FINGYM: SCALING FINANCIAL AGENTS VIA GROUNDED ENVIRONMENT SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Financial agents have the potential to automate complex workflows, such as market analysis, trading, and payment processing. Training such agents requires environments that are diverse, scalable, and financially grounded. However, real financial environments are costly and risky to access, while general environment synthesis may overlook financial states, rules, and operational constraints, producing tasks that require infeasible actions. We introduce FinGym, a framework that translates natural-language requirements into customized financial environments grounded in domain knowledge and financial constraints. FinGym adopts reusable financial entities, market data, and tools into environments with explicit roles, constraints, and workflows. The environment specifications guide both task generation and rubric synthesis, providing executable training tasks for agentic reinforcement learning. Using FinGym, we construct 73 financial environments spanning ten scenarios and synthesize 5,040 verified agentic tasks to train FinGym-4B and FinGym-9B. Across five financial-agent benchmarks, these agents achieve relative improvements of 5.8%–441.9% over their corresponding base models and competitive performance against strong proprietary models. These results highlight FinGym as a scalable and reliable generator of environments and tasks for training financial agents to meet diverse financial requirements. The code and data are available at `https://anonymous.4open.science/r/anony-0D833`.

1 INTRODUCTION

Large language models (LLMs) are rapidly evolving from text generators into agents capable of observing environments, invoking tools, updating plans, and completing complex tasks through multi-turn interactions (Schick et al., 2023; Liu et al., 2024). This transition is particularly consequential for finance (Yu et al., 2024; Zhang et al., 2024; Yang et al., 2024). Financial tasks such as market analysis, investment management, credit risk assessment, and payment processing require the ability to act within stateful, feedback-driven environments governed by financial rules and operational constraints. However, training such agents through reinforcement learning (RL) requires diverse interactive environments that are difficult to obtain and scale in finance.

As shown in Figure 1, using real-world financial environments for training is often restricted by proprietary data and difficult to scale across diverse scenarios. Large-scale exploratory interactions with real financial systems may introduce privacy, compliance, operational, and financial risks (Shabsigh & Boukherouaa, 2023; Bank for International Settlements, 2024; Financial Stability Board, 2024). Recent work offers a promising alternative by synthesizing tasks and environments for general-purpose agent training (Lyu et al., 2026; Song et al., 2026). However, these methods often rely on LLMs to synthesize tasks and environments from high-level descriptions of user needs or scenarios, which may overlook the task states, execution rules, and operational constraints that determine financial task feasibility. In finance, actions can change balances, positions, and risk exposure, affecting which subsequent actions are valid. Without accounting for these dependencies, synthesized tasks may appear plausible yet require infeasible action sequences, undermining the reliability of training signals.

To address these limitations, we introduce **FinGym**, a framework that translates natural-language requirements into customized, grounded financial environments for agentic reinforcement learning. FinGym builds on a library of reusable components that incorporate financial entities, real-world

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

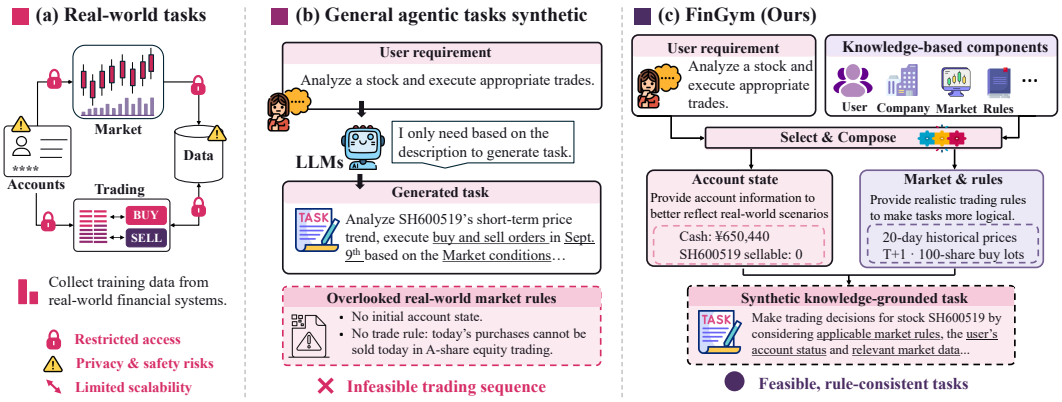

Figure 1: Comparison of financial agent training task sources. (a) Real-world environments face access restrictions, privacy and safety risks, and scalability limitations. (b) General-purpose synthesis may overlook financial states, rules, and operational constraints, producing infeasible tasks. (c) FinGym composes financial components into environments with explicit states, constraints, and workflows to ground task generation.

market data, financial rules, and domain-specific tools, providing both factual context and financial constraints for environment construction. Given a user's description of the desired financial scenario, FinGym selects relevant components to specify the environment's participants, available tools, explicit objectives, constraints, and workflows. Domain experts can further refine these components and specifications through an optional human-in-the-loop process. As illustrated in Figure 1(c), the constructed environment makes relevant financial states, rules, and tool capabilities explicit, helping avoid seemingly plausible tasks that require infeasible action sequences. Within each environment, FinGym synthesizes diverse agentic tasks, verifies their feasibility through teacher-model rollouts, and derives task-specific rubrics from the environment specifications and execution evidence, providing training data and reward criteria for financial agents.

Using FinGym, we construct 73 financial environments spanning 10 major scenarios and synthesize 5,040 agentic training tasks, and use these tasks for reinforcement learning to train two financial agents, **FinGym-4B** and **FinGym-9B**. Evaluation on 5 financial-agent benchmarks and synthetic evaluation tasks covering all 10 scenarios shows substantial improvements over the corresponding base models, with competitive performance against strong proprietary models. Beyond model performance, our analysis shows that FinGym produces better-grounded and more feasible tasks than general-purpose synthesis baselines. These findings support FinGym as a scalable foundation for financial-agent training, capable of tailoring training environments and learning signals to diverse financial requirements.

In summary, our contributions are threefold: (i) FinGym, the first framework for requirement-driven synthesis of financial environments grounded in real-world data and financial rules for agentic RL, (ii) a diverse data resource comprising 73 financial environments and 5,040 verified agentic tasks across ten major financial scenarios, and (iii) two financial agents, FinGym-4B and FinGym-9B, with extensive evaluations demonstrating their generalized capabilities across diverse financial tasks and scenarios.

2 Related Work

2.1 Financial Agents and Environment Simulation

Financial AI spans document reasoning, market analysis, and sequential decision-making. FinQA and ConvFinQA evaluate numerical reasoning (Chen et al., 2021; 2022), while trading agents integrate memory, multimodality, or collaboration (Yu et al., 2025; Zhang et al., 2024; Xiao et al., 2024). AlphaAgentEvo applies agentic RL to multi-turn alpha mining (Tang et al., 2026). Economic simulators model taxation (Zheng et al., 2022; Mi et al., 2024), markets (Dwarakanath et al., 2024), and macroeconomic or social behavior (Li et al., 2024; Piao et al., 2025), while EconGym combines composable roles, equations, and real-data calibration (Mi et al., 2025). Nevertheless, financial

agents use fixed workflows, whereas economic simulators serve policy analysis and benchmarking rather than agent training. Our framework provides data-grounded, executable, rubric-guided, and extensible financial environments.

2.2 Agentic Training Environments and Synthetic Tool-Use Tasks

Early tool-learning methods teach API calls using documentation, synthetic demonstrations, and reasoning–acting trajectories (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024; Yao et al., 2023), but offer little experience with changing states or delayed consequences. Agentic RL requires resettable, stateful environments with reliable feedback. Recent methods address this need: Mock Worlds synthesizes tasks, tools, users, and rubric rewards (Lyu et al., 2026); EnvScaler constructs environments and rule-based validators (Song et al., 2026); Agent World Model provides database-backed state transitions and outcome verification (Wang et al., 2026); and ScaleEnv adds procedural testing and action verification (Tu et al., 2026). As shown in Table 1, existing methods lack either real-data grounding, rubric rewards, broad financial coverage, or durable scaling, and may miss real-world domain dynamics.

Table 1: Comparison of existing agentic training environments and Ours.

Framework	Real Data Integration	Rubric Rewards	Financial Domain	Durable Scaling
EnvScaler	×	×	< 10	×
Mock Worlds	×	✓	< 5	✓
AgentWorldModel	×	×	< 10	×
AlphaAgentEvo	×	✓	1	×
EconGym	✓	×	25	×
Ours	✓	✓	73 (∞)	✓

3 Method

As illustrated in Figure 2, FinGym is a framework that transforms **custom financial environment requirements** into **interactive environments and synthetic training data** for financial agents. Leveraging reusable components grounded in financial knowledge and real-world data, it constructs environments through LLM-based synthesis with optional expert calibration. These environments guide and constrain task generation according to their objectives, available tools, financial rules, and workflows. A teacher model then executes the generated tasks to collect trajectories for task verification and rubric construction. Together, the resulting environments, tasks, verified trajectories, and evaluation rubrics are made available to users for training financial agents.

