoning and actions are supervised.

Fine-tuning. Qwen3.5-9B (Qwen Team, 2026) is fine-tuned with all parameters of its language model at a learning rate of 1×10^{-5}, on full trajectories. Qwen3.5-27B is fine-tuned with LoRA (rank 32, alpha 64) on all linear layers of its language model at a learning rate of 5×10^{-5}; trajectories longer than 48K tokens are truncated at a complete assistant turn. Both use a cosine schedule with a short warmup and BF16 precision, and are trained for four epochs and then continued from the epoch-4 weights to ten epochs.

Evaluation. Models before and after fine-tuning are evaluated in the same solving environment and under the same budget (Appendix B.1), with thinking enabled, greedy decoding, at most 32,768 output tokens per response, and a context of 262,144 tokens.

Leakage filter for external benchmarks. We do not have access to the full evaluation sets of the external benchmarks, so we evaluate on the questions each has released publicly. Public questions and their answers can leak through web search, so search results pass two filters during solving: a blacklist of domains, and a language model that checks every search result and every opened page and blocks copies of the questions, reference answers, and rubrics.

F STAGE REPLACEMENT AND PRODUCTION FINDINGS

F.1 STAGE REPLACEMENT STUDY

Setup. We took the complete production trajectory of one finance task in the Utilities industry group: built on the resource adequacy report of the Southwest Power Pool, it asks how the region's accredited capacity mix changes from 2025 to 2030. The trajectory has 104 steps, 82 carried out by models and 22 by code. One model step was affected by a fault in the execution environment and is left out; each of the other 81 is one replacement condition. A condition starts from a snapshot of the environment just before the step: the files and records the step could see and, for a step of the orchestrator, its conversation so far. Qwen3.8-27B redoes the step with the same instructions and tools, and later steps keep the outputs of the original run, so the conditions do not affect one another. For four orchestrator steps whose conversation exceeded Qwen's context, the history was written to files instead. Each step allowed up to 512 model requests and 24 hours, with no limit on tokens.

Review. Each of Qwen's outputs was checked against the step's instructions, its sources, and what the next step needs. The outputs of the original run had already been used by later steps and were judged by whether a problem surfaced later in the run. An output is usable as is, needs fixes (the main work can be kept), or is unusable (a key decision is wrong or no complete output is produced); a step whose job was to check counts an error it let pass as its own.

Results by stage. Table 16 splits the results by stage. The gap is widest in the stages of first version, difficulty, and review and of repairs and re-review, where most steps decide whether to let a task pass and to which step a defect returns; on problem family, compilation, and novelty the two runs are on par. Table 17 describes what each kind of error usually looks like.

Examples. *Ranking requirement.* The question asked for the changes by category to be ranked, while the scoring contract accepted only a ranking by absolute change in MW; ranking by percentage gives a different order. In the original run, several steps from compilation to formal review missed this, until the orchestrator's final disposition adopted an alternative reading raised by the blind review and repaired the question. At the same step, Qwen received the same counterexample and still judged the question sound because the scoring rule could be satisfied. *Date conflict.* The cover of the report is dated June 15, while the document's metadata shows that it became public on June 16. Qwen explained the conflict as an upload delay and let the task pass without evidence for that explanation. *Unfinished check.* Qwen read about 20,000 of the document's roughly 25,000 characters,

yet reported that the text matched the source exactly and, on that basis, ruled out a revision after the cutoff.

Usage and execution. On the 67 steps with comparable usage records on both sides, Qwen used about 3.8 times the output tokens and 1.8 times the input tokens of the original run. By its own run records, 76 of the 81 conditions ended normally, a status that says little about whether the output could be used. Of the three conditions without a complete output, two were cut off by the length limit of a single response and one failed its structured submission.

Table 16: Stage replacement results by stage: the number of steps whose output was usable as is, needed fixes, or was unusable, for the original run and for Qwen3.8-27B.

