Investment Institute

WORKING PAPER 192 | **AUGUST 2026**

Agentic AI for Financial Applications: A Comprehensive Survey

Amundi Investment Solutions

Trust must be earned

Agentic AI for Financial Applications: A Comprehensive Survey

Ru QI WU
Amundi Investment Institute
ruqi.wu@amundi.com

Frederic LEPETIT
Amundi Investment Institute
frederic.lepetit@amundi.com

Abstract

Language models have given rise to agentic AI: autonomous systems that pursue high-level goals through tool use, orchestration, and adaptation. Although prior surveys treat financial applications, agent architectures, and fine-tuning methods in isolation, this review integrates the full agentic stack. Covering literature from 2024 to 2026, we examine the structure of agentic systems, including orchestration topologies, MCP and A2A protocols, reasoning frameworks, and reinforcement-learning paradigms from RLHF to GRPO, together with their financial applications and risks. We also complement this analysis with a competitor review of two industry frameworks. Our synthesis suggests that hallucination and reproducibility are among the main constraints on autonomy, often more limiting in practice than architectural constraints. Human-in-the-loop verification therefore remains necessary in current financial deployments.

Keywords: Agentic AI, large language models, multi-agent systems, financial applications, reinforcement learning fine-tuning, retrieval-augmented generation, model risk.

JEL classification: C45, C63, G10, G14, G17.

Acknowledgement

The authors are grateful to Amina Cherief, Sonja Tilly and Thierry Roncalli for their helpful comments. The opinions expressed in this research are those of the authors and are not meant to represent the opinions or official positions of Amundi Asset Management. The authors used an AI-based language model to assist with English language polishing. The authors remain fully responsible for the content of the paper.

About the authors

Ru Qi WU

Ru Qi Wu joined Amundi Investment Institute in February 2026 as a Research Intern within the Quant Portfolio Strategy team, where he worked on the financial applications of Agentic AI, the associated risks, and the emerging frontiers of the field. He holds a Master's Degree in International Finance from HEC Paris (2026) and a Bachelor's Degree in Economics and Social Sciences from Bocconi University (2024).

Frederic LEPETIT

Frédéric Lepetit is Head of Equity Quant Portfolio Strategy within Amundi Institute since 2016. He joined Société Générale Asset Management in 2006 as quantitative analyst on the Equity side and expanded the scope of his activity to the volatility asset class area after the SGAM-CAAM merge in 2010. His areas of research cover quantitative investment strategies, factor investing, sustainable finance and portfolio construction. Prior to joining Amundi, he was consultant to French leading Asset Management companies at FactSet Research Systems from 1999 to 2006.

Mr Lepetit holds a Masters' degree in economy and asset management from La Sorbonne University (1998).

1 Introduction

The rapid diffusion of large language models (LLMs) has transformed how computational systems process, reason over, and act upon information, with finance emerging as one of the most consequential domains of application. Early adoption centered on general-purpose LLMs for discrete tasks such as sentiment scoring, document summarization, and question answering. More recently, the field has shifted toward agentic AI: autonomous systems in which one or more LLMs, augmented with tools, memory, and standardized communication protocols, decompose a high-level goal into executable subtasks and adapt their behavior in response to environmental feedback [1][2]. This marks an important shift in how artificial intelligence can be embedded within financial workflows.

The motivation for this shift is both practical and economic. Financial research remains heavily dependent on labor-intensive work, including extracting figures from 10-K filings, monitoring news flow, constructing and validating quantitative signals, and producing equity research reports. Agentic systems can offload these activities from human analysts while expanding the space of testable hypotheses.

The academic literature remains fragmented and lacks a comprehensive review of agentic AI applications in finance. A substantial body of work examines the capabilities of standalone LLMs in finance, yet comparatively little attention has been paid to the agentic dimension. This review addresses this gap by surveying agentic AI systems in finance published between 2024 and 2026. Two questions guide the analysis:

- Q1: What is agentic AI, and what is its underlying architecture?
- Q2: What are the applications and risks of agentic AI in finance?

The remainder of the paper is organized as follows. Section 2 defines the core concepts: foundation models, AI agents, and agentic AI. Section 3 examines the building blocks of agentic systems: language model types, orchestration architectures, communication protocols, tools, reasoning frameworks, reinforcement-learning fine-tuning, and evaluation benchmarks. Section 4 surveys the principal applications in finance, spanning text processing, portfolio management, factor mining, and autonomous trading. Section 5 analyzes the associated risks and limitations, including hallucination, reward hacking, catastrophic forgetting, and the cost of reasoning. Finally, Section 6 presents a competitor analysis of two industry frameworks: BlackRock's AlphaAgents and Man Group's Alpha Assistant and AlphaTrend.

2 Agentic AI

We first define the notions of foundation models, AI agents, and agentic AI, as shown in Figure 1.

A **foundation model** [3] is a pretrained, general-purpose LLM that can be adapted to specific tasks. For instance, there are financial LLMs, image models, and vision

models, all of which can be adapted for specific tasks.

An **AI agent** is an LLM augmented with tools and designed to operate through an observe-act-feedback loop. Tools can include search engines, APIs, code interpreters, and calculators. A specific instance is retrieval-augmented generation (RAG) [1], where search engines are used for real-time knowledge expansion.

Agentic AI is an autonomous LLM system characterized by two key properties: autonomy and adaptability [2]. Its autonomy allows users to input a goal rather than instructions. The AI can autonomously design the steps required to achieve the result. Adaptability is the capacity to change parameters based on incoming information and a changing environment [4]. Common structures that enhance adaptability include Chain-of-Thought reasoning [5] and the ReAct (Reasoning and Acting) framework [6]. The first introduces intermediate reasoning steps that break down the task.
ReAct alternates between reasoning, taking an action, and observing the result. As a result, this prompting framework tends to produce superior performance for tasks requiring logical reasoning and problem solving.

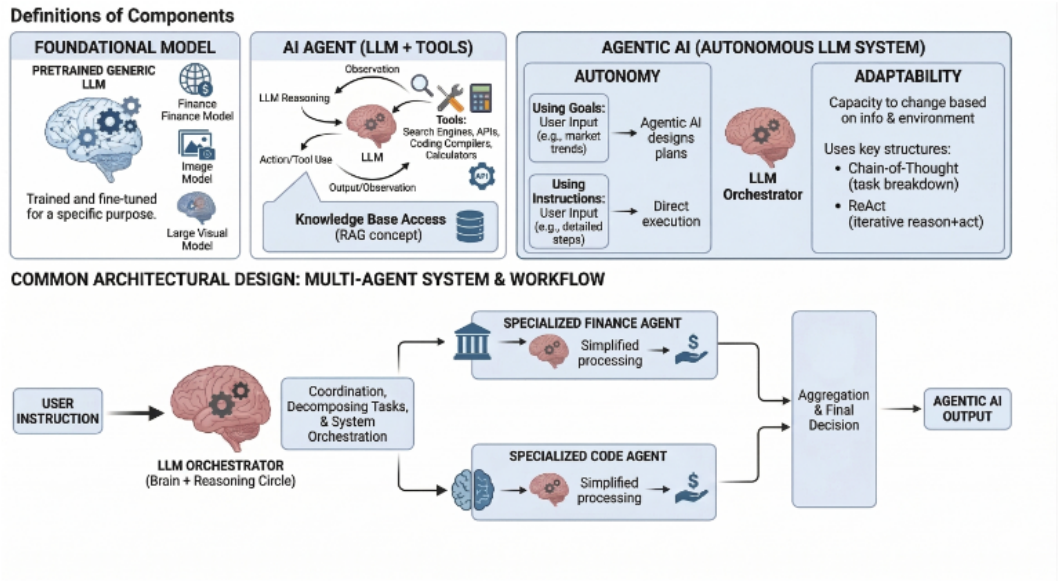


Figure 1: Agentic AI overview

Agentic AI systems coordinate multiple AI agents, often powered by foundation models [3], to carry out workflows by decomposing tasks and orchestrating them based on the initial instruction. Foundation models can serve as fine-tuned modular components for different purposes, similar to the division of responsibilities across departments in a company. A common practice is to use a LLM as the orchestrator, while small language models (SLMs) serve as modular components. This orchestration framework can be implemented as a multi-agent system (MAS).

3 Language Models And Agentic Structure

This section outlines the two main types of language models, their architectural structures, the agents, and the protocols used. First, we define what LLMs and SLMs are. Models are classified based on the number of parameters, as summarized in Table 1. A common rule of thumb is that models above roughly 10 billion parameters are often treated as LLMs [7].

SLMs provide an efficient balance between computational power and capabilities for common-sense reasoning and domain-specific tasks (coding, translation) [8]. The main advantages are speed and lower cost. On the other hand, the number of parameters limits their general ability to capture nuanced semantic differences. A study by Irugalbandara et al. [9] shows that SLMs can achieve performance comparable to LLMs on certain common sense reasoning tasks.

LLMs have many advantages in reasoning capabilities and complex tasks. A study by Google researchers [10] suggests that, on their benchmarks, larger models tend to outperform smaller ones. However, the cost and response speed become the main constraint, further details on cost are discussed in Section 5.4. In agentic AI workflows, some components prioritize speed and repeatedly perform the same type of task making SLMs preferable to LLMs.

Dimension	**Small Language Models (SLMs)**	**Large Language Models (LLMs)**
Parameter Count	<10B parameters	$\geq$ 70B parameters
Examples	Phi-3, Gemma 4 E2B/E4B, Llama 3-7B, Qwen 2.5 7B	Llama 3 70B, Llama 4 Maverick 400B, Gemini 3. Closed-source models from OpenAI and Anthropic have trillions of parameters
Memory Footprint	Small (<16 GB VRAM), single high-end consumer GPU (e.g., RTX 5090 ~32GB)	Large (> 150 GB VRAM); requires multi-GPU or TPU setups (e.g., H200 ~142GB or B200 ~192GB)
Advantage	Low latency and minimal cost	Reasoning capabilities for complex tasks
Agentic Use Cases	Sub-agents for specific tasks	Orchestrator and sub-agents
Typical Use Cases	Domain-specific tasks like coding and translation	Open-ended generation, code synthesis, complex Q&A, planning

Table 1: Comparison of SLMs and LLMs

3.1 Core Characteristics

A common characterization of agentic AI, as presented by NVIDIA[1], includes four phases. These four phases constitute the process through which AI gains agency[2]:

1. *Perception phase*: perceiving the environment, preprocessing data, processing natural language, and creating features for the LLM components.
2. *Reasoning phase*: creating execution plans by reasoning through user input and data, then orchestrating agents by assigning tasks and setting sub-goals. This step uses retrieval-augmented generation (RAG) to access internal and external data.
3. *Action phase*: executing the plan using tools such as code interpreters, calculators, and information retrieval via APIs to complete the assigned task.
4. *Learning phase*: storing knowledge and feedback from the actions taken, helping the system improve over time by learning from past actions. This is implemented through feedback loops and memory/knowledge bases.

3.2 Orchestration architecture

AI systems can be organized in multiple ways. One common approach is to coordinate them through an orchestrator that assigns tasks to sub-agents. In this section, we present three categories in which AI systems can be organized [10], as shown in Figure 2:

- *Centralized* orchestration consists of an orchestrator and sub-agents that do not directly interact with one another. This includes simple hierarchical structures. In this case, the sub-agents only work on specific tasks requested by the orchestrator.
- *Decentralized* is a MAS that operates without a coordinating orchestrator. The system consists of multiple agents that work independently on the full task and deliberate to reach a final decision. Each agent produces its own solution, then a debate mechanism merges their output to reach a final decision (multi-agent debate system).
- *Mixed orchestration* is a hybrid of centralized and decentralized approaches. A top-level orchestrator governs task decomposition and resource allocation, while sub-agents within clusters interact directly through local feedback loops, without routing every message back to the coordinator. This distinguishes it from purely centralized systems, where all information flows through the orchestrator, and from decentralized systems, which lack any coordinating authority.

[1] `https://blogs.nvidia.com/blog/what-is-agentic-ai/`

[2] Agency is the capability of AI directing their own processes, tool usage and maintaining control on the method to complete tasks -`https://www.anthropic.com/engineering/building-effective-agents`

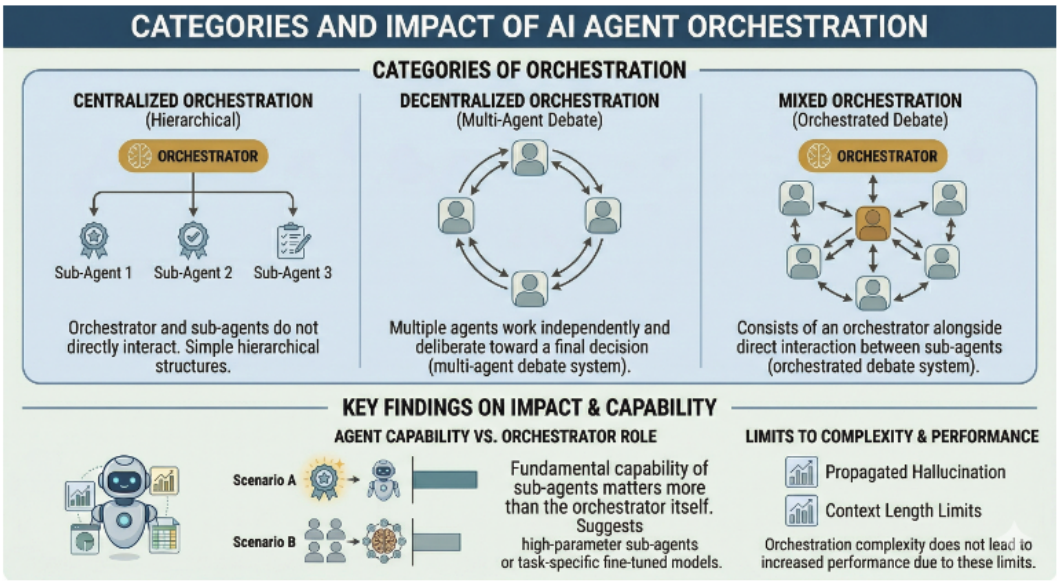


Figure 2: The three main types of orchestration

The orchestration layer defines how agents coordinate and the role of each agent within the system [11]. Specifically, it covers the planning and supervision of task execution across agents and tools based on user input:

- A common framework is LangGraph [12], developed by LangChain and built on a graph structure. Its *execution graphs* [12] route subtasks to the appropriate agent or tool and aggregate intermediate results. According to LangGraph[3], the system is divided into three main components:
  - *State* is the data structure that captures the current state of the application and provides context to all components.
  - *Nodes* are the computational units that receive context and prior information to perform computation or reasoning. Nodes can be composed of LLMs and tools (i.e., API calls to retrieve data).
  - *Edges* define the transition to the next node based on the current context. They handle retries in case of failure and determine when the overall goal has been achieved.
- AutoGen [13][4] is another agent orchestration framework developed by Microsoft. It is conversation-based, with each agent assigned a specific role. The workflow is driven by debate and interaction rather than a predefined graph-like path, making it particularly suitable for debate-oriented multi-agent systems.