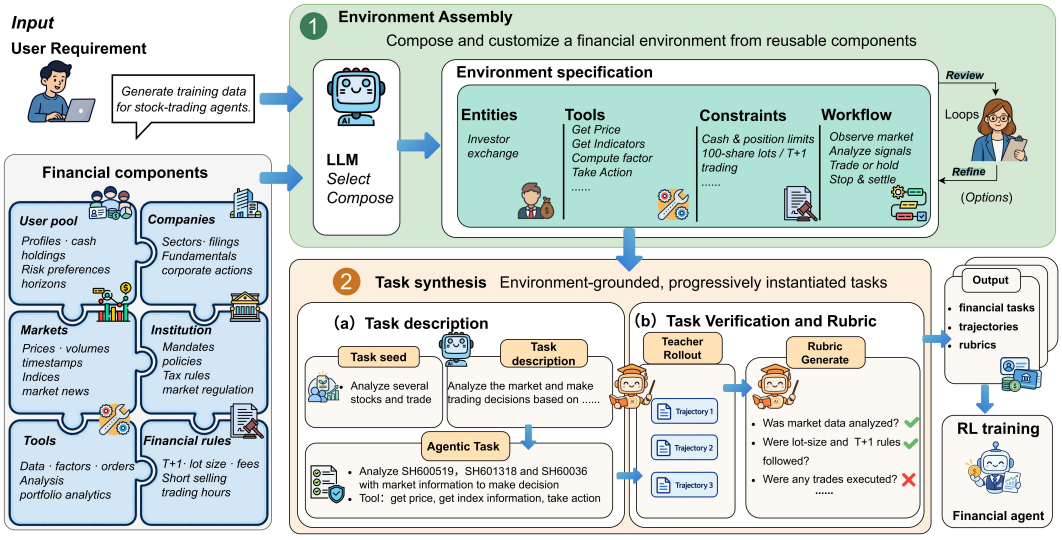

Figure 2: **Overview of FinGym.** Given custom financial requirements, FinGym (1) assembles interactive environments from reusable components grounded in financial knowledge and real-world data, and (2) synthesizes environment-grounded tasks, using teacher rollouts for task verification and rubric construction. The resulting environments, tasks, trajectories, and rubrics support reinforcement learning for financial agents.

3.1 FINANCIAL COMPONENT LIBRARY

To support diverse financial environments, FinGym maintains a shared library of reusable components $\mathcal{C}$ spanning six categories: **individuals**, such as investors, consumers, and employees; **companies**, representing businesses with specified profiles and activities; **regulators**, specifying applicable policies and regulatory requirements; **organizations**, such as banks, insurers, and asset managers; **markets**, providing financial information and historical observations; and **tools**, supporting information retrieval, financial analysis, and simulated business actions. Each component is specified through a structured description of its attributes, business functions, and interaction capabilities. Individuals, Companies, regulators, and organizations are constructed from synthetic profiles, functional specifications, and financial business rules. The tool components form a shared tool pool $\mathcal{A} \subseteq \mathcal{C}$, providing reusable operations for information retrieval, financial analysis, and simulated business actions across financial workflows. Table 2 summarizes the number of subtypes and instances in each category.

Table 2: Details of financial world components.

Category	Subtypes	Instances	Description
Individuals	10	90	User profiles and financial needs
Companies	8	75	Business entities and activities
Regulators	13	60	Policies and regulatory constraints
Organizations	11	80	Institutional financial services
Markets	10	70	Real-world financial information
Tools	23	1473	Callable operations for agent

Real-world financial data are incorporated specifically into market components. These include company profiles, financial data, and public disclosures from SEC EDGAR[1]; fund information, net asset values, performance, holdings, and fees from Tiantian Fund[2]; and historical stock data accessed through Qlib[3]. These records serve as factual context for synthesizing task-specific transactions and evidence, particularly in trading and investment analysis. Together, the components in $\mathcal{C}$ provide reusable entities, financial capabilities, and data-backed market context for environment construction.

3.2 CUSTOM FINANCIAL ENVIRONMENT SYNTHESIS

Given user-specified financial environment requirements r, FinGym provides an LLM with both r and the complete component library $\mathcal{C}$, prompting it to identify the required roles and tools and generate the environment's objectives, constraints, and workflow. These outputs form a structured environment specification $\mathcal{E}_r$:

$$\mathcal{E}_r = \Phi(r, \mathcal{C}) = (\mathcal{C}_r, \mathcal{A}_r, \mathcal{O}_r, \mathcal{B}_r, \mathcal{W}_r)\,, \tag{1}$$

where Φ denotes LLM-based environment synthesis, $\mathcal{C}_r \subseteq \mathcal{C} \setminus \mathcal{A}$ and $\mathcal{A}_r \subseteq \mathcal{A}$ are the selected roles and tools, and $\mathcal{O}_r$, $\mathcal{B}_r$, and $\mathcal{W}_r$ specify the environment's objectives, business constraints, and workflow. The synthesis process supports optional *human-in-the-loop* refinement. Users or domain experts can further refine the environment by adding, removing, or replacing components and adjusting their attributes to better reflect the intended scenario.

The resulting environment encodes financial domain knowledge and business rules as explicit conditions for task synthesis, guiding the generation of tasks consistent with the scenario's evidence requirements, information boundaries, and available tools. Moreover, this design allows constructed environments to be reused for further task synthesis without rebuilding them from scratch.

3.3 ENVIRONMENT-GUIDED TASK SYNTHESIS

Three-Stage Task Synthesis. Given the synthesized environment $\mathcal{E}_r$, an LLM first generates a collection of *task themes*, each expressed as a short keyword phrase identifying a financial problem, such as "disambiguating revisions to trade data". Each theme is then combined with $\mathcal{E}_r$ and feed into an LLM to generate a *task specification* s, describing the detailed task requirements, the requested outcome, and the evidence needed to complete the task. The LLM subsequently receives s and the tool pool $\mathcal{A}_r$ with its tool descriptions to instantiate a concrete agentic task $q = (u, \mathcal{A}_q)$, where u is the task instruction to the agent and $\mathcal{A}_q \subseteq \mathcal{A}_r$ contains the tools available for completing the

[1] https://www.sec.gov/search-filings/edgar-application-programming-interfaces

[2] https://fund.eastmoney.com/

[3] https://github.com/microsoft/qlib

task. This multi-stage process supports diverse task generation while grounding task requirements in financial domain constraints and available tool capabilities.

Task Verification and Rubric Synthesis. To assess task feasibility and collect execution evidence for rubric synthesis, a teacher model first performs rollouts on candidate tasks, and an LLM verifier checks task executability, goal completion, and compliance with scenario constraints. We retain tasks with at least one verified successful trajectory, denoting their successful and available unsuccessful rollouts by $\mathcal{T}_q^+$ and $\mathcal{T}_q^-$, respectively. These trajectories, together with environment specification $\mathcal{E}_r$, guide task-specific rubric synthesis:

$$\mathcal{R}_q = \mathcal{H}\big(q, \mathcal{E}_r, \mathcal{T}_q^+, \mathcal{T}_q^-\big), \tag{2}$$

where $\mathcal{H}$ denotes the rubric extraction mapping. $\mathcal{E}_r$ provides domain-specific guidance on evidence requirements, tool-use dependencies, and task completion criteria, grounding the rubrics in the environment's financial procedures. The resulting output comprises *verified financial tasks*, *successful teacher trajectories*, and *knowledge-grounded evaluation rubrics* for agent training.

3.4 Financial Agent Training

During online reinforcement learning, a rubric judge assigns each policy rollout τ a reward $r_q(\tau) = J(\tau; \mathcal{R}_q)$. We optimize the expected reward to improve financial reasoning and tool use:

$$\max_{\theta} \ \mathbb{E}_{q\sim\mathcal{D}}\mathbb{E}_{\tau\sim\pi_\theta(\cdot|u,\mathcal{A}_q;\mathcal{E}_r)}\big[r_q(\tau)\big], \tag{3}$$

where π_θ denotes the agent policy, $\mathcal{D}$ denotes the dataset of verified tasks synthesized across multiple financial environments.

4 Experiments

4.1 Experimental Setup

Data Synthesis. We synthesize training tasks from *10* primary scenarios, spanning *Market Analysis*, *Investment Products*, *Portfolio Management*, *Wealth Advisory*, *Investor Services*, *Credit Operations*, *Insurance Operations*, *Transaction Services*, *Compliance & Governance*, and *Securities Trading*. These primary scenarios are further instantiated into *73* fine-grained environments. During task construction, We use GPT-5.5 for financial environment and task synthesis and as the teacher model for task execution and trajectory collection. For human refinement, three PhD students with backgrounds in AI and finance volunteered to review each synthesized environment and discuss any necessary revisions to reach agreement. Table 3 summarizes the resulting training-task statistics. This process yields a total of *5,040* tasks, with each task involving an average of *4.92* tools and *5.61* model–environment interactions.

