

Finance Agent Benchmark: Benchmarking LLMs on Real-world Financial Research Tasks

Antoine Bigeard
Vals AI, Inc.
antoine@vals.ai

Rayan Krishnan
Vals AI, Inc.
rayan@vals.ai

Shirley Wu
Stanford University
shirwu@stanford.edu

Langston Nashold
Vals AI, Inc.
langston@vals.ai

Abstract

Artificial Intelligence (AI) technology has emerged as a transformative force in financial analysis and the finance industry, though significant questions remain about the full capabilities of Large Language Model (LLM) agents in this domain. We present the Finance Agent Benchmark, featuring challenging and diverse real-world finance research problems which require LLMs to perform complex analysis with the use of recent SEC filings. We construct the benchmark using a taxonomy of nine financial task categories, developed in consultation with experts from banks, hedge funds, and private equity firms. The dataset includes 537 expert-authored questions, covering tasks from information retrieval to complex financial modeling, where each question was validated through a rigorous review process to ensure accuracy and relevance. Moreover, we implement an agentic harness that equips LLMs with tools sufficient to produce an accurate response, including Google Search and EDGAR database access. Overall, Finance Agent Benchmark provides a comprehensive testbed for measuring the progress of LLM-driven finance agents. Our evaluation reveals significant limitations in current AI capabilities—even the best-performing model (OpenAI’s o3) achieved only 46.8% accuracy, at an average cost of \$3.79 per query. It underscores the need for further advancements before reliable deployment in high-stakes finance settings.

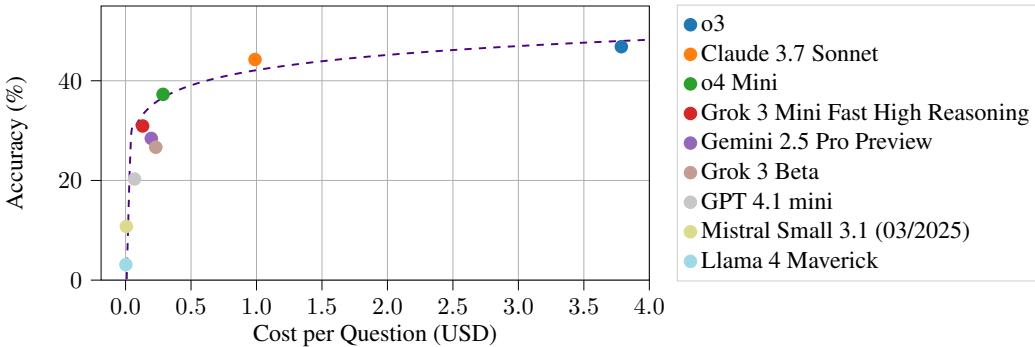


Figure 1: Cost-Accuracy pareto curve results on Finance Agent Benchmark. The Finance Agent Benchmark reveals a clear logarithmic relationship between accuracy and cost, with a sharp diminishing return beyond \$1 USD per question—highlighting that even today’s most sophisticated models struggle to achieve greater than 50% accuracy on real-world financial tasks.

1 Introduction

The finance sector is a critical domain with large economic impact and operational scale. With daily foreign exchange turnover averaging \$7.5 trillion [6], financial operations demand extensive human resources for routine tasks. Entry-level finance professionals sometimes spend up to 40% of their workweek on gathering data rather than analyzing it [21]—time that could be better used for

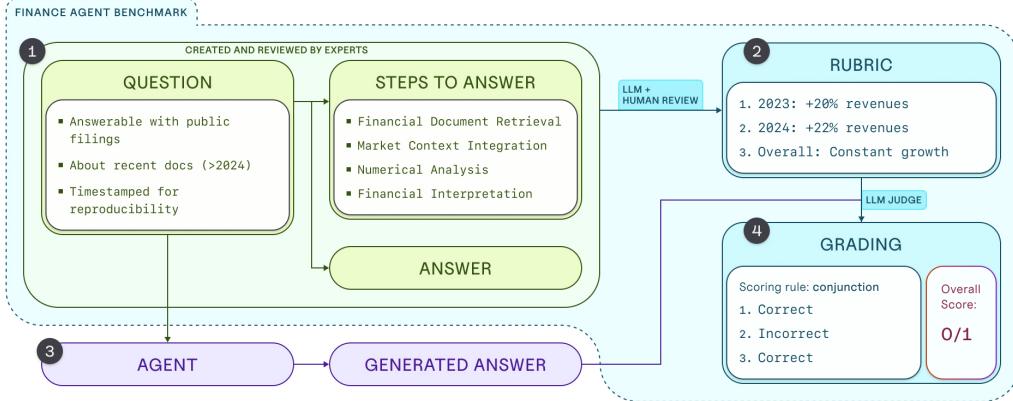


Figure 2: Architecture of the Finance Agent Benchmark. The framework features a structured evaluation process with four key steps: (1) Data Creation: experts identify practical and common financial questions requiring access to public financial documents and provide reference answers. (2) Rubric Development: expert-generated data is used to create robust rubrics with expected calculations and reasoning steps for standardized LLM evaluation. (3) Agent Evaluation: questions are processed through LLMs equipped with necessary tools to generate answers. (4) Answer Grading: an LLM-as-judge scoring system using LLM-as-judge applies conjunction rules to determine correctness across multiple criteria.

strategic analysis that drives business value. The repetitive and essential nature of these tasks requires significant workforce investment and creates inefficiencies.

Recent developments in autonomous AI agents offer a promising solution for financial workflows [33, 9]. These systems have shown advancements in their ability to perform complex tasks that involve unstructured data and multi-step reasoning—capabilities essential for financial analysis [1]. By automating routine yet time-consuming tasks, AI agents can greatly reduce human effort. However, despite this clear potential, there remains a need for robust, domain-specific benchmarks to assess the capabilities of financial agents.

Previous benchmarks often fail to address the interactive environments and nuanced reasoning an agent must undergo when completing industry-specific tasks, leaving performance in real-world scenarios uncertain [4]. This deficiency is particularly concerning given that AI models can generate plausible-sounding misinformation and hallucinations [28], which, if relied upon for high-stake, high-impact decisions, could lead to severe negative outcomes.

For instance, if an AI model misinterprets a company’s earnings report, incorrectly identifying a consistent earnings beat over four consecutive quarters, it could lead to misguided decision to invest in the company. Without proper evaluation, these systems cannot be trusted for high-stakes financial applications. A dedicated benchmark is essential for tracking the state-of-the-art financial agents and highlighting the remaining deficits in model capabilities in this domain.

To fill this gap, we present Finance Agent Benchmark (Figure 2), a novel, standardized framework to rigorously evaluate AI agents on real-world financial analysis tasks. Our benchmark offers:

- **Realistic queries representing real-world challenges:** We collaborated with seven domain experts across the financial services industry (banks, hedge funds, private equity) to identify and categorize common analytical tasks performed with public financial documents, see Table 1.
- **Step-by-step expert annotation for reliable validation:** We constructed this comprehensive dataset with questions, corresponding answers and reasoning trajectories (steps to answer) created by the experts. Each question is verifiable through public U.S. Securities and Exchange Commission (SEC)’s filings from the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database, official repository for public company filings, and underwent peer-review by the experts.
- **Multi-dimensional Evaluation Framework:** Our benchmark uses an LLM-as-judge approach with rubric-based assessment that evaluates answers against specific components of expert solutions rather than holistically. The framework incorporates multiple accuracy verification checks and a

contradiction detection mechanism to identify factual inconsistencies between generated and expert answers.

- **Model-Agnostic Evaluation Harness for Benchmark Baseline:** We developed an agentic evaluation framework (Figure 3) equipped with multiple tools, including Google Search and EDGAR Research Filing Search, enabling standard benchmarking and performance assessment across different LLMs. We established performance baselines using our evaluation harness, providing a foundation for future research in finance-focused AI agents.

Our analysis in Figure 1 indicates that substantial progress is still needed—the best-performing baseline model, o3, achieved only 46.8% accuracy. This highlights the benchmark’s value in testing the ability of models to perform time-intensive tasks expected of entry level finance analysts. Notably, no model surpassed 50% accuracy, underscoring the advancements required before these systems can be deployed autonomously and reliably in the financial sector.

Nevertheless, these models demonstrate notable efficiency advantages: even the most expensive model (o3) averages just 3.1 minutes per task and costs \$3.78, compared to human experts requiring 16.8 minutes and costing \$25.66¹ for equivalent analyses. This efficiency-to-performance ratio points to the promising potential of these models in supporting human analysts, even as work continues toward improving their accuracy for more autonomous applications.

2 Finance Agent Benchmark: High-quality Financial Benchmark

2.1 Dataset Creation Process

Expert Consultation and Taxonomy Development. We collaborated with financial industry experts from banks, hedge funds, and startups to identify common analytical tasks and develop a comprehensive taxonomy of finance analysis questions. Figure 2 shows the overall development process of the benchmark. The experts had at least 2-3 years of working experience at bulge bracket firms such as Goldman Sachs and J.P. Morgan. The resulting taxonomy categorizes tasks based on complexity and frequency in real investment banking contexts, ranging from basic retrieval to complex market analysis. See dataset taxonomy in Table 1.

Question Generation and Quality Control. The experts then crafted 537 questions across nine task categories, with instructions to create queries requiring multiple financial documents. Each entry includes the question, ground-truth answer, necessary source documents, and a step-by-step solution approach. Questions were designed to be answerable without additional information beyond what is available on the open internet or in public filings. The experts focused their question generation on documents published no earlier than 2024 (after most training cutoffs).

Rigorous Validation Process. Each question underwent peer review by a different expert to verify calculations and content validity. Questions containing errors were either corrected or removed. The authors of this paper conducted a final review to standardize formatting and ensure consistency throughout the benchmark. This multi-stage validation process ensured that our dataset represents real-world financial analysis challenges.