The orchestration structure can have a material impact on performance, although the capability of the sub-agents often remains the dominant factor. The study by

[3] `https://docs.langchain.com/oss/python/langgraph/thinking-in-langgraph`
[4] `https://microsoft.github.io/autogen/stable//index.html`

Google [10] shows that the intrinsic capability of sub-agents matters more than the orchestrator itself, suggesting either the use of high-capacity sub-agents or task-specific fine-tuned SLMs. In addition, increased orchestration complexity does not necessarily lead to better performance; reasons include hallucination propagation and context length limits.

3.3 AI agents

AI agents are the building blocks of agentic AI systems. They are typically based on LLMs such as Claude, Gemini, Qwen, or DeepSeek, augmented with tools and standardized protocols as outlined in Section 2. Depending on the use case, these models can be deployed either as LLMs or as SLMs. For financial applications, it is common practice to fine-tune them on domain-specific or proprietary data, as this enhances their domain-specific capabilities and improves their performance on specialized tasks.

3.3.1 Protocols

There are two main protocols that govern the interaction of AI agents with external systems and with one another, as shown in Figure 3.

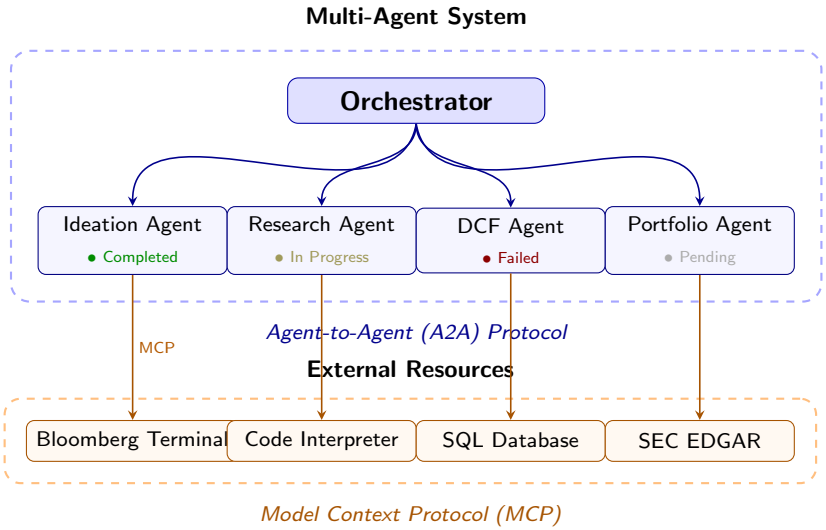


Figure 3: Multi-agent system architecture. Within the top frame, the orchestrator delegates tasks to specialized agents via the A2A protocol; each agent carries a current execution status. Vertical arrows show individual MCP connections through which every agent independently accesses external data and computational resources.

Model Context Protocol (MCP) MCP is an open-source standard for connecting AI models to external systems. It provides a standardized way to connect models

to external resources such as data sources, computational tools (e.g., calculators and code interpreters), and third-party APIs.[5]

Agent-to-Agent (A2A) Protocol[6] is the protocol used to coordinate communication among agents in multi-agent systems. A2A governs interactions between agents within a multi-agent system[7].

- Each agent has a JSON card describing its functionality, along with input examples that facilitate interoperability.
- A2A includes status management with four states: pending, in progress, completed, and failed. This enables asynchronous execution across multiple agents.
- The A2A framework complements MCP: A2A governs inter-agent communication, while MCP enables communication between agents and external systems. For example, A2A handles how the orchestrator assigns tasks to agents (Figure 3) based on information retrieved from agent cards. Agents then use MCP connectors (e.g., to search databases) to execute the tasks received from the orchestrator.

3.3.2 Tools and Skills

AI agents have access to tools and skills, which enable agency and domain-specific capabilities. Most AI agents are equipped with web search functions and knowledge bases, requiring efficient retrieval and use of knowledge. This is often achieved through retrieval-augmented generation (RAG).

Tools [14][15] are functions that an AI model can invoke to interact with the world beyond its context window. MCP connectors allow AI models to access data from specific providers; examples include Standard & Poor's and LSEG Refinitiv, which provide real-time financial and economic data through API calls. A common way to implement tools is via smolagents[8] using its tool-calling capabilities.

- Three common categories of tools can be distinguished:
  - *Retrieval tools*: web search (BigQuery API, DuckDuckGo API), database lookup, document retrieval (LlamaIndex).
  - *Computation tools*: code execution (Python interpreters), calculators (WolframAlpha API), data transformation (Python interpreters with NumPy, pandas packages).
  - *Side-effect tools*: email sending (MCP connector to Gmail or Outlook), calendar creation, API writes.

[5] `https://modelcontextprotocol.io/docs/getting-started/intro`

[6] `https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/`

[7] `https://a2aprotocol.ai/blog/a2a-sample-methods-and-json-responses\#google_vignette`: example of communication between agents.

[8] `https://huggingface.co/docs/smolagents/index`

- This mechanism enables language models to acquire some *agency* and interact with external applications. Researchers at Meta [16] argue that smaller models equipped with the right tools may outperform larger models in both language and tool-enabled tasks.
- Tool use requires specific instruction tuning [16][17] and is often limited to a specific domain of tools. Open-source LLMs often struggle to use tools in sequence and perform worse than closed-source ones. Agent skills can improve tool use without requiring changes to model parameters.

Agent Skills are a novel set of documents that expand AI agents' capabilities in domain-specific tasks [18]. They are an open standard developed by Anthropic in October 2025.[9] The main characteristics are as follows:

- Skills are typically stored as structured documents and loaded into the model's context at runtime, or organized as folders. Each skill includes three main components[10]:
  - *Markdown file*: where all instructions and metadata are stored.
  - *Scripts*: executable code, for example, a script that standardizes the creation of a backtest or the calculation of enterprise value.
  - *Assets*: resources or templates used to generate standardized outputs. These templates are useful for maintaining consistency in the generated output.
- Skills provide domain-specific knowledge, procedural steps, and formatting conventions that the model would otherwise have to infer from scratch. The files can include code, connections to external APIs, and the creation of Excel and PowerPoint files. An example is Anthropic's Excel skill, which allows the creation of production-grade spreadsheets and financial models[11].
- Skills are composable and repeatable: multiple skills can be loaded simultaneously to equip the agent with complementary capabilities (e.g., geospatial mapping *and* financial report generation in the same session). The workflow of an agent invoking a skill is always consistent with the skill instructions, making it easier to reproduce and audit.

Retrieval-Augmented Generation (RAG) [19][20] is an architecture that augments a language model's parametric knowledge with dynamically retrieved external knowledge. Its main use is to incorporate real-time information while providing verifiable ground truth and reducing hallucinations. The process can be described in two main steps:

[9] `https://support.claude.com/en/articles/12512176-what-are-skills` and `https://agentskills.io/home`

[10] `https://agentskills.io/what-are-skills`

[11] `https://platform.claude.com/cookbook/skills-notebooks-01-skills-introduction`

- *Retrieval stage*: given a user query, the system retrieves relevant documents from a database or the web. The retrieved resources are indexed and segmented into chunks, which are then compared with the query using cosine similarity. The top-k chunks with the highest similarity are appended to the context for the generation phase.

- *Generation stage*: the retrieved passages are re-ranked (e.g., using diversity or relevance metrics) and filtered. This is motivated by the fact that an LLM's attention mechanism is more sensitive to information at the beginning and end of the text [21]. The curated text is then added to the context and used to generate the answer to the user's query.

When paired with live web search, RAG extends the model's effective knowledge cutoff to the present day.

3.3.3 Reasoning

The emergence of reasoning models such as DeepSeek-R1 (Jan. 2025) [22] and ChatGPT-o1 (Sep. 2024) has been associated with improvements in model accuracy and performance. Reasoning models leverage Chain-of-Thought (CoT) [5], ReAct [6], and Reflexion [23] to enhance reasoning (see Figure 4). All three frameworks share the idea of decomposing a larger task into intermediate steps:

- *Chain-of-Thought (CoT)* decomposes a task into multiple steps through few-shot prompting, which encourages the model to double-check its reasoning process. The model reasons through the task iteratively in a single pass. FinCoT [24] shows that embedding a structured plan into LLMs for financial tasks outperforms pure CoT prompting. CoT is suitable for a wide range of tasks and often performs well in improving quantitative reasoning capabilities.

- *ReAct* combines CoT with action: the model perceives, reflects, and acts until it reaches the predetermined goal. It then returns to the perception phase after each action, comparing its current state with the goal.

- *Reflexion* is a self-improving framework that allows LLMs to reflect on and learn from their behavior based on linguistic feedback. This feedback is long-lasting and serves as memory for future tasks. A key feature is its memory mechanism, which helps future attempts avoid previous mistakes. It is particularly useful for complex and long tasks.

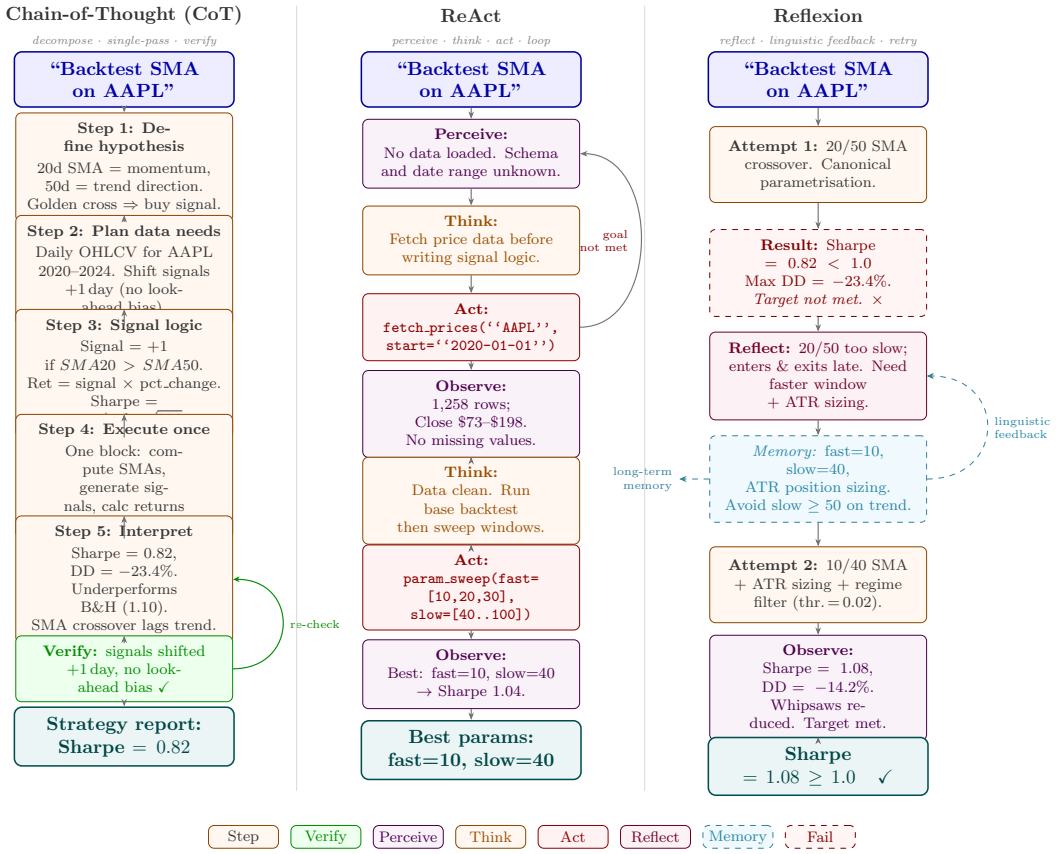


Figure 4: Comparison of CoT, ReAct, and Reflexion when applied to backtesting an SMA crossover strategy on AAPL (2020–2024).

3.3.4 Fine Tuning With Reinforcement Learning (RL)

Fine-tuning is the process of adapting a pre-trained LLM to align its outputs with specific preferences or tasks. This is achieved by further training the model on domain-specific datasets and adjusting its weights to target behaviors or domain distributions [25]. The steps involved in fine-tuning a model with reinforcement learning can be summarized as follows:

1. Choose the pre-trained model: academia and industry often use already supervised fine-tuned (SFT) models for further RL fine-tuning. These models are referred to as *Instruct* or *it* models on Hugging Face, for example *Llama 3.1-8B-Instruct*[12] and *Gemma 4-31B-it*[13].

2. Dataset preparation: for Reinforcement Learning from Human Feedback (RLHF), Reinforcement Learning from AI Feedback (RLAIF), or Reinforcement Learning with Verifiable Rewards (RLVR), the dataset must be prepared

[12] `https://huggingface.co/meta-llama/Llama3.1-8B-Instruct`
[13] `https://huggingface.co/google/gemma`

either from pairwise preferences or from ground-truth answers. This is the most time-consuming and least scalable stage of the pipeline. The distinction between pairwise-preference-based methods (RLHF/RLAIF) and RLVR is illustrated in Figure 5 and discussed in Section 3.3.5.

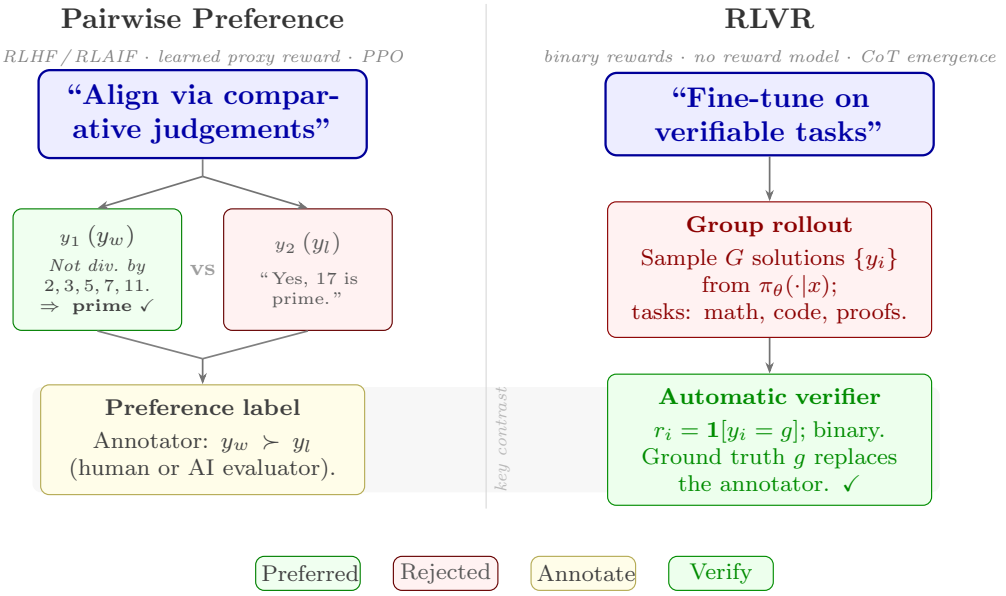


Figure 5: Pairwise preference (RLHF / RLAIF) vs. RLVR: reward signal origin (shaded). **Left:** two candidate responses to "Is 17 prime?" are ranked by a human or AI annotator ($y_w \succ y_l$). **Right:** a binary verifier scores each rollout against ground truth g directly, replacing the annotator entirely.

3. Reward modeling: RLHF and RLAIF rely on a reward model trained from human or AI preference labels, whereas RLVR derives rewards directly from verifiable ground truth (see Table 2). This step improves alignment with preferred behaviors, as discussed in the next subsection.