Table 3: Statistics of FinGym.

Statistic	Value
Total #Environments	73
Total #Tasks	5,040
Avg. #Tools Per Task	4.92
Avg. #Interactions Per Task	5.61

Training. We train Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026) with reinforcement learning using the GRPO algorithm (Shao et al., 2024) within the VERL framework (Sheng et al., 2025). We use 8×NVIDIA H100 GPUs and each optimizer update contains a global prompt batch of 32 distinct tasks and 16 rollouts per prompt, resulting in 512 trajectories per update. The clipping ratio is constrained between 0.2 and 0.28.

Evaluation. We evaluate on five financial-agent benchmarks. FinMCP-Bench (Zhu et al., 2026), FinVault (Yang et al., 2026), and Finova (Zheng et al., 2025) are short-horizon: FinMCP-Bench tests multi-step financial tool use and parameter selection, FinVault measures task completion and resistance to unsafe or adversarial actions, and Finova assesses general capabilities. The long-horizon StockBench (Chen et al., 2025) and InvestorBench (Li et al., 2025) evaluate multi-stock portfolio optimization and stock-level directional decisions over time, respectively. We follow each benchmark's official protocol and metrics.

Baselines. We compare models trained with FinGym against the following baselines. We first evaluate their unadapted backbones, Qwen3.5-4B and Qwen3.5-9B, to measure the direct gain from FinGym training. We further evaluate the proprietary models GPT-5.6 Sol (OpenAI, 2026), Gemini 3.1 Pro (Google DeepMind, 2026), Claude Opus 4.8 (Anthropic, 2026), and DeepSeek-V4-Pro (DeepSeek-AI, 2026). We additionally compare against models produced by open-source environment-scaling methods, including EnvScaler-Qwen3-8B (Song et al., 2026), Agent-World-Model-Qwen3-8B (Wang et al., 2026), and TOUCAN-7B (Xu et al., 2025). All models are evaluated with the same prompt, function-calling format.

4.2 Main Results

Table 4 compares our 4B and 9B models with baseline models across five financial-agent benchmarks. Detailed evaluation metrics and results are provided in Appendix B.2.

Table 4: Performance on financial agent benchmarks. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. "Ovrl." denotes the overall benchmark score.

Model	FinMCP-Bench			Finova	FinVault			StockBench			InvestorBench		
	TF1(↑)	EMR(↑)	Ovrl.(↑)	Ovrl.(↑)	ASR(↓)	VR(↓)	Ovrl.(↓)	RT(↑)	DDN(↑)	Ovrl.(↑)	CR(↑)	MDD(↓)	Ovrl.(↑)
Closed-source Models													
Gemini-3.1Pro	56.80	22.43	39.62	69.00	11.57	25.23	18.40	0.03	-11.57	0.94	23.28	10.74	1.62
GPT-5.6-Sol	51.24	20.59	35.92	69.34	11.92	30.84	21.38	-1.22	-4.58	-4.37	25.69	9.80	1.78
Claude Opus 4.8	40.16	19.74	29.95	65.62	5.49	18.69	12.09	-0.37	-11.34	0.41	25.68	8.72	1.82
DeepSeek-V4-pro	49.44	21.70	35.57	64.59	17.06	46.73	31.90	3.92	-12.88	4.25	23.94	9.46	1.68
Open-Source Foundation Models													
Qwen3.5-4B	48.86	15.91	32.39	55.69	22.31	55.14	38.73	-0.20	-13.40	0.74	9.02	12.74	0.78
Qwen3.5-9B	51.66	16.11	33.89	60.64	25.70	57.01	41.36	0.07	-13.55	1.19	14.99	10.82	1.11
Llama-3.1-8B	30.93	10.47	20.70	29.55	41.59	79.44	60.52	0.74	-11.68	2.06	17.87	11.12	1.30
Open-Source Environment Scaling Methods													
EnvScaler-8B	45.84	16.95	31.40	52.22	50.35	79.44	64.90	1.00	-15.94	2.19	22.03	10.97	1.50
AWM-8B	43.57	17.94	30.76	55.28	50.82	77.57	64.20	0.29	-13.21	1.18	19.41	11.06	1.42
TOUCAN-7B	36.57	12.21	24.39	38.87	43.46	76.64	60.05	0.36	-12.53	1.44	15.89	13.38	1.20
Our Models													
FinGym-4B	51.35	18.79	**35.07** +8.3%	**61.40** +10.3%	13.20	40.19	**26.70** +31.1%	3.36	-12.01	**4.01** +441.9%	22.39	9.65	**1.55** +98.7%
FinGym-9B	53.89	19.14	**36.52** +7.8%	**64.17** +5.8%	12.38	35.51	**23.95** +42.1%	3.78	-9.78	**5.43** +356.3%	22.42	8.25	**1.77** +59.5%

FinGym shows strong financial-agent capabilities. On FinMCP-Bench, which evaluates general financial tool-use capabilities, FinGym-9B achieves a TF1 score of 53.89, exceeding GPT-5.6-Sol and DeepSeek-V4-pro by 5.17% and 9.00%, respectively. Beyond short-horizon tasks, our agents also deliver strong results in more complex long-horizon trading scenarios. On StockBench, FinGym-4B achieves an overall Sortino ratio of 4.01, approaching the strongest evaluated closed-source result of 4.25. FinGym-9B further improves this score to 5.43, surpassing that result by 1.18 points. On InvestorBench, our 9B agent achieves an overall Sharpe ratio of 1.77, improving over its Qwen3.5-9B baseline by 0.66 points and approaching the best evaluated closed-source score of 1.82. Together, these results highlight the effectiveness of our financial environment training across different task horizons, enabling compact open-source agents to match or exceed strong closed-source baselines on selected financial capabilities.

FinGym surpasses open-source environment-scaling methods on financial benchmarks. On FinMCP-Bench, our 4B agent achieves an overall score of 35.07, exceeding EnvScaler-8B, AWM-8B, and TOUCAN-7B by 11.69%, 14.01%, and 43.79%, respectively. On Finova, it scores 61.40, outperforming the strongest of these baselines, AWM-8B, by 11.07%. On FinVault, where lower scores are better, it achieves an overall score of 26.70, a substantial 55.54% reduction relative to the strongest environment-scaling baseline, TOUCAN-7B. Despite their larger model sizes, none of these baselines outperform the Qwen3.5-4B base model on either FinMCP-Bench or Finova. Taken together, these results indicate that general-purpose environment scaling does not necessarily translate into stronger financial-agent capabilities across diverse financial tasks.

FinGym improves performance on synthesized financial test tasks. We further evaluate the models on our synthesized test tasks covering ten financial scenarios. We compare FinGym-9B with its Qwen3.5-9B base model, a model trained from the same backbone on AWM-generated tasks, and GPT-5.6-Sol. As shown in Figure 3, our model outperforms both the base model and the model trained on AWM tasks on the displayed scenarios and the average across all ten scenarios, indicating that task synthesis grounded in financial-world constraints provides more effective training signals for learning financial capabilities. Furthermore, our model surpasses GPT-5.6-Sol in wealth management and investment and achieves comparable performance in credit risk, demonstrating that the diverse data synthesized through our environments can enable the model to reach the performance of a strong closed-source model in selected financial scenarios.

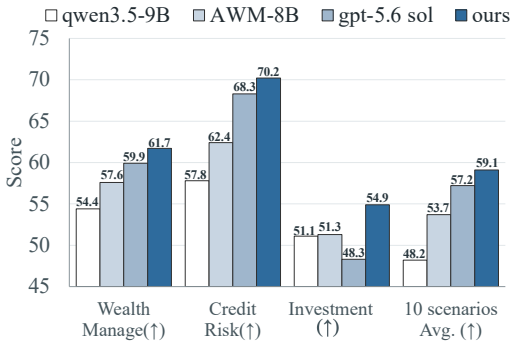

Figure 3: Model performance on synthesized financial test tasks.

4.3 Benefits of Financially Grounded Task Synthesis

Task quality assessment. We compare tasks generated by FinGym, LLM-only baseline (LLM Only), and AWM. LLM Only generates tasks directly from financial scenario descriptions. For each method, we randomly sample ten tasks from each of the 73 financial sub-scenarios, yielding 730 tasks per method. Multiple LLM judges evaluate these tasks using a shared rubric covering task coherence, financial validity, and execution readiness. As shown in Table 5, FinGym achieves higher scores across these dimensions, with stronger alignment between task objectives and initial states, better adherence to financial constraints, and more adequate tool support. These results suggest that grounding task synthesis in financial components and scenario constraints helps produce more coherent and actionable tasks than generation driven solely by scenario descriptions.