2.2 Evaluation Methodology

The LLM-as-judge methodology is becoming more widespread, as the complexity of evaluation scales with the complexity of tasks studied [13]. Our dataset employs a refined LLM-as-judge approach designed to maximize accuracy by systematically reducing both false positives and false negatives. We particularly focus on false negatives, which tend to occur more frequently when using language models as evaluators [19].

Rubric-Based Evaluation. Rather than evaluating answers holistically, we separate assessment into distinct rubrics that align with specific key points from expert answers [2]. For example, if an expert’s answer contains three main points, we check for the presence of each point in the generated answer separately. For straightforward responses (e.g., numerical values like “\$5,234,183”),

¹These calculations only account for time spent directly answering the question, and exclude the time to create the question, review the answer, etc.

Task Name	Question Example	Difficulty	Count
Quantitative Retrieval	<i>What was the quarterly revenue of Salesforce (NYSE:CRM) for the quarter ended December 31, 2024?</i>	Easy	102 (19%)
Qualitative Retrieval	<i>Describe the product offerings and business model of Microsoft (NASDAQ:MSFT)?</i>	Easy	97 (18%)
Numerical Reasoning	<i>What is % of revenue derived from AWS in each year and the 3 year CAGR from 2021-2024 of Amazon?</i>	Easy	83 (15%)
Complex Retrieval	<i>Please briefly summarize the most recent capital raise conducted by Viking Therapeutics (NASDAQ:VTKT).</i>	Medium	29 (6%)
Adjustments	<i>What is Lemonade Insurance’s Adjusted EBITDA for the year ended December 31, 2024?</i>	Medium	43 (8%)
Beat or Miss	<i>How did Lam Research’s revenue compare to management projections (at midpoint) on a quarterly basis in 2024? Format as % BEAT or MISS. Use guidance provided on a quarterly basis.</i>	Medium	69 (13%)
Trends	<i>Which Geographic Region has Airbnb (NASDAQ:ABNB) experienced the most revenue growth from 2022 to 2024?</i>	Hard	33 (6%)
Financial Modeling	<i>How much M&A firepower does Amazon have as of FY2024 end including balance sheet cash, non-restricted cash and other short term investments, and up to 2x GAAP EBITDA leverage? Round to nearest billion.</i>	Hard	47 (9%)
Market Analysis	<i>Compare the quarterly revenue growth of FAANG companies between 2022-2024.</i>	Hard	34 (6%)

Table 1: Task Types and Analyst Difficulty Levels

direct comparison methods are sufficient. This structured approach enhances evaluation reliability by ensuring thorough coverage of all critical aspects while maintaining high standards of factual accuracy and completeness.

Rubric Generation Process. We used GPT-4o to automatically extract initial rubrics from the experts’ questions and answers (see Appendix C.1 for the exact prompts used). Each question, answer, and its associated rubrics were then manually reviewed by the authors of this paper to verify they had accurately captured the key points of the answer written by the expert.

Contradiction Detection. To ensure factual accuracy, we implement a dedicated “contradiction rubric” that examines whether any part of the generated answer conflicts with the expert’s answer. This approach leverages the observation that identifying contradictions between texts is typically more reliable than confirming complete agreement on all points [26]. An example is provided in Appendix A.2.

Dataset Split. We divided the dataset into three parts: a public validation set (50 samples under the CC BY 4.0 License, available in the project’s repository), a private validation set (150 samples available to researchers upon request), and a test set (337 samples kept fully private to prevent future overfitting and contamination issues [30]). All metrics reported in this paper were calculated on the complete 537 samples. Further details about the split are available in Appendix B.

3 Financial Agent Harness

We created an agentic harness² for the LLMs shown in Figure 3 to produce the generations being evaluated. We built this infrastructure based on commonly used frameworks like ReAct [32] [12] [27], and informed by recent agentic evaluation benchmarks such as PaperBench [24] and SWE-Lancer [15]. This harness allows the models to be evaluated in an environment where they had access to a set of tools that is sufficient to produce an accurate response.

²<https://doi.org/10.5281/zenodo.15428823>

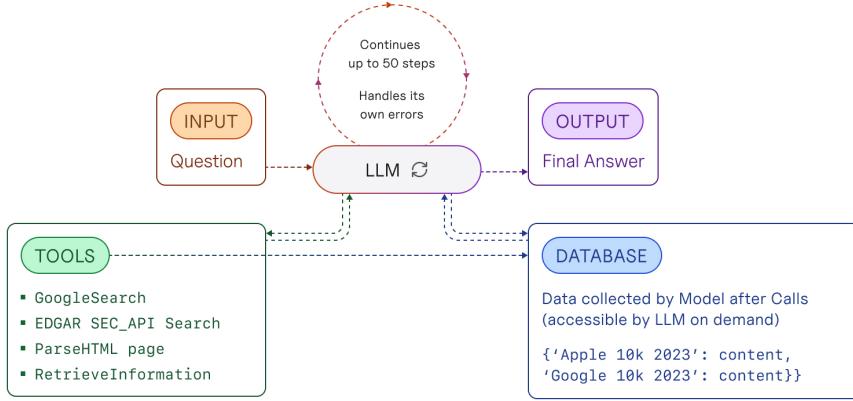


Figure 3: Financial Agent Harness architecture showing the interaction flow between LLM components, specialized tools, and information sources.

3.1 Tools Overview

The harness employs a variety of specialized tools to access and process financial information. Each tool is designed to perform a specific function:

- **GoogleSearch:** A tool to access general web search.
- **EdgarSearch:** A tool to access the EDGAR database, containing public SEC filings³.
- **ParseHTML:** A tool that extracts and saves content from HTML web pages in a key-value store database.
- **RetrieveInformation:** A tool to retrieve stored document content from the key-value database and send the retrieved content to the model.

The ParseHTML and the RetrieveInformation tools collectively allow the model to manage its own context window. The human experts did not make use of any additional tools when writing and answering their questions. More information about each tool is available in Appendix C.2.

3.2 Environment Feedback

Environment feedback is a critical aspect of agent environments [33]. As described in Figure 3, after any tool call, the model receives feedback on whether the call succeeded or failed, and the result of the call (on success).

In our case, most errors occurred when calling the underlying API for each tool. We categorize into two types: retryable and agent errors.

- **Retryable:** These primarily include rate limit errors from APIs. We do not treat them as agent errors; instead, we automatically retry the request using exponential backoff.
- **Agent errors:** These include issues such as exceeding token limits or providing incorrect arguments to tools. We consider these as model-generated errors and return them as tool responses.

Further details on our error-handling approach can be found in Appendix C.3.

These tools work together to enable the agent to research financial information effectively while gracefully handling various error conditions. We have released the full evaluation harness on our GitHub.

³<https://www.sec.gov/search-filings>

Model	Acc. (Class-Balanced) \uparrow	Acc. (Naive) \uparrow	Time per Query \downarrow	Cost per Query \downarrow	Reasoning
o3	46.8 ± 2.2	51.4 ± 2.2	186.5s (3.1m)	\$3.7861	✓
Claude 3.7 Sonnet (Thinking)	45.9 ± 2.2	52.0 ± 2.2	151.3s (2.5m)	\$1.0168	✓
Claude 3.7 Sonnet	44.3 ± 2.2	49.5 ± 2.2	121.0s (2.0m)	\$0.9886	✗
o4 Mini	37.3 ± 2.2	40.6 ± 2.1	164.4s (2.7m)	\$0.2863	✓
Grok 3 Mini Fast High Reasoning	30.9 ± 1.9	36.3 ± 2.1	253.0s (4.2m)	\$0.1305	✓
Gemini 2.5 Pro Preview	28.4 ± 2.0	33.0 ± 2.0	72.5s (1.2m)	\$0.1963	✗
GPT 4.1	26.7 ± 2.0	30.7 ± 2.0	61.3s (1.0m)	\$0.2309	✗
Grok 3 Beta	25.8 ± 1.9	29.4 ± 2.0	60.5s (1.0m)	\$0.4254	✗
o1	21.4 ± 1.7	25.9 ± 1.9	426.5s (7.1m)	\$1.4398	✓
GPT 4.1 mini	20.3 ± 1.7	25.0 ± 1.9	51.6s (0.9m)	\$0.0684	✗
GPT 4o (2024-08-06)	20.0 ± 1.7	24.6 ± 1.9	40.9s (0.7m)	\$0.2575	✓
Grok 3 Mini Fast Low Reasoning	17.6 ± 1.6	21.4 ± 1.8	80.5s (1.3m)	\$0.0667	✓
Gemini 2.0 Flash (001)	14.4 ± 1.4	18.2 ± 1.7	23.6s (0.4m)	\$0.0115	✗
Claude 3.5 Haiku Latest	13.1 ± 1.4	17.1 ± 1.6	46.8s (0.8m)	\$0.0665	✗
o3 Mini	12.8 ± 1.4	16.0 ± 1.6	146.1s (2.4m)	\$0.0472	✓
GPT 4o Mini	10.8 ± 1.2	15.3 ± 1.6	93.4s (1.6m)	\$0.0375	✗
Mistral Small 3.1 (03/2025)	10.8 ± 1.3	13.8 ± 1.5	39.1s (0.7m)	\$0.0061	✗
LLaMA 4 Scout	5.8 ± 1.0	7.4 ± 1.1	13.8s (0.2m)	\$0.0046	✗
Command A	4.6 ± 0.9	6.1 ± 1.0	94.9s (1.6m)	\$0.5628	✗
LLaMA 4 Maverick	3.1 ± 0.8	3.7 ± 0.8	11.8s (0.2m)	\$0.0023	✗
LLaMA 3.3 Instruct Turbo (70B)	2.8 ± 0.8	3.2 ± 0.8	3.3s (0.1m)	\$0.0030	✗
GPT 4.1 nano	2.4 ± 0.7	2.8 ± 0.7	65.5s (1.1m)	\$0.0032	✗
Expert	-	-	1010.5s (16.8m)	\$25.66	-

Table 2: Results Table for all the models. Best values in bold green, worst in bold red.

