FINCHAIN: A Symbolic Benchmark for Verifiable Chain-of-Thought Financial Reasoning

Anonymous ACL submission

Abstract

Multi-step symbolic reasoning is essential for robust financial analysis; yet, current benchmarks largely overlook this capability. Existing datasets such as FinQA and ConvFinQA emphasize final numerical answers while neglecting the intermediate reasoning required for transparency and verification. To address this gap, we introduce FINCHAIN, the first benchmark specifically designed for verifiable Chain-of-Thought (CoT) evaluation in finance. FINCHAIN spans 58 topics across 12 financial domains, each represented by parameterized symbolic templates with executable Python traces that enable fully machine-verifiable reasoning and scalable, contamination-free data generation. To assess reasoning capacity, we propose CHAINEVAL, a dynamic alignment measure that jointly evaluates both the final-answer correctness and the step-level reasoning consistency. Our evaluation of 26 leading LLMs reveals that even frontier proprietary LLMs exhibit clear limitations in symbolic financial reasoning, while domain-adapted and math-enhanced fine-tuned models can substantially narrow this gap. Overall, FINCHAIN exposes persistent weaknesses in multi-step financial reasoning and provides a foundation for developing trustworthy, interpretable, and verifiable financial AI. The code and data are uploaded to **Software** and **Data** on system for review.

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a wide range of tasks (Zhao et al., 2023a; Xie et al., 2023b), including applications in finance (Chen et al., 2024b). In this domain, effective analysis often requires synthesizing large volumes of textual information from reports, news, and social media, which reflect and influence financial phenomena such as market sentiment (Nie et al., 2024). Most prior work in financial NLP has focused on tasks with shallow supervision, including information extraction (Shah et al., 2023), sentiment analysis (Pei et al., 2022), and text classification (Sy et al., 2023). These tasks typically require models to produce short outputs and do not test whether they can perform transparent, multi-step financial reasoning. In contrast, many financial problems require generating structured chains of reasoning that justify each intermediate step, as illustrated in Figure 1.

FINCHAIN (Compound Interest)

#Question:
`investor_name` invested `principal` in `project_name`. The investment grows at an annual interest rate of `rate`% compounded annually over `time` years. Calculate the `compound interest (CI)`.

#Variables:
- `investor_name` = sample(investors)
- `project_name` = sample(projects)
- `principal` = range(1000, 5000)
- `rate` = uniform(2, 10)
- `time` = range(1, 5)

#Chain-of-Thought Solution:
Step 1: Compute the compound amount: $\texttt{amount} = \texttt{principal} \times \left(1 + \frac{\texttt{rate}}{100}\right)^{\texttt{time}}$
Step 2: Compute the compound interest: $\texttt{CI} = \texttt{amount} - \texttt{P}$

Figure 1: **Symbolic template for generating compound interest problems in FINCHAIN.**

Existing financial reasoning benchmarks such as FinQA (Chen et al., 2021) and ConvFinQA (Chen et al., 2022) primarily frame reasoning as a numerical question answering and emphasize final-answer correctness. While some examples include intermediate traces, these are neither systematically structured nor rigorously verified. As a result, such benchmarks cannot reliably diagnose where reasoning fails or distinguish genuine multi-step inference from surface-level pattern matching.

Inspired by the symbolic-template paradigm introduced in mathematical reasoning (Mirzadeh et al., 2024), we construct a financial reasoning benchmark entirely from scratch. As shown in Figure 1, each symbolic template encodes a parameterized financial problem with named variables and numeric inputs, paired with executable Python code that computes both intermediate steps and final results. This design supports scalable, contamination-free data generation grounded in explicit symbolic and numerical operations. Financial reasoning spans diverse topics and requires heterogeneous expertise. To capture this diversity, we organize our dataset using a fine-grained taxonomy (see Figure 2) covering 12 domains and 58 topics. For each topic, we design five parameterized templates of increasing difficulty, comprising two easy, two intermediate, and one advanced template. Each instantiated example consists of a scenario card specifying the topic, the difficulty, and some example inputs, together with an executable chain of reasoning steps grounded in domain-specific formulas. Because the gold reasoning traces are explicit and executable, the intermediate computations and the final results can be verified at the symbolic and the numerical levels, thus enabling automatic detection of incorrect or inconsistent reasoning steps. To support rigorous and interpretable evaluation, we introduce CHAINEVAL, a dynamic-alignment metric that jointly evaluates final-answer correctness and intermediate step faithfulness. Unlike conventional text similarity measures, CHAINEVAL explicitly accounts for both semantic correspondence and numerical consistency between predicted and reference reasoning chains. Using this benchmark and evaluation framework, we evaluate 26 proprietary and open-weight LLMs. We find that frontier LLMs perform best overall, yet consistently struggle with advanced multi-step symbolic financial reasoning, while fine-tuned compact models achieve only limited gains.

Our main contributions are as follows:

- We introduce the first from-scratch symbolic benchmark for financial reasoning, grounded in a fine-grained taxonomy spanning 12 domains and 58 topics.
- We propose CHAINEVAL, a verifiable reasoning measure that evaluates both step-level consistency and final-answer correctness, and shows the strongest correlation with expert human judgments.
- We benchmark 26 leading proprietary and open-weight LLMs, and find that even state-of-the-art LLMs struggle with verifiable multi-step financial reasoning, particularly on advanced symbolic templates.

2 Related Work

2.1 Financial NLP

Progress in financial NLP has been driven by both modeling and benchmarking. Early work focused on extraction and classification with models such as FinBERT (Araci, 2019), while later efforts expanded to personal finance (Hean et al., 2025), credit scoring (Feng et al., 2023), and risk-awareness benchmarking (Yuan et al., 2024). Datasets like FiNER-ORD, REFinD, FinARG, and ECTSum support tasks in NER, relation extraction, argument mining, and summarization (Shah et al., 2023; Kaur et al., 2023; Mukherjee et al., 2022; Xie et al., 2024). Large financial language models have further advanced the field. BloombergGPT (Wu et al., 2023) achieved broad in-domain performance, FinGPT (Liu et al., 2023) emphasized open-source adaptability, and FinMA (Xie et al., 2023a) delivered competitive results with a compact architecture. Corresponding benchmarks such as FLANG (Shah et al., 2022), FinBen (Xie et al., 2024), and FinMTEB (Tang and Yang, 2025) broadened evaluation coverage across diverse tasks, while BizBench (Koncel-Kedziorski et al., 2023) and PIXIU (Xie et al., 2023a) revealed the limitations of quantitative and multimodal reasoning.

Despite this progress, limitations remain in multi-step reasoning, long-context understanding, and cross-market generalization (Chen et al., 2024b). **These challenges motivate the need for benchmarks that assess the capability of LLMs to perform faithful, auditable reasoning that is grounded in financial knowledge.**

2.2 Financial Reasoning

Real-world financial problems require precise numerical reasoning. FinQA (Chen et al., 2021) and ConvFinQA (Chen et al., 2022) were developed before explicit Chain-of-Thought reasoning became a standard evaluation target. As a result, they supervise arithmetic program generation but offer only weak step-level signals, yielding traces that are

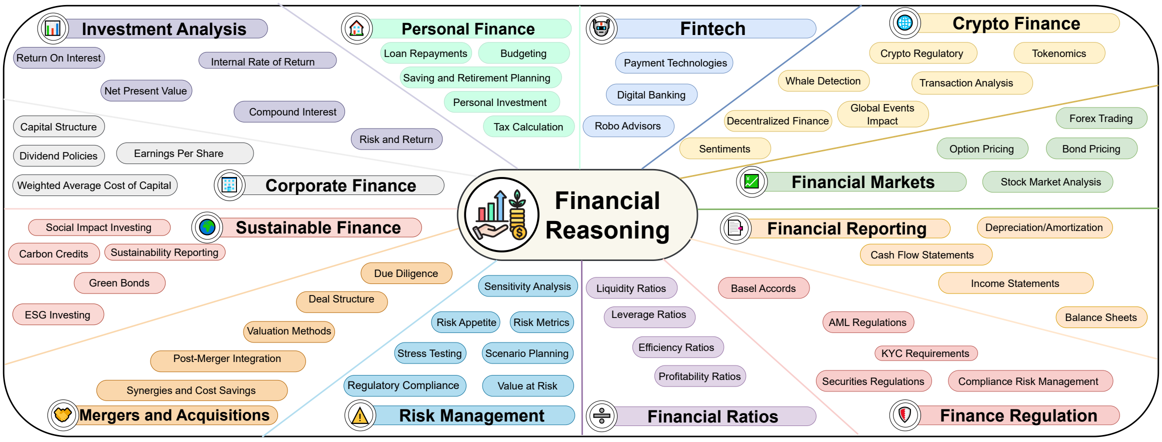


Figure 2: **FinChain taxonomy of financial reasoning topics.** Our benchmark spans 58 topics organized into 12 major domains, ranging from traditional areas like *Corporate Finance* and *Financial Reporting* to emerging fields such as *Crypto Finance* and *Sustainable Finance*. This hierarchical structure enables fine-grained evaluation of symbolic reasoning across diverse financial domains.

neither explicit nor verifiable. FinTextQA (Chen et al., 2024a) introduces long-form financial questions from textbooks and regulatory sources and focuses on explanatory retrieval rather than traceable computation. Bridging text and numerical reasoning, TAT-QA (Zhu et al., 2021) and MultiHiertt (Zhao et al., 2022) combine textual and tabular evidence, while DocMath-Eval (Zhao et al., 2023c) and FinanceMath (Zhao et al., 2023b) move toward interpretable symbolic evaluation. However, these datasets remain largely domain-agnostic and lack explicit, step-level supervision grounded in financial formulas. More recently, FinanceReasoning (Tang et al., 2025) improves answer-level numerical reliability by introducing executable Python solutions, but does not provide systematic verification of step-level reasoning alignment.

FinChain addresses this gap by introducing a symbolic, executable benchmark with explicit intermediate supervision and automatic alignment-based evaluation, spanning 58 topics across 12 financial domains.

3 FinChain

3.1 Data Creation Process

We begin by identifying and defining financial domains based on established literature (Bodie et al., 2025) and expert input within the team, resulting in 12 distinct domains. Within each domain, we propose candidate financial topics with LLM assistance and curate them with financial experts, yielding a total of 58 topics (mean 4.8 per domain). The resulting taxonomy is illustrated in Figure 2.

Statistic	Basic	Intermediate	Advanced
#Templates	116	116	58
Avg. steps	2.01	2.97	3.90

Table 1: Dataset statistics of FinChain.

Following Mirzadeh et al. (2024), we instantiate each topic through parameterized symbolic templates that define both the question structure and an executable Chain-of-Thought solution grounded in domain-specific formulas. We implement these templates as executable Python programs that generate both intermediate reasoning steps and final answers, thereby enabling fully machine-verifiable evaluation. For each topic, we design five templates spanning three difficulty levels (two basic, two intermediate, and one advanced), where we control difficulty by the number and the complexity of required reasoning steps. Table 1 summarizes the dataset statistics and shows increasing reasoning depth across difficulty levels. An example symbolic template is shown in Figure 1. We generate the templates with LLM assistance and subsequently curate them with domain experts to ensure correctness, consistency, and balanced difficulty. We provide detailed prompt designs and generation procedures in Appendix A.1.

3.2 Data Validation and Expert Review

To ensure data quality and consistency, we apply a set of validation constraints covering numerical precision, unit consistency, input completeness, and reasoning step informativeness. Templates that fail validation are revised prior to expert review. A detailed description of the validation criteria is provided in Appendix A.3. We ask financial experts to review all validated templates following a calibrated annotation protocol. Specifically, all reviewers first participated in a pilot calibration phase, during which they jointly reviewed a shared subset of templates and discussed discrepancies to align on annotation standards. After this calibration phase, reviewers independently assessed the remaining templates, evaluating both the correctness of reasoning steps and the final numerical results under the agreed-upon criteria. Further details on annotator backgrounds, annotation procedures, and quality control are provided in Appendix A.4.

4 ChainEval

We propose ChainEval, an evaluation framework that jointly assesses reasoning-step alignment and final-answer correctness. Building on prior work on reasoning consistency (Lyu et al., 2023; Golovneva et al., 2023), our approach explicitly verifies intermediate results via step–answer matching while also checking the final numerical outcome.