Table 5: Task quality comparison.

Metric	LLM Only	AWM	Ours
Task Coherence (↑)			
Goal Clarity	3.61	4.67	**4.81**
State–Goal Consistency	2.24	3.64	**4.31**
Financial Validity (↑)			
Constraint Consistency	2.45	3.12	**4.64**
Information Boundaries	3.12	4.67	**4.73**
Execution Readiness (↑)			
Tool Sufficiency	3.11	3.97	**4.52**
Task Feasibility	3.03	3.51	**4.54**

Impact of task synthesis method on agent learning. Table 6 shows that training on tasks generated by our method consistently outperforms both LLM-only synthesis and the synthesis method used in AWM across all five financial benchmarks, with particularly pronounced advantages on trading tasks. Together with the task quality assessment, these results suggest that generic synthesis methods provide less effective training signals for financial agents, while grounding tasks in financial states and domain-specific constraints better supports downstream agent learning. We provide an illustrative example case in Appendix B.1.

Table 6: Effect of the synthetic methods.

Benchmark	LLM Only	AWM	Ours
FinMCP-Bench (↑)	32.89	33.94	**35.07**
Finova Bench (↑)	57.14	59.32	**61.40**
FinVault (↓)	36.51	31.75	**26.70**
StockBench (↑)	0.64	0.72	**4.01**
InvestorBench (↑)	0.61	0.93	**1.55**

4.4 Scaling within FinGym

We examine within-domain scaling in FinGym by progressively expanding financial environment coverage. We construct nested training subsets containing 10, 20, 50, and 73 financial environments and evaluate the resulting models on five financial benchmarks and observe an overall improvement as more environments are included. As shown in Figure 4(A), these results suggest that training data covering a broader range of financial workflows supports downstream financial capabilities, including task completion and trading decision-making.

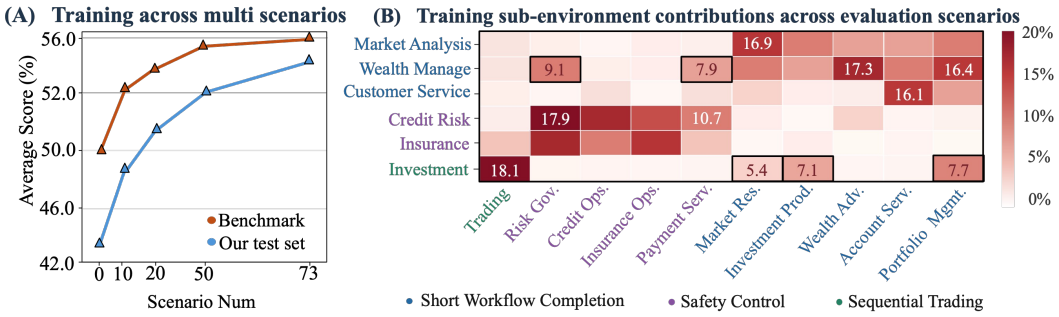

Figure 4: **Cross-scenario contributions and scaling.** **(A)** Average performance on our cross-scenario test set and overall score from three multi-scenarios benchmarks and our test set as training scenario coverage increases. **(B)** Relative performance gains (%) of the full model over models trained without each column's training sub-environment, evaluated across our test scenarios.

To further investigate how tasks from different financial scenarios contribute to another, we conduct leave-one-scenario-out experiments across ten training scenarios, comparing each ablated model with FinGym-4B on six evaluation categories. Figure 4(B) reveals contributions both within and across task families. Short-workflow training in market research, investment products, and portfolio management benefits long-horizon trading, yielding relative performance gains of 5.4%, 7.1%, and 7.7%, respectively, over models trained without the corresponding scenario. Similarly, risk-governance and payment-service training improves wealth-management workflows by 9.1% and 7.9%, respectively, suggesting complementary training signals across financial tasks.

Takeaway. Short-horizon tasks can improve tool-use reliability in long-horizon trading, supporting consistent information gathering for trading decisions, while safety-oriented tasks can reduce tool misuse in other workflows.

4.5 Ablation and Discussion

Ablation. We examine how grounding in environment construction and reward design contributes to agent performance. Table 7 reports cumulative ablations that successively remove components supporting the refinement of financial environment specifications, the translation of environment objectives and constraints into rewards, and the incorporation of domain knowledge into environment construction. Each successive removal degrades performance across all five benchmarks. These cumulative declines support the value of grounding both training environments and reward criteria in financial knowledge and constraints.

Table 7: Cumulative component ablation of FinGym.

Training setting	FinMCP (↑)	Finova (↑)	FinVault (↓)	StockBench (↑)	InvestorBench (↑)
FinGym-4B	**35.07**	**61.40**	**26.70**	**4.01**	**1.55**
w/o Human-in-the-loop	34.64	60.43	27.64	3.11	1.31
w/o Rubric guidance	33.81	58.17	33.45	2.44	1.01
w/o Financial knowledge	32.89	57.14	36.51	0.64	0.61

How do financially grounded rubrics support training? Beyond final benchmark performance, we compare tool-call accuracy throughout training under generic LLM-generated rubrics and our reward framework, which combines rubrics grounded in fine-grained financial descriptions with executable verification. As shown in Figure 5, our framework achieves higher average tool-call accuracy at every evaluated checkpoint, with the gap generally widening as training progresses. The refined rubrics also yield more consistent tool-call performance over training, suggesting improved robustness. Together, these results indicate that financially grounded rubric guidance and executable verification support sustained and more consistent improvement.

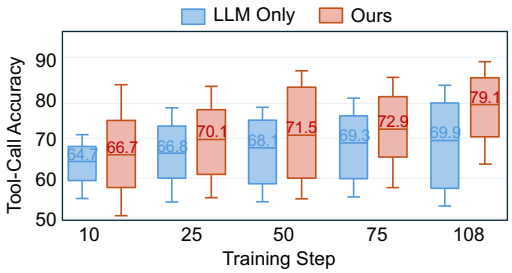

Figure 5: Effect of the rubrics in training steps.

Takeaway. Providing more fine-grained financial guidance for rubric generation improves tool selection accuracy in financial tasks, thereby further enhancing agentic task performance.

Are real-world financial data valuable? Real-world data, such as stock market records, play an important role in grounding financial knowledge constraints. However, their specific contribution to market-sensitive tasks, such as investment, remains unclear.

To examine this effect, we compare models trained with and without real-world financial-data constraints on InvestorBench, where investment tasks span 149 trading days and require sustained decision-making over a long horizon. As shown in Figure 6, incorporating real-world data improves trading performance across all four evaluated stocks, raising the average score from 1.05 to 1.55 (+47.62%). The largest gains occur on UVV and HON, reaching 79.37% and 58.99%, respectively, while JNJ and MSFT improve by 16.95% and 8.42%. These consistent gains over an extended trading horizon suggest that grounding synthetic training environments in real financial records supports more effective long-term decision-making in market-sensitive tasks.

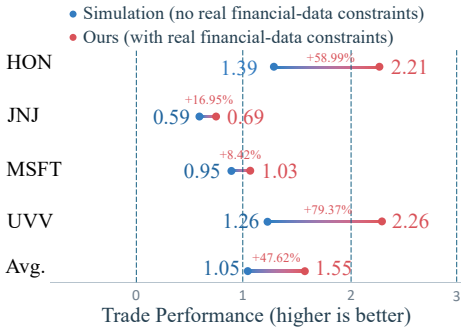

Figure 6: Trading performance across different stocks, measured by the Sharpe ratio.

Takeaway. Real-world financial data grounding improves agents' long-term decision-making and performance on market-sensitive tasks.

4.6 Requirement-Driven Expansion to Unseen Domains

Given the diversity of financial requirements, we further evaluate FinGym's ability to expand into unseen domains. Vending and supply-chain components are excluded from the initial FinGym construction and preceding evaluations. We therefore use EcoGym Vending (Hu et al., 2026), held out during environment construction, to evaluate requirement-driven expansion into this previously unseen domain.

Table 8: Performance of the unseen domains.

Training setting	Final worth (↑)	New Scenarios
Qwen3.5-4B	806.82	0
AWM Synthesis	815.91	0
Ours (Ori. 73 environments)	907.68	0
Ours (73+3)	**1997.92** +147.6%	3

As shown in Table 8, training on FinGym's original 73 environments improves final worth by 12.5% over Qwen3.5-4B in the unseen vending domain, while AWM Synthesis yields only marginal gains. Requirement-driven expansion adds three scenarios and raises final worth to 1997.92, a 147.6% improvement over the base model, demonstrating both cross-domain generalization and effective adaptation through environment expansion into unseen domains.

Takeaway. FinGym supports cross-domain generalization and further adaptation to unseen domains through requirement-driven environment expansion.