4 Experiments and Results

4.1 Experiments Setup

Setup and Hyperparameters. All models were evaluated under consistent conditions to ensure a fair comparison:

- **Prompting:** All models were prompted identically using the instruction provided in Appendix C.1. No system prompt was used, as not all providers or models support this feature.
- **Temperature:** The temperature was set to 0 for all models that supported this parameter to ensure better reproducibility of results [18]. For models without configurable temperature, the default temperature (typically 1) was used.
- **Token Limit:** The maximum token limit was set sufficiently high (16,384 tokens by default) to avoid truncation before response completion. In practice, most responses did not approach this limit.
- **Compute Environment:** All experiments were lightweight in terms of memory and compute, as they primarily involved API calls. They were conducted on an AWS t2.2xlarge EC2 instance.

Evaluation Metrics. We report two types of accuracy:

- **Class-Balanced accuracy:** An average accuracy was computed for each category of question. The per-category scores were averaged, with equal weighting given to each.
- **Naive accuracy:** The percentage of questions the model got right, regardless of category.

4.2 Results

We find the class-balanced accuracy to be more representative of the model’s general agentic financial capabilities. This is because some retrieval categories are overrepresented and often require significantly fewer tool calls or agentic behavior. If not otherwise specified, the figures below show results on the Class-Balanced Accuracy.

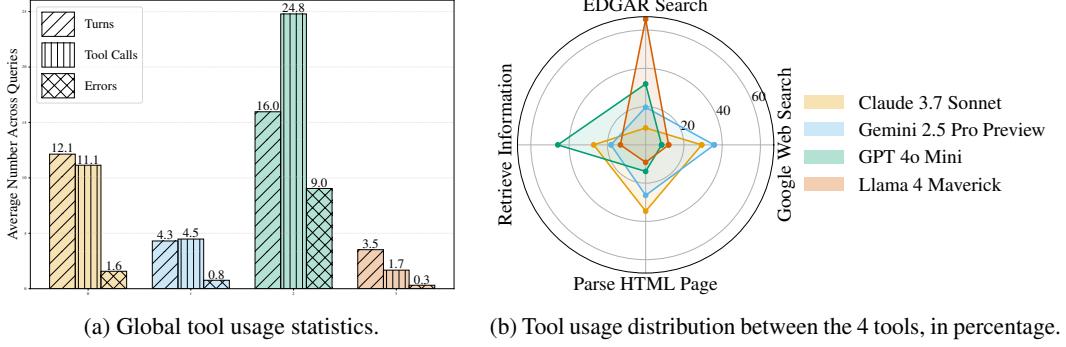


Figure 4: Overall tool use analysis. The best models not only better understand how to use the tools, they also tend to search deeper and be more persistent before settling on an answer. GPT 4o Mini is an outlier in this behavior, with a high number of unsuccessful tool calls.

Figure 1 shows the subset of models that define the accuracy-cost Pareto curve (see the complete results in Figure 7). The models that define the Pareto curve and achieve over 20% accuracy are o3, o4-mini, and GPT 4.1 mini. There is also a very clear logarithmic trend of cost versus accuracy, although some noticeable outliers include OpenAI’s o3-mini and Cohere Command A, which fall significantly below the curve.

4.2.1 Agent Tool Calling Analysis

Figure 4 presents an overview of tool usage behavior across four representative models, chosen due to their markedly different strategies. A more exhaustive version covering all models is provided in the Appendix in Figure 11. Subfigure 4a shows global statistics, including the number of conversational turns and tool calls, as well as error rates. The number of turns taken to reach an answer ranges from 3.5 (LLaMA 4 Maverick) to 12.1 (Claude 3.7 Sonnet), while tool calls vary from 1.7 (LLaMA 4 Maverick) to an unusually high 24.8 (GPT-4o Mini).

Insight 1: More exploratory models tend to perform better. Claude 3.7 Sonnet, which ranks among the top performers, makes significantly more tool calls than either LLaMA 4 or Gemini, suggesting that more extensive exploration contributes to improved results. This behavior pattern also holds for the top performing model, o3 (see Figure 11).

Insight 2: High tool usage without precision leads to failure. GPT-4o Mini stands out as an outlier, issuing a large number of tool calls, often in batches. However, this model suffers from the highest error rate by far, indicating poor tool utilization. It frequently fails by calling the same tool repeatedly (e.g., EDGAR Search) despite persistent errors, without adjusting its strategy.

Insight 3: Balanced tool usage correlates with performance. In Subfigure 4b, which breaks down tool use across tools, higher-performing models such as Claude 3.7 Sonnet and Gemini 2.5 Pro Preview demonstrate balanced use across retrieval, search, and parsing operations. This indicates a nuanced understanding of when and how to apply specific tools.

Insight 4: Tool misuse accounts for poor performance in some models. LLaMA 4 Maverick, despite making fewer tool calls overall, shows disproportionate use of retrieval operations. In practice, this results in the model hallucinating documents and attempting to extract information from them, even when no such documents exist—highlighting a core misunderstanding of the tool interface.

4.2.2 Case Study Examples

To highlight behavioral differences in tool usage, Figure 5 presents agent trajectories for the task: *Due to its business combinations, what is RTX Corp’s (NYSE: RTX) projected future contractual obligation consumption for 2025 - 2029? Provide the amount for each year.* We visualize the interaction flow of Claude 3.7 Sonnet, o1, Gemini 2.5 Pro Preview, and LLaMA 4 Maverick, showing the sequence and timing of tool calls with outcomes.

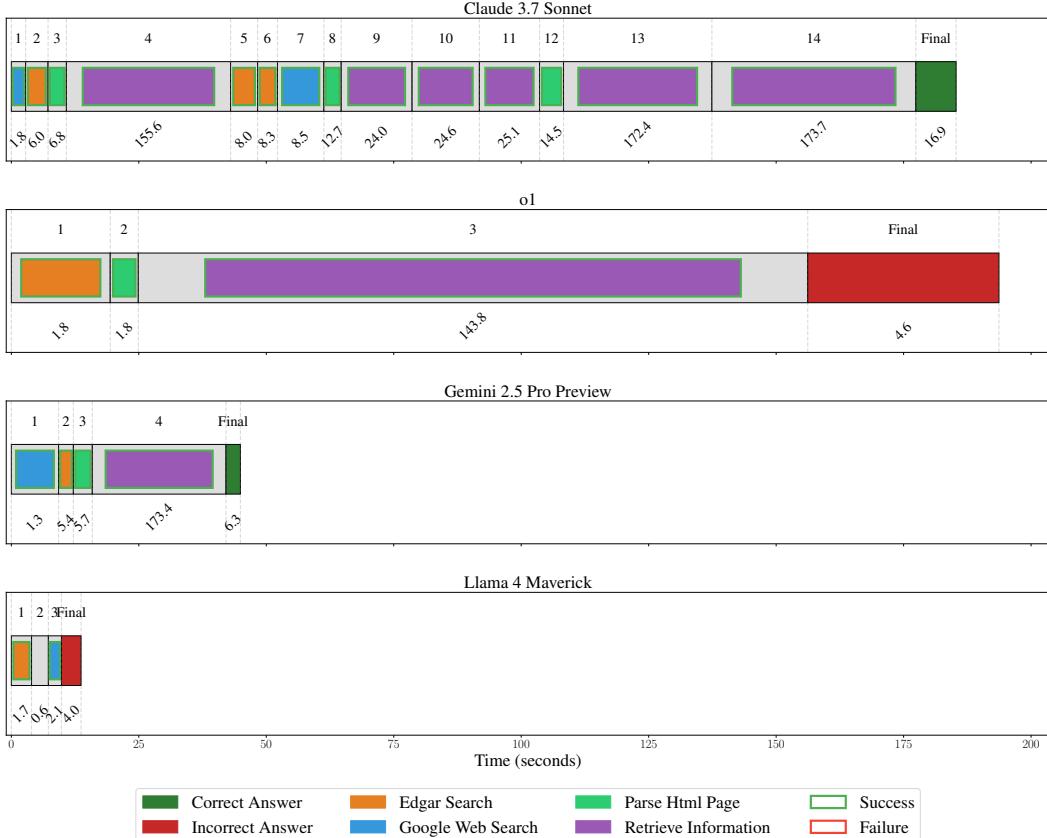


Figure 5: Agent trajectories for various models. On each subplot, the numbers above the row are the turn index, and the numbers below are the number of tokens used for the turn (in thousands).

As noted in Section 4.2.1, Claude 3.7 Sonnet explores tools extensively and efficiently. It follows a highly iterative strategy, repeatedly engaging with multiple tools even after successful calls. This verification approach aligns with its high tool call count and low error rate. Due to its speed, Claude 3.7 makes significantly more tool calls than o1 within the same 200-second timeframe, yet o1 fails to get the correct answer despite using the right tool sequence.

Gemini 2.5 Pro Preview displays the most streamlined trajectory, performing a minimal yet sufficient sequence to produce the correct result. While effective here, this tendency to use fewer tools reduces accuracy across the dataset.

LLAMA 4 Maverick misunderstands the purpose of the tools, calling EDGAR and Google searches but providing an answer without properly parsing and retrieving information from the results.

5 Related Work

General Autonomous-Agent Evaluation. LLMs have evolved from text predictors to autonomous agents [5, 31] capable of performing complex, multi-step tasks across domains. Modern agent frameworks enhance LLMs with access to external tools—such as calculators, search engines, and APIs—and allow them to reason, act, and decide when to stop autonomously [32, 16, 22]. This shift has spurred a wave of general-purpose agent benchmarks.