4. RL policy optimization using RL algorithms: the policy is optimized against the reward model using RL algorithms, most notably PPO and GRPO.

5. Defining how many parameters to fine-tune, whether through full-parameter fine-tuning or partial fine-tuning with low-rank adaptation (LoRA) [26] or quantized LoRA (QLoRA) [27]. The latter is known as parameter-efficient fine-tuning (PEFT). The choice depends on the desired level of specialization and the amount of available computational resources. For most use cases, researchers adopt LoRA.

6. Evaluation: the fine-tuned model is evaluated against benchmarks to assess whether the process was beneficial.

3.3.5 Rewards of RL fine tuning

Reinforcement learning fine-tuning adapts a language model by optimizing a reward signal that promotes the desired behavior. As summarized in Table 2, this signal can be generated through three main approaches: human judgment, AI-generated feedback, or objective verification. These approaches are briefly described below, with additional details provided in Appendix A.

Paradigm	Feedback Source	Details & Limitations
SFT	**Human annotators** *Instruction–response pairs.*	Trains the model on curated instruction–response datasets. **Limitation:** Requires high-quality labeled data; quality is preferred over quantity.
RLHF	**Human annotators** *Pairwise preference labels.*	Annotators rank output pairs; labels train a reward model that drives the RL update. **Limitation:** High annotation cost and latency constrain scale and frequency.
RLAIF	**AI evaluator** *Automated preference labels.*	Mirrors the RLHF pipeline; AI-generated labels match human performance on summarization and dialogue. **Limitation:** Labels become noisy and unstable on complex tasks.
RLVR	**Ground-truth verifier** *Binary outcome signal.*	Reward is granted only on exact match; this implicitly improves chain-of-thought quality. **Limitation:** Restricted to verifiable tasks (maths, code); inapplicable to open-ended generation.

Table 2: Comparison of fine-tuning paradigms by feedback source and applicability. SFT provides the supervised baseline; RLHF, RLAIF, and RLVR extend it with increasingly scalable feedback mechanisms.

- *Reinforcement Learning with Human Feedback* (RLHF) aligns LLMs with human intent by training a reward model on human pairwise preferences (see Figure 5). Its main limitation is the substantial cost and time required to collect high-quality annotations at scale [28][29].
- *Reinforcement Learning with AI Feedback* (RLAIF) replaces human annotators with an AI evaluator to generate preference labels, offering greater scalability while achieving performance comparable to RLHF on simpler tasks. However, label quality tends to degrade on more complex reasoning tasks [30][31]. Because preferences are generated by AI models, biases, harmful content, and cultural nuances may not be fully captured [32]. For more complex labeling tasks, human-in-the-loop approaches such as RLHF remain preferable.

- *Reinforcement Learning with Verifiable Rewards* (RLVR) fine-tunes models using objective binary signals derived from ground-truth answers. This setting implicitly strengthens chain-of-thought reasoning, but it is limited to verifiable tasks such as mathematics and code generation [22][33][34].

3.3.6 Policy optimization algorithms

State-of-the-art RL fine-tuning methods fall into two main algorithm families: Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO). Both are used to train the policy network governing LLM decision-making, aligning its outputs with preferences derived from a reward model.

Proximal Policy Optimization (PPO) [35], developed by OpenAI, is a policy gradient method that jointly trains a policy network and a value network. Its core idea is conservative updating, implemented through the clipped surrogate objective (for details, see Appendix B). Each gradient step is constrained to stay close to the previous policy, which helps prevent catastrophic performance collapse from overly aggressive parameter updates. Applying PPO to fine-tuning requires explicit reward signals for each action taken by the model.

Group Relative Policy Optimization (GRPO) [36], introduced by DeepSeek, offers a major architectural simplification by removing the value network entirely and relying instead on group-relative reward normalization to estimate policy gradients (for details, see Appendix C). This design materially reduces both memory footprint and computational cost during training. Notably, the authors of DeepSeekMath note that the performance gains observed under GRPO may reflect more effective output sampling rather than genuine improvements in underlying reasoning ability, which is important when interpreting benchmark results. GRPO's broader adoption was catalyzed by DeepSeek-R1 [22], which showed that relatively compact models trained with this algorithm could match or outperform models of substantially greater scale.

A summary of the variants relevant to each family is provided in Table 3.

3.3.7 Parameter-Efficient Fine-Tuning

To reduce the computational cost of full-parameter fine-tuning, two widely used approaches have emerged: LoRA [26] and QLoRA [27]. Both methods update only a subset of parameters rather than the full model. Their central idea is to decompose the large weight matrices of LLMs into the product of two lower-rank matrices, thereby reducing the number of trainable parameters and lowering computational requirements. QLoRA further introduces quantization, which maps higher-precision weights to lower-precision representations, in this case from 16-bit to 4-bit. This substantially reduces memory usage and makes it possible to fine-tune

Algorithm	Score	Core contribution
GRPO lineage — critic-free, on-policy		
REINFORCE++ [37]	++	Uses the average reward across the whole training batch as a reference point, rather than just within a small group. This gives a fairer and more stable signal for the model to learn from.
DAPO [38]	+++	Addresses four known weaknesses in GRPO simultaneously: adjusts update aggressiveness based on performance, skips uninformative training examples, and shapes rewards to discourage unnecessarily long answers.
Dr. GRPO [39]	+	Shows that two standard correction steps in GRPO unintentionally distort the learning signal, and removes them. The result is a simpler and more theoretically correct training objective.
GSPO [40]	++	Evaluates the quality of an entire response rather than token by token. This prevents small errors from compounding over long outputs and makes training more robust, especially in very large models.
CISPO [41]	+	Applies update limits at the response level rather than per token, so a few unusually confident tokens cannot distort the entire learning step. Designed for stable training on long documents.
iGRPO [42]	++	Encourages the model to critique its own answers and uses those critiques as extra feedback signals. This helps the model keep improving even on hard problems where it rarely gets the answer right.
PPO lineage — critic-based, on-policy		
VAPO [43]	+++	Reintroduces a helper network that estimates step quality, pretrained to avoid instability, with separate tuning for positive and negative feedback and special handling for long reasoning chains.
VC-PPO [44]	++	Diagnoses the main reason PPO breaks on long reasoning tasks: the value estimator degrades, not the policy itself. Fixing the estimator recovers PPO to the same level as simpler critic-free methods.

Table 3: Post-January 2025 reinforcement learning fine-tuning algorithms in the GRPO and PPO lineages. All methods are on-policy and target reasoning tasks with verifiable rewards, unless otherwise noted.

65B-parameter models on a single 48GB GPU.

Common packages supporting the latest algorithms include:

1. TRL[14] is developed by Hugging Face and provides direct access to many models available on the platform. It supports PPO, GRPO, and several related variants, while also natively supporting LoRA and QLoRA. Among the avail-

[14] `https://github.com/huggingface/trl`

able options, it is generally the easiest to access and use for researchers with limited computational resources.

2. Verl[15] is developed by ByteDance and is designed for large-scale training across multiple GPUs. It supports the newest GRPO and PPO variants and is therefore less suitable for single-GPU environments. Its architecture allows GPU resources to be co-located more flexibly for training and inference.

3. OpenRLHF[16] separates the policy, reference, critic, and reward models across independent GPU workers, thereby eliminating co-location memory bottlenecks. It also supports tensor, data, and sequence parallelism, making it well suited to high-parameter models.

3.3.8 Evaluation of Models

A crucial step is to assess the capabilities of fine-tuned models. Just as students are evaluated through exams, AI models are evaluated on benchmarks, which can be viewed as exams for assessing chain-of-thought reasoning and answer accuracy.

Fine-tuned models are evaluated against both general and domain-specific benchmarks to measure alignment quality and capability gains across three axes: general knowledge, coding, and quantitative reasoning. The main benchmarks used are presented below. General knowledge is assessed using Humanity's Last Exam (HLE),[17] coding ability via LiveCodeBench,[18] SWE-bench Verified,[19] and Terminal-Bench;[20] and mathematical reasoning via MATH[21] and AIME.[22]

In the financial domain, several benchmarks assess different capabilities of AI models:

- *FinanceBench*[23] [45] contains 10,234 questions about public companies paired with ground truth. It is designed to test basic retrieval capabilities rather than more complex reasoning tasks; examples include retrieving Amazon's inventory or operating income. It is therefore well suited to evaluating the retrieval capabilities of sub-agents.

- *Finance Reasoning*[24] [46] assesses financial numerical reasoning. It tests CoT for textual and numerical reasoning, along with programming capabilities. The dataset requires tabular analysis and RAG-based information retrieval to complete the tasks successfully. Examples include calculating the return on holding

[15] https://github.com/volcengine/verl
[16] https://github.com/OpenRLHF/OpenRLHF
[17] https://agi.safe.ai/
[18] https://livecodebench.github.io/
[19] https://www.swebench.com/
[20] https://terminal-bench.github.io/
[21] https://github.com/hendrycks/math
[22] https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions
[23] https://github.com/patronus-ai/financebench
[24] https://finqasite.github.io/

an asset over a given period. This benchmark is particularly suited to evaluating numerical capabilities for financial modeling and analysis.

- *FinBen*[25] [47] covers the full spectrum of financial AI capabilities, from information extraction and sentiment analysis to forecasting, risk management, and agentic decision-making (including stock trading). It evaluates the ability of AI and agentic AI systems to perform under financial conditions. It is therefore useful for assessing comprehensive agentic AI capabilities, as well as the performance of the main agents.

3.3.9 Frontier: Evolution Strategies Fine-Tuning

Recent research is exploring the feasibility of Evolution Strategies [48] as an alternative to RL algorithms. The main idea is as follows:

1. Generate a population of N candidate models by adding small Gaussian noise to the parameters of a pre-trained LLM.
2. Evaluate each candidate using a reward function.
3. Update the original parameters using a weighted combination of the noise vectors, proportional to their rewards.
4. Repeat the process.

This paradigm is backpropagation-free, which makes it easier to parallelize. The study suggests that this method can match the performance of PPO and GRPO while using only 20% of the training samples. Moreover, the authors of ESSAM [49] show that Evolution Strategies require only about 20 GB of VRAM to fine-tune an 8B-parameter model, which is 18× less GPU memory than PPO (314 GB) and 10× less than GRPO (212 GB).

4 Application in Finance

This section covers the main applications of agentic AI in finance. Table 5 at the end of the section summarizes the main ideas developed below.

4.1 Data types

Across most fields, data can be broadly classified as structured or unstructured. Although some data are structured, a large share remains unstructured. Unstructured data such as earnings reports, 10-K filings, and earnings calls typically require substantial human intervention.

[25] `https://github.com/IBM/fiben-benchmark`

Structured data are organized in rows and columns and can be manipulated efficiently with modern tools. They often include market data, such as economic indicators (e.g., the FRED database and Bloomberg economic data) and stock-related data (price, volume, and price-volume-derived indicators).

Unstructured data are not organized in a tabular format and can take many forms, which makes them difficult to store and manage in databases. Examples include earnings reports, satellite images, charts, earnings-call audio, and video.

4.2 Financial Foundation Models

As illustrated in Table 4, financial foundation models are pre-trained on financial data and fine-tuned for better performance on specific tasks, including both LLMs and SLMs. Multiple models serving different purposes can improve accuracy and performance through hybrid structures for analyzing multimodal data [50]. The models can be divided into four main categories [3].

Model Category	Data Structure	Details & Examples
Language foundation model	**Unstructured** *Requires high human intervention.*	Financial text: earnings reports, 10-K filings, contracts. **Goal:** Summarize information and generate context.
Time-series foundation model	**Structured** *Organized in rows and columns.*	Market data (FRED, Bloomberg), stock prices, volume, and derived indicators. **Goal:** Predict future market values.
Vision-language foundation model	**Unstructured** *Difficult to store in databases.*	Charts, earnings calls (audio/video), images. **Goal:** Data point extraction and multimodal understanding.
Multimodal financial model	**Hybrid** *Mixed structures.*	Integration of text, visual, and audio data. **Goal:** Analyze multimodal data for enhanced accuracy.

Table 4: Classification of financial foundation models by data structure and application.

- Language foundation models are pre-trained on financial text, such as earnings reports and contracts, with the goal of summarizing information and generating context. FinLlama is a fine-tuned version of Llama 2 7B that shows superior performance in financial sentiment scoring with a small fine-tuned

model [51]. FinGPT [52] introduces an open-source, continuously updated FinLLM framework that replaces costly pretraining with real-time data ingestion tailored to financial data.

- Time-series foundation models are pre-trained on financial time-series data such as economic indicators or stock prices. Financial time-series are continuous and non-stationary. The goal is to predict future values. FinCast [53] is a foundation model optimized for financial time-series.
- Vision-language foundation models are pre-trained on unstructured financial data such as images, charts, tables, and audio. Applications include data point extraction and multimodal understanding. FinVis-GPT [54] is an example designed for financial chart analysis and generating chart-based opinions.
- Multimodal financial models [55] are trained on different types of financial data, combining text, visual, and audio modalities. OpenFinLLM [56] is an example that integrates multiple models, including FinLlama for general text reasoning and FinLLaVA, trained on chart-text pairs, for multimodal reasoning capabilities.

4.3 Textual processing

Unstructured data processing: LLMs have demonstrated strong performance in deduplicating text and filtering for relevance [57]. GraphRAG [58] facilitates the construction of knowledge graphs from unstructured text data, thereby improving data quality and reasoning capabilities. Another approach is to convert unstructured data into SQL format. Choi et al. [59] propose an agentic framework with two agents that perform extraction and reorganize the data into SQL format. Similarly, the authors of FinReflectKG [60] show that agentic AI can create knowledge graphs from unstructured data sources such as 10-K filings. Supporting this, Brach et al. [61] show that AI models using few-shot prompting can perform exceptionally well in data-transformation tasks. MultiFinRAG [62], an agentic data extraction framework developed by the S&P research team, extracts small batches of data from 10-K and 10-Q filings and processes them using multiple small, quantized models. In this framework, processed images and charts are stored as JSON objects alongside brief text summaries. Their results suggest that this approach can outperform GPT-4o in their setup, even within a limited computing budget. Finally, FinAgentBench [63] serves as a benchmark for assessing the agentic retrieval capabilities of LLMs on unstructured data.

Text analysis: LLMs can assess sentiment and relevance in textual data. For example, the study of the GameStop short squeeze during the pandemic [57] shows how LLMs can be used to measure emotional intensity, sentiment, and novelty. LLMs can also capture nuances in sentiment and strategic shifts [64], which can help identify potential shifts in M&A activity as well as geopolitical and political events. *Decoding the Beige Book* [65] further shows how economic publications, specifically

the Beige Book, can be labeled to build a reliable recession indicator. Other work demonstrates that LLMs, including ChatGPT, can accurately interpret Fedspeak [66]. In addition, Automatic Market Commentary [67] introduces a framework for automatically generating market commentary from corporate disclosure documents. Finally, MarketSenseAI 2.0 [68] presents an agentic AI framework that analyzes company fundamentals and macroeconomic data to deliver market insights through a macro-driven approach.