4.1 Preliminaries

We define the gold solution S^* and the predicted solution $\hat{S}$ as sequences of m and n reasoning steps, respectively:

$$S^* = (s_1^*, \dots, s_m^*), \quad \hat{S} = (\hat{s}_1, \dots, \hat{s}_n), \tag{1}$$

where s_i^* and $\hat{s}_j$ denote individual reasoning steps in the gold and in the predicted solutions, respectively. Each step s_i produces an intermediate result,

$$\mathrm{StepRes}(s_i) = a_i, \tag{2}$$

representing the numerical or the symbolic value computed at that step.

To evaluate the reasoning faithfulness, we compare these sequences both semantically and numerically. In addition, we apply *Dynamic Time Warping* (DTW) to capture the global structural alignment between step sequences. DTW provides an order-preserving but flexible alignment that accommodates insertions, deletions, or small reordering of steps while maintaining the overall sequence coherence.

4.2 Reasoning Step Alignment

We assess the consistency between gold and predicted reasoning traces through two complementary components: semantic similarity and answer-level agreement, combined within a DTW-based alignment framework.

Semantic Similarity. Each step is encoded using a sentence encoder $\mathrm{Enc}(\cdot)$, and the pairwise semantic similarity between the gold and the predicted steps is computed as

$$\mathrm{SS}(s_i^*, \hat{s}_j) = \cos(\mathrm{Enc}(s_i^*), \mathrm{Enc}(\hat{s}_j)), \tag{3}$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity and $\mathrm{SS} \in [0, 1]$.

Answer Match. For the intermediate results produced by each step, we evaluate numeric or symbolic consistency:

$$\mathrm{StepRes}(s_i^*) = a_i^*, \qquad \mathrm{StepRes}(\hat{s}_j) = \hat{a}_j.$$

We then define the answer-matching function:

$$\mathrm{AM}(s_i^*, \hat{s}_j) = \begin{cases} \mathbb{I}\left(\frac{|\hat{a}_j - a_i^*|}{|a_i^*|} \leq \epsilon\right), \\ \quad \text{if both are numeric,} \\ \\ \mathbb{I}(\hat{a}_j = a_i^*), \ \text{otherwise.} \end{cases} \tag{4}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function, and $\epsilon = 0.05$ permits up to a 5% relative numerical deviation to account for rounding or error propagation. This design choice is motivated by financial auditing standards[1], in which materiality thresholds are commonly defined as between 5% and 10% of a base metric such as earnings, and deviations below 5% are generally considered immaterial.

Gated Step-Level Similarity. To ensure that a pair of steps is considered consistent only when both their semantics and results agree, we define a gated score:

$$\mathrm{Score}_{\text{gate}}(i, j) = \mathrm{SS}(s_i^*, \hat{s}_j) \ \times \ \mathrm{AM}(s_i^*, \hat{s}_j). \tag{5}$$

This score forms the basis of the DTW alignment matrix.

Dynamic Sequence Alignment. To capture global reasoning consistency, we align the two step sequences using *Dynamic Time Warping* (DTW). DTW searches for an optimal monotonic alignment path between $(S^*, \hat{S})$ that minimizes the cumulative cost while respecting the step order. This formulation naturally handles local insertions, deletions, and step compressions, providing a structured measure of how closely the reasoning flows align.

---

[1] `https://www.materialitytracker.net/standards/financial-thresholds/`

DTWGate Alignment Score. We transform the minimal DTW cost into a normalized similarity measure as follows:

$$\mathrm{DTWGate}(S^*, \hat{S}) = 1 - \frac{\mathrm{Cost}_{\mathrm{DTW}}}{L_{\mathrm{path}}}, \quad (6)$$

where $\mathrm{Cost}_{\mathrm{DTW}}$ denotes the total alignment cost and L_{path} represents the length of the optimal alignment path. The resulting score lies in the range $[0, 1]$, with higher values indicating stronger reasoning alignment between the gold and the predicted solutions.

4.3 Final Answer Correctness

Beyond step-level reasoning alignment, we also assess the correctness of the final predicted outcome. Let s^*_m and $\hat{s}_n$ denote the last steps of the gold and the predicted solutions, respectively. We define the **Final Answer Correctness (FAC)** metric as

$$\mathrm{FAC}(S^*, \hat{S}) = \begin{cases} \mathbb{I}\left(\frac{|\hat{a}_n - a^*_m|}{|a^*_m|} \le \epsilon\right), \\ \text{if both are numeric,} \\ \mathbb{I}(\hat{a}_n = a^*_m), \text{otherwise,} \end{cases} \quad (7)$$

using the same tolerance $\epsilon = 0.05$ as before. FAC measures whether the model's final computation aligns with the correct end result, complementing the DTW-based metric that evaluates reasoning faithfulness throughout the entire solution sequence. Therefore, we have

$$\textsc{ChainEval} = (1-\alpha)\cdot\mathrm{DTWGate} + \alpha\cdot\mathrm{FAC}, \quad (8)$$

which accounts for both reasoning correctness and final answer's correctness. We set $\alpha = 0.1$, selected via grid search on a subset to maximize the correlation with human evaluations, as explained in Appendix D. We further verify that the final measure best reflects true reasoning quality by comparing it with human evaluations, as described in § 5.2.

5 Experiments and Results

5.1 Evaluated Models

We evaluate a total of 26 LLMs, grouped into four categories according to their capability and relevance to financial reasoning. (1) **Frontier proprietary models**, including `GPT-{5, 4.1, 5-mini, 4.1-mini}` (OpenAI, 2025a,b), `Claude Sonnet {4.5, 4, 3.7}` (Anthropic, 2025b,c,a), `Gemini 2.5 {Pro, Flash}` (Comanici et al., 2025), `DeepSeek {V3.2, V3.1, R1}` (Liu et al., 2024; Guo et al., 2025), and `Grok 4 {Heavy, Fast}` (xAI, 2025). (2) **Finance-specific models**, including `Fino1` (Qian et al., 2025), `FinR1` (Liu et al., 2025), `DianJin-R1` (Zhu et al., 2025), and `WiroAI Finance-{LLaMA, Qwen}` (Abdullah Bezir, 2025). (3) **Math-enhanced models**, including `WizardMath` (Luo et al., 2023), `MetaMath` (Yu et al., 2023), `Mathstral` (Mistral, 2024), and `Qwen2.5-Math` (Yang et al., 2024). (4) **General-purpose open-weight models**, including `LLaMA 3.1` (Grattafiori et al., 2024) and `Qwen {2.5, 3}` (Qwen, 2024, 2025). Detailed configurations and model sources are described in Appendix C.

5.2 ChainEval Validation

Before conducting large-scale experiments, we validated the proposed ChainEval through a controlled expert evaluation. We randomly sampled 20 instantiated questions from the dataset and generated answers using five models of different capacities and training paradigms, namely `GPT-5`, `GPT-4.1 mini`, `MetaMath`, `Fino1`, and `LLaMA 3.1`, to ensure a clear range of reasoning quality. This process produced 100 model-generated responses, which were then randomized and anonymized for human assessment. Financial experts independently evaluated each response with respect to the corresponding question and gold-standard reasoning trace. They rated each output along two dimensions, *Reasoning Process Quality* and *Final Answer Accuracy*, on a five-point scale as described in Appendix B.4. We observed a strong association between these dimensions, with Spearman's ρ exceeding 0.94, showing that coherent reasoning typically leads to accurate final answers. Consequently, subsequent analysis below will focus on *Reasoning Process Quality* as the primary evaluation dimension. We further computed several variants of ChainEval and compared their correlations with expert process scores against reference-based evaluation measures such as ROUGE-2 and ROUGE-L (Lin, 2004). Among all variants, the DTWNormGate formulation with inclusion of final answer's correctness showed the highest correlation with human judgments, capturing both semantic and numeric consistency across reasoning steps. Therefore, we adopted it as the primary evaluation measure in this study, with detailed ablation results reported in Appendix D.

5.3 Experimental Setup

We instantiate the benchmark by sampling 10 instances per symbolic template with distinct random seeds, yielding 58 topics × 5 templates × 10 instances = 2,900 test cases. We evaluated all models under a unified decoding configuration: temperature = 0.7, top-p = 0.95, and a maximum token limit of 4,096 unless these parameters are unavailable for a given model. We use a zero-shot setup with a standardized reasoning prompt:

```
Please answer the given question and provide a
step-by-step solution.
Use the format: Step 1: ..., Step 2: ..., ...
The question is: {q}
```

We use ChainEval as the primary evaluation measure, as it jointly measures final-answer correctness and alignment of the intermediate reasoning steps. We post-processed model outputs with regular expressions to extract the ordered list of reasoning steps, accommodating common variations such as "Step x:", "Step x", or "stepx". For comparison, we also report results using frequently used reference-based measures (Xie et al., 2023c) to assess surface-level quality, including ROUGE-2 and ROUGE-L and BERTScore (Zhang et al., 2020).

5.4 Results

5.4.1 Overall Model Performance

As shown in Table 2, frontier proprietary models achieve the strongest overall performance under ChainEval, indicating higher alignment quality in step-level symbolic reasoning. The performance differences between frontier models are not strictly monotonic. For example, GPT-5-mini slightly outperforms GPT-5, despite comparable or lower scores on surface-level metrics. Among open-source systems, Fin-R1 and Mathstral achieve ChainEval scores of 58.14 and 59.87 respectively, approaching frontier-level alignment despite their substantially smaller scale. These results suggest that targeted fine-tuning and symbolic supervision can substantially improve multi-step reasoning fidelity beyond what model size alone provides. However, the effectiveness of fine-tuning varies with adaptation scope: models trained on broad mathematical corpora (e.g., Mathstral) exhibit stronger generalization across reasoning styles, whereas models tuned narrowly for finance (e.g., Finance-LLaMA and Finance-Qwen) display more variable alignment quality. We also evaluate Grok 4 Heavy, which was among the strongest available systems at the time of evaluation. Due to its high inference cost (approximately $0.5 per sample), we only assessed this system on a randomly sampled subset rather than the full benchmark, with results reported in Appendix E.1. Overall, these results show that while frontier models lead in verifiable symbolic reasoning, model scale alone is insufficient, and structured supervision plays a critical role in achieving high step-level alignment.

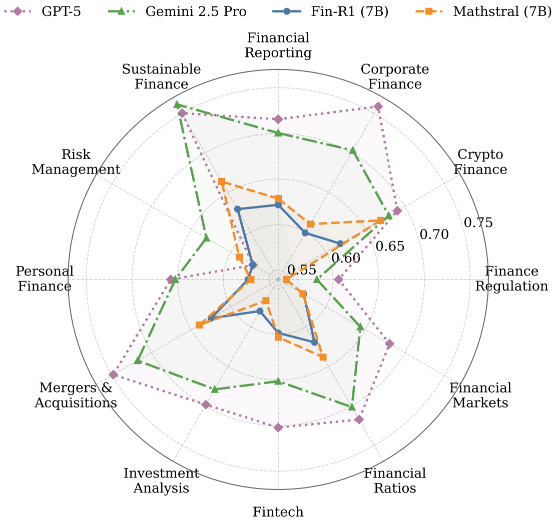


Figure 3: **Domain-level performance across financial domains.** Radar plot showing ChainEval scores across twelve financial domains for four representative models: GPT-5, Gemini 2.5 Pro, Fin-R1, and Mathstral.