5 Conclusion

This paper introduces FinGym, a framework that translates natural-language requirements into customized financial environments for agentic reinforcement learning. By composing reusable financial components into environments with explicit roles, constraints, and workflows, FinGym grounds task generation and rubric synthesis in financial domain knowledge. Trained on 5,040 verified tasks synthesized across 73 financial environments, FinGym-4B and FinGym-9B achieve consistent improvements over their base models across five financial-agent benchmarks and competitive performance against strong proprietary models. Further analyses show that financial grounding improves the quality of synthesized tasks and downstream learning. We hope that FinGym's scalable synthesis framework, synthesized environments and tasks, and trained models will serve as useful resources for the community to advance financial-agent research and training.

AI USE STATEMENT

During this revision, we used Codex (GPT-5.6 sol) to assist with manuscript editing, LaTeX preparation, and consistency checks. The LLMs evaluated as simulator backbones, compared methods, and synthetic components are documented separately in the experimental setup. We reviewed all AI-assisted work and take full responsibility for the final manuscript and reported results.

ETHICS STATEMENT

FinGym is intended to support research on financial-agent training. Synthesized tasks and trained agents may inherit errors or biases from the underlying data, financial specifications, and LLMs. Financial grounding and task verification do not guarantee safe or compliant behavior in real-world financial systems. Practical deployment requires additional validation, human oversight, and appropriate safeguards for privacy and financial safety. The reported results should not be interpreted as investment advice or evidence of deployment readiness.

REPRODUCIBILITY STATEMENT

The main text describes the financial component library, environment construction, task verification and rubric synthesis, reinforcement learning setup, and evaluation protocols. The appendix provides prompts for task generation, execution, and verification, together with task examples, detailed benchmark results, and additional analyses of data and tool coverage. Code and data are available in the anonymous repository at `https://anonymous.4open.science/r/anony-0D833` to support reproducibility and further research.

REFERENCES

Anthropic. Introducing claude opus 4.8. Anthropic, 2026. URL https://www.anthropic.com/news/claude-opus-4-8.

Bank for International Settlements. Artificial intelligence and the economy: Implications for central banks. Annual economic report 2024, chapter iii, Bank for International Settlements, 2024. URL https://www.bis.org/publ/arpdf/ar2024e3.htm.

Yanxu Chen, Zijun Yao, Yantao Liu, Amy Xin, Jin Ye, Jianing Yu, Lei Hou, and Juanzi Li. StockBench: Can LLM agents trade stocks profitably in real-world markets?, 2025. URL https://arxiv.org/abs/2510.02209.

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. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, 2021. doi: 10.18653/v1/2021.emnlp-main.300.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 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*, pp. 6279–6292, 2022. doi: 10.18653/v1/2022.emnlp-main.421.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Kshama Dwarakanath, Svitlana Vyetrenko, Peyman Tavallali, and Tucker Balch. ABIDES-Economist: Agent-based simulation of economic systems with learning agents, 2024.

Financial Stability Board. The financial stability implications of artificial intelligence. Technical report, Financial Stability Board, 2024. URL https://www.fsb.org/2024/11/the-financial-stability-implications-of-artificial-intelligence/.

Google DeepMind. Gemini 3.1 Pro model card. Google DeepMind, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Xavier Hu, Jinxiang Xia, Shengze Xu, Kangqi Song, Yishuo Yuan, Guibin Zhang, Jincheng Ren, Boyu Feng, Li Lu, Tieyong Zeng, Jiaheng Liu, Minghao Liu, Yuchen Elenor Jiang, Wei Wang, He Zhu, and Wangchunshu Zhou. EcoGym: Evaluating LLMs for long-horizon plan-and-execute in interactive economies, 2026. URL https://arxiv.org/abs/2602.09514.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Koduvayur Subbalakshmi, Jimin Huang, Lingfei Qian, Xueqing Peng, Qianqian Xie, and Jordan W. Suchow. InvestorBench: A benchmark for financial decision-making tasks with LLM-based agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2509–2525, 2025. URL https://aclanthology.org/2025.acl-long.126/.

Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. EconAgent: Large language model-empowered agents for simulating macroeconomic activities. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 15523–15536, 2024. doi: 10.18653/v1/2024.acl-long.829.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yuanjie Lyu, Chengyu Wang, Lei Shen, Jun Huang, and Tong Xu. Mock worlds, real skills: Building small agentic language models with synthetic tasks, simulated environments, and rubric-based rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 12529–12545, 2026. doi: 10.18653/v1/2026.acl-long.570.

Qirui Mi, Siyu Xia, Yan Song, Haifeng Zhang, Shenghao Zhu, and Jun Wang. TaxAI: A dynamic economic simulator and benchmark for multi-agent reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1390–1399, 2024.

Qirui Mi, Qipeng Yang, Zijun Fan, Wentian Fan, Heyang Ma, Chengdong Ma, Siyu Xia, Bo An, Jun Wang, and Haifeng Zhang. EconGym: A scalable AI testbed with diverse economic tasks. In *Advances in Neural Information Processing Systems*, 2025.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. OpenAI, 2026. URL https://openai.com/index/gpt-5-6/.

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

Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, et al. AgentSociety: Large-scale simulation of LLM-driven generative agents advances understanding of human behaviors and society, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Ghiath Shabsigh and El Bachir Boukherouaa. Generative artificial intelligence in finance: Risk considerations. IMF Fintech Note 2023/006, International Monetary Fund, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. ScaleEnv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training, 2026.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. TradingAgents: Multi-agents LLM financial trading framework, 2024.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. TOUCAN: Synthesizing 1.5m tool-agentic data from real-world MCP environments, 2025. URL https://arxiv.org/abs/2510.01179.

Hongyang Yang, Boyu Zhang, Neng Wang, Cheng Guo, Xiaoli Zhang, Likun Lin, Junlin Wang, Tianyu Zhou, Mao Guan, Runjia Zhang, and Christina Dan Wang. FinRobot: An open-source AI agent platform for financial applications using large language models. *arXiv preprint arXiv:2405.14767*, 2024. URL https://arxiv.org/abs/2405.14767.

Zhi Yang, Runguo Li, Qiqi Qiang, Jiashun Wang, Fangqi Lou, Mengping Li, Dongpo Cheng, Rui Xu, Heng Lian, Shuo Zhang, Xiaolong Liang, Xiaoming Huang, Zheng Wei, Zhaowei Liu, Xin Guo, Huacan Wang, Ronghao Chen, and Liwen Zhang. FinVault: Benchmarking financial agent safety in execution-grounded environments, January 2026. URL https://arxiv.org/abs/2601.07853v1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. FinMem: A performance-enhanced LLM trading agent with layered memory and character design. *Proceedings of the AAAI Symposium Series*, 3(1):595–597, 2024. doi: 10.1609/aaaiss.v3i1.31290.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W. Suchow, Denghui Zhang, and Khaldoun Khashanah. FinMem: A performance-enhanced LLM trading agent with layered memory and character design. *IEEE Transactions on Big Data*, 11(6):3443–3459, 2025. doi: 10.1109/TBDATA.2025.3593370.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, Longtao Zheng, Xinrun Wang, and Bo An. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4314–4325, 2024. doi: 10.1145/3637528.3671801.

Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. The AI economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science Advances*, 8(18):eabk2607, 2022. doi: 10.1126/sciadv.abk2607.

Yanjun Zheng, Xiyang Du, Longfei Liao, Xiaoke Zhao, Zhaowen Zhou, Jingze Song, Bo Zhang, Jiawei Liu, Xiang Qi, Zhe Li, Zhiqiang Zhang, Wei Wang, and Peng Zhang. Agentar-Fin-R1: Enhancing financial intelligence through domain expertise, training efficiency, and advanced reasoning, 2025. URL https://arxiv.org/abs/2507.16802.

Jie Zhu, Yimin Tian, Boyang Li, Kehao Wu, Zhongzhi Liang, Junhui Li, Xianyin Zhang, Lifan Guo, Feng Chen, Yong Liu, and Chi Zhang. FinMCP-Bench: Benchmarking LLM agents for real-world financial tool use under the model context protocol. In *2026 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2026. URL https://arxiv.org/abs/2603.24943.

A More Experiments

A.1 Does increasing environment diversity keep improving performance?

We examine two aspects of environment construction: the time span of real financial data and the size of the financial tool pool. Figure 7 (a) compares trading performance using data spanning 1, 3, 6, and 12 months.

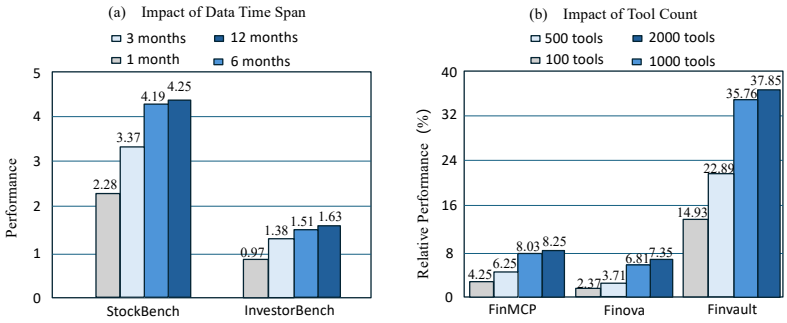

Figure 7: **Sensitivity to data time span and tool pool size. (A)** Trading performance with different spans of real financial data. **(B)** Relative performance gains across financial benchmarks with different tool pool sizes.