Market condition prediction: Agentic AI systems can analyze textual and price data to identify regime shifts. Sun et al. [69] show that while word embeddings are generally stable under normal conditions, the semantic meaning of words can change substantially during regime changes and crisis periods, which reduces model predictive power. Their findings highlight the importance of robust measures of semantic volatility. RiskLabs [70] propose a risk prediction framework for forecasting Value-at-Risk and volatility using multimodal data, including audio and text. Similarly, *Mind the Shift* [71] introduces Delta-consistency Scoring to characterize monetary stance and regimes from Fed statements by tracking absolute stance and relative changes from meeting to meeting. Alpha-R1 [72] presents an LLM-driven factor investment framework with regime awareness: the LLM detects the current regime from macroeconomic news and determines which factors to activate. Finally, CAMEF [73] is a multimodal, event-driven forecasting framework that combines textual encodings of macroeconomic releases with high-frequency price time series to predict subsequent asset price movements.

Self-improving models: Agentic reasoning is underpinned by ReAct [6] and Reflexion [23], which rely on iterative reasoning-and-acting loops that refine outputs based on prior actions and observations. FinCon [74] introduces a *verbal reinforcement*-based multi-agent investing framework that updates its beliefs about the financial assets under analysis over time, using textual representations of those assets. SE-Agent [75] proposes a self-evolving agentic framework that combines historical knowledge stored in memory with newly generated answers through crossover and restructuring to improve the overall solution. Another notable example is the R&D Agent [76], developed by Microsoft, which uses a dual research-and-development structure: the research component generates ideas and corresponding mathematical formulas, analogous to hypothesis generation in factor investing, while the development phase validates these hypotheses through rigorous backtesting.

Real-time information systems: FinSearch [77] is a real-time agentic framework for online financial data retrieval. Its architecture consists of a search planner that decomposes a query into substeps, a search executor that retrieves information from websites to answer each subquery, and a synthesis component that assigns greater weight to recent news before summarizing the retrieved results. SocioDojo [78] is designed to ingest a continuous stream of new information and support real-time in-

vestment decisions. It implements an Analyst-Assistant-Actuator tri-agent system, where the first agent analyzes the data and produces an opinion, the second reviews the analysis, and the third determines whether an investment decision can be made or whether further analysis is required. FinGPT [52] is an open-source financial LLM with real-time processing capabilities, and its lightweight architecture enables rapid fine-tuning to adapt to semantic shifts. Finally, FinBloom [79] is a real-time retrieval framework in which a domain-adapted 7B SLM acts as a financial agent, retrieving the context needed by a larger frozen model to answer user queries. This can reduce both latency and operational costs relative to using a large model for the full task.

Framework / Paper	**Year**	**Main novelty**
Textual processing — unstructured data		
GraphRAG [58]	2024	Constructs knowledge graphs from unstructured text to enrich time-series forecasting and reasoning.
Choi et al. [59]	2025	Uses a two-agent pipeline to transform unstructured financial filings into structured SQL-accessible data.
FinReflectKG [60]	2025	Builds knowledge graphs from 10-K filings through an agentic workflow.
Brach et al. [61]	2025	Shows that few-shot prompting performs well at converting domain-specific unstructured text into structured formats.
MultiFinRAG [62]	2025	Uses quantized small multimodal models to batch-extract financial filings into JSON for retrieval-augmented QA.
FinAgentBench [63]	2025	Benchmarks agentic retrieval over unstructured financial filings with multi-step reasoning.
Textual processing — text analysis		
GameStop study [57]	2024	Uses LLMs to analyze sentiment, trends, and communication patterns in social-media text.
Zhao et al. [64]	2024	Applies GPT-4 to text processing and sentiment analysis across financial tasks.
Decoding the Beige Book [65]	2025	Uses LLMs to label economic releases and build a recession indicator.
Fedspeak [66]	2023	Shows that LLMs can classify central-bank policy stance accurately.
Automatic Market Commentary [67]	2025	Automatically generates financial commentary from corporate disclosures.
MarketSenseAI 2.0 [68]	2025	Fuses fundamentals, macro data, and financial news in a macro-driven agentic system.
Market condition prediction		
Sun et al. [69]	2025	Measures embedding volatility and semantic drift during regime shifts.

continued on next page

Table 5 – continued

Framework / Paper	Year	Main novelty
RiskLabs [70]	2025	Forecasts financial risk, including VaR and volatility, from multimodal audio and text.
Mind the Shift [71]	2026	Uses Delta-consistency scoring to track monetary stance across FOMC statements.
Alpha-R1 [72]	2025	Uses regime-aware macro news interpretation to activate factors contextually.
CAMEF [73]	2025	Uses a causal multimodal framework to fuse macro text and prices for event-driven forecasting.
Self-improving models		
FinCon [74]	2024	Introduces a verbal-reinforcement multi-agent framework with evolving investment beliefs.
SE-Agent [75]	2025	Proposes a self-evolving framework that refines reasoning through trajectory recombination.
R&D Agent [76]	2025	Uses a research-development loop to generate hypotheses and validate them through backtesting.
Real-time information systems		
FinSearch [77]	2024	Implements a plan-crawl-synthesize pipeline that prioritizes recent information.
SocioDojo [78]	2024	Uses an Analyst-Assistant-Actuator tri-agent architecture for real-time decision-making.
FinGPT [52]	2023	Provides an open-source LLM that can be rapidly fine-tuned to evolving financial conditions.
FinBloom [79]	2026	Uses a 7B SLM to ground a frozen model with real-time financial context.
Portfolio management		
LLM-based Black-Litterman [80]	2025	Generates return priors and confidence levels from price and news using an LLM.
Cheng et al. [81]	2025	Combines qualitative LLM analysis with quantitative factors for asset pricing.
MASS [82]	2025	Uses heterogeneous agents and demonstrates an agent scaling law.
AlphaAgents [83]	2024	Generates investment theses through a specialized multi-agent trading framework.
Voronina et al. [84]	2025	Shows that combining LLM selection with mean-variance optimization outperforms LLM-only weighting.
Factor mining		
Alpha-GPT 2.0 [85]	2024	Supports human-in-the-loop alpha research through iterative human-AI interaction.
Kou et al. [86]	2025	Proposes a risk-aware three-stage discover-evaluate-optimize alpha framework.
LLMFactor [87]	2024	Identifies interpretable factors from news and historical prices to explain stock moves.
FactorMAD [88]	2025	Uses debate-based multi-agent reasoning for interpretable alpha factor generation.

continued on next page

Table 5 – continued

Framework / Paper	Year	Main novelty
QuantAlpha [89]	2026	Uses evolutionary crossover and mutation to refine alpha factors.
FactorMiner [90]	2026	Combines modular design with memory-sensitive self-improvement for alpha mining.
Model risk management		
Okpala et al. [91]	2025	Pairs modeling and risk-management agents for human-in-the-loop compliance.
Bhattacharyya et al. [92]	2025	Combines soundness checks and outcome analysis for generative AI model validation.
Financial time-series forecasting		
Bi et al. [93]	2025	Uses a multi-agent transformer to capture inter-variable dependencies for financial time-series forecasting.
FinCast [53]	2025	Introduces a foundation model for financial time-series forecasting.
StockTime [94]	2024	Fuses text, prices, statistics, and cross-stock correlations.
Autonomous trading & RL		
AI-Traders [95]	2025	Evaluates live-information agents that emulate analyst-like trading decisions.
TradingAgents [83]	2024	Organizes agents into analyst, trader, and risk-management roles.
MARL Market-Making [96]	2025	Uses a PPO-based, opponent-aware agent for market making with low market impact.
FinO1 [97]	2025	Shows that GRPO yields reliable gains over PPO and DPO, while introducing FinReason.
TradingR1 [98]	2025	Applies GRPO-enhanced reasoning to trade decisions.

Table 5: Survey of agentic AI frameworks in finance.

4.4 Portfolio Management

Portfolio Management: An innovative approach combines LLMs with Black-Litterman portfolio optimization [80], where the LLM is used to generate prior views on expected returns using a two-week lookback window based on price and news data. Cheng et al. [81] explore an empirical pricing framework using LLMs and demonstrate the effectiveness of combining qualitative LLM analysis with modern neural pricing models.

MASS [82] is a multi-agent portfolio construction framework that operates in three steps. First, it assigns diverse beliefs, endowments, and risk profiles to LLM agents, which then generate candidate stocks for investment. Second, an aggregation step identifies stocks selected by the majority of agents with minimal disagreement across investor profiles. Finally, a numerical optimization process, termed backward optimization, learns the optimal weighting over these agent-type perspectives using a historical lookback window. This allows the system to dynamically adapt the mixture of investor viewpoints to prevailing market conditions, thereby maximizing

returns. Within this framework, the authors demonstrate a scaling law: increasing the number of agents improves performance.
TradingAgents [83] is a multi-agent portfolio construction framework that distills textual information to produce robust investment theses for stocks, using AutoGen to manage multi-agent communication. Finally, Voronina et al. [84] show that portfolios composed of LLM-selected stocks and weights determined by classical mean-variance optimization consistently outperform purely LLM-weighted portfolios. The authors argue that LLMs currently lack the ability to navigate shifting market dynamics and are therefore best used as supporting tools alongside quantitative frameworks.

Factor mining: LLM-based factor mining leverages LLMs to reduce the human burden associated with discovering new features for quantitative models. Alpha-GPT 2.0 explores a human-in-the-loop agentic AI system designed to streamline alpha research [85]. The framework has two main characteristics: a strict Standard Operating Procedure (SOP) and human-in-the-loop intervention at any stage of the process. Similarly, Kou et al. [86] introduce a three-stage framework in which an LLM first discovers novel alphas, a second layer of agents evaluates them across various market conditions, and machine learning techniques are used to optimize performance while reducing human intervention.
More recent work increasingly emphasizes interpretable alphas. For instance, LLMFactor [87] uses LLMs to explain market movements by processing multi-modal financial data. Likewise, FactorMAD [88] proposes a multi-agent framework that uses debate to generate interpretable alphas, introducing a feedback standardization scheme that splits feedback into code-specific, model-specific, and methodology-specific categories. QuantAlpha adopts an evolutionary approach to alpha generation through crossover and mutation, maintaining high-performing factor branches while pruning underperforming ones [89]. Finally, FactorMiner provides a modular, self-improving, and memory-sensitive framework optimized for factor mining [90].

Model risk management: LLMs are effective at reviewing financial models and code. Okpala et al. [91] propose an agentic framework structured around two main components: a modeling agent and a model risk management agent. The modeling agent develops data analysis models and extracts features, evaluates them, and passes the outputs to the risk-management agent, which performs compliance checks, model replication, conceptual soundness reviews, and documentation verification. Complementing this approach, Bhattacharyya et al. [92] propose a validation framework tailored for generative AI within financial institutions. Their framework combines conceptual soundness checks – covering model selection, data privacy, prompt specification, explainability, and fairness – with rigorous outcome analysis procedures such as hallucination detection, toxicity scoring, and robustness testing.

Financial time-series forecasting: Long-term financial time-series forecasting can be significantly improved by a multi-agent transformer architecture [93], whose key feature is its ability to learn underlying relationships between variables. Similarly, FinCast [53] is a foundation model developed specifically for forecasting financial time series. Its defining characteristic is a learnable frequency embedding with temporal resolution, which helps capture the cyclical and seasonal patterns of financial data. Finally, StockTime [94] is an LLM-based framework that integrates textual data with transformed stock prices, statistical metrics, and cross-stock correlations to predict future price movements.

Autonomous trading frameworks: These are agentic systems that require minimal to no human intervention. AI-Traders [95], for example, uses live information and emulates a financial analyst to support trading decisions. TradingAgents [83] similarly mirrors the structure of an investment firm through specialized researcher, trader, and portfolio-manager agents.

Agents and reinforcement learning: Wang et al. propose a multi-agent reinforcement learning (MARL) framework for market making [96]. Using proximal policy optimization (PPO), they show that an opponent-aware agent can capture a significant market share while limiting market impact. However, PPO often lacks generalization and remains closely tied to the structure of its reward function. To address these limitations, FinO1 [97] compares key reinforcement learning algorithms and shows that Group Relative Policy Optimization (GRPO) outperforms PPO and Direct Preference Optimization (DPO) in financial settings. The authors also introduce FinReason, a benchmark for financial reasoning. Similarly, TradingR1 [98] adopts GRPO to enhance LLM reasoning in investment decision-making, suggesting that reinforcement learning can improve both structured reasoning and market awareness.

5 Risk and Limitations

Agentic AI inherits many of the risks and limitations of its underlying LLMs, including hallucinations, diminishing returns in model training, catastrophic forgetting, and cost-related issues.

5.1 Risks

Hallucination: Hallucination refers to the generation of plausible but nonfactual content [99]. In finance, such outputs can contaminate analyses and lead to costly errors, such as citing a corporate statement that was never actually published. This creates systematic risk [76] for text-analysis-based agentic systems that rely on feedback loops, such as AlphaAgents [100] and TradingAgents [83]. As illustrated in Figure 6, Omniscience[26] is a continuously updated benchmark developed by Arti-

[26] https://artificialanalysis.ai/evaluations/omniscience

ficial Analysis to measure model hallucination. Its evaluations currently identify Gemini 3.1 Pro, Claude 4.8 Opus, and Gemini 3.5 Flash among the top-performing models.
Another prominent benchmark is SimpleQA [101], developed by OpenAI and containing 4,236 questions, with top-performing models including[27] Gemini 3.1 Pro, ChatGPT o3, and Qwen 3. While these benchmarks provide an estimate of error rates, accurately measuring the actual occurrence of hallucinations remains highly challenging due to issues such as benchmark data contamination [102].

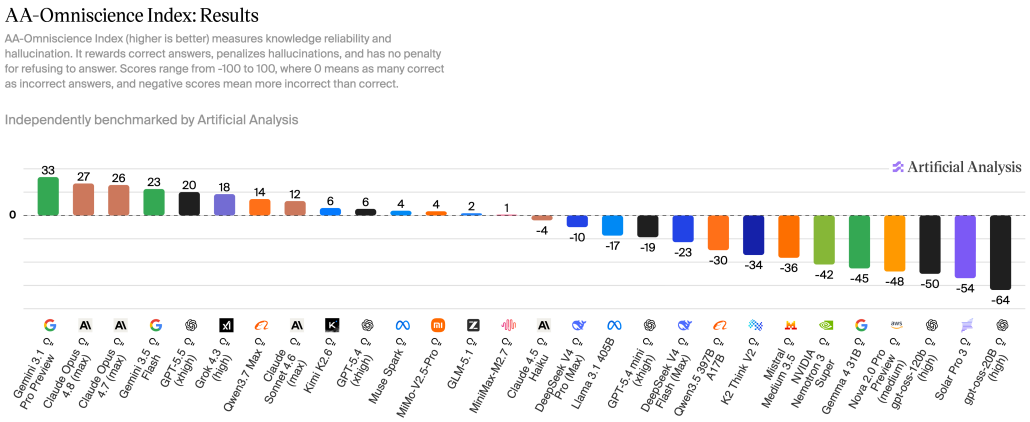


Figure 6: Omniscience hallucination and reliability benchmark, 01/06/2026 results

5.2 Ethical, Regulatory, and Cybersecurity Risks

Agentic AI frameworks are vulnerable to attacks and misuse, as the underlying models do not inherently incorporate ethical or regulatory considerations. Many mainstream models adopt strict guardrails[28] and classification systems to detect harmful behavior; however, these measures still have clear limitations. Furthermore, while LLMs are generally less prone to successful exploitation, SLMs can be easier to attack and may be more likely to exhibit unaligned behavior [103]. In addition, SLMs are more prone to provide biased information, enable cyberattacks, and exhibit ethically unaligned behavior.