5.4.2 Performance Across Domains

Figure 3 shows the domain-level ChainEval performance across twelve financial categories for four representative models: GPT-5, Gemini 2.5 Pro, Fin-R1, and Mathstral. Among them, GPT-5 demonstrates consistently strong performance across most domains, forming a relatively stable upper envelope with limited variation. Gemini 2.5 Pro follows a similar overall trend, achieving competitive scores across domains while exhibiting moderate domain-level differences. The two open-weight models show more heterogeneous domain-specific patterns. Fin-R1 achieves relatively higher scores in domains such as Financial Reporting, Sustainable Finance, and Risk Management, while exhibiting lower performance in several quantitatively intensive or structurally complex domains. In contrast, Mathstral performs competitively in quantitatively orientated domains, including Financial Ratios and Investment Analysis, but shows weaker performance in domains with heavier textual or

Model	Size	CHAINEVAL ↑	FAC ↑	ROUGE R_2 ↑	ROUGE R_L ↑	BERTScore ↑
Frontier Proprietary LLMs						
GPT-5	N/A	$66.57^{10.64}$	$82.03^{32.40}$	$\mathbf{28.84}^{12.30}$	$\mathbf{42.77}^{12.91}$	$\mathbf{88.77}^{2.39}$
GPT-4.1	N/A	$65.34^{9.36}$	$\mathbf{84.66}^{29.26}$	$19.38^{8.91}$	$30.12^{10.57}$	$86.04^{2.09}$
GPT-5-mini	N/A	$\mathbf{67.17}^{11.06}$	$80.28^{32.34}$	$26.48^{11.99}$	$39.74^{12.83}$	$88.18^{2.35}$
GPT-4.1-mini	N/A	$65.06^{8.88}$	$84.59^{30.23}$	$18.67^{8.86}$	$29.05^{10.51}$	$86.05^{2.09}$
Claude Sonnet 4.5	N/A	$66.33^{9.44}$	$83.34^{31.79}$	$19.69^{7.77}$	$29.37^{8.81}$	$86.07^{2.00}$
Claude Sonnet 4	N/A	$66.20^{9.66}$	$82.62^{31.96}$	$19.87^{7.82}$	$29.50^{8.59}$	$86.38^{1.83}$
Claude Sonnet 3.7	N/A	$65.51^{9.30}$	$83.14^{31.00}$	$19.49^{7.77}$	$29.36^{8.56}$	$86.38^{1.82}$
Gemini 2.5 Pro	N/A	$66.04^{9.71}$	$84.34^{30.99}$	$17.61^{7.07}$	$27.36^{8.28}$	$85.94^{1.88}$
Gemini 2.5 Flash	N/A	$65.96^{10.10}$	$83.90^{30.61}$	$18.98^{8.05}$	$29.22^{9.32}$	$86.34^{2.02}$
DeepSeek v3.2	N/A	$65.23^{9.63}$	$84.17^{31.23}$	$21.73^{10.32}$	$32.85^{11.31}$	$86.66^{2.15}$
DeepSeek v3.1	N/A	$65.29^{9.62}$	$84.34^{31.02}$	$21.72^{10.29}$	$32.87^{11.24}$	$86.68^{2.14}$
DeepSeek R1	N/A	$51.22^{11.04}$	$28.97^{30.29}$	$8.67^{7.21}$	$12.93^{9.72}$	$84.39^{1.79}$
Grok 4 Fast	N/A	$60.69^{16.26}$	$66.54^{42.54}$	$21.33^{11.09}$	$32.25^{13.30}$	$86.83^{3.06}$
Finance-Specific LLMs						
Fin-o1	8B	$41.50^{12.49}$	$\mathbf{52.79}^{27.89}$	$3.47^{1.55}$	$6.35^{2.32}$	$83.55^{1.50}$
Fin-R1	7B	$\mathbf{58.14}^{7.32}$	$52.76^{28.04}$	$5.70^{2.44}$	$9.22^{3.33}$	$\mathbf{84.30}^{1.34}$
DianJin-R1	7B	$51.95^{8.98}$	$37.69^{23.73}$	$6.28^{2.95}$	$10.79^{4.19}$	$83.12^{1.32}$
Finance-LLaMA	8B	$41.35^{10.49}$	$25.21^{25.64}$	$9.39^{4.69}$	$16.19^{5.84}$	$83.48^{2.09}$
Finance-Qwen	7B	$34.57^{11.01}$	$31.62^{25.62}$	$\mathbf{9.50}^{4.26}$	$\mathbf{16.46}^{5.44}$	$83.35^{1.70}$
Math-Enhanced LLMs						
WizardMath	7B	$24.33^{15.00}$	$41.28^{35.69}$	$11.66^{6.57}$	$20.72^{7.83}$	$84.78^{2.36}$
MetaMath	7B	$7.93^{9.43}$	$23.97^{28.76}$	$11.45^{7.36}$	$21.08^{9.24}$	$84.86^{2.99}$
Mathstral	7B	$\mathbf{59.87}^{10.02}$	$54.03^{36.61}$	$\mathbf{16.79}^{7.82}$	$\mathbf{26.97}^{9.34}$	$\mathbf{86.13}^{2.18}$
Qwen2.5-Math	7B	$55.35^{14.98}$	$\mathbf{62.62}^{34.56}$	$11.74^{5.87}$	$20.56^{7.61}$	$83.45^{1.85}$
General-Purpose Open LLMs						
LLaMA 3.1 Instruct	8B	$53.99^{6.02}$	$32.72^{27.46}$	$4.61^{2.28}$	$8.09^{3.02}$	$83.35^{1.36}$
Qwen 2.5 Instruct	7B	$\mathbf{60.35}^{7.47}$	$\mathbf{65.41}^{32.53}$	$\mathbf{9.20}^{4.51}$	$\mathbf{15.26}^{5.85}$	$\mathbf{84.22}^{1.78}$
Qwen 3	8B	$43.32^{11.81}$	$32.28^{28.58}$	$4.05^{1.69}$	$6.61^{2.14}$	$83.58^{1.24}$

Table 2: **Zero-shot performance across financial, mathematical, and general reasoning benchmarks.** Scores are reported as percentages, with standard deviation in superscript. Model size (*N/A*) denotes proprietary or undisclosed configurations. Within each model group, the best-performing system for each metric is highlighted in bold.

regulatory components. In general, domain-level performance is not uniform across models and relative rankings vary by financial category. These observations highlight systematic variation across domains and motivate further diagnostic analysis in subsequent sections. Additional domain-level results for the remaining models are provided in Appendix E.2. Overall, these results indicate that symbolic financial reasoning ability is highly domain-dependent, and that strong aggregate performance does not guarantee robust reasoning across diverse financial categories.

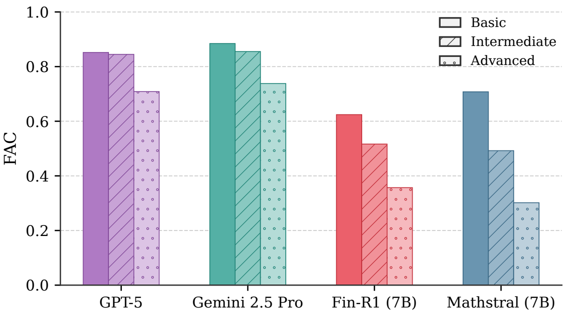


Figure 4: **Final answer correctness (FAC) across difficulty levels.** FAC for representative models on *Basic*, *Intermediate*, and *Advanced* FINCHAIN instances.

5.4.3 Performance Across Difficulty Levels

To assess the model robustness under increasing reasoning complexity, we group FINCHAIN instances into three predefined difficulty tiers: *Basic*, *Intermediate*, and *Advanced* (§ 3). Each tier corresponds to an increase in the number of required reasoning steps and the depth of symbolic and numerical operations. Figure 4 reports model performance across difficulty levels using final answer correctness (FAC), isolating end-task success from partial step-level alignment captured by CHAINEVAL. Across all tiers, frontier proprietary models achieve the highest correctness, with GPT-5 and Gemini 2.5 Pro maintaining relatively strong performance as task difficulty increases. Nevertheless, even these models exhibit a clear degradation on advanced instances, highlighting the challenge

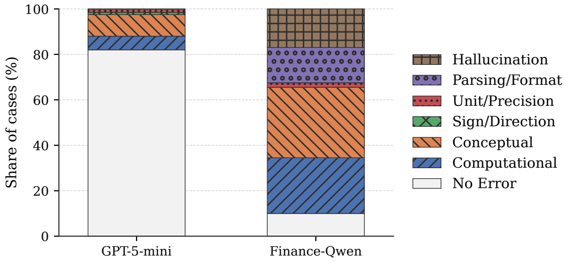


Figure 5: **Error type distribution on a randomly sampled subset of 200 questions.**

of solving complex, multi-step financial reasoning problems to completion. In contrast, open-weight and fine-tuned models show substantially steeper declines as difficulty increases. Mathstral performs competitively on *Basic* and *Intermediate* tasks, suggesting that mathematical fine-tuning improves structured numerical reasoning, but its performance drops markedly on *Advanced* instances. Fin-R1 displays similar behavior, achieving reasonable accuracy on simpler finance-oriented queries while degrading more sharply on tasks requiring longer reasoning chains. Overall, FAC reveals a pronounced gap between frontier and non-frontier models on advanced symbolic reasoning tasks. Taken together with the more gradual degradation observed under CHAINEVAL, these results suggest that completing long, multi-step reasoning chains remains a key bottleneck. We report the corresponding difficulty breakdown under CHAINEVAL in Appendix E.3.

5.5 Error Analysis

We conduct a targeted error analysis on a randomly sampled subset of 200 questions from FINCHAIN, using the same question set for GPT-5-mini and Finance-Qwen. These two models serve as contrasting points on the performance spectrum under an identical evaluation setup, with GPT-5-mini as a strong frontier-aligned reference and Finance-Qwen as a finance-tuned model with lower overall accuracy. This shared subset enables a direct comparison of error profiles without confounding effects from domain coverage or evaluation scale. We manually inspect model outputs and assign them to a small set of coarse-grained error categories capturing common reasoning failures in financial problem solving. Our taxonomy includes *Computational* errors (incorrect numerical execution), *Conceptual* errors (incorrect application of financial concepts), *Sign/Direction* errors, *Unit/Precision* mismatches, *Parsing/Format* issues that prevent reliable step alignment, and *Hallucination*, which includes content unsupported by the input. Outputs that are fully correct within tolerance are labeled as *No Error*. A detailed description of each category is provided in Appendix F.1. During expert review, a small number of borderline cases initially categorized as computational errors were revised to *No Error* after manual inspection, primarily when numerical deviations were deemed immaterial. All reported statistics reflect these expert-validated labels. Figure 5 summarizes the resulting error distributions. GPT-5-mini produces correct solutions for most cases, with remaining failures concentrated in numerical computation and conceptual reasoning. Other error types are rare, and no hallucinated or unparsable outputs are observed on this subset. In contrast, Finance-Qwen exhibits errors on most evaluated cases, with failures spread across multiple categories. Conceptual and computational errors dominate, while a substantial fraction involves formatting issues or hallucinated information. Compared to GPT-5-mini, Finance-Qwen errors are less concentrated in a single mode and reflect more diverse failure behaviors. Examples of representative errors are provided in Appendix F.2. Overall, this analysis shows that symbolic financial reasoning errors are heterogeneous and model-dependent, with weaker models exhibiting more diverse and compounding failure modes, helping explain the aggregate performance gaps.

6 Conclusions and Future Work

We introduced FINCHAIN, a symbolic benchmark for verifiable Chain-of-Thought financial reasoning, spanning 58 topics across 12 domains and three difficulty levels. To support step-level evaluation, we proposed CHAINEVAL, which jointly assesses intermediate reasoning consistency and final-answer correctness. Our results showed that while frontier LLMs perform best overall, even the strongest models struggle with complex symbolic reasoning, and fine-tuned open-source systems narrow down, but do not close this gap.

In future work, we plan to extend FINCHAIN to multilingual and region-specific settings and to incorporate problems grounded in real-world financial documents. This direction aims to bridge symbolic reasoning and factual verification (Xie et al., 2025), advancing more interpretable and reliable financial AI systems.

Limitations

This work has several limitations, which can be addressed in future research. First, our dataset is entirely synthetic and generated from symbolic templates. While this design enables controllable, contamination-free generation and automatic verification, it may lack the linguistic diversity and contextual richness of real-world financial texts. Future work could incorporate real financial documents as seed inputs for semi-structured generation while preserving symbolic grounding.

Second, the benchmark focuses on symbolic numerical reasoning and does not capture qualitative or strategic aspects of financial decision-making, such as risk assessment or market sentiment. Extending the benchmark to cover these higher-level reasoning dimensions remains an open challenge.

Third, FINCHAIN is limited to English and primarily reflects U.S.-centric financial conventions, which restricts its applicability to multilingual and regional contexts. Expanding to additional languages and financial systems is an important direction for future work.