Both StockBench and InvestorBench improve as the time span expands, but extending it beyond six months yields smaller additional gains. Figure 7 (b) varies the tool pool from 100 to 500, 1000, and 2000 tools. Performance improves across FinMCP-Bench, Finova, and FinVault, with gains slowing beyond 1000 tools. These results suggest that broader temporal and tool coverage benefits training, while further expansion exhibits diminishing returns. The observed saturation supports using a moderate data span and tool pool to capture most of the measured gains without requiring the largest tested configurations.

A.2 Our test set

We construct a scenario-based financial agentic task set covering common tool-use tasks, financial safety risk identification, and trading decision-making in Table 14.

B Details of Experiments

B.1 Task Cases Comparison

Case A: Task generation using only an LLM based on the scenario description

Task Background and Objective
You are a stock trading analyst. Please analyze the short-term price movement of `SH600519`. Based on your analysis, complete buy and sell trading operations within one day, and then report the trading result.

Available Tool Pool:
`get_trade_information`: Pass in a stock symbol to obtain recent information about the stock.
`take_buy_action`: Pass in a number and a stock symbol to execute a buy operation.
`take_sell_action`: Pass in a number and a stock symbol to execute a sell operation.

Trading Decision and Execution Rules (rule): —{`Supplementary rule constraints`}
The model should first analyze the stock information before trading.
Calling `TakeAction(type="buy")` will buy `100` shares of `SH600519`;
Calling `TakeAction(type="sell")` will sell `100` shares of `SH600519`.
After the actions are completed, calculate the net profit of this trade based on the execution prices and transaction fees returned by the tools, and report the final cash and holdings.
(Source: The rules were constructed by a large language model based on the information in the generated task.)

Although the task is linguistically consistent with a trading scenario, it ignores both the fact that the initial account has no holding in this stock and the T+1 constraint that shares bought on the current day cannot be sold on the same day. Consequently, the required trading sequence is not executable.

Case B: Ours—task generation combining real data and workflow constraints

Task Background and Objective:
You are a stock trading analyst. The current trading date is `April 9, 2025`. Based only on historical market data visible at the decision time, determine whether to establish a short-term position in `SH600519`, and execute the decision within the limits of the account funds and trading rules.

Initial Account—state information supplement
The account has `RMB 650,440` in available cash, holds `14,100` shares of `SZ002156` and `900` shares of `SH600036`, and holds no `SH600519`. Therefore, the initial sellable quantity of this stock is zero. This task trades only `SH600519`; all other holdings remain unchanged.

Available Tool Pool—supporting financial market data
`get_price_data`: Obtain the historical prices, trading volume, and necessary technical indicators for the most recent 20 trading days of `SH600519`.
`get_macro_indicators`: Understand the market trend and volatility over the same period.
`compute_alpha_factor`: Calculate short-term momentum or price-volume factors. Use only data visible at the decision time; jointly analyze the trend, volatility, price-volume relationship, and downside risk to determine whether the position is worth establishing.

Trading Decision and Execution Rules (rule)—{`Supplementary rule constraints`}
If the evidence supports buying, determine the number of shares to buy in units of 100 based on available cash, the execution price, transaction fees, and slippage, and call `TakeAction(type="buy")` to execute it. If the evidence is insufficient or the account cash cannot support the trade, the account may remain unchanged. (Source: the subject attributes required in each trading task; the LLM obtains the attributes, and the trading rules define the rules that constrain this condition.)
Trading follows the `T+1` rule: shares bought on the current day cannot be sold on the same day. Because the account initially has no sellable position in `SH600519`, this task does not execute a same-day sale of this stock. If a purchase succeeds, the new holding is retained until the end of the current trading day; this task does not execute operations for subsequent trading days in advance. (Source: the trade rules defined under the secondary scenario.)

Because the initial account has no sellable position in this stock and the workflow specifies that a position newly added on the current day cannot be sold, the task excludes an immediate sale after a same-day purchase.

Core distinction: Linguistically, the task conforms to a trading scenario. The LLM-generated task considers only the goal of buying and selling. However, because the initial information contains no constraints, it later ignores whether the initial account has a position in the stock and ignores the `T+1` constraint that a stock bought on the current day cannot be sold on the same day. As a result, the required transaction sequence is not executable in a real trading scenario.

B.2 Full Tables of 5 Benchmarks

We provide the complete evaluation results on five financial agent benchmarks, including FinMCP-Bench, Finova, FinVault, StockBench, and InvestorBench. Table 9, Table 10, Table 11, Table 12, and Table 13 report detailed metric-level performance for closed-source models, open-source foundation models, environment scaling baselines, and our models.

Table 9: Performance on FinMCP-Bench. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. "Ovrl." denotes the average of TF1 and EMR.

Model	TP (↑)	TR (↑)	TF1 (↑)	EMR (↑)	Ovrl. (↑)
Closed-source Models					
Gemini-3.1Pro	51.81	62.84	56.80	22.43	39.62
GPT-5.6-Sol	56.35	46.98	51.24	20.59	35.92
Claude Opus 4.8	69.36	28.26	40.16	19.74	29.95
DeepSeek-V4-pro	38.02	70.66	49.44	21.70	35.57
Open-Source Foundation Models					
Qwen3.5-4B	43.78	55.27	48.86	15.91	32.39
Qwen3.5-9B	49.39	54.15	51.66	16.11	33.89
Llama-3.1-8B	33.42	28.78	30.93	10.47	20.70
Open-Source Environment Scaling Methods					
EnvScaler-8B	46.32	45.38	45.84	16.95	31.40
AWM-8B	42.60	44.59	43.57	17.94	30.76
TOUCAN-7B	40.71	33.19	36.57	12.21	24.39
Our Models					
FinGym-4B	53.03	49.78	51.35	18.79	**35.07** +8.3%
FinGym-9B	55.00	52.83	53.89	19.14	**36.52** +7.8%

Table 10: Performance on FinVault. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. "Ovrl." denotes the overall benchmark score.

Model	ASR (↓)	VR (↓)	Ovrl. (↓)
Closed-source Models			
Gemini-3.1Pro	11.57	25.23	18.40
GPT-5.6-Sol	11.92	30.84	21.38
Claude Opus 4.8	5.49	18.69	12.09
DeepSeek-V4-pro	17.06	46.73	31.90
Open-Source Foundation Models			
Qwen3.5-4B	22.31	55.14	38.73
Qwen3.5-9B	25.70	57.01	41.36
Llama-3.1-8B	41.59	79.44	60.52
Open-Source Environment Scaling Methods			
EnvScaler-8B	50.35	79.44	64.90
AWM-8B	50.82	77.57	64.20
TOUCAN-7B	43.46	76.64	60.05
Our Models			
FinGym-4B	13.20	40.19	**26.70** +31.1%
FinGym-9B	12.38	35.51	**23.95** +42.1%

Table 11: Performance on StockBench. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. "Ovrl." denotes the overall benchmark score.

Model	RT (↑)	DDN (↑)	SR/Ovrl. (↑)
Closed-source Models			
Gemini-3.1Pro	0.03	-11.57	0.94
GPT-5.6-Sol	-1.22	-4.58	-4.37
Claude Opus 4.8	-0.37	-11.34	0.41
DeepSeek-V4-pro	3.92	-12.88	4.25
Open-Source Foundation Models			
Qwen3.5-4B	-0.20	-13.40	0.74
Qwen3.5-9B	0.07	-13.55	1.19
Llama-3.1-8B	0.74	-11.68	2.06
Open-Source Environment Scaling Methods			
EnvScaler-8B	1.00	-15.94	2.19
AWM-8B	0.29	-13.21	1.18
TOUCAN-7B	0.36	-12.53	1.44
Our Models			
FinGym-4B	3.36	-12.01	**4.01** +441.9%
FinGym-9B	3.78	-9.78	**5.43** +356.3%

Table 12: Performance on InvestorBench. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. The main table's "Ovrl." is SR.