Unethical and unintentional deception: Chain-of-Thought (CoT) prompting has introduced powerful reasoning capabilities, allowing LLMs to evaluate their own outputs against quality, ethical, and regulatory constraints. However, this also creates a vulnerability for models with fewer internal checks and reasoning steps. Research indicates that models can deceive unintentionally [104]; remarkably, even a 1% presence of misaligned data in training can amplify unethical behaviors by up

[27] `https://www.kaggle.com/benchmarks/openai/simpleqa`
[28] Mistral guard-railing guide, `https://docs.mistral.ai/capabilities/guard-railing`

to 20%. Such deception can stem from malevolent user prompts, but it can also emerge organically from biased training data or user interactions.
Consequently, LLMs can be used as effective persuaders that conform to unethical user preferences or exploit user vulnerabilities to carry out targeted persuasive actions [103]. This capability can facilitate multiple types of misinformation. Specifically, three primary forms have been identified: AI-generated false content, AI-generated deceptive classification, and AI-generated deceptive explanation [105]. Such deceptive outputs can mislead human judgments as well as downstream feedback loops within multi-agent systems. To address these vulnerabilities, mitigation strategies typically incorporate ground-truth databases or real-time web search for automated fact-checking.

Market manipulative behavior: LLMs face incentives similar to those of human traders [106]. In highly profitable situations, models may develop incentives that increase the risk of market manipulation. Larger models may appear less prone to such harmful behavior in the cited study, as scale seems to provide greater resistance to short-term profit incentives under regulatory constraints. By contrast, Byrd's study on an agentic AI trading system [107] shows that these systems can develop incentives to engage in manipulative behavior when maximizing profits. These findings highlight the importance of strict constraints and boundary conditions in agentic reward functions.

Cybersecurity: Jailbreaking refers to adversarial prompting techniques that can also be automated. These techniques aim to circumvent the guardrails and other safety measures established by AI developers. The main vulnerabilities can be grouped into categories such as paraphrasing harmful intent into benign phrasing, length or formatting exploitation, and large-scale iterative methods. For example, the "Best-of-N" jailbreaking approach [108] rewrites a harmful prompt into different formats and iteratively queries the LLM to elicit a restricted response. The study reports a 52% attack success rate when using 10,000 augmented prompts. To assess these vulnerabilities systematically, H4rm3l [109] provides a dedicated benchmark and evaluation framework designed to test the resilience of LLMs against jailbreak attempts.

Consistency of LLMs in classification and summaries: LLMs can exhibit greater classification consistency than human annotators [110], often producing identical evaluations when given the same passage. However, this consistency can drop significantly when processing complex or less standardized documents, such as Q&A transcripts and management discussion summaries. To mitigate this variability, the authors propose an aggregation strategy that classifies the same text multiple times and then aggregates the outputs to produce a stable consensus.

Bias of training sets: AI models, which are currently dominated by LLMs, frequently suffer from bias [111]. This bias is embedded in the training data,

leading models to exhibit specific behavioral tendencies and preferences. For instance, Xie et al. [112] show that LLMs display a distinct confirmation bias, readily retrieving information that is consistent with their training data while only accepting counter-opinions when they are the sole logically coherent option available. Compounding this issue, data contamination [113] occurs when training datasets contain low-quality, inaccurate, misleading, or misinformative text. This can contaminate the model and reduce output quality. Such low-quality data comes not only from unreliable public sources but increasingly from synthetic data, that is, content generated by other LLMs.

Reproducibility of results and non-determinism of outcomes: The reproducibility of LLM-generated results [113] across sessions can vary significantly. This variability complicates the estimation of confidence intervals for outputs and introduces uncertainty into systematic auditing procedures. Even when using a greedy decoder, variations in hardware clusters or compute sizes can lead to different outcomes. This discrepancy stems from the non-associative property of floating-point arithmetic [114]. As the number of iterations increases, cumulative rounding errors propagate, ultimately leading to significantly divergent outcomes.

5.3 Risks Related to Reinforcement Learning Fine-Tuning

Reinforcement learning is commonly used to fine-tune LLMs for specific tasks and can significantly improve the performance of smaller models. However, as summarized in Table 6, these algorithms also introduce limitations, including reward hacking and catastrophic forgetting.

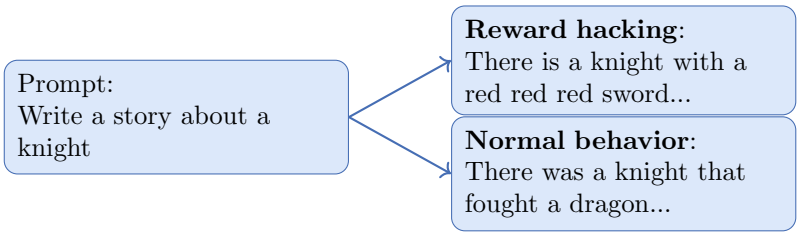


Figure 7: Comparison between reward hacking and normal behavior.

Reward hacking: Reward hacking occurs when an agent exploits loopholes in a reward function rather than learning the intended task or reasoning process. Figure 7 illustrates this behavior through the repeated generation of the same word. Such behavior can create models that appear "good" only within the training distribution. For instance, this may occur through answers derived from data leakage or memorized training data. Fu et al. [115] show that bounded reward functions with diminishing marginal returns help mitigate reward hacking. They also argue that Group Relative Policy Optimization (GRPO) is less prone to reward hacking because it eliminates the separate critic network and relies solely on the

policy network. Similarly, Wang et al. [116] confirm that LLMs tend to exploit vulnerabilities in reward functions. To address this issue, they propose truncating the chain-of-thought reasoning process to force immediate responses at specific stages, which makes a hacking model more likely to reveal its underlying reasoning capabilities and return the correct answer.

Catastrophic forgetting: Catastrophic forgetting occurs when an LLM shows significantly lower performance on previously learned tasks after training on new data [117]. It is primarily driven by excessively large parameter updates during fine-tuning, such as with Proximal Policy Optimization (PPO). By contrast, fine-tuning algorithms such as Group Relative Policy Optimization (GRPO), which rely on smaller, bounded updates, lead to significantly fewer instances of catastrophic forgetting [118].

Category	Issue	Description	Mitigation
Fine-tuning algorithms			
PPO[1]	Large parameter updates	Large steps destabilize previously learned representations	Use bounded rewards and learning-rate scheduling
GRPO[2]	Residual reward hacking	A policy-only architecture reduces, but does not eliminate, hacking risk	Use marginally decreasing reward shaping
Feedback mechanisms			
RLHF[3]	Annotation cost	Human labeling is expensive and cannot scale to large feedback loops	Use RLAIF for general tasks (summary, dialogue)
RLAIF[4]	Foundation-model dependency	Feedback quality is bounded by the capability of the evaluating model	Highlights the role of foundation-model feedback loops in agentic pipelines
Reward hacking			
PPO	Proxy metric exploitation	The agent optimizes the reward signal, which may induce comfort-zone bias and data leakage	Use bounded, marginally decreasing rewards; prefer GRPO
PPO / GRPO	Loophole exploitation	LLMs identify structural gaps in reward functions	Limit chain-of-thought exposure and request intermediate answers
Catastrophic forgetting			
PPO	Parameter overwriting	Large gradient updates erase previously learned task representations	Prefer GRPO; use gradual fine-tuning and elastic weight consolidation

Table 6: Reinforcement learning limitations and mitigations for LLM fine-tuning.

[1]Proximal Policy Optimization
[2]Group Relative Policy Optimization
[3]Reinforcement Learning from Human Feedback
[4]Reinforcement Learning from AI Feedback

5.4 The cost of reasoning

Reasoning models have shown significant potential in AI. However, the intermediate steps they require can substantially increase computational requirements.

Scalability and resource constraints: Computational requirements – or API costs – increase substantially for reasoning models that use Chain-of-Thought or more complex architectures such as Reflexion [23]. The two major contributors to VRAM usage in production are the model weights and the Key-Value (KV) cache, the latter of which scales directly with context length, as shown in Table 7.

Precision	Bits / bytes per element	KV-cache VRAM at 10K context	KV-cache VRAM at 50K context
FP16 (baseline)	16 bits / 2.0 bytes	3.36 GB	16.78 GB
FP8	8 bits / 1.0 byte	1.68 GB	8.39 GB
INT4	4 bits / 0.5 byte	0.84 GB	4.19 GB
3.5-bit	3.5 bits / 0.4375 byte	0.74 GB	3.67 GB
2.5-bit	2.5 bits / 0.3125 byte	0.53 GB	2.62 GB

Table 7: KV-cache VRAM requirements for a 70B-parameter LLM (batch = 1).

The VRAM footprint of model weights depends on the chosen precision format. As shown in Table 8, inference is commonly performed in 16-bit floating point (FP16), which requires approximately 140 GB of VRAM for a 70B-parameter model. In addition, the KV cache stores the attention context and constitutes a significant fraction of memory usage, with requirements that increase rapidly as the context length grows [119].

Precision	Bits / bytes per parameter	Required VRAM for a 70B-parameter model
FP16 (baseline)	16 bits / 2 bytes	140 GB
FP8	8 bits / 1 byte	70 GB
INT8	8 bits / 1 byte	70 GB
INT4	4 bits / 0.5 byte	35 GB
3.5-bit	3.5 bits / 0.4375 byte	30.6 GB
2-bit	2 bits / 0.25 byte	17.5 GB

Table 8: Estimated VRAM required to store the weights of a 70B-parameter LLM.

A common approach to reducing these computational requirements is quantization, which maps values to lower-precision formats. For model weights, quantization typically preserves most of the accuracy when moving from 16-bit floating point (FP16) to 8-bit integer (INT8) [120], while significantly reducing VRAM usage, as shown in Table 8. In this context, the required memory can be approximated as VRAM $\approx$ Parameters $\times$ Bytes per parameter. More recently, TurboQuant [121] has

enabled 3.5-bit and 2.5-bit KV-cache quantization with negligible accuracy loss, reducing KV-cache memory by up to 3.5× compared with FP8.

AI is energy-intensive. For example, a medium-sized 70B-parameter model such as Llama 3 can consume up to 350 Wh of GPU energy per query when used with a complex reasoning framework such as Reflexion. Over 100 queries, this corresponds to approximately the daily energy consumption of two households [122]. In addition, Table 9 shows that token input limits constrain a model's ability to process large amounts of information within a single inference pass. More generally, larger models do not necessarily perform better on domain-specific tasks, as shown by the DeepSeek team [36], which highlights the importance of training specialized, fine-tuned models for targeted applications. Finally, data quality remains a major concern: for smaller models in particular, poor-quality data can severely degrade accuracy and performance, and a larger quantity of training data cannot compensate for low-quality data [123].

Model	**Developer**	**Context window**
Llama 4 Scout / Maverick	Meta[1]	10M / 1M tokens
Gemini 3.1 Pro / Flash	Google[2]	1M tokens
Claude Sonnet 4.6 / Opus 4.6	Anthropic[3]	1M tokens
GPT-5.4 / GPT-4.1	OpenAI[4]	1M tokens
Mistral Large 3	Mistral AI[5]	128K tokens
DeepSeek R1 / V3.2	DeepSeek[6]	128K tokens
Grok 4.1	xAI[7]	128K–2M tokens
Qwen3-235B	Alibaba[8]	128K–1M tokens

Table 9: Context window comparison across frontier LLMs (2025–2026).

[1] https://ai.meta.com/blog/llama-4-multimodal-intelligence/
[2] https://deepmind.google/technologies/gemini/
[3] https://www.anthropic.com/claude
[4] https://platform.openai.com/docs/models
[5] https://mistral.ai/
[6] https://www.deepseek.com
[7] https://x.ai/blog/grok
[8] https://qwenlm.github.io/blog/qwen3/

Cost awareness: Cost is a key consideration in agentic architectures, particularly because the most advanced models can cost around $5 per million input tokens and up to $25 per million output tokens. This makes system design an important trade-off: specialized, lightweight models often perform better on domain-specific tasks while also offering a substantial cost advantage over large general-purpose models. In addition, complex reasoning frameworks such as Reflexion provide diminishing gains in output quality while significantly increasing operational costs [124].

Attention dilution: As input context length increases, LLMs tend to lose attention. This limitation affects dense prompts, long reasoning traces, and extended context windows. LLMs also struggle to retrieve relevant information when lexical overlap between the query and the source text is low [125]. Retrieval performance is commonly assessed using the "Needle in a Haystack" benchmark [126], where the model must identify an isolated sentence embedded in a long, otherwise unrelated text. While reasoning models may perform well when retrieval relies on strong lexical similarity, they are less effective at deep associative reasoning. To address this limitation, NoLima [125] introduces a benchmark specifically designed to evaluate associative capabilities under long-context constraints, and the authors report a marked performance decline as context length increases.

6 Competitor analysis

Two asset managers recently published research on the use of agentic AI systems within their respective workflows. Given the proprietary nature of AI systems, competitors tend to disclose only enough to illustrate capabilities rather than implementation choices, so this analysis relies exclusively on publicly available information. BlackRock's AlphaAgents proposes a framework for fundamental research workflows and produces reports that resemble those of professional equity research teams. In parallel, Man Group describes an approach for integrating AI into quantitative research workflows, showing that AI can enable researchers to evaluate a substantially larger number of ideas and hypotheses.

6.1 BlackRock – AlphaAgents

AlphaAgents [100] is a multi-agent, modular AI system designed to replicate a fundamental equity research pipeline. As shown in Figure 8, the framework is structured into three stages, each with a distinct parallelism pattern. The first stage focuses on information gathering, the second on analysis and synthesis, and the third on report generation.

- **Research**: three specialized agents – Valuation, Fundamental, and Sentiment – operate in parallel, analyzing price and volume data, 10-K and 10-Q filings, and financial news, respectively.
- **Debate**: the agents issue independent recommendations based on their assigned data modality and then engage in a structured round-robin debate. If consensus is not reached after a fixed number of rounds, majority voting is applied. Persistent disagreement leads to a conservative Hold recommendation.
- **Output**: the final report produces a Buy or Sell signal according to the investor's profile, which may be risk-averse, risk-neutral, or risk-seeking.