		Original run			Qwen3.8-27B		
Stage	Steps	Usable	Fixes	Unusable	Usable	Fixes	Unusable
Sources, candidates, early screening	15	11	3	1	6	6	3
Integration, evidence, rebuild inputs	9	5	4	0	3	3	3
Independent rebuild and comparison	7	5	1	1	4	1	2
Problem family, compilation, novelty	7	5	0	2	5	1	1
First version, difficulty, review	13	9	2	2	3	6	4
Repairs and re-review	10	7	0	3	1	4	5
Wording check and formal review	8	3	3	2	1	4	3
Ranking repair and final review	12	12	0	0	4	8	0
All	81	57	13	11	27	33	21

Table 17: Outputs of the original run and of Qwen3.8-27B at the 81 model steps of one finance task. The original run used GPT-5.6 Sol for most steps and Claude Opus 5 for several reviews. The lower block counts the steps with each kind of error.

	Original run	Qwen3.8-27B	
Usable as is	**57** (70%)	**27** (33%)	
Needs fixes	13 (16%)	33 (41%)	The main work can be kept
Unusable	11 (14%)	21 (26%)	A key decision is wrong, or no complete output is produced
Kind of error			What these errors usually look like
Unstated requirement	14	14	Checks that each scoring item has matching words in the question, not whether a solver reading only the question would know to answer that way; leaves a boundary or definition incomplete
Wrong decision	3	11	Notices a conflict or counterexample but explains it away with an earlier conclusion, a program flag, or a scoring rule the solver cannot see, and lets the task pass; or returns a defect to too early a step
Unverified claim	2	8	Reports an unfinished or unprovable check as confirmed: claims a full match after reading part of a document, takes the date of an event for the date of publication, or states a guessed cause as fact
Incomplete handoff	4	7	Passes on materials that are missing or misdescribed, or reports a step as done when only part of it is
Wrong format	1	6	Breaks the agreed structure of the output, e.g., submits an object as a string
Factual slip	0	5	Computes correctly but attaches a value to the wrong item, or contradicts itself
No output	0	3	The output is cut off for length or the structured submission fails, leaving no result

F.2 Findings from Production

The answer comes before the question. During development we made 20 question-first attempts: a question was written first, and an agent then searched the sources for its answer. Fifteen reached an answer that appeared to pass verification, but five of them first needed the comparison basis, a threshold, or the bounds of the conclusion corrected, four more than under the current engine. In one banking task on a community program, the question passed independent reproduction, yet its hidden scoring rejected an answer based on the California amount, which a primary source allowed by the question explicitly supports: the reproducer had worked only within the evidence the author selected and could not see that the task admitted broader evidence and another valid answer.

Length is not difficulty. Table 18 groups the finance tasks by the number of items in their scoring contracts. Tasks with fewer than five items are easier; beyond that, more items leave the pass rate between 33% and 40%. Two rounds of experiments extended tasks that the constructing model had already solved, adding reasoning layers that each consume the result of the previous one. In the first round, 16 of 29 designs passed a structural review and three yielded complete new relationships, adding two, one, and one layers; a further round took one task from four to six layers. In the second, larger round, 11 of 59 designs passed the structural review, and two tasks passed answer verification, going from five to seven and from five to eight layers. Across 12 runs on these six extended tasks, the constructing model's core conclusion was always correct; for one task, the solving time rose from 188 to 297 seconds. The difficulty screen had predicted success rates of 17% to 78% for them.

Table 18: Pass rate of the six models, pooled, by the number of items in the scoring contract, with each group's share of the finance tasks.

Contract items	Share of tasks	Pass rate
1–4	8.6%	48.8%
5–6	22.2%	40.3%
7–8	18.1%	40.2%
9–11	20.0%	33.4%
12–15	12.7%	39.5%
16–20	10.5%	32.9%
21 or more	7.9%	33.0%

Where models do fail. From tasks the constructing model had answered wrongly, we derived five kinds of judgment that change a conclusion: the scope of a risk stock against a loss flow, a qualitative boundary the issuer leaves undefined, double recognition across ordered accounting adjustments, the version and revision scope of a disclosure, and exact thresholds applied to rounded inputs. Searching for new instances of these five kinds produced, for the first time, errors in the core conclusion on newly mined tasks (four tasks, eight runs).

Other observations. Adding material makes a task harder only when it changes a verifiable relationship in the answer; more documents or background alone do not. The difficulty screen orders candidates well but underestimates how often models succeed. A validity review confirms that a task stands, not that it is hard.