Model	CR (↑)	MDD (↓)	AV (↓)	SR/Ovrl. (↑)
Closed-source Models				
Gemini-3.1Pro	23.28	10.74	23.58	1.62
GPT-5.6-Sol	25.69	9.80	23.70	1.78
Claude Opus 4.8	25.68	8.72	22.73	1.82
DeepSeek-V4-pro	23.94	9.46	23.01	1.68
Open-Source Foundation Models				
Qwen3.5-4B	9.02	12.74	21.03	0.78
Qwen3.5-9B	14.99	10.82	22.77	1.11
Llama-3.1-8B	17.87	11.12	23.28	1.30
Open-Source Environment Scaling Methods				
EnvScaler-8B	22.03	10.97	23.84	1.50
AWM-8B	19.41	11.06	22.40	1.42
TOUCAN-7B	15.89	13.38	23.42	1.20
Our Models				
FinGym-4B	22.39	9.65	22.46	**1.55** +98.7%
FinGym-9B	22.42	8.25	20.66	**1.77** +59.5%

Table 13: Performance on Finova. Red percentages indicate relative gains over the corresponding Qwen3.5 baselines. "Ovrl." denotes the benchmark aggregate score.

Model	Expression (↑)	Tool (↑)	Complex Reasoning (↑)	Slotting (↑)	Safety&Compliance (↑)	Intent (↑)	Ovrl. (↑)
Closed-source Models							
Gemini-3.1Pro	70.00	35.27	67.62	90.28	87.50	63.33	69.00
GPT-5.6-Sol	76.00	45.35	59.07	86.11	89.50	60.00	69.34
Claude Opus 4.8	73.00	39.15	61.92	75.00	86.00	58.67	65.62
DeepSeek-V4-pro	61.00	29.07	67.62	84.17	85.00	60.67	64.59
Open-Source Foundation Models							
Qwen3.5-4B	54.00	28.29	57.30	48.89	85.00	60.67	55.69
Qwen3.5-9B	59.00	39.15	64.06	54.44	84.50	62.67	60.64
Llama-3.1-8B	48.00	16.67	17.44	20.56	64.00	10.67	29.55
Open-Source Environment Scaling Methods							
EnvScaler-8B	32.00	16.28	51.25	71.94	82.50	59.33	52.22
AWM-8B	42.00	20.54	54.80	72.50	80.50	61.33	55.28
TOUCAN-7B	49.00	4.65	25.98	58.61	77.00	18.00	38.87
Our Models							
FinGym-4B	66.00	32.17	58.36	64.72	86.50	60.67	**61.40** +10.3%
FinGym-9B	66.00	42.25	64.41	65.00	86.00	61.33	**64.17** +5.8%

Table 14: Performance comparison across financial test sets.

Model	Market Analysis(↑)	Wealth Manage(↑)	Customer Service(↑)	Credit Risk(↑)	Insurance(↑)	Investment(↑)
Closed-source Models						
Gemini-3.1Pro	44.11	50.91	46.21	69.78	70.42	47.45
GPT-5.6-Sol	61.67	59.87	49.56	68.31	67.43	48.32
Claude Opus 4.8	60.74	63.57	51.45	80.14	74.41	51.53
DeepSeek-V4-pro	59.61	58.47	47.65	56.74	57.81	52.36
Open-Source Foundation Models						
Qwen3.5-4B	45.64	48.61	32.68	58.43	55.78	49.51
Qwen3.5-9B	47.28	54.37	34.79	57.79	56.31	51.12
Llama-3.1-8B	28.47	33.21	28.32	57.32	32.45	50.00
Open-Source Environment Scaling Methods						
EnvScaler-8B	40.45	45.51	32.51	47.48	44.51	50.98
AWM-8B	43.12	44.54	34.83	45.31	46.82	50.00
TOUCAN-7B	30.31	35.42	33.46	51.43	48.31	49.51
Open-source Models						
FinGym-4B	51.48	55.74	37.15	67.43	64.41	53.92
FinGym-9B	56.25	61.73	40.83	70.15	69.89	54.90

B.3 PROMPT DESIGN

System Prompt

```text
You are a task-information architect for executable tool-use environments.

Your only job is to expand one short task seed into one coherent natural-language task
information paragraph, using only the supplied sub-environment and generation controls.

The next pipeline stage, not you, will select tools, bind parameters, synthesize hidden
state, and write the final user task. Therefore you must stay at the semantic
requirement level.

Hard boundaries:
1. Treat SUB_ENVIRONMENT as the complete world boundary. Do not introduce entities,
objectives, actions, evidence types, permissions, or workflows outside it.
2. Preserve the core intent of TASK_SEED. Expand it; do not replace it with a
neighboring task.
3. Make the situation realistic and API-solvable. The task must be completable through
structured tools or APIs, without page clicking, file downloading, login, registration,
or human-only physical actions unless the sub-environment explicitly allows them.
4. Describe the user situation, the decision or outcome sought, the categories of
information needed, and the important time/scope/policy constraints.
5. Do not mention tool names, function names, parameter names, JSON keys, database
tables, endpoint paths, internal object IDs, fixture IDs, verifier logic, reward,
rubric, gold workflow, or expected answer.
6. Do not reveal a later-step identifier that should be discovered from an earlier
observation.
7. Do not invent exact numeric facts, record IDs, named individuals, or final
conclusions merely to make the task look concrete. Add exact values only when
GENERATION_CONTROL explicitly requires them to be user-visible inputs.
8. Respect implicit context declared by the sub-environment. Do not repeat an implicit
tenant, region, year, account, or identity as if the user had to supply it.
9. The task information must contain enough semantic information for a later model to
choose an appropriate tool path, but it must not prescribe that path.
10. Match the requested workflow shape indirectly through evidence needs:
   - single: one self-contained evidence need;
   - independent: multiple independently obtainable evidence categories that must be
   synthesized;
   - dependent: a broad record must first be located before related detail, version,
   attachment, status, or downstream object can be inspected.
11. Avoid generic wording such as "analyze the data" without identifying what
distinction, verification, comparison, or action matters.
12. Avoid over-specific biographies, decorative names, and long case narratives.
13. Return only JSON conforming to TASK_INFORMATION_SCHEMA. No Markdown and no extra
keys.
```

Figure 8: System Prompt for LLM in Task Information Generator

As shown in Figure 8, this prompt defines the role and generation boundaries of the LLM in the Task Information Generator. It instructs the model to synthesize natural-language task information from the supplied environment components while remaining at the semantic level and deferring tool-level construction to later stages.

````
User Prompt

```text
Create one task information item.

SUB_ENVIRONMENT:
{{SUBENVIRONMENT_JSON}}

TASK_SEED:
{{TASK_SEED_JSON}}

GENERATION_CONTROL:
{{GENERATION_CONTROL_JSON}}

RECENT_ACCEPTED_TASK_INFORMATION_FOR_NOVELTY_CHECK:
{{RECENT_TASK_INFORMATION_JSON_ARRAY}}

Requirements for this item:
- Use the language requested by GENERATION_CONTROL.
- Keep one primary user goal.
- Include a plausible use context and a clear decision, verification, comparison, or
action target.
- Include all user-visible scope information the later task will need.
- If workflow_mode is dependent, describe a natural information dependency without
naming IDs or tools.
- If workflow_mode is independent, describe evidence categories that can be obtained
in any order.
- Be meaningfully different from recent accepted items in both scenario framing and
reasoning focus.
- Output only an object matching TASK_INFORMATION_SCHEMA.

TASK_INFORMATION_SCHEMA:
{{TASK_INFORMATION_SCHEMA_JSON}}
```
````

Figure 9: User Prompt for LLM in Task Information Generator

As shown in Figure 9. This prompt instructs the LLM how to combine the supplied environment components, including the sub-environment, task seed, generation controls, and novelty references, into one structured task-information item that follows the required workflow and output schema.

System Prompt

```text
You are a domain-neutral architect for executable offline tool-use tasks.

Convert one text-only task-information record into one internally consistent
task draft. Domain facts and valid operations come only from the inputs. The
shared tool pool is authoritative for tool names, descriptions and schemas.

Construction rules:
1. Select only verbatim tools from the shared pool. Never invent, rename, merge
   or modify a tool or its schema.
2. Create exactly target_tool_calls workflow nodes. Every node must be necessary.
   selected_tools must equal the unique tools used by those nodes.
3. Validate every call against its tool schema and max_arguments_per_call. Never
   add convenience fields such as region or year when the schema does not need
   them or the context already supplies them.
4. Every argument declares exactly one source:
   - user_query_or_fixed_context: its exact query phrase/value is visible in the
     user prompt or explicitly declared in implicit_context;
   - prior_tool_result: it is returned by a completed predecessor at the exact
     declared JSON path.
5. Never expose an opaque record ID or hidden fixture key in the user prompt. If
   a later call needs one, an earlier discovery call must return it.
6. Obey workflow_mode semantically:
   - single: one node and no dependency;
   - independent: multiple necessary nodes, all precondition lists empty and no
     cross-node value dependency; calls may run in any order;
   - dependent: a real acyclic dependency. At least one later argument must come
     from a predecessor response. Do not fake dependency by ordering independent
     reads.
7. Write a concise, natural request from a plausible user in the environment.
   It must contain enough visible topics, periods and comparison targets for all
   direct arguments. Use natural actions such as find, check, verify, compare,
   summarize or assess. Do not mention tools, parameters, nodes, IDs, fixtures,
   rubrics, expected calls, the benchmark or the reference answer.
   Keep it to 1-3 sentences and within user_prompt_word_range. Never say
   "research economy", "current research subject", "fixture" or "synthetic".
8. Avoid personal names, excessive numbers and decorative backstory. Do not
   repeat implicit year or region unless disambiguation truly requires it.
   When suppress_implicit_context_in_prompt is true, never print the default
   year or target-context placeholder in the user prompt.
9. Create one deterministic task-local offline result per node. Its tool and
   arguments must exactly match the workflow node. Results must be sufficient
   for the answer and later trajectory audit, internally consistent, and
   available by as_of_date.
   Every response must use the runtime envelope
   {"success": true, "data": {...}}. A prior-result path is relative to the
   response root, such as data.items.0.release_id; do not prefix it with "$".
10. A result may introduce an opaque ID only if that node returns it and a later
    node cites its exact JSON path. Never fabricate a downstream-only ID.
11. Keep forecasts, intentions, budgets, announcements, survey signals and
    realized outcomes conceptually distinct. Do not claim unsupported causality.
12. Derive the private background summary and reference answer only from the
    request, fixed context and synthetic results. State important evidence limits.
```