AgentBench [12] evaluates LLM agents in interactive simulations spanning reasoning, web navigation, and games. GTA (General Tool Agents) [27] introduces real-user tool-augmented tasks covering web search, summarization, and booking. For coding tasks, SWE-Lancer [15] offers 1,488 real-world freelance problems sourced from Upwork. Web-based benchmarks such as GAIA [14] and WebVoyager [8] evaluate agentic browsing and user emulation across diverse sites.

Despite their diversity, these benchmarks rarely include fresh, time-sensitive questions. As a result, it’s often unclear whether a model retrieved information during execution or memorized it during pretraining. Moreover, there is a lack of benchmarks targeting in-depth financial analysis—a critical domain where real-time, tool-augmented reasoning is essential. Our proposed FinanceAgent Benchmark addresses this gap with live EDGAR access, expert-written contemporary tasks, and strict grounding in evidence.

Finance-based NLP & QA Datasets. Since their emergence into mainstream awareness, LLMs have been extensively investigated for their potential applications in financial services and analysis [17], to the point of training models for this sole purpose [29]. Earlier work on finance QA centered on static datasets such as FinQA [3] and TAT-QA [34], which require numerical and hybrid reasoning over financial reports. These benchmarks contributed to specialized modeling approaches but did not support agentic planning or tool use.

Recent efforts have moved toward agent-style frameworks. [35] combines expert and critic LLMs for answer verification over structured filings, improving factual accuracy via multi-agent collaboration. FailSafeQA [20] evaluates robustness in finance QA with realistic, noisy queries based on EDGAR filings, but still assumes a static input and lacks autonomous planning. SWE-Lancer [15] is also relevant as a large-scale task benchmark, though it focuses on software engineering rather than finance. FinAgent [36] and MarS [10] explores the ability of an LLM-based agent to make decisions on the stock market based on multimodal observations: however, both focus on numerical data analysis rather than large documents.

In contrast, our FinanceAgent Benchmark evaluates autonomous agents on real-world financial workflows. It features 537 expert-authored, validated questions across nine categories, emphasizing multi-step reasoning, real-time retrieval from live SEC filings, and evidence-backed answers. Agents are equipped with tools like EDGAR search, calculators, and web access, making this the first benchmark to comprehensively test finance-specific agent capabilities in realistic, high-stakes environments.

6 Conclusion

Finance Agent Benchmark demonstrates significant limitations in current AI financial analysis capabilities. The best-performing model achieved only 46.8% accuracy, highlighting a substantial gap between AI and human expert performance. The benchmark proved challenging across all tested systems, with performance varying dramatically between models (from below 3% to 46.8%).

Despite these limitations, all models completed tasks significantly faster than human experts, indicating potential productivity benefits even with current technological constraints. Performance followed a clear logarithmic cost-accuracy relationship, with specific models (o3, Claude 3.7 Sonnet, o4 Mini) defining the efficiency frontier. The stark performance differences across task types—with models handling simple retrieval better than complex financial modeling or market analysis—suggests that while AI agents show promise for automating routine financial tasks, considerable advancement is needed before these systems can reliably operate autonomously in high-stakes financial environments.

This benchmark provides a critical measure for tracking progress as foundation models continue to evolve toward meeting real-world financial analysis requirements. For future work, we recommend deeper investigation into model performance on structured tabular data or using more complex agents. Refer to detailed discussion in Appendix D.

Acknowledgments and Disclosure of Funding

Thanks to the following people for their support: Alfston Thomas, Andrew Schettino, Kathy Ye, Kyle Jung, Matthew Friday, Michael Xia, and Nicholas Crawley-Brown.

This report was funded by Vals AI, a startup dedicated to evaluating Large Language Models

References

- [1] Siyu An, Qin Li, Junru Lu, Di Yin, and Xing Sun. Finverse: An autonomous agent system for versatile financial analysis, 2024.

- [2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.
- [3] Zichao Chen, Wenhui Chen, Yuwei Fang, Xiaocheng Feng, and Heng Ji. Finqa: A dataset of numerical reasoning over financial data, 2021.
- [4] Zichen Chen, Jiaao Chen, Jianda Chen, and Misha Sra. Position: Standard benchmarks fail – llm agents present overlooked risks for financial applications, 2025.
- [5] Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and Xiuqiang He. Exploring large language model based intelligent agents: Definitions, methods, and prospects, 2024.
- [6] Bank for International Settlements. Triennial survey shows global foreign exchange trading averaged \$7.5 trillion a day in april 2022, 2022.
- [7] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations, 2023.
- [8] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024.
- [9] Sayash Kapoor, Benedikt Stroebel, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter, 2024.
- [10] Junjie Li, Yang Liu, Weiqing Liu, Shikai Fang, Lewen Wang, Chang Xu, and Jiang Bian. Mars: a financial market simulation engine powered by generative foundation model, 2025.
- [11] Tianyang Liu, Fei Wang, and Muhaoo Chen. Rethinking tabular data understanding with large language models, 2023.
- [12] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023.
- [13] Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing*, 2023.
- [14] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023.
- [15] Samuel Miserendino, Michele Wang, Tejal Patwardhan, and Johannes Heidecke. Swe-lancer: Can frontier llms earn \$1 million from real-world freelance software engineering?, 2025.
- [16] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.
- [17] Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. A survey of large language models for financial applications: Progress, prospects and challenges, 2024.
- [18] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. An empirical study of the non-determinism of chatgpt in code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–28, January 2025.
- [19] Aditya Pathak, Rachit Gandhi, Vaibhav Uttam, Devansh, Yashwanth Nakka, Aaryan Raj Jindal, Pratyush Ghosh, Arnav Ramamoorthy, Shreyash Verma, Aditya Mittal, Aashna Ased, Chirag Khatri, Jagat Sesh Challa, and Dhruv Kumar. Rubric is all you need: Enhancing llm-based code evaluation with question-specific rubrics, 2025.

- [20] Rahul Vallath Prabhakar, Jiaqi Zhou, Zichen Chen, and Misha Sra. Failsafeqa: Evaluating the reliability of llms on financial tasks, 2024.
- [21] PWC. Stepping up how finance functions are transforming to drive business results, 2017.
- [22] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023.
- [23] serpapi.com. Serpapi.
- [24] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research, 2025.
- [25] Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. Struc-bench: Are large language models really good at generating complex structured data?, 2024.
- [26] Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges, 2025.
- [27] Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: A benchmark for general tool agents, 2024.
- [28] Eric Williamson. Understanding ai hallucinations: What's the problem and how can we address it?, 2024.
- [29] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabrowski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. wu2023bloombergptlargelanguagemodel, 2023.
- [30] Cheng Xu, Shuhao Guan, Derek Greene, and M-Tahar Kechadi. Benchmark data contamination of large language models: A survey, 2024.
- [31] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024.
- [32] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.
- [33] Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents, 2025.
- [34] Xinyu Zang, Zichao Li, Wenhao Yu, Xiaodong Liu, Ivan Evtimov, Muhamao Chen, Yejin Choi, and Hannaneh Hajishirzi. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of ACL*, 2022.
- [35] Bo Zhang, Wenhao Yu, Weixin Liang, and Wenhui Chen. Enhancing financial question answering with a multi-agent reflection framework, 2024.
- [36] Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, Longtao Zheng, Xinrun Wang, and Bo An. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist, 2024.

Appendices

A Dataset Creation

Task Name	Description	Difficulty Level	Count
Quantitative Retrieval	Direct extraction of numerical information from one or more documents without any post-retrieval calculation or manipulation.	Easy	102 (19%)
Qualitative Retrieval	Direct quotation or summarization of non-numerical information from one or more documents.	Easy	97 (97%)
Numerical Reasoning	Calculations or aggregation of key numbers to produce an answer.	Easy	83 (15%)
Complex Retrieval	Numerical or non-numerical retrieval or content summarization requiring synthesis of information from multiple documents.	Medium	29 (6%)
Adjustments	Quantitative and qualitative analysis of reporting context bridging GAAP and Non-GAAP Financial Metrics.	Medium	43 (8%)
Beat or Miss	Comparison of forward management guidance versus actuals, synthesized by reconciling sequential quarterly reporting documents.	Medium	69 (13%)
Trends	Analyze patterns within a single company's reporting structure or calculate and contextualize evolving performance, key metrics or business composition.	Hard	33 (6%)
Financial Modeling	Complex numerical reasoning calculations which require additional financial expertise to define and evaluate.	Hard	47 (9%)
Market Analysis	Advanced analysis of one or more companies using various documents, requiring normalization of comparison metrics, or complex reasoning and usage of causality to contextualize drivers of business changes or competition dynamics.	Hard	34 (6%)

Table 3: Task Types, Descriptions, Analyst Difficulty Levels and Distribution

A.1 Rubric Generation Prompts

To generate the rubrics for the questions and before the final review, we used GPT-4o with the following prompts that make use of the *Questions*, *Reasoning Steps*, and *Answers*:

System Prompt

You are an expert at converting answers into structured evaluation checks.
Your task is to analyze an answer and break it down into individual checks that can be automatically evaluated.

Each check must use one of these available operators:

- edgar_research_operator: This operator checks that the given criteria is present in the text as a complete, meaningful concept. It verifies factual content such as numerical figures (accepting rounded values), names, dates, and relationships between facts. Each check should represent a complete piece of information rather than fragmenting related facts into separate checks. The format, writing style, and length of the answer do not affect this check.

Guidelines:

1. Break down complex answers into multiple simple checks only when the answer contains distinct, separable components.
2. Create checks that are specific, measurable, and objective.
3. Ensure each criteria is clear, precise, and unambiguous. Do not write full sentences for the criteria if not necessary. It can just be figures or phrases.