Finally, our evaluation relies on automatic parsing of model-generated reasoning chains, which can be sensitive to formatting variations or extraneous text. Improving the robustness of step-level alignment through more structured output formats or tighter integration with symbolic execution is another possible direction for future work.

Ethical Statement and Broad Impact

This work uses only synthetic data generated through templated code and language model outputs. No private, sensitive, or copyrighted content was used. Our benchmark is designed for transparency and reproducibility in financial AI. However, caution should be taken when deploying LLMs in real-world financial decision-making, especially where symbolic correctness and regulatory compliance are critical. We believe FINCHAIN will support research toward more interpretable, verifiable, and safe reasoning systems in high-stakes domains.

Data License The FINCHAIN dataset and accompanying code will be released under the MIT License.

Acknowledgments

We thank all our financial experts for their assistance in reviewing the templates. This project would not have been possible without their support. We also express our sincere gratitude to our student volunteers for their interest in this project and for developing the live demo and demonstration website. Finally, we thank the reviewers for their valuable feedback, which helped us improve the quality of this paper.

References

Cengiz Asmazoğlu Abdullah Bezir, Furkan Burhan Türkay. 2025. Wiroai/wiroai-finance-qwen-7b. *Hugging Face Hub*.

Anthropic. 2025a. Claude 3.7 sonnet system card. https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf.

Anthropic. 2025b. Claude sonnet 4.5 system card. https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf.

Anthropic. 2025c. System card: Claude opus 4 & claude sonnet 4. https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *ArXiv preprint*, abs/1908.10063.

Zvi Bodie, Robert C. Merton, and Richard T. Thakor. 2025. *Principles of Finance*. Cambridge University Press.

Jian Chen, Peilin Zhou, Yining Hua, Loh Xin, Kehui Chen, Ziyuan Li, Bing Zhu, and Junwei Liang. 2024a. FinTextQA: A dataset for long-form financial question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6047, Bangkok, Thailand. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhiyu Zoey Chen, Jing Ma, Xinlu Zhang, Nan Hao, An Yan, Armineh Nourbakhsh, Xianjun Yang, Julian McAuley, Linda Petzold, and William Yang Wang. 2024b. A survey on large language models for critical societal domains: Finance, healthcare, and law. *ArXiv preprint*, abs/2405.01769.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Duanyu Feng, Yongfu Dai, Jimin Huang, Yifang Zhang, Qianqian Xie, Weiguang Han, Zhengyu Chen, Alejandro Lopez-Lira, and Hao Wang. 2023. Empowering many, biasing a few: Generalist credit scoring through large language models. *ArXiv preprint*, abs/2310.00566.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Oudom Hean, Utsha Saha, and Binita Saha. 2025. Can ai help with your personal finances? *Applied Economics*, pages 1–9.

Simerjot Kaur, Charese Smiley, Akshat Gupta, Joy Sain, Dongsheng Wang, Suchetha Siddagangappa, Toyin Aguda, and Sameena Shah. 2023. Refind: Relation extraction financial dataset. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 3054–3063. ACM.

Rik Koncel-Kedziorski, Michael Krumdick, Viet Lai, Varshini Reddy, Charles Lovering, and Chris Tanner. 2023. Bizbench: A quantitative reasoning benchmark for business and finance. *ArXiv preprint*, abs/2311.06602.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiao-Yang Liu, Guoxuan Wang, Hongyang Yang, and Daochen Zha. 2023. Fingpt: Democratizing internet-scale data for financial large language models. *ArXiv preprint*, abs/2307.10485.

Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, and 1 others. 2025. Fin-r1: A large language model for financial reasoning through reinforcement learning. *ArXiv preprint*, abs/2503.16252.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *ArXiv preprint*, abs/2308.09583.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *ArXiv preprint*, abs/2410.05229.

AI Mistral. 2024. Mathstral. 7B parameter model for mathematical reasoning, released under Apache 2.0 license.

Rajdeep Mukherjee, Abhinav Bohra, Akash Banerjee, Soumya Sharma, Manjunath Hegde, Afreen Shaikh, Shivani Shrivastava, Koustuv Dasgupta, Niloy Ganguly, Saptarshi Ghosh, and Pawan Goyal. 2022. ECTSum: A new benchmark dataset for bullet point summarization of long earnings call transcripts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10893–10906, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. A survey of large language models for financial applications: Progress, prospects and challenges. *ArXiv preprint*, abs/2406.11903.

OpenAI. 2025a. Gpt-5 system card. `https://cdn.openai.com/gpt-5-system-card.pdf`. System card describing GPT-5 variants (thinking/main/mini/-nano).

OpenAI. 2025b. Introducing gpt-4.1 in the api. `https://openai.com/index/gpt-4-1/`. Product research post introducing GPT-4.1, 4.1 mini, and 4.1 nano.

Yulong Pei, Amarachi Mbakwe, Akshat Gupta, Salwa Alamir, Hanxuan Lin, Xiaomo Liu, and Sameena Shah. 2022. TweetFinSent: A dataset of stock sentiments on Twitter. In *Proceedings of the Fourth Workshop on Financial Technology and Natural Language Processing (FinNLP)*, pages 37–47, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Lingfei Qian, Weipeng Zhou, Yan Wang, Xueqing Peng, Jimin Huang, and Qianqian Xie. 2025. Fino1: On the transferability of reasoning enhanced llms to finance. *ArXiv preprint*, abs/2502.08127.

Team Qwen. 2024. Qwen2.5 technical report. *ArXiv preprint*, abs/2412.15115.

Team Qwen. 2025. Qwen3.

Agam Shah, Ruchit Vithani, Abhinav Gullapalli, and Sudheer Chava. 2023. Finer: Financial named entity recognition dataset and weak-supervision model. *ArXiv preprint*, abs/2302.11157.

Raj Shah, Kunal Chawla, Dheeraj Eidnani, Agam Shah, Wendi Du, Sudheer Chava, Natraj Raman, Charese Smiley, Jiaao Chen, and Diyi Yang. 2022. When FLUE meets FLANG: Benchmarks and large pretrained language model for financial domain. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2335, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Eugene Sy, Tzu-Cheng Peng, Shih-Hsuan Huang, Heng-Yu Lin, and Yung-Chun Chang. 2023. Fine-grained argument understanding with BERT ensemble techniques: A deep dive into financial sentiment analysis. In *Proceedings of the 35th Conference on Computational Linguistics and Speech Processing (ROCLING 2023)*, pages 242–249, Taipei City, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Yixuan Tang and Yi Yang. 2025. Finmteb: Finance massive text embedding benchmark. *ArXiv preprint*, abs/2502.10990.

Zichen Tang, Haihong E, Ziyan Ma, Haoyang He, Jiacheng Liu, Zhongjun Yang, Zihua Rong, Rongjin Li, Kun Ji, Qing Huang, Xinyang Hu, Yang Liu, and Qianhe Zheng. 2025. FinanceReasoning: Benchmarking financial numerical reasoning more credible, comprehensive and challenging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15721–15749, Vienna, Austria. Association for Computational Linguistics.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *ArXiv preprint*, abs/2303.17564.

xAI. 2025. Grok 4 model card. `https://data.x.ai/2025-08-20-grok-4-model-card.pdf`. Accessed: 2025-10-17.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, and 1 others. 2024. The finben: An holistic financial benchmark for large language models. *ArXiv preprint*, abs/2402.12659.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023a. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *ArXiv preprint*, abs/2306.05443.

Zhuohan Xie, Trevor Cohn, and Jey Han Lau. 2023b. The next chapter: A study of large language models in storytelling. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 323–351, Prague, Czechia. Association for Computational Linguistics.

Zhuohan Xie, Miao Li, Trevor Cohn, and Jey Lau. 2023c. DeltaScore: Fine-grained story evaluation with perturbations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5317–5331, Singapore. Association for Computational Linguistics.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. 2025. FIRE: Fact-checking with iterative retrieval and verification. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2901–2914, Albuquerque, New Mexico. Association for Computational Linguistics.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv preprint*, abs/2409.12122.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *ArXiv preprint*, abs/2309.12284.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. 2024. R-judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1467–1490, Miami, Florida, USA. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023a. A survey of large language models. *ArXiv preprint*, abs/2303.18223.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. *arXiv preprint arXiv:2206.01347*.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2023b. Financemath: Knowledge-intensive math reasoning in finance domains. *arXiv preprint arXiv:2311.09797*.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2023c. Docmath-eval: Evaluating math reasoning capabilities of llms in understanding long and specialized documents. *arXiv preprint arXiv:2311.09805*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

Jie Zhu, Qian Chen, Huaixia Dou, Junhui Li, Lifan Guo, Feng Chen, and Chi Zhang. 2025. Dianjin-r1: Evaluating and enhancing financial reasoning in large language models. *ArXiv preprint*, abs/2504.15716.

A FINCHAIN Construction Details

A.1 Template Creation Prompt

To construct symbolic financial reasoning benchmarks analogous to GSM-Symbolic, we design a structured prompt that guides the generation of executable financial templates. These templates support variable-based instantiation, symbolic stepwise supervision, and controlled perturbations for robustness evaluation. Below, we present the prompt used for template construction.

System Instruction: You are a financial NLP expert developing symbolic reasoning datasets. Your task is to construct symbolic templates for financial reasoning problems. Each template should support (i) controlled generation of diverse question instances, (ii) executable reasoning traces for automatic verification, and (iii) systematic variation in surface and logical complexity.

Please follow the steps below:

1. **Identify a financial reasoning task:** For example, compound interest, IRR, ROI, NPV, breakeven analysis, loan amortization, etc.

2. **Write a natural language question template:** Formulate the question using variable placeholders instead of fixed values. For instance, use `{principal}`, `{rate}`, `{years}`, etc.

3. **Define variables and constraints:** Specify the domain (e.g., numerical range or categorical values) for each variable. Add algebraic constraints to ensure the question is solvable and the answer valid.

4. **Write a symbolic solution trace:** Provide a step-by-step solution using the variables. Ensure the reasoning chain is executable in Python for automatic evaluation.

5. **Vary difficulty levels:** For each task, generate 10 templates with increasing complexity. Longer and more compositional reasoning chains should correspond to harder levels.

A.2 Template Examples

Here, we present example templates of three compound interest (CI) financial questions, grouped by difficulty level, including basic, intermediate, and advanced.

Basic Level.

```
def template_ci_quarterly_compounding():
    """Basic: Compound Interest with Quarterly
      Compounding"""
    investor_name = random.choice(
     investor_names)
    project_name  = random.choice(
     project_names)

    # Parameters
    principal = random.randint(1_000, 7_000)
         # $
    rate      = round(random.uniform(2, 8), 2)
         # annual %, two decimals
    time      = random.randint(1, 3)
         # years
    n         = 4
          # quarterly

    # ---------- Question ----------
    question = (
        f"{investor_name} invests ${principal}
      in {project_name}. "
        f"The account earns {rate:.2f}%
     interest per year, compounded quarterly, "
        f"for {time} years. What is the total
     compound interest earned "
    )

    # ---------- Reasoning ----------
    # Step 1: future (compound) amount
    future_value = principal * (1 + rate /
     (100 * n)) ** (n * time)
    # Step 2: compound interest
    ci = future_value - principal

    # Round only for display
    fv_display = f"${future_value:.2f}"
    ci_display = f"${ci:.2f}"

    # ---------- Solution ----------
    solution = (
        "Step 1. Compute the future value with
      quarterly compounding:\n"
        "  n = 4 periods per year.\n"
        "  Future Value = P × (1 + r / (100 ×
      n))^(n × t)\n"
        f"              = ${principal} × (1 +
     {rate:.2f}% / (100 × 4))^(4 × {time})\n"
        f"              = ${principal} × (1 +
     {rate / (100 * n):.4f})^{4*time}\n"
        f"              = {fv_display}\n\n"
        "Step 2. Find the compound interest
     earned:\n"
        "  Compound Interest = Future Value —
     Principal\n"
        f"                    = {fv_display} —
     ${principal}\n"
        f"                    = {ci_display}"
    )

    return question, solution
```