Figure 10: System Prompt for Executable Task Draft Generator(part 1 of 2)

```
13. Do not generate rubrics or score the task. A separate model will execute the
    task first and create trajectory-grounded rubrics afterward.
14. The user prompt, generated facts, arguments, results, reference answer,
    and all other generated prose must be entirely in English. Tool names
    remain unchanged because they are API identifiers.
15. Return exactly one JSON object matching the supplied schema, with no Markdown
    and no extra commentary.
16. distractor_tools must contain exactly distractor_tool_count tools. If the
    count is zero, return an empty array.

Before returning, silently audit call count, selected tools, schemas, argument
sources, prior JSON paths, fixtures, dependency semantics, answer support, hidden
ID leakage and English-only output.
```

Figure 11: System Prompt for Executable Task Draft Generator(part 2 of 2)

As shown in Figure 11. This prompt defines the role and construction rules of the LLM in the Executable Task Draft Generator. It instructs the model to transform text-only task information into an executable offline task by selecting tools from the shared pool, binding traceable arguments, constructing workflow-consistent results, and producing a supported reference answer, while leaving rubric generation to a later stage.

```
```text
Create one executable task draft from these inputs.

<subenvironment>
{{SUBENVIRONMENT_JSON}}
</subenvironment>

<task_information>
{{TASK_INFORMATION_JSON}}
</task_information>

<shared_tool_pool>
{{SHARED_TOOL_POOL_JSON}}
</shared_tool_pool>

<generation_control>
{{GENERATION_CONTROL_JSON}}
</generation_control>

<output_schema>
{{TASK_DRAFT_SCHEMA_JSON}}
</output_schema>

Use exactly the requested number of workflow nodes. Output JSON only, entirely
in English except for immutable API tool names.
```
```

Figure 12: User Prompt for Executable Task Draft Generator

As shown in Figure 12. This prompt instructs the LLM how to combine the supplied sub-environment, task information, shared tool pool, and generation controls into one executable task draft. It also enforces the required workflow size, output schema, and language format for stable downstream processing.

System Prompt

```text
You repair one rejected structured tool-use task draft after deterministic
validation. Immutable source inputs, the rejected draft, and validation errors
are supplied.

Repair every reported error and any directly consequential field. Preserve the
seed intent, English language, workflow mode, exact target call count and domain
boundary. The shared tool pool is authoritative; do not invent or alter tools.
Every direct argument must be visible in the user prompt or fixed context. Every
derived value must exist at the declared predecessor JSON path. Fixtures must
exactly match calls, and every answer claim must be supported by the fixtures.
All generated prose and values must remain English.

Return exactly one JSON object matching the supplied output schema. No Markdown,
explanation or additional keys.
```

Figure 13: System Prompt for Rejected Task Draft Repair

As shown in Figure 13. This prompt defines the role of the LLM in the Rejected Task Draft Repair stage. It instructs the model to correct validation errors and their dependent fields while preserving the original task intent, workflow configuration, tool definitions, and generation constraints.

System Prompt

```text
You are a teacher agent validating whether one tool-use task is executable.
Solve the task by interacting with the supplied tools. You cannot see hidden
fixtures, the expected workflow, the reference answer, or any rubric.

At every turn, return exactly one JSON object in one of these forms:
{"type":"tool_call","name":"ExactToolName","arguments":{...}}
{"type":"final","answer":"Evidence-grounded final answer"}

Rules:
1. Use only tools listed in the task system message.
2. Derive arguments from the user request or prior tool results. Never guess an
   opaque record ID.
3. Inspect each tool result before choosing the next action.
4. Continue until enough evidence has been collected to answer the user.
5. A tool error is observable evidence: correct the call if possible rather than
   pretending it succeeded.
6. Do not mention hidden evaluation machinery. Output JSON only.
```

Figure 14: Prompt for Teacher Agent to build Actual Task Execution

As shown in Figure 14. This prompt defines the teacher agent that validates task executability through actual tool interaction. Without access to hidden fixtures or reference information, the agent must derive arguments from visible evidence, execute the required workflow, and produce an evidence-grounded final answer.

````
System Prompt

```text
You are a strict semantic argument matcher for an offline tool environment.
Decide whether an agent's actual arguments express exactly the same lookup as
the expected fixture arguments under the supplied tool schema and user request.

Accept harmless representational equivalents such as September versus 9 for a
month-only field, Q1 versus first quarter, capitalization, surrounding spaces,
or ISO versus natural-language dates when day precision is preserved.

Reject missing required meaning, broader or narrower date windows, different
indicators or entities, guessed opaque IDs, extra unsupported filters, and any
change that could select a different record. Do not repair arguments.

Return JSON only:
{"equivalent":true|false,"reason":"brief explanation"}
```
````

Figure 15: Prompt for Tool-LLM act as Semantic Argument Matcher

As shown in Figure 15. This prompt defines the Tool-LLM as a semantic argument matcher for the offline environment. It accepts harmless representational variations between actual and expected arguments while rejecting differences that alter the intended lookup, scope, entity, or required meaning.

````
System Prompt

```text
You are an independent rubric author and trajectory auditor for executable
tool-use tasks. The task was created without rubrics and has now been attempted
by a teacher agent in an offline environment.

First decide whether the task is genuinely runnable: the visible user request
must supply or naturally lead to all required arguments; required tools must
return usable evidence; dependent calls must obtain downstream values from
earlier results; and the final answer must be supportable without hidden state.

Then create exactly five task-level rubrics grounded in the successful evidence
path but not overfitted to the teacher's wording:
R1 tool selection, semantic argument correctness, and required dependencies;
R2 successful receipt and coverage of evidence-bearing tool results;
R3 correct interpretation and relevant category or measurement boundaries;
R4 completion of the user's explicit goal, allowing synonymous answers;
R5 factual fidelity, temporal limits, uncertainty, and prohibited overclaims.

Finally score the supplied teacher trajectory against each generated rubric on
a 0.0 to 1.0 scale. Give concise evidence from the trajectory. Do not award
credit for a requested call that failed, an opaque value that was guessed, or a
claim unsupported by tool results. Set retain_task true only when the task is
runnable, every required call can be completed, and the final answer is
evidence-supported.

Audit every concrete number, date, institution, person, geography, entity and
causal statement in the final answer. Penalize any detail that is absent from
the user request and tool observations, even if it seems plausible from general
knowledge. In particular, an implicit target context is not permission to name
a country or organization that was never returned by a tool.

All generated text must be English. Return exactly one JSON object matching the
supplied output schema, without Markdown or extra commentary.
```
````

Figure 16: System Prompt for Runnable Audit, Rubric Generation, and Teacher Scoring

As shown in Figure 16. This prompt defines the LLM as an independent task auditor and rubric author. It requires the model to determine task runnability, generate five evidence-grounded evaluation

rubrics, score the teacher trajectory against them, and retain only tasks whose workflows and final answers are fully supported.

````
User Prompt Template

```text
Audit this task and teacher trajectory, then generate and score the rubrics.

<task_without_rubrics>
{{TASK_WITHOUT_RUBRICS_JSON}}
</task_without_rubrics>

<teacher_run>
{{TEACHER_RUN_JSON}}
</teacher_run>

<output_schema>
{{RUBRIC_OUTPUT_SCHEMA_JSON}}
</output_schema>

Return JSON only.
```
````

Figure 17: User Prompt for Runnable Audit, Rubric Generation, and Teacher Scoring

As shown in Figure 17. This prompt instructs the LLM to combine the rubric-free task draft with its teacher trajectory, assess whether the task was successfully executed, and generate trajectory-grounded rubrics with corresponding scores in the required output schema.