User Prompt

Convert this answer into a list of specific evaluation checks.

Break down complex requirements into multiple simple checks where appropriate.
Return the result as a JSON array of objects with 'operator' and 'criteria' fields.

Important: The question and reasoning are provided ONLY for context to help you understand the answer.

Your checks must ONLY evaluate the answer itself - not the question or reasoning.
The checks will be applied exclusively to the answer text.

Create meaningful checks that capture substantive elements of the answer. Each check should:

- Evaluate a significant aspect of the answer
- Be clearly defined and testable
- Make sense as a standalone evaluation criterion

Very important:

- Do not split sentences or phrases into multiple checks if they are related to the same underlying concept.
- Some answers might be a bit verbose and contain more information than originally asked in the question. Do not make more checks than what is asked in the question. For example, if a question asks about a specific number, and the answer contains the number but also the calculation, do not make two checks: one for the number and one for the calculation. Make only one check for the number.
- Particularly make sure that the logical connections are kept within the same check and not split into multiple checks.

Data to evaluate:

Question: {question}

Reasoning: {reasoning}

Answer: {answer}

A.2 Evaluation Example

To illustrate our approach, consider the following question from our dataset:

“How has Carvana’s (NYSE: CVNA) gross profit per unit changed from 2019-2024?”

The expert’s answer contains several key facts:

“CVNA has increased its gross profit per unit from \$2,852 in 2019 to \$7,196 in 2024 representing a 20.3% CAGR. It has driven this metric through improved operating and technology efficiencies as well as introducing new products such as origination fees from financing.”

Instead of evaluating this answer as a whole, we decompose it into five distinct checks:

1. CVNA gross profit per unit in 2019 was \$2,852
2. CVNA gross profit per unit in 2024 was \$7,196
3. CVNA gross profit per unit increased at a 20.3% CAGR from 2019 to 2024
4. Increase driven by improved operating and technology efficiencies
5. Increase driven by new products such as origination fees from financing

Each point is evaluated separately using a dedicated evaluation check. Additionally, we employ a contradiction check, against the entire expert answer to ensure the generated response doesn’t contain any conflicting information.

B Dataset Access

B.1 Accessibility and Reproducibility

The dataset and code are released under permissive open-source licenses: CC BY 4.0 for the dataset and MIT License for the code. Users are free to use and adapt the resources, provided they include appropriate citation. A DOI is provided for the dataset on HuggingFace: <https://doi.org/10.57967/hf/5514>. A different DOI is provided via Zenodo for the harness: <https://doi.org/10.5281/zenodo.15428823>.

The public validation set and agent code are open source and accessible at: <https://github.com/vals-ai/finance-agent>. This repository includes the dataset, simulation environment, evaluation scripts, and detailed documentation to support reproducibility and further research.

The dataset is provided in standard CSV format, accompanied by a structured README and usage examples. The simulation environment is implemented in Python using standard libraries, with clear setup instructions.

We provide documentation based on the Datasheets for Datasets framework, outlining data collection, intended uses, and limitations. All data used is sourced from public, license-compatible domains. The authors bear full responsibility for the dataset and confirm compliance with all applicable rights and licensing.

To ensure long-term accessibility, the dataset is hosted on both GitHub and Zenodo. Structured metadata is included via Zenodo to support discoverability. All experiments in the paper can be reproduced using the provided code and instructions.

B.2 Dataset Split Correlation Plots

The datasets were split to maintain a consistent distribution of question categories across the training, validation, and test sets. In Figure 6, we plot the accuracies of various models on the Validation set (Public + Private) against their corresponding accuracies on the Test set, with each point representing one model. We observe a strong linear relationship, achieving a Pearson correlation coefficient of 0.98 and an R^2 value of 0.97.

The experiments and results described in this document were conducted on all 537 samples, as no data had been made public at the time of experimentation. We maintain a public leaderboard showing model performance on **the test set only** at vals.ai.

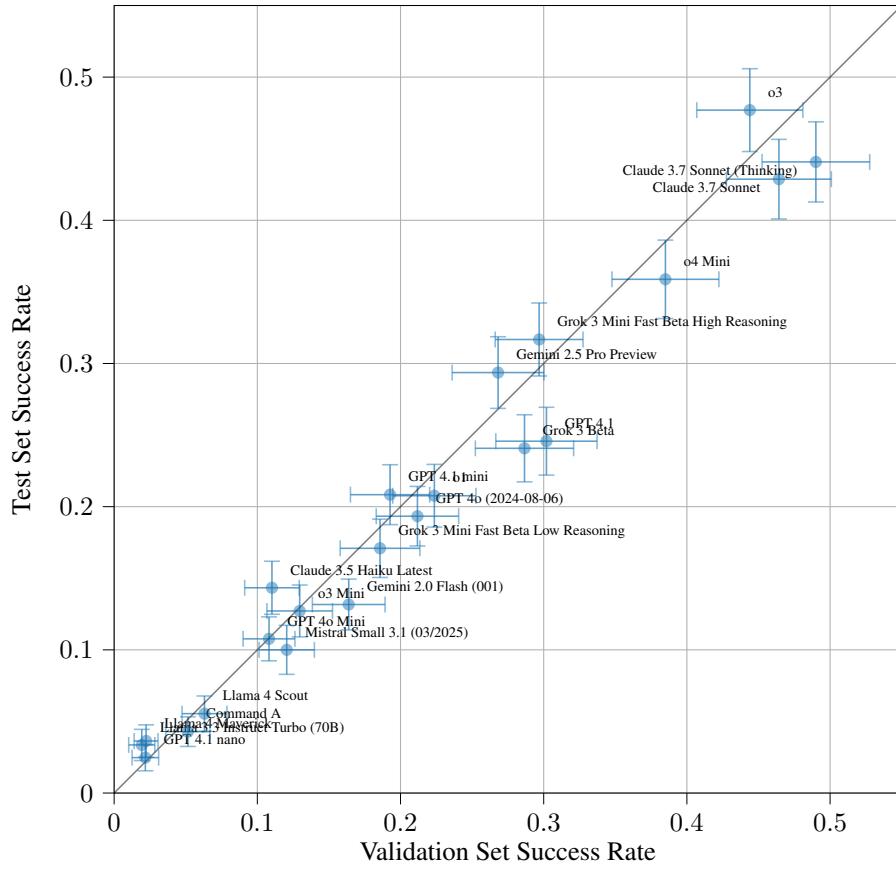


Figure 6: Correlation Plot between Validation and Test sets for Class-Balanced accuracy.

C Harness Details

C.1 Instruction Prompt

The models were all prompted with the following instructions, as well as the list of tools available. We provide the current date to models in the prompt, because otherwise they tend to guess the date, and are not always correct.

Instruction Prompt

You are a financial agent. Today is {current_date}. You are given a question and you need to answer it using the tools provided. You may not interact with the user.

When you have the answer, you should respond with 'FINAL ANSWER:' followed by your answer.

At the end of your answer, you should provide your sources in a dictionary with the following format:

```
{  
    "sources": [  
        {  
            "url": "https://example.com",  
            "name": "Name of the source"  
        },  
        ...  
    ]  
}
```

Question:
{question}

C.2 Tool Use

C.2.1 Google Web Search

A tool that leverages SerpAPI [23] to perform web searches.

Input Arguments:

- `search_query` (string, required): The query to search for information

Implementation Details:

- Returns up to 10 results by default
- Uses SerpAPI to access Google Search results
- Results include title, link, and snippet information

C.2.2 EDGAR Search

A tool for searching the SEC's EDGAR database using the SEC API.

Input Arguments:

- `query` (string, required): Keywords or phrases to search (e.g., "substantial doubt" OR "material weakness")
- `form_types` (array, required): SEC form types to limit search to (e.g., ["8-K", "10-Q"])
- `ciks` (array, required): Company CIK numbers to filter results
- `start_date` (string, required): Start date in yyyy-mm-dd format
- `end_date` (string, required): End date in yyyy-mm-dd format
- `page` (string, required): Pagination parameter
- `top_n_results` (integer, required): Number of top results to return

Implementation Details:

- Returns filing metadata, not the full filing text
- Optimized for financial document discovery

C.2.3 Parse HTML Page

A tool that extracts and saves content from web pages.

Input Arguments:

- `url` (string, required): The URL of the HTML page to parse
- `key` (string, required): The key to use when saving the result in the data structure

Implementation Details:

- Uses BeautifulSoup to parse HTML content
- Extracts clean text by removing scripts, styles, and formatting
- Saves content to a key-value store for later retrieval

C.2.4 Retrieve Information

A tool that retrieves stored document content and processes it through the LLM.

Input Arguments:

- `prompt` (string, required): The prompt containing placeholder(s) in format `{key_name}`
- `input_character_ranges` (object, optional): Dictionary mapping keys to character ranges [start, end]

Implementation Details:

- Replaces `{key_name}` placeholders with actual document content
- Allows extracting specific portions of documents using character ranges
- Makes a call to the same LLM that's leading the agent to process the information

C.3 Error Handling

C.3.1 Rate Limit Errors

- Implements exponential backoff with randomized jitter for 429 errors
- Automatically retries API calls when rate limits are encountered
- Maximum of 8 retry attempts with increasing delays

C.3.2 Token Limit Errors

- **During retrieval:** Errors from documents that are too large are returned to the agent, which can retry by sending partial document chunks
- **During conversation:** If the conversation exceeds the token limit, older messages are removed until the entire exchange fits within the allowed limits. We chose not to implement a more sophisticated long-short term memory mechanism, as the model rarely exhausted its context window during typical conversations. In most instances where this did occur, it was due to the model repeatedly attempting—unsuccessfully—to use the same tool, a behavior we categorize as a model error mode.