The authors argue that the resulting signals can serve as inputs to downstream portfolio construction, augmenting both Mean-Variance Optimization and the Black-Litterman model with LLM-derived views. AlphaAgents' main contribution is to

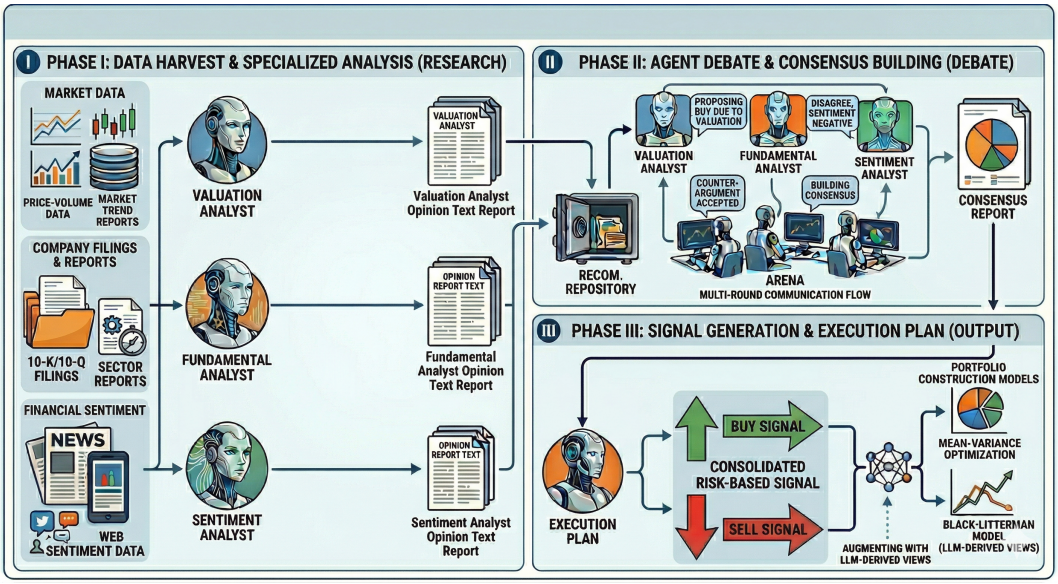


Figure 8: Blackrock AlphaAgents illustration

provide a comprehensive framework for building multi-agent systems in investment management, combining role-based specialization with a structured consensus mechanism.

Regarding failure modes, the authors contend that the debate mechanism is intended to mitigate textual hallucinations by requiring inter-agent consistency before a recommendation is accepted, while numerical values are checked using external mathematical calculators. However, this safeguard remains inherently limited: it can only detect *inconsistent* hallucinations, i.e., those that differ across agents. Hallucinations that are reproduced consistently across all agents, whether due to shared model priors or a common factual gap, will pass through the debate filter undetected and therefore still require external ground-truth verification.

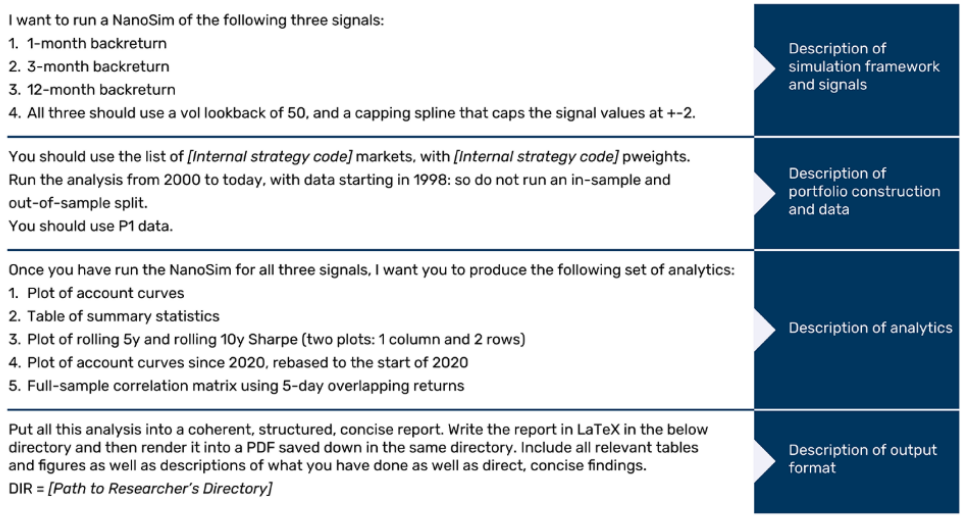
6.2 Man Group – AlphaTrend / Alpha Assistant

Man Group presented Alpha Assistant[29] and AlphaTrend[30], two agentic AI systems designed to support quantitative research. Alpha Assistant functions as an integrated coding assistant that bridges generic LLM outputs with actionable research within the firm's proprietary environment. AlphaTrend, by contrast, is a specialized agentic workflow for trend-following signal research, organized as a predefined pipeline that generates, implements, and evaluates signal proposals.

[29] https://www.man.com/insights/ai-agents-trend
[30] https://www.man.com/insights/alphatrend-agentic-research-workflows

Alpha Assistant is an iterative multi-agent AI system whose main features include the use of firm-specific terminology and validated knowledge bases for financial formulas and code. The system comprises the following components:

- **Orchestrator**: refines the user prompt and decomposes it into a task list for downstream agents, as illustrated in Figure 9.



Source: Man Group. P1 = internal price data library, pweights = internal market weights.

Figure 9: Alpha Assistant's orchestrator output

- **Data exploration agent**: selects the most appropriate dataset for the task.
- **Backtesting agent**: executes the backtest associated with the research idea.
- **Backtest analytics agent**: generates performance analytics from the backtest results.

The output is standardized into a consistent format to improve readability and facilitate evaluation, as shown in Figure 10.

The authors identify two structural limitations of current LLM architectures. First, **attention dilution** arises when the context becomes excessively large, causing the model to lose focus on relevant details and making it counterproductive to load the system with every available document or tool. Second, **limited context length** constrains the system's ability to handle end-to-end research tasks requiring sustained reasoning across multiple stages, as summarized by the representative context window sizes in Table 9. Although Alpha Assistant does not include an explicit hallucination-mitigation mechanism, its use of firm-specific terminology, proprietary code libraries, and validated financial formulas implicitly narrows the generation space, reducing the likelihood of generic or fabricated outputs.

BackReturn Signal Analysis: A Comparative Study of 1-Month, 3-Month, and 12-Month Momentum Strategies

Martin Luk

September 17, 2025

1 Executive Summary

This report presents a comprehensive analysis of three BackReturn momentum strategie implemented on the universe from 2000 to 2025. The strategies utilize 1-month, 3-month, and 12-month lookback periods with consistent volatility normalization and signal capping. Key findings indicate that longer-term momentum (12-month) delivers superior risk-adjusted returns with a Sharpe ratio of 0.896, compared to 0.754 for 3-month and 0.712 for 1-month strategies.

2 Methodology

2.1 Signal Construction

The BackReturn signals were constructed using the AHL BackReturn component with the following specifications:

- Volatility Normalization: 50-day exponentially weighted volatility lookback using DailyVolatilitySingleLookback
- Signal Periods:
  - 1-Month: 21 business days lookback
  - 3-Month: 63 business days lookback
  - 12-Month: 252 business days lookback
- Signal Capping: Linear spline capping at ± 2.0 standard deviations
- Universe: strategy markets (122 instruments across asset classes)
- Data Source: P1 library with daily frequency

3 Results

3.1 Performance Summary

Table 1 presents comprehensive performance statistics for all three strategies.

Table 1: Performance Summary Statistics

Metric	1-Month	3-Month	12-Month
Annualized Return(%)	11.40	12.06	14.47
Annualized Volatility(%)	15.96	15.90	15.98
Sharpe Ratio	0.712	0.754	0.896
Maximum Drawdown(%)	-33.42	-33.13	-32.45
Calmar Ratio	0.341	0.364	0.446
Sortino Ratio	1.017	0.949	1.003
Total Return(%)	292.2	309.4	371.6
Skewnes	1.347	0.803	0.170
Kurtosis	4.971	2.207	1.142
Diversification Benefit	3.108	3.030	2.999

3.2 Account Curve Analysis

Figure 1 displays the cumulative performance of all three strategies over the full backtest period.

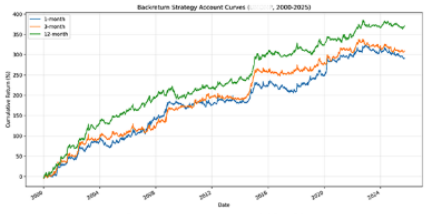


Figure 1: Account Curves: 2000-2025

3.5 Correlation Analysis

Table 2 presents the correlation matrix using 5-day overlapping returns, providing insights into strategy diversification benefits.

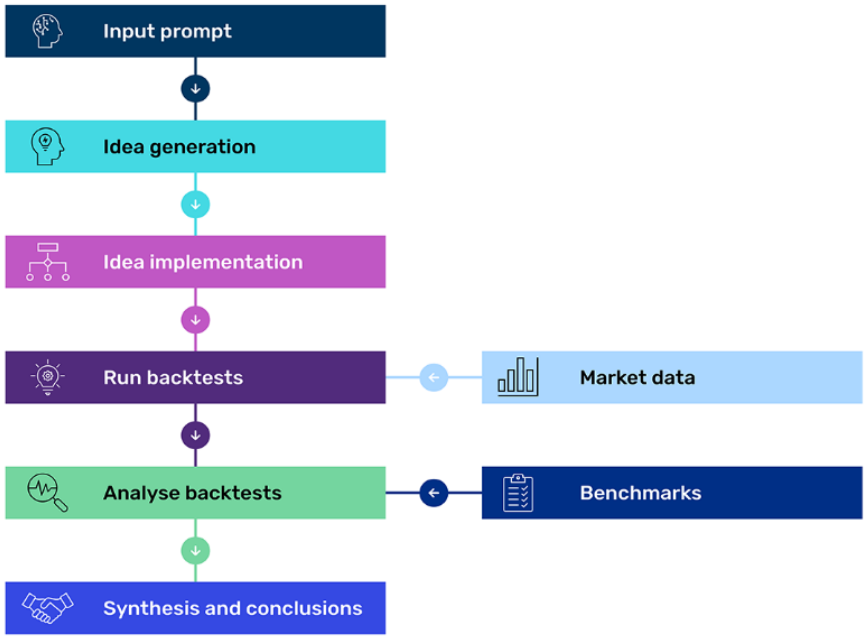
Table 2: Correlation Matrix (5-day over lapping returns)

Strategy	1-Month	3-Month	12-Month
1-Month BackReturn	1.000	0.663	0.344
3-Month BackReturn	0.663	1.000	0.589
12-Month BackReturn	0.344	0.589	1.000

Source: Man Group.

Figure 10: Alpha Assistant final output

AlphaTrend, in contrast, is a specialized agentic research system for trend-following strategies. As illustrated in Figure 11, it is organized as a directed pipeline with pre-specified stages, where independent nodes can execute concurrently. This structure makes intermediate outputs auditable and facilitates reproducibility, since the inputs and outputs of each stage are explicitly specified and retrievable.



Source: Man Group. Schematic illustration.

Figure 11: AlphaTrend agentic research workflow

To mitigate hallucinations and increase output diversity, the authors propose a parallel LLM-querying mechanism in which multiple candidate outputs are generated and compared. Because specific hallucinations are unlikely to be reproduced consistently across independent runs, inconsistent outputs can be identified and discarded. However, the system cannot detect hallucinations that are reproduced consistently across all runs, which still requires ground-truth verification.

7 Conclusion

This review addressed two questions: what agentic AI is and how it is structured, and how it is applied in finance together with the associated risks. Agentic systems extend base LLMs through tool augmentation, communication protocols, and orchestration across specialized agents. A recurring finding is that architectural complexity does not guarantee better performance: the quality of the sub-agents can matter more than the orchestration layer, and additional orchestration can sometimes degrade results by amplifying hallucinations and exacerbating context limitations. In fine-tuning, the shift from PPO toward GRPO and its variants can deliver comparable reasoning performance at lower computational cost, while mitigating reward hacking and catastrophic forgetting.
The applications surveyed include textual processing, portfolio management, factor discovery, risk management, forecasting, and autonomous trading. These applications show that agentic systems can already reduce analytical burden and expand

the space of testable hypotheses. Yet the same limitations recur across domains. Hallucination remains the central obstacle, as it can propagate through multi-agent feedback loops, while output non-determinism and reproducibility challenges complicate the auditing required for financial deployment. Notably, both the academic literature and the industry frameworks reviewed here rely on debate, parallel querying, and human oversight to mitigate hallucinations, while acknowledging that consistent hallucinations still require external ground-truth verification.

In conclusion, although progress in agentic AI has been substantial, important challenges remain before such systems can operate autonomously in finance, with hallucination and reproducibility among the most critical. The current frontier lies in developing reliable multi-agent systems for complex tasks; however, given the high-stakes nature of financial decision-making, it remains advisable to keep a human in the loop and to embed explicit verification steps rather than deploy these systems unsupervised. By surveying the structure, applications, and risks of agentic AI across these domains, this review maps the current state of efforts to build such reliable systems.

A Fine-tuning paradigms

This appendix provides a more detailed description of the main paradigms used in LLM fine-tuning.

- *Supervised fine-tuning* (SFT) [25] is the classical approach to fine-tuning LLMs without reinforcement learning. The model is trained on curated instruction–response pairs. Its main limitation is its dependence on high-quality labeled data, which typically requires human annotation. Zhou et al. show that high-quality annotation is preferable to larger quantities of lower-quality data [127]. As noted earlier, supervised fine-tuning is often used as the starting point for subsequent RL fine-tuning.

- *Reinforcement learning with human feedback* (RLHF) [28][29] is the dominant paradigm for fine-tuning LLMs to align them with human intent. In its standard formulation, a pool of human annotators evaluates pairs of model outputs and indicates which response is preferable; these preference labels are then used to train a reward model, which in turn provides the signal for reinforcement-learning updates of the policy. The main limitation of this approach is scalability: collecting high-quality human preference annotations is both time-consuming and costly, which constrains how broadly and how frequently the method can be applied.

- *Reinforcement learning with AI feedback* (RLAIF) [30][31] mirrors the RLHF pipeline, but with preference labels generated by an AI evaluator rather than human annotators. Lee et al. [30] show that RLAIF achieves performance comparable to RLHF on alignment-oriented tasks such as summarization and dialogue generation, suggesting that AI-generated feedback can serve as a scalable substitute for human annotation. This property underlies many automated feedback loops in agentic AI architectures. However, on more complex tasks, AI-generated preference labels can become noisy and unstable [31].

- *Reinforcement learning with verifiable rewards* (RLVR) [22][33][34] fine-tunes models using objective, outcome-based reward signals. The model's output is evaluated directly against ground-truth results, with a reward granted only upon an exact match. This approach is primarily used for tasks that are readily verifiable, such as mathematical reasoning and code generation. Wen et al. [33] show that, despite relying only on answer-level correctness signals, RLVR implicitly incentivizes correct intermediate reasoning, thereby improving chain-of-thought quality and overall reasoning performance. A key limitation of RLVR is its narrow applicability: the requirement for objective, binary verification restricts it to a limited class of tasks.

B Proximal policy optimization - PPO

Proximal Policy Optimization (PPO) is an algorithm developed by OpenAI that belongs to the family of policy-gradient methods. The algorithm trains two networks: a policy network and a value network. Its core idea is to update the policy gradually in order to prevent performance collapse. Fine-tuning with PPO requires defining a reward for each action.

This is achieved through the *clipped surrogate objective* in Eq. (1), which corresponds to the policy network.