Intermediate Level.

```
def template_ci_rate_and_total_known():
    """Intermediate: Compound Interest with
     nominal rate, time, and frequency known"""

    investor_name = random.choice(
     investor_names)
    project_name  = random.choice(
     project_names)

    # ---------- Parameters ----------
    total_amount = random.randint(5_000, 15
    _000)          # Final amount A ($)
    rate         = round(random.uniform(2, 10),
     2)        # Nominal annual rate %
    time         = random.randint(1, 5)
                 # Years
    freq_name, n = random.choice(
        [("semi-annually", 2), ("quarterly",
     4), ("monthly", 12)]
    )

    # ---------- Question ----------
    question = (
        f"{investor_name} received a total
     amount of ${total_amount:,.2f} "
        f"from their investment in {
     project_name}. "
        f"The investment grew at a nominal
     annual interest rate of {rate:.2f}% "
        f"compounded {freq_name} for {time}
     years. "
        f"Calculate the compound interest
     earned (in dollars)."
    )

    # ---------- Reasoning ----------
    # Step 1: periodic rate and growth factor
    periodic_rate = round(rate / 100 / n, 6)
                 # r_p
    growth_factor = round((1 + periodic_rate)
     ** (n * time), 6)

    # Step 2: principal P
    principal = round(total_amount /
     growth_factor, 2)     # 2 dp dollars

    # Step 3: compound interest CI
    ci = round(total_amount - principal, 2)

    # ---------- Solution ----------
    solution = (
        "Step 1 — Find the periodic rate and
     growth factor\n"
        f"  Periodic rate  = {rate:.2f}% ÷ {n}
      = {periodic_rate*100:.4f}%\n"
        f"  Growth factor  = (1 + {
     periodic_rate:.6f})^{n*time} = {
     growth_factor:.6f}\n\n"
        "Step 2 — Compute the initial
     principal\n"
        f"  P = A ÷ growth factor = "
        f"${total_amount:,.2f} ÷ {
     growth_factor:.6f} = ${principal:,.2f}\n\n
     "
        "Step 3 — Calculate the compound
     interest\n"
        f"  CI = A — P = ${total_amount:,.2f}
     — ${principal:,.2f} = ${ci:,.2f}"
    )

    return question, solution
```

Advanced Level.

```
def template_ci_with_additional_deposit():
    """Advanced: Compound Interest with a Mid
     Term Additional Deposit (needs 4 steps)"""
    investor_name = random.choice(
     investor_names)
    project_name  = random.choice(
     project_names)

    # --- parameters ---
    principal = random.randint(2000, 8000)
          # initial $
    rate      = round(random.uniform(3, 10),
     2)      # % p.a.
    time      = random.randint(3, 7)
          # total years (>=3 so a mid deposit
     makes sense)
    n         = random.choice([1, 2, 4, 12])
          # compounds per year

    deposit       = random.randint(500, 4000)
          # extra $
    deposit_time  = random.randint(1, time -
     1)      # year when deposit is made

    # ---------- Question ----------
    question = (
        f"{investor_name} initially invested $
     {principal} in {project_name} at an annual
      "
        f"rate of {rate:.2f}%, compounded {n}
     times a year, for a total of {time} years.
      "
        f"Exactly {deposit_time} years after
     the start, they added an extra ${deposit}
     "
        f"to the same account under the same
     rate and compounding frequency. "
        f"Calculate the total compound
     interest earned by the end of the {time}
     years."
    )

    # ---------- Reasoning ----------
    # Step 1 — periodic rate
    periodic_rate = round(rate / (100 * n), 4)

    # Step 2 — grow the original principal for
      the full period
    periods_principal = n * time
    fv_principal = round(principal * (1 +
     periodic_rate) ** periods_principal, 2)

    # Step 3 — grow the later deposit for the
     remaining (time - deposit_time) years
    remaining_years = time - deposit_time
    periods_deposit = n * remaining_years
    fv_deposit = round(deposit * (1 +
     periodic_rate) ** periods_deposit, 2)

    # Step 4 — combine amounts and find
     compound interest
    total_future_value   = round(fv_principal
     + fv_deposit, 2)
    total_contributions  = principal + deposit
    compound_interest    = round(
     total_future_value - total_contributions,
     2)

    # ---------- Solution ----------
    solution = (
        "Step 1 — Periodic rate:\n"
        f"  r = {rate:.2f}% / (100 x {n}) = {
     periodic_rate:.4f}\n\n"
        "Step 2 — Future value of the original
      principal:\n"
        f"  Periods = {n} x {time} = {
     periods_principal}\n"
        f"  FV1 = ${principal} x (1 + {
     periodic_rate:.4f})^{periods_principal} =
     "
        f"${fv_principal:.2f}\n\n"
        "Step 3 — Future value of the
     additional deposit:\n"
        f"  Remaining years = {time} - {
     deposit_time} = {remaining_years}\n"
        f"  Periods = {n} x {remaining_years}
     = {periods_deposit}\n"
        f"  FV2 = ${deposit} x (1 + {
     periodic_rate:.4f})^{periods_deposit} = "
        f"${fv_deposit:.2f}\n\n"
        "Step 4 — Total compound interest:\n"
        f"  Total FV = FV1 + FV2 = ${
     fv_principal:.2f} + ${fv_deposit:.2f} = "
        f"${total_future_value:.2f}\n"
        f"  Contributions = ${principal} + ${
     deposit} = ${total_contributions}\n"
        f"  Compound Interest = Total FV -
     Contributions = "
        f"${total_future_value:.2f} - ${
     total_contributions} = ${compound_interest
     :.2f}"
    )

    return question, solution
```

A.3 Data Validation Criteria

Directly prompting large language models to generate symbolic financial reasoning templates can lead to inconsistencies or incomplete specifications. To address these issues, we apply the following validation constraints prior to expert review.

Cross-national inconsistencies. Generated questions occasionally contained country-specific financial conventions (e.g., currencies, indices, or terminology). All such cases are standardized to U.S.-based financial settings.

Precision mismatch. In some cases, displayed values were rounded while computations used full precision. We align computational outputs with the displayed numerical precision.

Incomplete input specification. Some questions omitted variables required for computation. These cases are revised to include all necessary inputs.

Unit consistency. Currency symbols and units were inconsistently applied across questions and solutions. All templates are standardized to consistent units.

Non-informative steps. Certain generated solutions decomposed simple calculations into trivial substeps or omitted intermediate reasoning. These solutions are revised to reflect substantive reasoning steps.

Multiple targets. Some templates requested multiple output values, complicating evaluation. We constrain all templates to require a single target value.

A.4 Expert Review and Annotation Protocol

To further enhance data quality, we recruited ten financial experts to review all validated templates. The expert panel consists of seven graduate students in economics, finance, and related quantitative disciplines, and three industry professionals with experience in quantitative research, financial engineering, and risk management. Annotators were selected through an internal vetting process to ensure domain expertise and professional credibility. Additional demographic details are provided in § B.1.

Annotation Platform. We developed a Streamlit-based annotation platform to facilitate efficient expert review. Implementation details are provided in § B.2.

Pilot Study. Prior to full annotation, we conduct a pilot study in which 20 templates are reviewed by all annotators to calibrate evaluation standards. After calibration, all annotators agreed on the correctness of the pilot templates.

Main Annotation. The remaining 270 templates are randomly distributed among annotators, with each template reviewed by a single expert. Out of 290 total templates, 29 are identified as incorrect and subsequently revised by financial experts. Summary statistics of identified issues are reported in § B.3.

B Annotation and Quality Control

B.1 Financial Expert Demography

To ensure the reliability and domain robustness of our benchmark, all annotations were conducted by a diverse team of financial experts and advanced students with strong quantitative and economic backgrounds. The annotators collectively represent three major categories: (1) industry professionals in quantitative research and financial engineering, (2) postgraduate students specializing in finance, economics, and auditing, and (3) experienced annotators trained in data labeling and financial analysis.

Several annotators have extensive industry experience across financial technology, quantitative research, and trading, with prior roles in investment banks, hedge funds, and fintech companies. Others are graduate students conducting research in finance, economics, and auditing, contributing academic rigor and theoretical grounding. Together, they bring complementary expertise that enhances both the practical and analytical aspects of our benchmark construction.

Summary. Our benchmark construction relies on a team of ten highly qualified annotators, including three industry professionals with prior experience in quantitative research or trading, and seven academic annotators who are graduate students in finance, economics, and auditing. This balanced composition, encompassing strong and diverse backgrounds in computer science, mathematics, statistics, and finance, ensures both professional authenticity and academic depth. Their combined expertise provides a robust foundation for high-quality, domain-consistent annotations, contributing to the overall reliability of FINCHAIN. The following are the details for each of them.

Annotator A: Currently pursuing a Ph.D. at a leading university in Asia, this annotator previously worked as a quantitative researcher at a fintech company, with experience across multiple financial markets including domestic equities, U.S. equities, Hong Kong equities, and cryptocurrencies. Their research focused on financial data generation, risk modeling, and trading strategies. They have also served as a research lead in risk management at a cryptocurrency investment fund. This blend of academic research and cross-market industry practice enhances the robustness and domain relevance of the benchmark annotations.

Annotator B: A Master's student at a leading university with a strong undergraduate background in finance. They previously interned in the equity financing division of a major securities firm, contributing practical insights into capital markets and investment banking.

Annotator C: A Master's student at a top institution, holding a bachelor's degree in economics. Their training bridges theoretical economics and applied policy research, enriching the annotation process with domain-specific understanding.

Annotator D: Holds a bachelor's degree in economics and has received graduate admission offers from top international institutions. Their interdisciplinary background strengthens the dataset's coverage of trade and international finance contexts.

Annotator E: Holds a bachelor's degree in economics, providing a solid foundation in macroeconomic theory and financial principles that supports reliable annotation and consistency across financial texts.

Annotator F: A Master's student at a well-known university specializing in auditing and intelligent systems, with a research focus on large language model evaluation and its applications in auditing. Their familiarity with both auditing and financial concepts supports the annotation of financial news and auditing benchmarks from a research-oriented perspective.

Annotator G: A Master's student at a university recognized for its auditing and financial programs, with strong grounding in auditing, financial analysis, and data quality control. Their prior participation in annotation projects ensures consistent standards for annotation accuracy.

Annotator H: A quantitative analyst with an MSc-equivalent degree in financial technology from a top UK university. They have prior experience at major global financial institutions, focusing on stochastic modeling, risk management, and process automation. They also contribute to research on large language models in finance and are advancing toward professional certification in investment analysis.

Annotator I: A quantitative researcher at a global investment firm with prior experience at quantitative research and technology companies. Their work spans cross-asset systematic strategies, portfolio optimization, and machine learning applications in trading. They also serve as a teaching assistant for a postgraduate course on systematic trading strategies.

Annotator J: A quantitative trading analyst focused on equity derivatives, holding a postgraduate degree in financial engineering and risk management from a top European university and a bachelor's degree from a globally recognized institution. Their professional experience includes roles at several financial institutions across asset management, banking, and fintech, covering alpha-signal development, portfolio optimization, and derivatives trading.

B.2 Annotation Platform

We developed a custom annotation platform to evaluate the correctness of Python templates that generate financial questions and solutions. Each template corresponds to a financial scenario (e.g., investment analysis, compound interest, deposits, or ratio calculations). Annotators are instructed to review the code and determine whether both the financial framework and its implementation are correct, and whether the output representation (e.g., units, rounding) complies with the annotation policy.

The annotation task requires a binary verdict: *Correct* or *Incorrect*. Templates labeled as *Correct* need no modifications, though annotators may optionally provide comments. Templates labeled as *Incorrect* must be associated with one or more issue tags, accompanied by a minimal code correction and a brief explanation.

We defined two verdict categories. A template is considered *Correct* when its financial framework, calculations, and representation fully conform to the policy. It is marked as *Incorrect* if any substantive flaw is present in framework selection, mathematical logic, representation, robustness, or clarity. To facilitate consistent labeling, we introduced five issue tags:

- **Formula Choice Error**: An incorrect financial framework or formula is applied (e.g., simple vs. compound interest).
- **Math/Logic Error**: Arithmetic or algorithmic errors within the chosen formula (e.g., $r \times n$ instead of r/n).
- **Representation Error**: Inconsistent or incorrect handling of numbers, units, or rounding (e.g., annual vs. monthly mismatch).
- **Robustness Error**: Failures on boundary or extreme inputs (e.g., division by zero, negative values).
- **Clarity Issue**: Ambiguous variable names or comments that hinder auditability, even if the numerical results are correct.