C.3.3 Argument Formatting Errors

- Input validation errors are caught and returned to the agent
- Detailed error messages help the agent understand and correct formatting issues
- Handles special cases like JSON string parsing for arrays

D Limitations

D.1 Dataset

Our current dataset emphasizes questions that can be answered with relatively short passages or that focus on the final output of a more complex reasoning process. During development, domain experts highlighted that a significant portion of their workflows involves interacting with structured tabular data, such as spreadsheets or CSV files. While we included a few examples of such queries, future work could more deeply investigate model performance in synthesizing or reasoning over entire tabular documents and financial datasets. Prior research has begun exploring this direction, demonstrating the challenges and opportunities of reasoning over structured financial data [25, 11].

Moreover, although we asked models to provide citations or sources for their answers, we did not evaluate the accuracy or reliability of these sources. This is an important avenue for future work, especially in high-stakes domains. Recent work on source attribution and citation accuracy in language models could guide such evaluation [7].

D.2 Agentic Harness

The agentic harness used in our experiments follows a relatively simple architecture inspired by the ReAct framework, enabling models to interleave reasoning and action [32]. Our setup is comparable in spirit to benchmarks such as PaperBench [24] and SWE-Lancer [15]. While sufficient to evaluate general capabilities, it is not representative of the full complexity of commercial retrieval-augmented generation (RAG) systems or proprietary agentic tools developed by industry leaders such as OpenAI, Google, or xAI.

Future research should explore how these more advanced systems—many of which integrate proprietary retrieval mechanisms, structured data backends, or user feedback loops—perform on similar document understanding tasks. Additionally, our models only had access to publicly available SEC filings and were not integrated with private knowledge bases or advanced retrieval systems. Enabling access to broader and deeper data sources, including internal contracts and proprietary databases, could significantly impact performance in real-world use cases.

E Additional Analysis

E.1 General Analysis

Figure 8 shows the average distribution of tool usage across the four available tools, for each model. In general, tool usage varied significantly by model. For example, mistral barely used the Edgar Search tool, whereas other models like LLaMA Maverick used it significantly. Tool usage also varied significantly even between models from the same generation or provider - GPT 4.1 Nano uses significantly more RetrieveInformation calls than GPT 4.1.

For the rest of the detailed metrics, we present the metrics described above categorized by question type. The results are displayed using two model subsets:

- **Reasoning Models:** These are displayed separately due to their increasing prominence in recent research.
- **Full Range Subset:** This subset includes six models spanning the performance spectrum, from the highest-performing to one of the lowest-performing, with several intermediate performers included.

E.1.1 Time and Cost Analysis

The overall takeaway from Figure 9 and Figure 10 is that all models are significantly more time effective than human experts, ranging from two times faster to more than ten times faster.

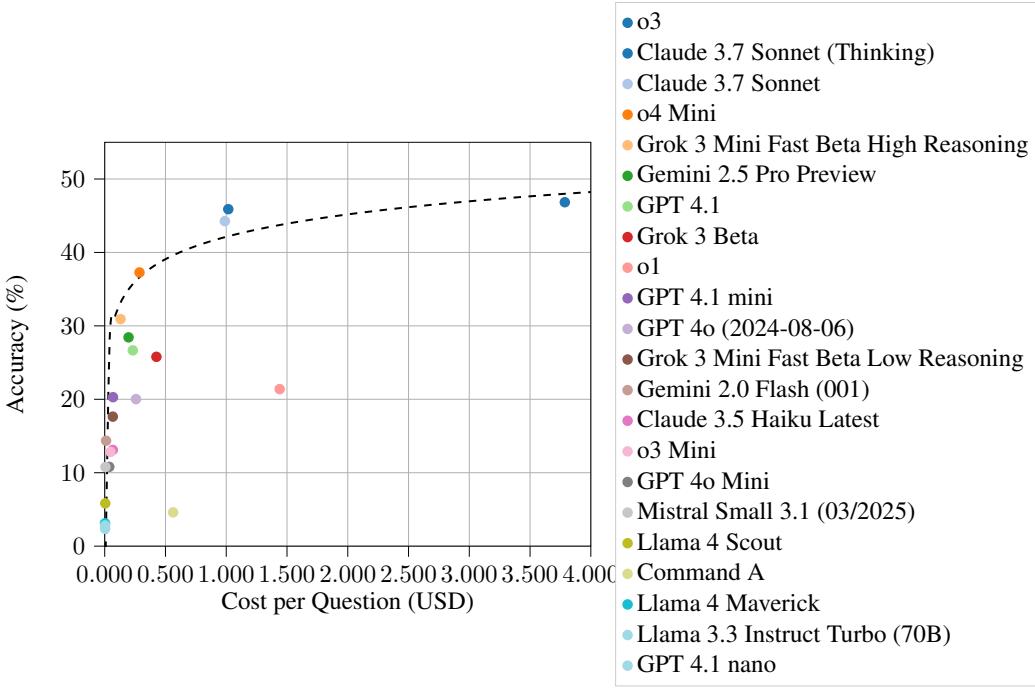


Figure 7: Cost-Accuracy pareto curve results on Finance Agent Benchmark.