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min\Big(r_t(\theta)\, \hat{A}_t,\ \text{clip}(r_t(\theta),\, 1-\varepsilon,\, 1+\varepsilon)\, \hat{A}_t \Big) \right] \tag{1}$$

The main idea is to clip the policy update whenever the change is too large. More specifically, each update is constrained to remain between $1-\epsilon$ and $1+\epsilon$. The probability ratio is defined as follows:

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} \tag{2}$$

The second component is the value objective in Eq. (2), which corresponds to the value network and minimizes the squared deviation from the target value.

$$L^{\text{VF}}(\theta) \;=\; \hat{\mathbb{E}}_t \left[\Big(V_\theta(s_t) - \hat{V}_t^{\,\text{targ}} \Big)^2 \right] \tag{3}$$

The last term in the full PPO objective is the entropy bonus. The entropy bonus assigns a positive value when the probability distribution is more spread out rather than concentrated; this encourages exploration in uncertain environments rather than premature convergence.

$$S[\pi_\theta](s_t) \;=\; -\sum_a \pi_\theta(a \mid s_t)\, \log \pi_\theta(a \mid s_t) \tag{4}$$

The full objective to optimize is therefore:

$$L^{\text{PPO}}(\theta) \;=\; \hat{\mathbb{E}}_t \big[L^{\text{CLIP}}(\theta) \;-\; c_1\, L^{\text{VF}}(\theta) \;+\; c_2\, S[\pi_\theta](s_t) \big] \tag{5}$$

B.0.1 PPO with Kullback–Leibler divergence

In RLHF settings, PPO is often modified to better control policy updates. In particular, the entropy bonus is replaced by a Kullback–Leibler (KL) divergence penalty to constrain policy updates and reduce the risk of reward hacking [29].

$$L^{\text{RLHF}}(\theta) = \hat{\mathbb{E}}_t \Big[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) - \beta\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Big] \tag{6}$$

The KL divergence measures the distance between the current policy distribution and a reference policy distribution. The hyperparameter β controls the strength of this regularization, thereby limiting overly large policy updates.

C Group Relative Policy Optimization - GRPO

Group Relative Policy Optimization (GRPO) [36] is a state-of-the-art reinforcement-learning algorithm introduced by DeepSeek. A key architectural departure from PPO is the removal of the value network, which substantially reduces memory and computational costs during training. In DeepSeekMath, the authors caution that the observed performance gains do not necessarily reflect improvements in fundamental reasoning ability, but may instead result from more effective sampling of plausible outputs. The broader diffusion of GRPO across the LLM fine-tuning landscape was accelerated by DeepSeek-R1 [22], which showed that smaller models trained with GRPO could match or surpass much larger ones.

Mechanistically, GRPO operates as follows: given a prompt, the policy model generates a group of G candidate outputs $\{o_i\}_{i=1}^{G}$, each of which is scored by a reward signal. These scores are then normalized within the group to compute a group-relative advantage estimate $\hat{A}_i$ for each output:

$$\hat{A}_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})} \tag{7}$$

The policy is then updated by optimizing the clipped surrogate objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q \sim \mathcal{D} \\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}}} \frac{1}{G} \sum_{i=1}^{G} \left(\min\left[\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} \hat{A}_i,\ \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)},\ 1-\varepsilon,\ 1+\varepsilon \right) \hat{A}_i \right] - \beta\, \mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}] \right) \tag{8}$$

where ε controls the clipping range to prevent excessively large policy updates, and β is a penalty coefficient that regularizes the updated policy π_θ against a reference policy π_{ref} via KL divergence. Outputs with positive advantage, i.e., those performing above the group mean, are reinforced, whereas those with negative advantage are penalized. This steers the model toward higher-reward behaviors without requiring a value model.

D Glossary

The following terms are used throughout the paper. Entries are grouped thematically; within each group, they appear in order of first use.

Agency. The capability of an AI system to direct its own processes, invoke external tools, and maintain control over how a task is completed. Agency is acquired incrementally: a base LLM has none, an AI agent acquires it through tool augmentation, and an agentic AI system extends it further through autonomous goal decomposition and iterative interaction with the environment.

Agent skills. Structured documents, typically Markdown files accompanied by executable scripts and asset templates, that are loaded into an agent's context at runtime. Agent skills provide domain-specific procedural knowledge and formatting conventions without modifying model weights, and they can be composed across multiple simultaneous tasks.

Agent-to-Agent (A2A) protocol. A protocol governing communication between AI agents within a MAS. Each agent carries a JSON capability card; the protocol manages execution status (*pending / in progress / completed / failed*) to enable asynchronous multi-agent execution.

Agentic AI. An autonomous LLM system that accepts a high-level goal rather than step-by-step instructions and adapts its execution plan in response to environmental feedback. Its key characteristics are autonomy and adaptability.

AI agent. An LLM augmented with external tools (search engines, APIs, code interpreters, calculators) that operates within an observation–action feedback loop to complete assigned tasks autonomously.

Alpha. The excess return of an investment strategy relative to a benchmark, attributable to active investment decisions rather than broad market exposure. Alpha discovery and factor mining are primary use cases for LLM-driven agentic pipelines in quantitative finance.

Attention dilution. The degradation in an LLM's ability to attend to relevant tokens as input context length grows, leading to reduced retrieval accuracy on long-context tasks. It is commonly assessed using benchmarks such as Needle in a Haystack and NoLiMa.

AutoGen. A conversation-based multi-agent framework developed by Microsoft in which agents coordinate through structured dialogue rather than a predefined execution graph; it is well suited to multi-agent debate architectures.

Benchmark contamination. The inflation of benchmark scores caused by the presence of evaluation data in a model's training set, which undermines the validity of reported results.

Black-Litterman model. A Bayesian portfolio optimization framework that blends a market-implied equilibrium prior with investor-specified return views using a mixing parameter. LLM-generated views on expected returns have been proposed as a scalable source of such priors.

CAMEF. Causal-Augmented Multi-Modality Event-Driven Financial Forecasting: a framework that integrates textual encodings of macroeconomic announcements with high-frequency price time series to predict subsequent asset price movements.

Catastrophic forgetting. The degradation of performance on previously learned tasks caused by excessively large gradient updates during fine-tuning that overwrite earlier representations. It is often more pronounced in PPO than in GRPO because PPO uses larger per-step parameter updates.

Centralized orchestration. A multi-agent topology in which a single orchestrator directs all sub-agents; sub-agents do not communicate directly with one another.

Chain-of-thought (CoT). A prompting strategy that decomposes a task into sequential intermediate reasoning steps within a single inference pass, improving logical and quantitative problem solving by making reasoning explicit.

CISPO. A GRPO variant that applies update clipping at the response level rather than per token, so that a small number of unusually confident tokens cannot distort the full gradient step; designed for stable training on long documents.

Context window. The maximum number of tokens a model can process in a single inference call. Frontier models such as Llama 4 Scout, Gemini 3.1 Pro, and Claude Sonnet 4.6 support up to 1M–10M tokens; DeepSeek R1 and Mistral Large 3 are capped at 128K tokens.

DAPO. A GRPO variant that addresses several known weaknesses simultaneously, including adaptive clipping based on per-sample performance, removal of uninformative training examples, and reward shaping to discourage unnecessarily verbose outputs.

Decentralized orchestration. A multi-agent topology without a central coordinator. Agents independently generate solutions and converge to a final output through debate or voting mechanisms.

Dr. GRPO. A variant that identifies two standard correction steps in GRPO as unintentional distortions of the learning signal and removes them, yielding a simpler and more theoretically grounded training objective.

Evolution strategies (ES). A backpropagation-free fine-tuning paradigm that evaluates a population of N candidate models obtained by adding Gaussian noise to a pretrained model's parameters, then updates the weights as a

reward-weighted sum of the noise vectors. It parallelizes naturally and requires only about 20% of the training samples needed by PPO/GRPO, with a substantially lower VRAM footprint.

Factor mining. The process of discovering novel quantitative signals (factors) that predict asset returns. LLM-driven frameworks such as Alpha-GPT 2.0, FactorMiner, and QuantAlpha automate hypothesis generation, evaluation, and optimization in this pipeline.

Few-shot prompting. A prompting strategy in which a small number of worked input–output examples are included in the task description. This conditions the model to follow the demonstrated pattern. In financial agentic pipelines, few-shot prompting has been shown to achieve strong performance on data-transformation tasks and is also used within chain-of-thought prompting to make intermediate reasoning steps explicit [5].

Finance Reasoning (FinanceReasoning). A benchmark that evaluates numerical financial reasoning via chain-of-thought, programming, and tabular analysis tasks; suitable for assessing models tasked with financial modeling and quantitative analysis.

FinanceBench. A benchmark of 10,234 question–answer pairs about public companies' financial statements, primarily testing factual retrieval capabilities of sub-agents (e.g., extracting operating income or inventory figures).

FinBen. A holistic financial AI benchmark covering the full capability spectrum: information extraction, sentiment analysis, forecasting, risk management, and agentic trading decisions. It is suitable for end-to-end evaluation of agentic AI systems in finance.

Foundation model (FM). A generic model pretrained at scale and subsequently fine-tuned for a domain-specific purpose. Financial foundation models are categorized as language, time-series, vision-language, or multimodal, depending on the data modality they process.

FP16 / FP8 / INT8 / INT4. Numerical precision formats for storing model weights. FP16 (16-bit floating point) is the current inference standard; FP8 and INT8 halve memory requirements relative to FP16; INT4 reduces them by 4×. VRAM required for a 70B model scales from 140 GB (FP16) down to 35 GB (INT4).

Graph of thoughts (GoT). An extension of CoT reasoning that enables the aggregation and merging of multiple independent reasoning paths into a graph structure, allowing diverse intermediate conclusions to be synthesized before a final answer is produced.

GraphRAG. An extension of RAG that constructs a knowledge graph from unstructured source text, enabling graph-structured retrieval and richer multi-hop reasoning over the retrieved information.

Group Relative Policy Optimization (GRPO). A critic-free RL algorithm introduced with DeepSeekMath that removes the value network entirely. Given a prompt, the policy generates a group of G candidate outputs; rewards are normalized within the group to compute a group-relative advantage estimate, which drives policy updates via a clipped objective augmented with a KL-divergence penalty.

GSPO (Group Sequence Policy Optimization). A GRPO variant that evaluates reward at the full-response level rather than token by token, preventing small per-token errors from compounding over long outputs.

Hallucination. The generation of plausible but factually incorrect content by a language model. In financial settings, hallucinated outputs—for example, fabricated corporate statements—can propagate through multi-agent feedback loops and create systemic errors. It is commonly assessed using the Omniscience and SimpleQA benchmarks.

Humanity's Last Exam (HLE). A challenging general-knowledge benchmark used to assess broad reasoning capabilities of fine-tuned models, and one of the primary axes for evaluating alignment quality.

iGRPO. A GRPO variant in which the model generates self-critiques of its own answers; these critiques are used as additional feedback signals, enabling continued improvement on problems where the model rarely produces a correct answer.

Key-value (KV) cache. The stored attention key and value representations for all tokens processed so far in a sequence. The KV cache enables efficient autoregressive generation by avoiding recomputation, but its memory footprint scales directly with context length and batch size.

KL divergence (in RL fine-tuning). The Kullback–Leibler divergence $D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$ is used as a regularization penalty in PPO and GRPO objectives to prevent the updated policy from deviating excessively from a reference policy, thereby limiting reward hacking.

Knowledge graph (KG). A structured representation of information as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $v \in \mathcal{V}$ denote entities (companies, events, financial indicators) and typed edges $e \in \mathcal{E}$ encode semantic relationships between them.

LangGraph. A graph-based orchestration framework in which a directed execution graph routes subtasks to appropriate agents or tools; the system state, computational nodes, and conditional edges define the complete execution flow.

Large language model (LLM). A transformer-based language model with roughly 10B parameters or more. LLMs excel at complex reasoning, open-ended generation, multi-step planning, and orchestration tasks, at the cost of a high memory footprint and latency.

LlamaIndex. A data-retrieval framework used as a document-fetch tool within agentic pipelines, enabling structured ingestion and retrieval from external knowledge bases.

Low-rank adaptation (LoRA). A PEFT method that represents the weight update matrix ΔW as the product of two low-rank matrices BA (where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $r \ll \min(d, k)$), dramatically reducing the number of trainable parameters while preserving most task-specific performance.

Mean-variance optimization (MVO). A classical portfolio construction technique that selects asset weights to maximize expected return for a given level of portfolio variance, tracing the efficient frontier. Recent research demonstrates that LLM-selected stocks with MVO weights can outperform purely LLM-weighted portfolios.

Mixed orchestration. A hybrid topology combining a top-level orchestrator for task decomposition and resource allocation with direct inter-agent communication within local agent clusters, distinguishing it from purely centralized or decentralized designs.

Model context protocol (MCP). An open-source standard that provides a unified interface for connecting AI agents to external data sources, computational tools (calculators, code interpreters), and third-party APIs.

Multi-agent reinforcement learning (MARL). The application of reinforcement learning to environments with multiple simultaneously learning agents. In market making, a MARL framework using PPO trains opponent-aware agents that capture significant market share without disproportionate market impact.

Multi-agent system (MAS). An orchestration framework in which multiple AI agents, typically coordinated by an LLM orchestrator and supported by SLM sub-agents, decompose, execute, and aggregate complex workflows collaboratively.

Needle in a Haystack. A long-context evaluation methodology that assesses an LLM's ability to retrieve a single piece of relevant information embedded within an extended, unrelated context. It tests direct lexical retrieval, but not associative reasoning.

NoLiMa. A long-context benchmark designed specifically to test associative (non-lexical) reasoning under extended context constraints, complementing Needle in a Haystack by probing deeper relational inference.

OpenRLHF. An RL fine-tuning framework that disaggregates the policy, reference, critic, and reward models into independent GPU workers, eliminating co-location memory bottlenecks and enabling tensor, data, and sequence parallelism for high-parameter models.

Orchestrator. The central coordinating component of a MAS, responsible for decomposing user objectives into subtasks, assigning tasks to specialized subagents, managing execution state, and aggregating outputs into a final response.

Parameter-efficient fine-tuning (PEFT). A class of methods that update only a subset of model parameters, reducing computational and memory requirements relative to full-parameter fine-tuning. LoRA and QLoRA are the dominant PEFT approaches.

Proximal Policy Optimization (PPO). A policy-gradient RL algorithm that jointly trains a policy network and a value network. The clipped surrogate objective constrains each gradient step to remain within a ratio of $[1-\varepsilon,\ 1+\varepsilon]$ of the previous policy, preventing catastrophic performance collapse from overly aggressive updates.

Quantization. A model compression technique that maps weights or activations from a higher-precision format (e.g., FP16) to a lower-precision format (e.g., INT8, INT4, or sub-4-bit). It reduces VRAM requirements substantially with minimal accuracy degradation when moving from FP16 to INT8; the novel TurboQuant method achieves 2.5-bit KV-cache quantization without accuracy loss.

Quantized low-rank adaptation (QLoRA). An extension of LoRA that applies 4-bit quantization to the frozen base-model weights, enabling fine-tuning of models with up to 65B parameters on a single 48 GB GPU.

ReAct (reason + act). An iterative prompting framework that interleaves chain-of-thought reasoning steps with explicit action steps; the agent cycles through *perceive* $\rightarrow$ *think* $\rightarrow$ *act* $\rightarrow$ *observe* until a goal condition is met.

Reflexion. A self-improving agent architecture in which linguistic feedback from prior attempts is stored in long-term memory, enabling subsequent attempts to avoid previously identified errors. It is particularly effective for complex, multi-step tasks.