To further support annotators, the platform provides curated reference cases across five templates within a single finance topic: compound interest. Each case includes (1) a question example, (2) a

Issue Type	Count	Proportion (%)
Representation Error	12	41.4
Clarity Issue	9	31.0
Formula Choice Error	5	17.2
Math/Logic Error	3	10.3
Robustness Error	2	6.9
Total Tagged Templates	**29**	**100.0**

Table 3: Distribution of issue types among annotated templates.

potential error type aligned with one of the defined issue tags, (3) a bad solution illustrating the error, and (4) a minimal code fix. Figure 6 shows two representative cases: a *Formula Choice Error*, where simple interest is incorrectly applied in a compound interest setting, and a *Math/Logic Error*, where the exponent is omitted. Such examples provide concrete guidance for annotators, ensuring consistency and reliability.

After reviewing these reference cases, annotators proceed to the main annotation interface, where they evaluate unseen templates (Figure 7). For each template, annotators must issue a binary verdict, select one or more issue tags if applicable, and provide a minimal code correction with a short justification. The interface presents the Python template and its generated question on the left, while the right panel allows annotators to record their verdict, choose tags, and edit the code directly. This design mirrors realistic auditing conditions and ensures that annotations capture both error identification and corrective reasoning.

B.3 Annotated Template Issue Statistics

Out of 290 templates, 29 (10%) were tagged as containing errors during the annotation process. Table 3 summarizes the distribution of issue types. Most problems stem from representation and clarity errors, followed by formula selection, logical inconsistencies, and robustness issues.

B.4 Review Rubrics

To ensure fair and interpretable human evaluation, each model response is assessed along two complementary dimensions: *Reasoning Process Quality* and *Final Answer Accuracy*. For each question, reviewers are provided with the question itself, the standard reference answer, and the generated responses from different models. They independently assign scores on a 1–5 scale for each dimension, following the detailed rubrics below.

B.5 Reasoning Process Quality

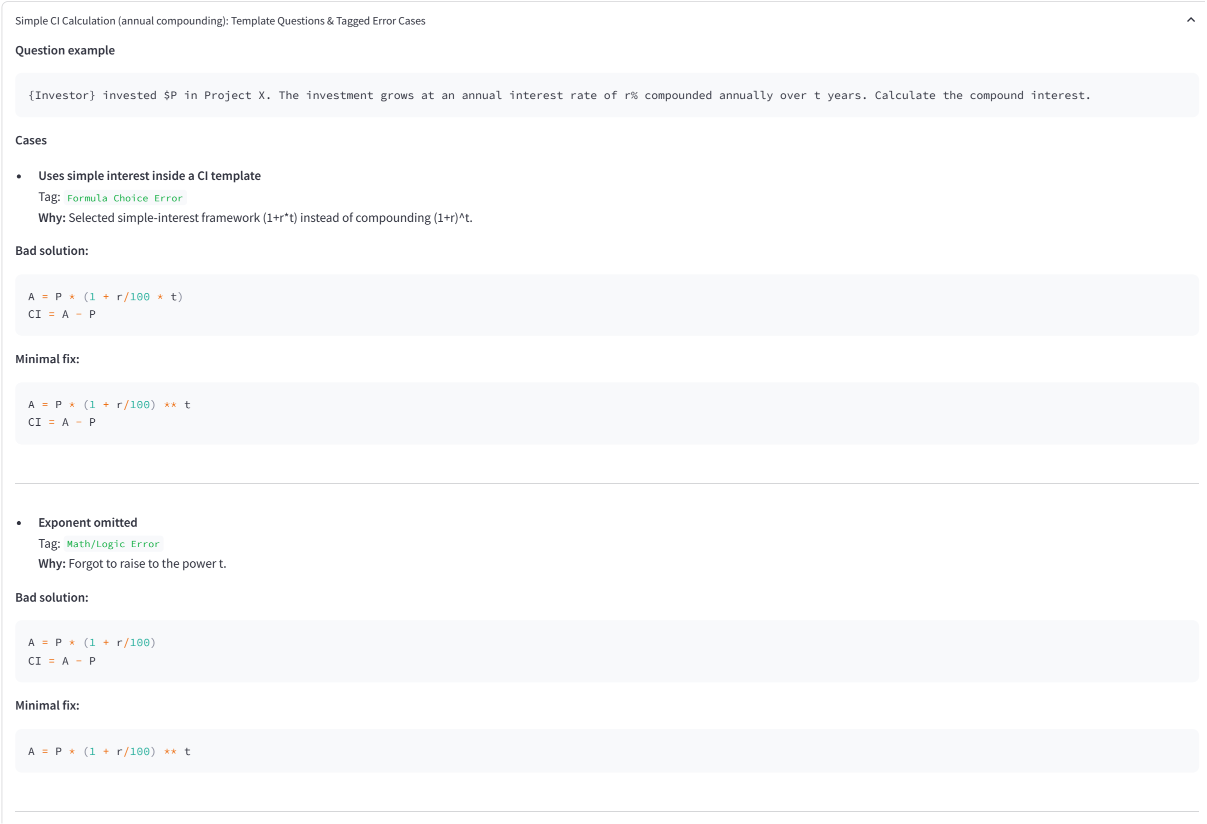
This dimension evaluates how clearly, logically, and correctly the model articulates its reasoning steps leading to the final answer. High-quality reasoning should demonstrate coherent logical flow, factual correctness, and consistency with valid domain principles.

- **1 (Unacceptable):** Illogical, incoherent, or irrelevant reasoning; missing steps or severe conceptual errors.
- **2 (Poor):** Some reasoning attempt but with major factual or procedural flaws; inconsistent or unclear Chain-of-Thought.
- **3 (Fair):** Partial understanding with mixed correct and incorrect reasoning; superficial or incomplete explanation.
- **4 (Good):** Mostly correct and coherent reasoning with minor inaccuracies or unclear phrasing; logical flow generally sound.
- **5 (Excellent):** Clear, well-structured, and logically consistent reasoning throughout; fully correct and well-justified steps.

B.6 Final Answer Accuracy

This dimension evaluates the correctness and completeness of the model's final output relative to the reference solution. Reviewers compare each model's final answer with the standard answer to determine whether the model's conclusion is correct and sufficiently supported.

- **1 (Unacceptable):** Completely incorrect or missing answer; no alignment with the reference solution.
- **2 (Poor):** Largely incorrect due to major conceptual or computational errors.
- **3 (Fair):** Partially correct; captures some relevant elements but omits or distorts key aspects of the correct solution.
- **4 (Good):** Largely correct and complete with only minor inaccuracies that do not affect the main result.
- **5 (Excellent):** Fully correct, precise, and complete; matches the reference solution exactly or with an equivalent formulation.



Simple CI Calculation (annual compounding): Template Questions & Tagged Error Cases

Question example

```
{Investor} invested $P in Project X. The investment grows at an annual interest rate of r% compounded annually over t years. Calculate the compound interest.
```

Cases

- **Uses simple interest inside a CI template**
  Tag: `Formula Choice Error`
  Why: Selected simple-interest framework (1+r*t) instead of compounding (1+r)^t.

Bad solution:

```
A = P * (1 + r/100 * t)
CI = A - P
```

Minimal fix:

```
A = P * (1 + r/100) ** t
CI = A - P
```

- **Exponent omitted**
  Tag: `Math/Logic Error`
  Why: Forgot to raise to the power t.

Bad solution:

```
A = P * (1 + r/100)
CI = A - P
```

Minimal fix:

```
A = P * (1 + r/100) ** t
```

Figure 6: Reference examples for compound interest templates, illustrating typical annotation cases with error tags, flawed solutions, and minimal fixes.

C Model Detail Information

We provide the details of the evaluated models in Table 4.

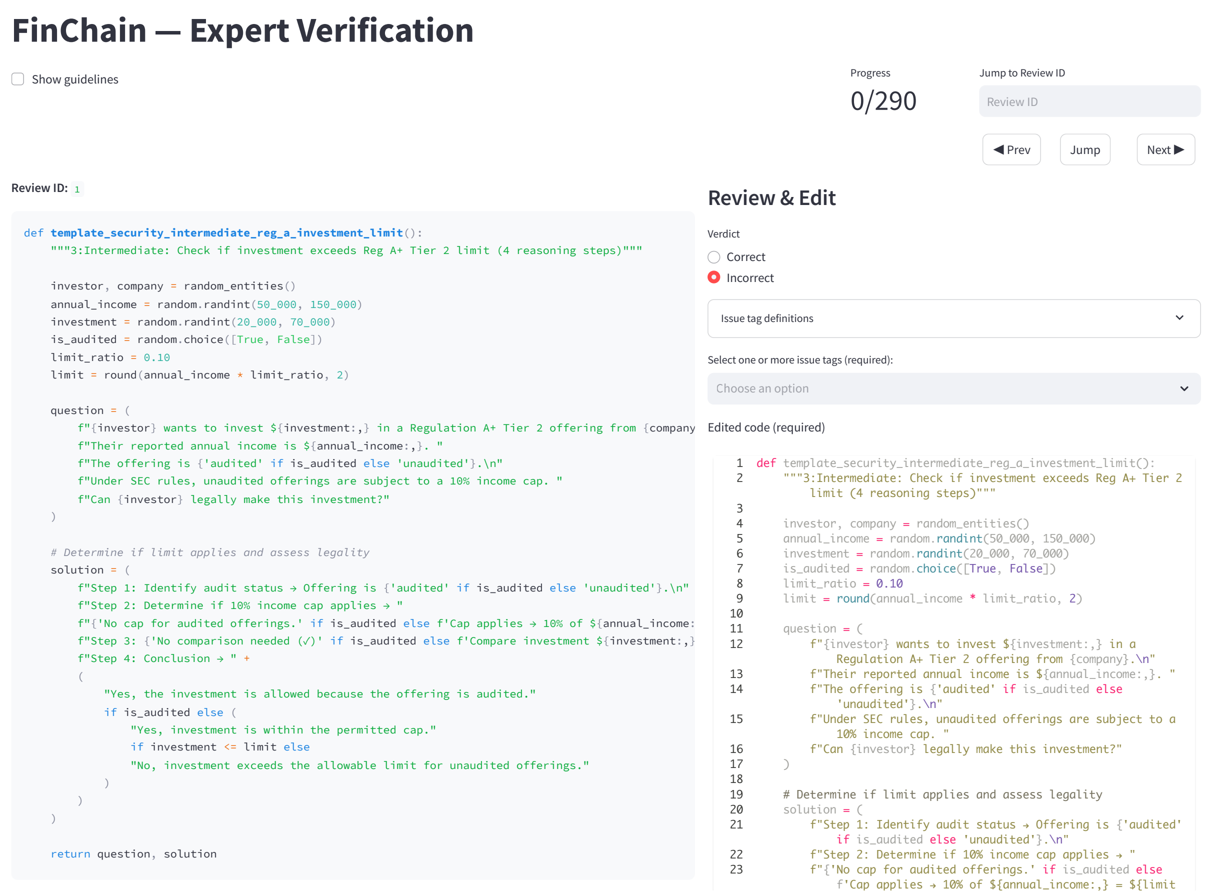
D Metric Evaluation and Ablations

To better understand the behavior of our proposed metric, we conducted a series of ablation experiments and comparative analyses. All quantitative results reported in this section are benchmarked against expert human evaluations of reasoning quality (see § B.1 for expert details).