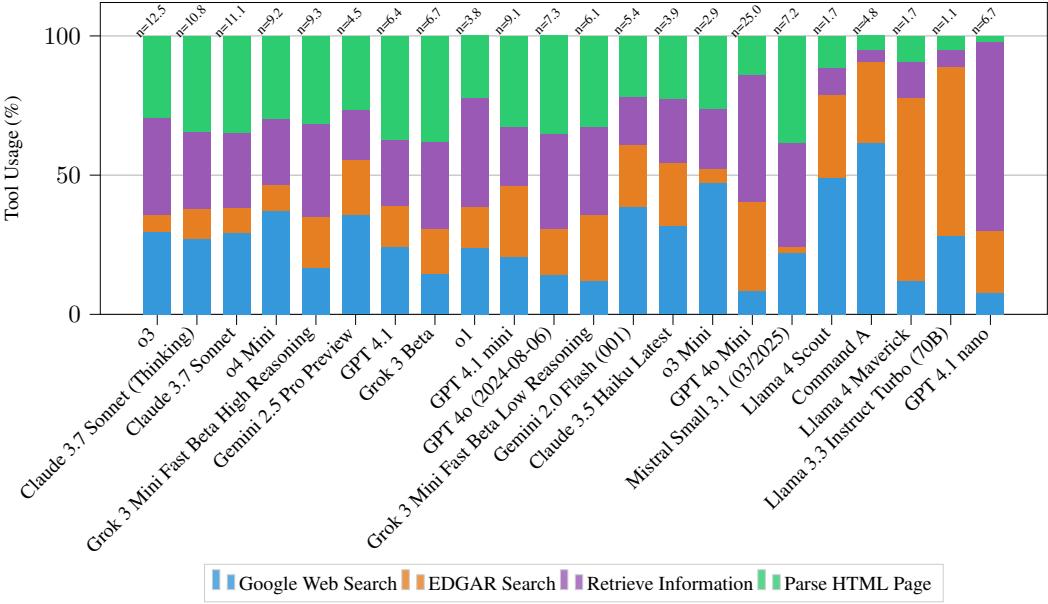


Figure 8: Tool Usage Statistics

E.2 Tool Call Analysis by Question Type: Figure 12, Figure 13 and Figure 11

E.3 Time Analysis by Question Type: Figure 14 and Figure 15

E.4 Cost Analysis by Question Type: Figure ?? and Figure 17

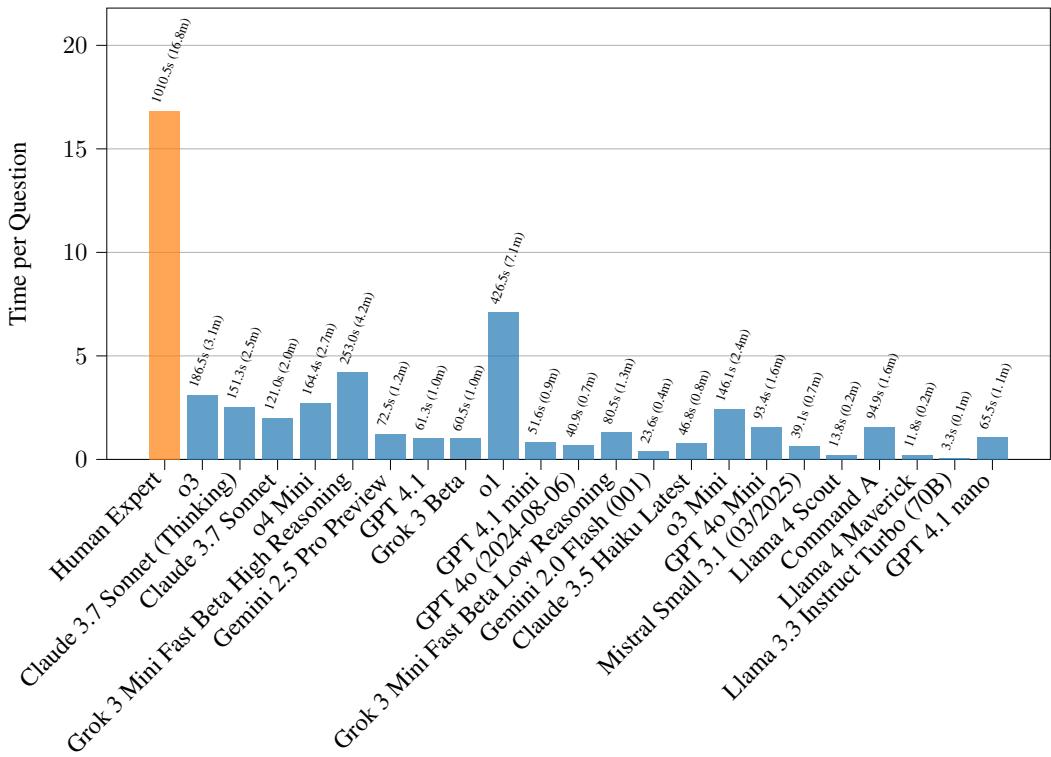


Figure 9: Overall Time Analysis Compared to Expert

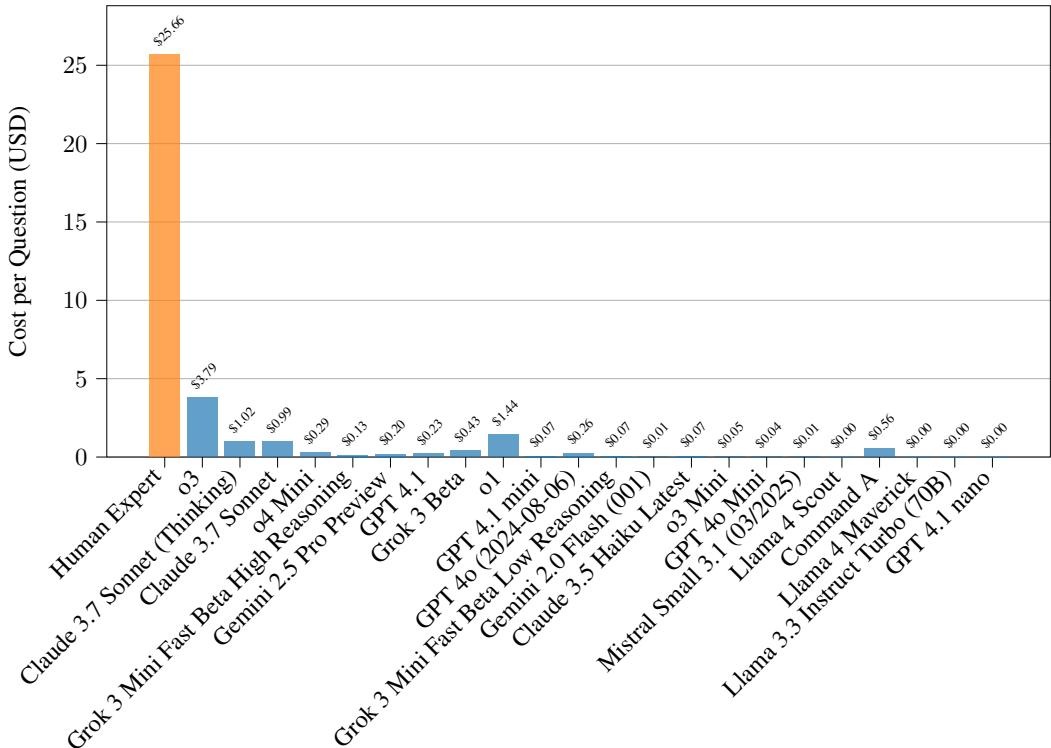


Figure 10: Overall Cost Analysis Compared to Expert (avg \$91.4 /h)