G Extended Related Work

Data Engine with an execution oracle. SWE-Gym (Pan et al., 2025) collects real GitHub issues with executable environments and trains smaller models on teacher trajectories that pass the tests. SWE-smith (Yang et al., 2025) removes the dependence on human-written issues by synthesizing bugs into existing repositories through LM-based modification, procedural edits, and PR mirroring, yielding tens of thousands of instances; R2E-Gym (Jain et al., 2025) likewise builds executable environments procedurally and adds execution-free verifiers for test-time scaling. SWE-Next (Liang et al., 2026) mines merged pull requests at scale and keeps only commit pairs whose tests strictly improve without regressions, so that each task verifies itself. CLI-Universe (Hua et al., 2026) synthesizes terminal tasks from technical materials with fail-to-pass tests and hint-assisted teacher execution. TaskCraft (Shi et al., 2026) composes tool-answerable atomic operations through depth and width expansion and validates by rejection sampling; APIGen-MT (Prabhakar et al., 2025) generates

multi-turn tool-use data through blueprint-then-execution, with blueprints validated by a committee of LM reviewers. In all of these the reference is fixed by executing something; the training recipe they share, filtered-trajectory fine-tuning (Zeng et al., 2024), is the one we adopt, with the scoring contract as the filter.

Data Engine without an execution oracle. When the deliverable cannot be executed, existing methods obtain verifiability by restricting it. FinForge (Matlin et al., 2026) pairs an expert-curated financial corpus with LM-generated multiple-choice questions. AGORA (Guo et al., 2026) builds cross-document questions over workplace document collections in eight domains, with independent solving, leakage checking, and answer verification, but the deliverable is a single value. SPARK (Park et al., 2026) and WebSailor-V2 (Li et al., 2026) derive questions from knowledge graphs—over scientific literature and over web-expanded entities respectively—so that the graph fixes the answer; SPARK checks answers against graph paths and trains proposer and solver by self-play, while WebSailor-V2 scores its answers with a language-model judge. QUEST (Xie et al., 2026) keeps a long-form deliverable and grades it against rubric trees, but the rubrics are produced by the same generation process that writes the task. SWE-QA-Pro (Cai et al., 2026) likewise asks for free-form answers, to repository-level questions seeded from real issues; it drops questions a direct-answer baseline can solve and trains with fine-tuning followed by reinforcement learning from AI feedback. VADE shares the information-asymmetry principle of SPARK and the source grounding of AGORA, but its construction unit is a complete professional analysis, and its reference is reproduced independently from the sources by a second worker rather than checked against a graph or trusted from the generator that wrote it.

Expert-authored professional benchmarks. GDPval (Patwardhan et al., 2026) compares model deliverables with those of occupational experts in blinded pairwise judgments, and APEX (Bennett et al., 2026) grades long-horizon professional tasks against expert-audited rubrics. In finance, Finance Agent Benchmark (Bigeard et al., 2025; Vals AI, 2026) defines nine task categories and an agentic harness over SEC filings; BigFinanceBench (Wang et al., 2026) and FrontierFinance (Zhang et al., 2026b) grade open-ended financial-research tasks with point-weighted and source-attributed rubrics respectively; Hedge-Bench (Cho et al., 2026) derives tasks from the recorded reasoning of analysts. In law, LAB (Grupen et al., 2026) assigns agents a client matter and grades the work product against expert-written criteria. These benchmarks fix the shape of the task and of the rubric that VADE reproduces without expert authoring; we adopt their solving environments and categories (§4, §3.3) and use their public questions as transfer targets (§5).

Learning from constructed tasks. We fine-tune on solver trajectories that pass the scoring contract, the rejection-sampling recipe of STaR (Zelikman et al., 2022) applied to agents by Agent-Tuning (Zeng et al., 2024), SWE-Gym (Pan et al., 2025), and APIGen-MT (Prabhakar et al., 2025), with the contract replacing execution as the filter. The neighbouring construction methods go further and train by reinforcement learning on their own tasks: TaskCraft and QUEST on rubric or answer rewards, WebSailor-V2 in a simulated web environment, and SPARK by proposer–solver self-play. The item-level contract of VADE is a verifiable reward of the same kind, and we leave reinforcement learning on it to future work.