D.1 Ablations of the DTW-Based Metric

Our main evaluation metric, the **Normalized DTW Alignment Score (Gate Mode)**, measures both local semantic-numeric agreement and global sequence-level alignment between predicted and gold reasoning traces. To assess its robustness and the effect of its design choices, we considered several variants:

- **DTW Gate Mode.** This is the primary formulation used in the paper. Semantic similarity and numeric agreement are combined multiplicatively, i.e., $\mathrm{Score}_{\mathrm{gate}}(i,j) = \mathrm{SS}(s_i^*, \hat{s}_j) \times \mathrm{AM}(s_i^*, \hat{s}_j)$. This "gating" ensures that steps only contribute when both semantic meaning and intermediate results align.
- **DTW Soft Mode.** A more permissive variant that blends semantic and numeric agreement through a weighted combination: $\mathrm{Score}_{\mathrm{soft}}(i,j) = \alpha\,\mathrm{SS}(s_i^*, \hat{s}_j) + \beta\,\mathrm{AM}(s_i^*, \hat{s}_j)$, with weights $\alpha = 0.85$ and $\beta = 0.15$. This "soft" formulation captures cases where partial numeric agreement still reflects correct reasoning, providing smoother sensitivity to small deviations. In other words, while the Gated version will assign 0 to a sequence of aligning reasoning steps, which resulted in a wrong answer (which can be a case if an LLM fails mathematics behind the solution), Soft version will still give a higher score.
- **DTW Precision, Recall, and F1.** In addition to the normalized alignment score, we derive DTW-based precision, recall, and F1 measures that quantify step-level coverage under the DTW alignment path. These provide a



Progress 0/290

Review ID: 1

```
def template_security_intermediate_reg_a_investment_limit():
    """3:Intermediate: Check if investment exceeds Reg A+ Tier 2 limit (4 reasoning steps)"""

    investor, company = random_entities()
    annual_income = random.randint(50_000, 150_000)
    investment = random.randint(20_000, 70_000)
    is_audited = random.choice([True, False])
    limit_ratio = 0.10
    limit = round(annual_income * limit_ratio, 2)

    question = (
        f"{investor} wants to invest ${investment:,} in a Regulation A+ Tier 2 offering from {company
        f"Their reported annual income is ${annual_income:,}. "
        f"The offering is {'audited' if is_audited else 'unaudited'}.\n"
        f"Under SEC rules, unaudited offerings are subject to a 10% income cap. "
        f"Can {investor} legally make this investment?"
    )

    # Determine if limit applies and assess legality
    solution = (
        f"Step 1: Identify audit status → Offering is {'audited' if is_audited else 'unaudited'}.\n"
        f"Step 2: Determine if 10% income cap applies → "
        f"{'No cap for audited offerings.' if is_audited else f'Cap applies → 10% of ${annual_income:
        f"Step 3: {'No comparison needed (✓)' if is_audited else f'Compare investment ${investment:,}
        f"Step 4: Conclusion → " +
        (
            "Yes, the investment is allowed because the offering is audited."
            if is_audited else (
                "Yes, investment is within the permitted cap."
                if investment <= limit else
                "No, investment exceeds the allowable limit for unaudited offerings."
            )
        )
    )

    return question, solution
```

Verdict

○ Correct
● Incorrect

Select one or more issue tags (required):

Edited code (required)

```
def template_security_intermediate_reg_a_investment_limit():
    """3:Intermediate: Check if investment exceeds Reg A+ Tier 2
        limit (4 reasoning steps)"""

    investor, company = random_entities()
    annual_income = random.randint(50_000, 150_000)
    investment = random.randint(20_000, 70_000)
    is_audited = random.choice([True, False])
    limit_ratio = 0.10
    limit = round(annual_income * limit_ratio, 2)

    question = (
        f"{investor} wants to invest ${investment:,} in a
            Regulation A+ Tier 2 offering from {company}.\n"
        f"Their reported annual income is ${annual_income:,}. "
        f"The offering is {'audited' if is_audited else
            'unaudited'}.\n"
        f"Under SEC rules, unaudited offerings are subject to a
            10% income cap. "
        f"Can {investor} legally make this investment?"
    )

    # Determine if limit applies and assess legality
    solution = (
        f"Step 1: Identify audit status → Offering is {'audited'
            if is_audited else 'unaudited'}.\n"
        f"Step 2: Determine if 10% income cap applies → "
        f"{'No cap for audited offerings.' if is_audited else
            f'Cap applies → 10% of ${annual_income:,} = ${limit
```

Figure 7: Expert annotation interface. Annotators review each template, assign a verdict, select issue tags, and provide minimal code corrections.

finer breakdown of reasoning alignment quality.

To also capture the correctness of the final result, we also tried a weighted sum of the DTW-based scores and the final answer's correctness. We used all possible α's in range $0.1 - 0.9$ to identify the best proportion. Based on these experiments, the α of 0.1 resulted in the best Spearman ρ and it was used for further comparison.

D.2 Comparative Evaluation

We also evaluated a range of traditional text-similarity and reasoning metrics, including ROUGE-2, ROUGE-L, step-level precision and recall (marked as 'non-DTW' in the table), BERTScore, and our weighted sum of DTWNorm-Gate and final answer's correctness (DTWNorm-Gate+FAC in the table). Each metric was correlated with expert-assigned *Reasoning Process Quality* score. Table 5 summarizes the top Spearman correlations with expert process judgments.

As additional validation, we also measure other correlation metrics: Kendall tau (which measures similarity of rankings) and Pearson correlation. As shown in Table 6 and Table 7, our suggested metric still holds high correlation position even with other correlation metrics.

D.3 Discussion

The DTW-based variants consistently achieve the highest correlation with expert judgments, with the **Normalized DTW Alignment Score (Gate Mode) with FAC** emerging as the most reliable indicator of reasoning faithfulness. Non-FAC and the "Soft" variant yields slightly lower but still strong correlations, suggesting that the gating formulation better captures strict consistency, while the soft variant provides smoother sensitivity to near-correct reasoning. Compared to traditional metrics such as ROUGE or simple step-level precision and recall, DTW captures not only semantic similarity but also the structural coherence and numerical consistency of reasoning chains. These results highlight the usefulness of our proposed metric.

Model	Organization	Size	Backbone	Source
Frontier Proprietary LLMs				
GPT-5	OpenAI	N/A	–	`gpt-5-2025-08-07`
GPT-4.1	OpenAI	N/A	–	`gpt-4.1-2025-04-14`
GPT-5 mini	OpenAI	N/A	–	`gpt-5-mini-2025-08-07`
GPT-4.1 mini	OpenAI	N/A	–	`gpt-4.1-mini-2025-04-14`
Claude Sonnet 4.5	Anthropic	N/A	–	`claude-sonnet-4-5-20250929`
Claude Sonnet 4	Anthropic	N/A	–	`claude-sonnet-4-20250514`
Claude Sonnet 3.7	Anthropic	N/A	–	`claude-3-7-sonnet-20250219`
Gemini 2.5 Pro	Google	N/A	–	`Last Update: June 2025`
Gemini 2.5 Flash	Google	N/A	–	`Last Update: June 2025`
DeepSeek V3.2	DeepSeek	N/A	–	`Last Update: Sep 29 2025`
DeepSeek V3.1	DeepSeek	N/A	–	`Last Update: Sep 22 2025`
DeepSeek R1	DeepSeek	N/A	–	`Last Update: Jan 20 2025`
Grok 4 Heavy	xAI	N/A	–	`grok-4-0709`
Grok 4 Fast	xAI	N/A	–	`grok-4-fast-reasoning`
Finance Specific LLMs				
Fino1	TheFinAI	8B	`meta-llama/Llama-3.1-8B`	`TheFinAI/Fin-o1-8B`
Fin-R1	SUFE-AIFLM-Lab	7B	`Qwen/Qwen2.5-7B-Instruct`	`SUFE-AIFLM-Lab/Fin-R1`
DianJin-R1	Qwen DianJin Team	7B	`Qwen/Qwen2.5-7B-Instruct`	`DianJin/DianJin-R1-7B`
Finance-LLaMA	Wiro AI	8B	`deepseek-ai/DeepSeek-R1-Distill-Llama-8B`	`WiroAI/WiroAI-Finance-Llama-8B`
Finance-Qwen	Wiro AI	7B	`Qwen/Qwen2.5-7B`	`WiroAI/WiroAI-Finance-Qwen-7B`
Math Enhanced LLMs				
WizardMath	WizardLM Team	7B	`mistralai/Mistral-7B-v0.1`	`WizardLMTeam/WizardMath-7B-V1.1`
MetaMath	MetaMath Project	7B	`EleutherAI/llemma 7b`	`meta-math/MetaMath-7B-V1.0`
Mathstral	Mistral AI	7B	`mistralai/Mistral-7B-v0.1`	`mistralai/Mathstral-7B-v0.1`
Qwen2.5-Math	Qwen Team	7B	`Qwen/Qwen2.5-7B`	`Qwen/Qwen2.5-Math-7B-Instruct`
General Purpose Open LLMs				
LLaMA 3.1	Meta	8B	-	`meta-llama/Llama-3.1-8B`
Qwen 2.5	Qwen Team	7B	-	`Qwen/Qwen2.5-7B-Instruct`
Qwen 3	Qwen Team	8B	-	`Qwen/Qwen3-8B`

Table 4: Details of the organization and model source (i.e. model version for proprietary models, and HuggingFace model name for open-source models) for the LLMs evaluated in FINCHAIN.

Metric	Spearman ρ
DTWNormGate+FAC	**0.655**
DTWNormGate	0.640
DTW Precision (Soft)	0.625
DTW Precision (Gate)	0.622
DTW F1 (Soft)	0.619
DTW F1 (Gate)	0.618
Step Precision (non-DTW)	0.604
DTWNormSoft	0.592
DTW Recall (Gate)	0.573
Step Recall (non-DTW)	0.570
DTW Avg. Path Score (Gate)	0.529
DTW Avg. Path Score (Soft)	0.526
DTW Recall (Soft)	0.512
ROUGE-2	0.469
ROUGE-L	0.434
BERTScore	0.287

Table 5: Spearman correlation with expert evaluation of the reasoning process.

Metric	Pearson ρ
DTWNormGate+FAC	**0.584**
DTW F1 (Soft)	**0.584**
DTW Precision (Soft)	0.578
DTW F1 (Gate)	0.568
DTW Avg. Path Score (Gate)	0.556
DTW Recall (Gate)	0.550
Step Recall (non-DTW)	0.549
DTW Precision (Gate)	0.548
Step Precision (non-DTW)	0.530
DTWNormSoft	0.515
DTW Avg. Path Score (Soft)	0.515
DTW Recall (Soft)	0.511
DTWNormGate	0.492
ROUGE-L	0.421
ROUGE-2	0.420
BERTScore	0.300

Table 6: Pearson correlation with expert evaluation of the reasoning process.

D.4 Suggestions For Reproducibility

Although our approach shows a strong correlation with human evaluations, it is not without limitations.

The way in which the reasoning steps are parsed from the model's output plays an important role in the quality estimation. In this work we combine both LLM-based parsing and parsing using regular expressions. We try to split the response on reasoning steps using regular expressions, which capture 'Step X'-like patterns in the response, if such patterns are not found, we instruct LLM to split the answer on steps, copy-pasting the whole step string.

Metric	Kendall τ
DTW Precision (Gate)	**0.511**
DTWNormGate+FAC	0.509
Step Precision (non-DTW)	0.505
DTW F1 (Gate)	0.503
DTWNormGate	0.503
DTW Precision (Soft)	0.494
DTW F1 (Soft)	0.487
Step Recall (non-DTW)	0.477
DTW Recall (Gate)	0.463
DTWNormSoft	0.460
DTW Avg. Path Score (Gate)	0.420
DTW Avg. Path Score (Soft)	0.401
DTW Recall (Soft)	0.389
ROUGE-2	0.360
ROUGE-L	0.330
BERTScore	0.213

Table 7: Kendall τ correlation with expert evaluation of the reasoning process.

We also use LLM to parse the final answer from the response. We acknowledge that such approaches depend on the prompting strategy, and there is a chance that other parsing and comparison methods (like the use of executable symbolic engine such as SymPy) can produce varying results.