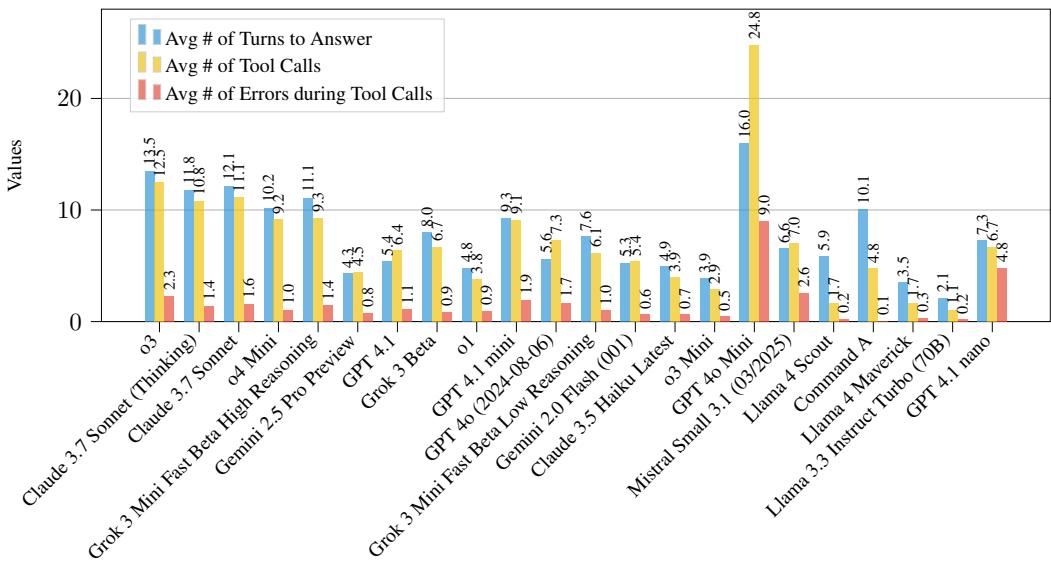


Figure 11: Tool Usage Statistics on All Models

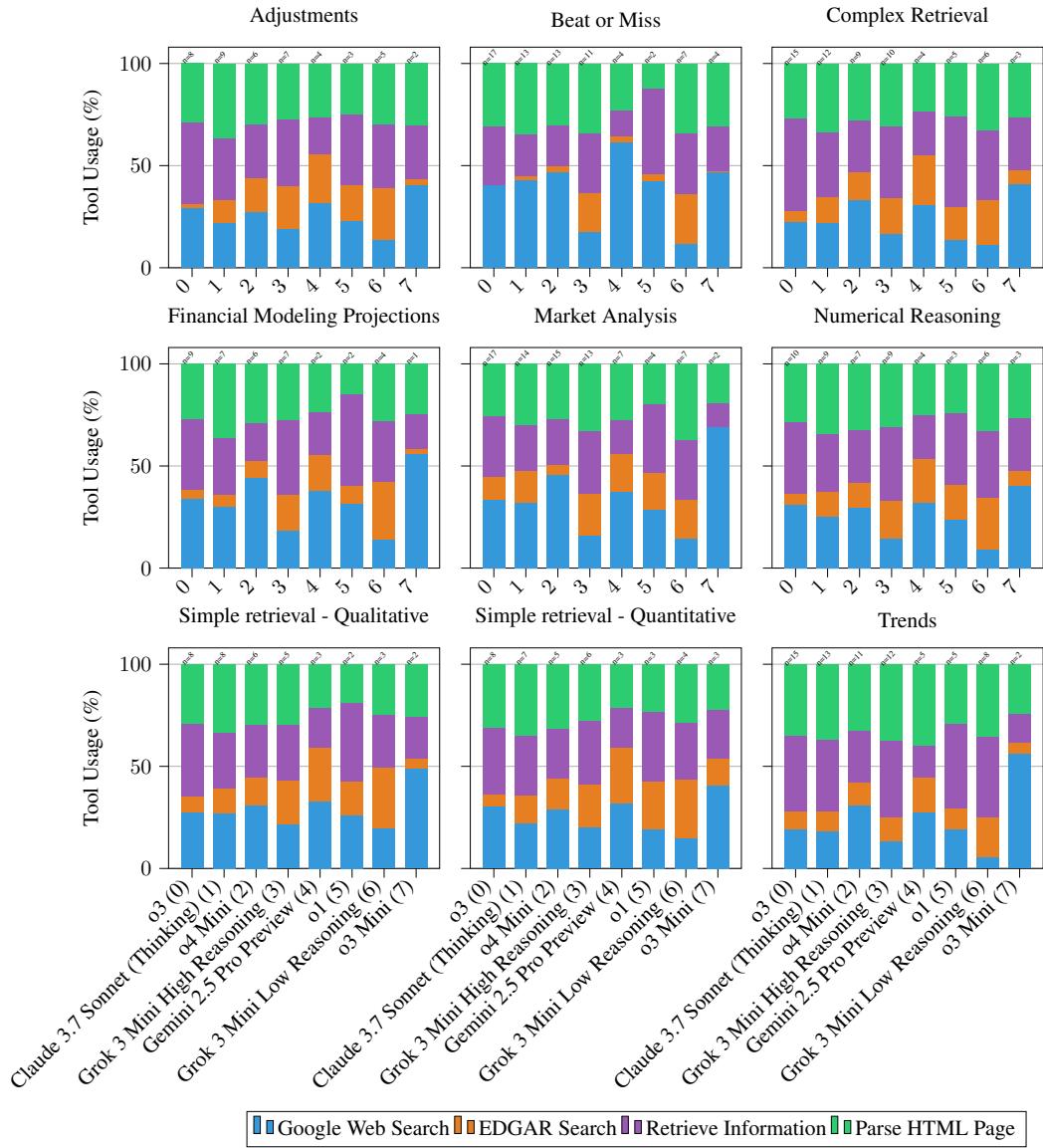


Figure 12: Reasoning Models Tool Usage by Question Type

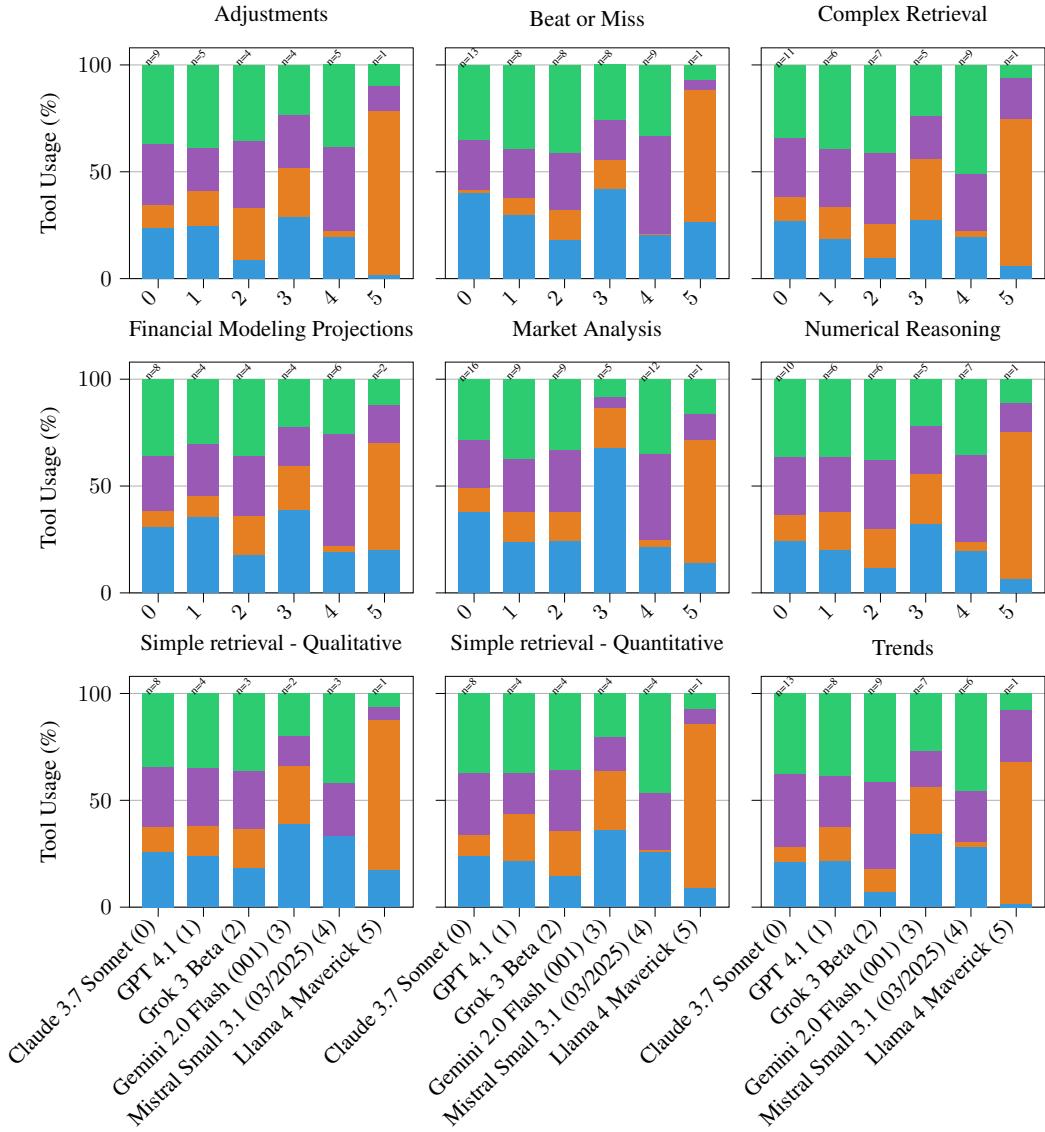


Figure 13: Full Range Subset Models Tool Usage by Question Type

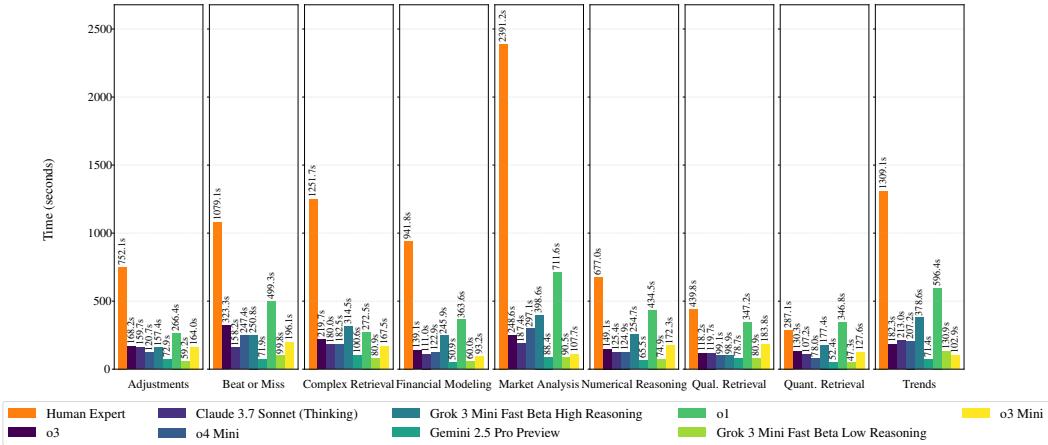


Figure 14: Reasoning Models Time by Question Type

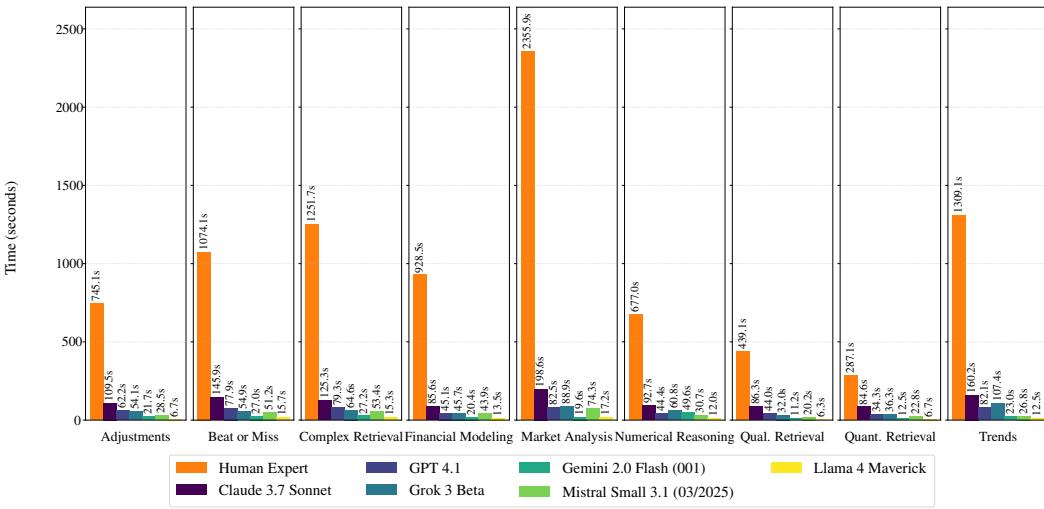


Figure 15: Full Range Subset Models Time by Question Type

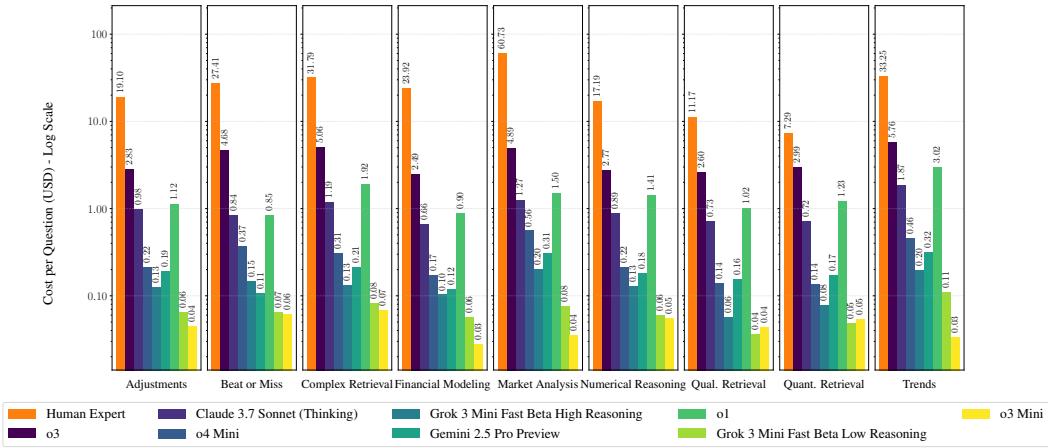


Figure 16: Reasoning Models Cost by Question Type

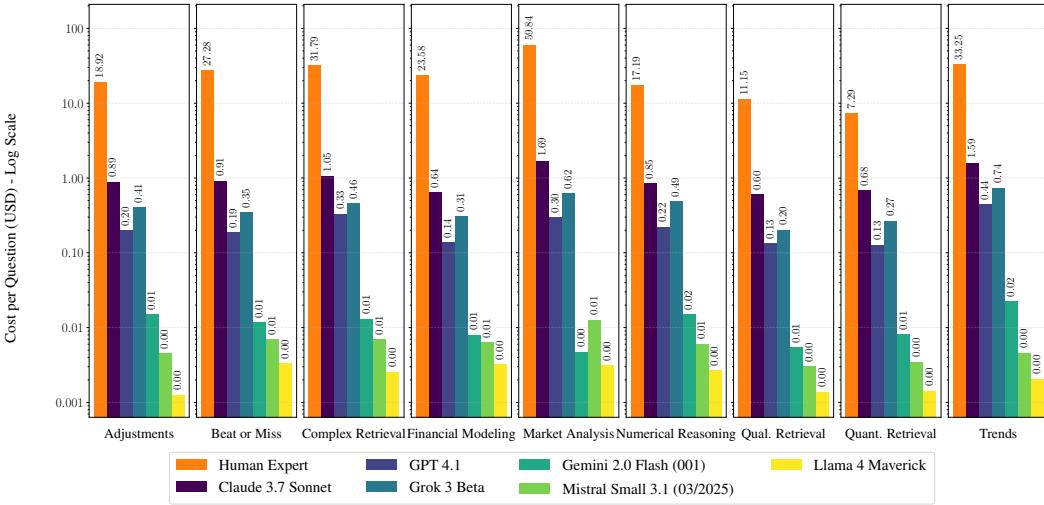


Figure 17: Full Range Subset Models Cost by Question Type