For better reproducibility we share our parsing model details and prompts used to extract final answer and reasoning steps. We used GPT-4.1 model. Results were parsed using OpenAI's output schema, which allowed us to avoid inconsistency in outputs. As a system prompt we used the following text:

```
You are a strict financial reasoning parser.
Your task is to convert a noisy, long,
conversational Chain-of-Thought solution
into a clean and structured JSON object.
Follow these rules exactly:
1. No Calculations
Do NOT compute anything.
Do NOT recompute numbers from the text.
Do NOT round, simplify,
or adjust any numeric value.
2. Number Extraction (CRITICAL)
You must copy numbers
EXACTLY as they appear in the text.
Keep the sign intact
(do NOT remove or change negative signs).
Keep decimal precision exactly as written.
Remove commas, dollar signs,
and percent signs only as formatting,
not as value changes.
3. Step Extraction
Identify only the actual reasoning steps.
Ignore filler text: summaries,
meta-comments, confidence statements,
restatements, chatter.
Produce concise step descriptions
capturing the logic of each step.
For each reasoning step, output an object with:
{
”index”: <step_number>,
”text”: ”<short description of the step>”,
”value”: <numeric result of that step or null>
}
4. How to Determine value
If the step contains a
computation result, copy the
final numeric result in that step.
Usually this is the last
number in the step (e.g., number after ”=”).
If the step has no numeric result,
set ”value”: null.
5. Final Answer Extraction
”final_answer” must be copied exactly
as the last numeric value in the entire solution.
Apply the same numeric-copy rules as above.
6. JSON Output Format
Return ONLY a valid JSON object of the form:
{
”steps”: [...],
”final_answer”: <number>
}
7. Forbidden Behaviors
Do NOT change ”-0.2636” into ”2636.0”.
Do NOT remove negative signs.
Do NOT round 1.514016 to 1.51.
Do NOT compute new numbers.
Do NOT invent numbers not present in the text.
Do NOT choose earlier numbers
if a later number is clearly the result.
Accuracy is measured strictly.
Copy numbers exactly.
```

As user prompt we used:

```
Here is the full solution text
to parse into reasoning steps
and a final answer:
f”{text}\n\n”
Remember: follow the instructions and
return ONLY the JSON object.
```

In case if the steps were parsed with regular expression correctly, the final answer was retrieved using the following prompt:

```
You are given a full solution text.
Ignore step segmentation.
Your ONLY job is to extract
'final_answer' as the last numeric
value in the solution,
following the same numeric-copy
rules from the system prompt.
Return ONLY a JSON object of the form:
”{ \”final_answer\”: <number or null> }\”
Here is the solution text:
```

E Complementary Results

E.1 Results on a Sampled Subset

Due to the high inference cost of Grok 4 Heavy, we evaluate this model on a randomly sampled subset of 200 instances rather than the full benchmark. The results in Table 8 are reported to provide a coarse reference for this cost-limited setting and should not be directly compared with the full-benchmark results in Table 2.

Model	CHAINEVAL	FAC	ROUGE R_2
Grok 4 Heavy	65.64	**81.00**	23.87
GPT-5	**68.42**	69.50	**28.37**
Gemini 2.5 Pro	67.92	73.00	18.85
Fin-R1	56.44	34.00	5.43
Mathstral	64.02	51.00	18.99

Table 8: Performance comparison on a randomly sampled 200-instance subset using CHAINEVAL, final answer correctness (FAC), and ROUGE R_2. These results are reported for reference only and are not directly comparable to full-benchmark results.

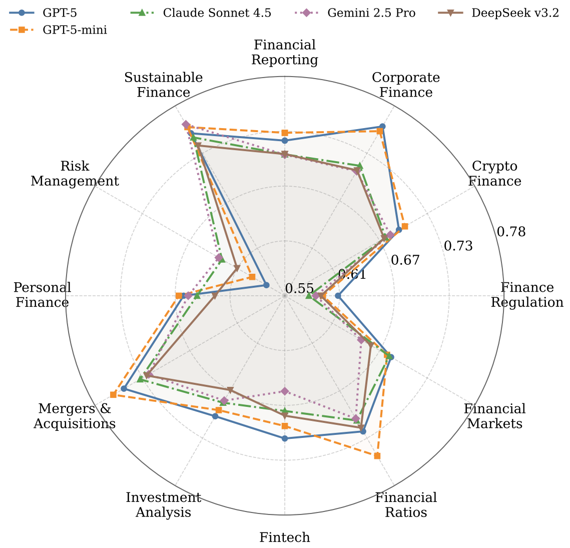


Figure 8: **Domain-level performance of frontier proprietary models.** Radar plot showing CHAINEVAL scores across twelve financial domains for GPT-5, GPT-5-mini, Claude Sonnet 4.5, Gemini 2.5 Pro, and DeepSeek v3.2.

E.2 Complementary Domain-Level Results

We report additional domain-level results for the remaining models grouped by model category. These figures are provided for completeness and transparency, and no further analysis is conducted. Figure 8 reports domain-level CHAINEVAL scores for additional frontier proprietary models, showing broadly consistent performance patterns across financial domains with moderate variation. Figure 9 presents domain-level performance for finance-tuned models, illustrating heterogeneous patterns across domains. Figure 10 shows domain-level results for math-enhanced models, with performance varying across financial domains. Figure 11 reports domain-level performance for general-purpose open models, serving as reference baselines across domains.

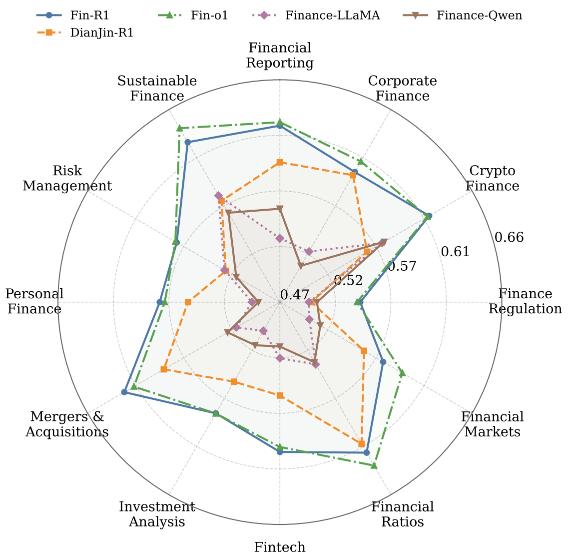


Figure 9: **Domain-level performance of finance-tuned models.** Radar plot showing CHAINEVAL scores across twelve financial domains for Fin-R1, DianJin-R1, Fin-o1, Finance-LLaMA, and Finance-Qwen.

E.3 Difficulty Breakdown under CHAINEVAL

As shown in Figure 12, CHAINEVAL exhibits a more gradual change across difficulty tiers than FAC. This indicates that models may maintain partial step-level alignment on harder instances even when end-task success decreases, consistent with CHAINEVAL assigning partial credit to intermediate reasoning.

F Error Analysis Supplementary

F.1 Error Taxonomy

This appendix defines the error categories used in the diagnostic error analysis. Each model output is assigned to a single category based on the primary criterion violated in the predicted reasoning or final result.

No Error. The reasoning process and final result match the gold computation within the predefined tolerance.

Computational. The applied formula or method is appropriate, but one or more intermediate or final numerical computations are incorrect.

Conceptual. The reasoning violates a problem constraint or applies a financial rule or formula inconsistently with the gold specification.

Sign / Direction. The sign or directional definition of a quantity is inconsistent with the gold formulation, such as reversing subtraction order or treating a decrease as an increase.

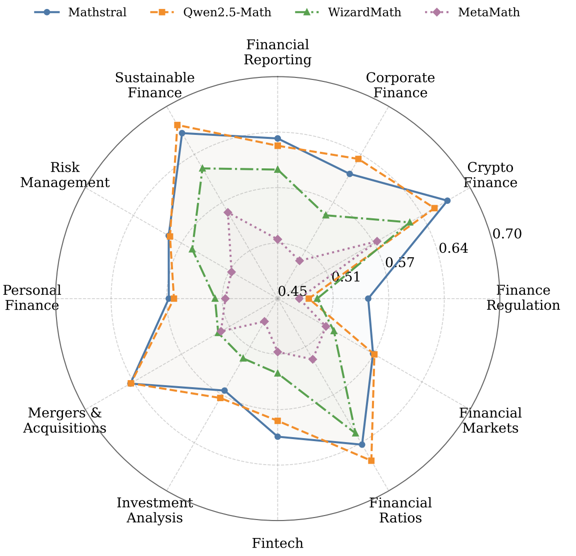


Figure 10: **Domain-level performance of math-enhanced models.** Radar plot showing ChainEval scores across twelve financial domains for Mathstral, Qwen2.5-Math, WizardMath, and MetaMath.

Unit / Precision. Units, scale, or numerical precision are handled inconsistently with the problem specification, for example percent versus decimal or dollars versus millions.

Parsing / Format. The output structure is malformed or inconsistent, preventing reliable parsing or alignment of reasoning steps.

Hallucination. One or more variables, assumptions, or quantities are introduced that are not supported by the original question or provided context.

F.2 Error Examples

Table 9 presents one representative example for each error category used in the diagnostic analysis. Each example consists of a short excerpt from the model output and a concise description of the specific criterion under which the output is labeled as erroneous. The descriptions focus on observable mismatches between the model output and the corresponding gold computation or definition, such as numerical inconsistency, violation of problem constraints, or unsupported quantities. The examples are intended solely to illustrate how the error taxonomy is applied in practice. They do not aim to explain the underlying causes of model behavior, assess error frequency, or attribute failures to model training, architecture, or reasoning capacity.

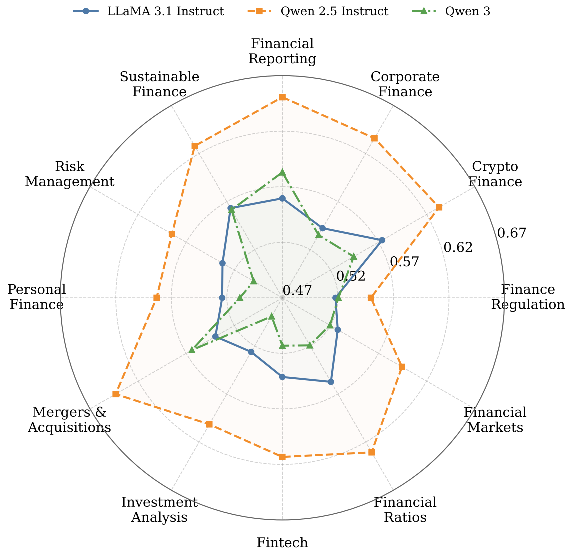


Figure 11: **Domain-level performance of general-purpose open models.** Radar plot showing ChainEval scores across twelve financial domains for LLaMA 3.1 Instruct, Qwen 2.5 Instruct, and Qwen 3.

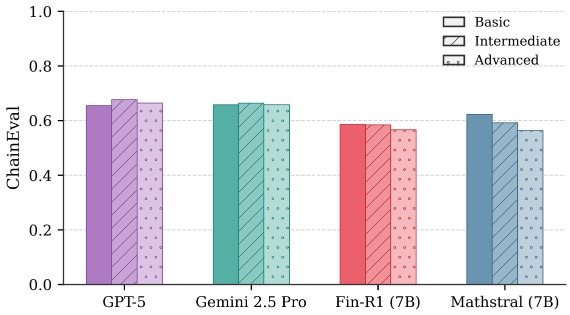


Figure 12: **ChainEval across difficulty levels.** ChainEval scores for representative models on *Basic*, *Intermediate*, and *Advanced* FinChain instances. Compared to FAC in the main text, ChainEval varies more gradually across difficulty, reflecting partial credit from step-level alignment.

Error Type	**Model**	**Model Output (snippet)**	**Error Description**
Computational	GPT-5-mini	“Total cash = 3,592,610.3 × 1.8632 = \$6,693,751.51.”	Incorrect multiplication; correct total is approximately \$6.68M.
Conceptual	GPT-5-mini	“$47408 + x = 0.8(60427 + x) \Rightarrow x = 4668$.”	Violates conservation of total portfolio value during rebalancing.
Sign / Direction	GPT-5-mini	“Change = new - original ≈ -0.123.”	Uses an inconsistent sign convention; gold definition is old - new.
Unit / Precision	GPT-5-mini	“175 (whole credits) ... total rebate = 175 × 2.03 = \$355.25.”	Applies unjustified rounding to a fractional quantity (175.5 credits).
Parsing / Format	Finance-Qwen	“... on the investment at the end of 3 years?”	Output is malformed and cannot be reliably parsed into steps.
Hallucination	Finance-Qwen	“Assume 500M shares ... Market cap = EPS × P/E = \$33.74B.”	Introduces an unsupported quantity and applies an inconsistent valuation formula.

Table 9: Representative error examples with model outputs (expert-audited sample).