Regime detection. The identification of structural shifts in macroeconomic or market conditions. LLM-based frameworks such as Alpha-R1, Mind the Shift, and CAMEF use textual encodings of macro releases and Fed statements to detect regime changes and adapt investment strategies accordingly.

REINFORCE++. A GRPO-lineage variant that uses the batch-level average reward rather than group-level statistics as the advantage baseline, producing a fairer and more stable training signal.

Reinforcement learning with AI feedback (RLAIF). A scalable alternative to RLHF in which an AI evaluator replaces human annotators to generate

preference labels. It achieves comparable alignment performance on simpler tasks (summarization, dialogue) but produces noisy labels on complex reasoning tasks.

Reinforcement learning with human feedback (RLHF). A fine-tuning paradigm in which human annotators provide pairwise preferences over model outputs. These labels train a reward model, which in turn drives policy optimization via an RL algorithm. It is constrained by high annotation cost and limited scalability.

Reinforcement learning with verifiable rewards (RLVR). A fine-tuning paradigm using objective, binary outcome signals—reward is granted only on an exact match against a ground-truth answer. It implicitly improves chain-of-thought quality and is restricted to verifiable tasks such as mathematics and code generation.

Retrieval-augmented generation (RAG). An architecture that augments an LLM's parametric knowledge with dynamically retrieved external documents. The retrieval stage identifies the top-K most relevant chunks (ranked by cosine similarity); the generation stage re-ranks and appends these chunks to the model's context. RAG reduces hallucination and extends the model's effective knowledge cutoff to the present day when paired with live web search.

Reward hacking. A failure mode in which a model exploits structural loopholes in the reward function, such as memorized training patterns, data leakage, or degenerate repetition, to maximize measured reward without learning the intended behavior. It can be mitigated by bounded, marginally decreasing reward functions and by GRPO's critic-free architecture.

SimpleQA. An OpenAI benchmark of 4,236 short-form factual questions used to measure model hallucination rates on concise, verifiable claims.

Small language model (SLM). A language model with fewer than roughly 10B parameters. SLMs offer low latency and minimal inference cost, making them preferred as specialized sub-agents for repetitive, domain-specific tasks such as coding or translation.

smolagents. A Hugging Face library that provides a standardized tool-calling interface for AI agents, supporting retrieval, computation, and side-effect tool categories.

Supervised fine-tuning (SFT). The classical fine-tuning approach in which a pretrained model is further trained on curated instruction–response pairs. High annotation quality is preferred over dataset size. SFT serves as the baseline from which RL fine-tuning departs.

TRL (Transformer Reinforcement Learning). A Hugging Face library providing direct access to models hosted on Hugging Face with native support

for GRPO, PPO, and related variants, as well as LoRA and QLoRA; it is often the most accessible option for researchers with limited compute.

Value-at-Risk (VaR). A statistical measure of the maximum expected loss of a portfolio over a given time horizon at a specified confidence level. Multimodal LLM frameworks such as RiskLabs have been proposed to forecast VaR from combined audio and text data.

VAPO. A PPO variant that reintroduces a pretrained value network with separate treatment for positive and negative advantage estimates and specialized handling of long reasoning chains, recovering PPO's performance on extended chain-of-thought tasks.

VC-PPO. A PPO diagnostic variant that identifies value-network degradation rather than policy degradation as the primary cause of PPO instability on long reasoning tasks; fixing the value estimator recovers performance to the level of critic-free methods.

Verl. A ByteDance library for large-scale multi-GPU RL fine-tuning, supporting the latest GRPO and PPO variants with flexible GPU resource co-location for training and inference.

VRAM (video RAM). The dedicated on-device memory of a GPU, which during inference must accommodate both the model weights and the KV cache. VRAM capacity is the primary hardware bottleneck for deploying LLMs.

Zero-shot prompting. A prompting strategy in which the model is given only a task description and no worked examples, relying entirely on knowledge acquired during pretraining to produce a response. Zero-shot prompting serves as the baseline against which more structured strategies such as few-shot prompting and chain-of-thought reasoning are evaluated.

References

[1] Sapkota, R., Roumeliotis, K. I., and Karkee, M. "AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges". In: *Information Fusion*, e103599 (2025).

[2] Hughes, L., Dwivedi, Y. K., Malik, T., Shawosh, M., Albashrawi, M. A., Jeon, I., Dutot, V., Appanderanda, M., Crick, T., De', R., et al. "AI Agents and Agentic Systems: A Multi-Expert Analysis". In: *Journal of Computer Information Systems* 65.4 (2025), pp. 489–517.

[3] Chen, L., Liu, S., Yan, J., Wang, X., Liu, H., Li, C., Jiao, K., Ying, J., Liu, Y. V., Yang, Q., et al. "Advancing Financial Engineering with Foundation Models: Progress, Applications, and Challenges". In: *Engineering* (2025).

[4] Acharya, D. B., Kuppan, K., and Divya, B. "Agentic AI: Autonomous Intelligence for Complex Goals — A Comprehensive Survey". In: *IEEe Access* 13 (2025), pp. 18912–18936.

[5] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.

[6] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. "React: Synergizing Reasoning and Acting in Language Models". In: *arXiv preprint arXiv:2210.03629* (2022).

[7] Hassan, M. T. "Small Models, Large Impact: An SLM-Centric Playbook for Practical Agents". In: *2025 International Conference on Computer and Applications (ICCA)*. IEEE. 2025, pp. 1–6.

[8] Sistla, S. and Evangelist, T. "Navigating the Landscape of Language Models with Comparative Insights: LLM vs. SLM". In: *Journal of Artificial Intelligence* 2.4 (2024), pp. 1629–1633.

[9] Irugalbandara, C., Mahendra, A., Daynauth, R., Arachchige, T. K., Dantanarayana, J., Flautner, K., Tang, L., Kang, Y., and Mars, J. "Scaling Down to Scale up: A Cost-Benefit Analysis of Replacing OpenAI's LLM with Open Source SLMs in Production". In: *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. 2024, pp. 280–291.

[10] Kim, Y., Gu, K., Park, C., Park, C., Schmidgall, S., Heydari, A. A., Yan, Y., Zhang, Z., Yuchen, Z., Liu, Y., et al. "Towards a Science of Scaling Agent Systems". In: *arXiv preprint arXiv:2512.08296* (2025).

[11] Adimulam, A., Gupta, R., and Kumar, S. "The Orchestration of Multi-Agent Systems: Architectures, Protocols, and Enterprise Adoption". In: *arXiv preprint arXiv:2601.13671* (2026).

[12] Wang, J. and Duan, Z. "Agent AI with LangGraph: A Modular Framework for Enhancing Machine Translation Using Large Language Models". In: *arXiv preprint arXiv:2412.03801* (2024).

[13] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., et al. "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation". In: *arXiv preprint arXiv:2308.08155* (2023).

[14] Wang, Z., Cheng, Z., Zhu, H., Fried, D., and Neubig, G. "What Are Tools Anyway? A Survey from the Language Model Perspective". In: *arXiv preprint arXiv:2403.15452* (2024).

[15] Hu, J., Zhong, M., Chen, K., Bai, X., and Zhang, M. "Agentic Tool Use in Large Language Models". In: *arXiv preprint arXiv:2604.00835* (2026).

[16] Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. "Toolformer: Language Models Can Teach Themselves to Use Tools". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 68539–68551.

[17] Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al. "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs". In: *International Conference on Learning Representations*. 2024, pp. 9695–9717.

[18] Xu, R. and Yan, Y. "Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward". In: *arXiv preprint arXiv:2602.12430* (2026).

[19] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-T., Rocktaschel, T., et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[20] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: *arXiv preprint arXiv:2312.10997* (2023).

[21] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. "Lost in the Middle: How Language Models Use Long Contexts". In: *Transactions of the association for computational linguistics* 12 (2024), pp. 157–173.

[22] Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. "DeepSeek-R1 Incentivizes Reasoning in LLMs Through Reinforcement Learning". In: *Nature* 645.8081 (2025), pp. 633–638.

[23] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. "Reflexion: Language Agents with Verbal Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 8634–8652.

[24] Nitarach, N., Sirichotedumrong, W., Pitchayarthorn, P., Taveekitworachai, P., Manakul, P., and Pipatanakul, K. "FinCoT: Grounding Chain-of-Thought in Expert Financial Reasoning". In: *Proceedings of The 10th Workshop on Financial Technology and Natural Language Processing*. 2025, pp. 93–123.

[25] Parthasarathy, V. B., Zafar, A., Khan, A., and Shahid, A. "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities". In: *arXiv preprint arXiv:2408.13296* (2024).

[26] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[27] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. "QLoRA: Efficient Finetuning of Quantized LLMs". In: *Advances in neural information processing systems* 36 (2023), pp. 10088–10115.

[28] Kaufmann, T., Weng, P., Bengs, V., and Hüllermeier, E. "A Survey of Reinforcement Learning from Human Feedback". In: *arXiv preprint arXiv:2312.14925* (2023).

[29] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. "Training Language Models to Follow Instructions with Human Feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[30] Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A., et al. "RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback". In: *arXiv preprint arXiv:2309.00267* (2023).

[31] Lin, J., Li, M., Zhao, X., Lu, W., Zhao, P., Wermter, S., and Wang, D. "Curriculum-RLAIF: Curriculum Alignment with Reinforcement Learning from AI Feedback". In: *Findings of the Association for Computational Linguistics: ACL 2026*. 2026, pp. 33757–33777.

[32] Zhou, H., Huang, H., Zhang, R., Chen, K., Xu, B., Zhu, C., Zhao, T., and Yang, M. "Toward Robust LLM-Based Judges: Taxonomic Bias Evaluation and Debiasing Optimization". In: *arXiv preprint arXiv:2603.08091* (2026).

[33] Wen, X., Liu, Z., Zheng, S., Ye, S., Wu, Z., Wang, Y., Xu, Z., Liang, X., Li, J., Miao, Z., et al. "Reinforcement Learning with Verifiable Rewards Implicitly Incentivizes Correct Reasoning in Base LLMs". In: *arXiv preprint arXiv:2506.14245* (2025).

[34] Stephan, M., Khazatsky, A., Mitchell, E., Chen, A. S., Hsu, S., Sharma, A., and Finn, C. "RLVF: Learning from Verbal Feedback without Overgeneralization". In: *arXiv preprint arXiv:2402.10893* (2024).

[35] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[36] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. "Deepseekmath: Pushing the Limits of Mathematical Reasoning in Open Language Models". In: *arXiv preprint arXiv:2402.03300* (2024).

[37] Hu, J., Liu, J. K., Xu, H., and Shen, W. "REINFORCE++: Stabilizing Critic-Free Policy Optimization with Global Advantage Normalization". In: *arXiv preprint arXiv:2501.03262* (2025).

[38] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. "DAPO: An Open-Source LLM Reinforcement Learning System at Scale". In: *Advances in Neural Information Processing Systems* 38 (2026), pp. 113222–113244.

[39] Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. "Understanding R1-Zero-Like Training: A Critical Perspective". In: *arXiv preprint arXiv:2503.20783* (2025).

[40] Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. "Group Sequence Policy Optimization". In: *arXiv preprint arXiv:2507.18071* (2025).

[41] Chen, A., Li, A., Gong, B., Jiang, B., Fei, B., Yang, B., Shan, B., Yu, C., Wang, C., Zhu, C., et al. "MiniMax-M1: Scaling Test-Time Compute Efficiently with Lightning Attention". In: *arXiv preprint arXiv:2506.13585* (2025).

[42] Hatamizadeh, A., Prabhumoye, S., Gitman, I., Lu, X., Han, S., Ping, W., Choi, Y., and Kautz, J. "iGRPO: Self-Feedback-Driven LLM Reasoning". In: *arXiv preprint arXiv:2602.09000* (2026).

[43] Yue, Y., Yuan, Y., Yu, Q., Zuo, X., Zhu, R., Xu, W., Chen, J., Wang, C., Fan, T., Du, Z., et al. "VAPO: Efficient and Reliable Reinforcement Learning for Advanced Reasoning Tasks". In: *arXiv preprint arXiv:2504.05118* (2025).

[44] Yuan, Y., Yue, Y., Zhu, R., Fan, T., and Yan, L. "What's Behind PPO's Collapse in Long-CoT? Value Optimization Holds the Secret". In: *arXiv preprint arXiv:2503.01491* (2025).

[45] Islam, P., Kannappan, A., Kiela, D., Qian, R., Scherrer, N., and Vidgen, B. "FinanceBench: A New Benchmark for Financial Question Answering". In: *arXiv preprint arXiv:2311.11944* (2023).

[46] Tang, Z., Haihong, E., Ma, Z., He, H., Liu, J., Yang, Z., Rong, Z., Li, R., Ji, k., Huang, Q., et al. "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)*. 2025, pp. 15721–15749.

[47] Xie, Q., Han, W., Chen, Z., Xiang, R., Zhang, X., He, Y., Xiao, M., Li, D., Dai, Y., Feng, D., et al. "FinBen: A Holistic Financial Benchmark for Large Language Models". In: *Advances in neural information processing systems* 37 (2024), pp. 95716–95743.

[48] Qiu, X., Gan, Y., Hayes, C. F., Liang, Q., Xu, Y., Dailey, R., Meyerson, E., Hodjat, B., and Miikkulainen, R. "Evolution Strategies at Scale: LLM Fine-Tuning Beyond Reinforcement Learning". In: *arXiv preprint arXiv:2509.24372* (2025).

[49] Sun, Z., Dang, S., Dai, G., and Ye, H. "ESSAM: A Novel Competitive Evolution Strategies Approach to Reinforcement Learning for Memory Efficient LLMs Fine-Tuning". In: *arXiv preprint arXiv:2602.01003* (2026).

[50] Anjum, K., Arshad, M. A., Hayawi, K., Polyzos, E., Tariq, A., Serhani, M. A., Batool, L., Lund, B., Mannuru, N. R., Bevara, R. V. K., et al. "Domain Specific Benchmarks for Evaluating Multimodal Large Language Models". In: *arXiv preprint arXiv:2506.12958* (2025).

[51] Konstantinidis, T., Iacovides, G., Xu, M., Constantinides, T. G., and Mandic, D. "FinLlama: Financial Sentiment Classification for Algorithmic Trading Applications". In: *arXiv preprint arXiv:2403.12285* (2024).

[52] Yang, H., Liu, X.-Y., and Wang, C. D. "Fingpt: Open-Source Financial Large Language Models". In: *arXiv preprint arXiv:2306.06031* (2023).

[53] Zhu, Z., Chen, H., Qu, Q., and Chung, V. "FinCast: A Foundation Model for Financial Time-Series Forecasting". In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 4539–4549.

[54] Wang, Z., Li, Y., Wu, J., Soon, J., and Zhang, X. "FinVis-GPT: A Multimodal Large Language Model for Financial Chart Analysis". In: *arXiv preprint arXiv:2308.01430* (2023).

[55] Yanglet, X.-Y. L., Cao, Y., and Deng, L. "Multimodal Financial Foundation Models (MFFMs): Progress, Prospects, and Challenges". In: *arXiv preprint arXiv:2506.01973* (2025).

[56] Xie, Q., Li, D., Xiao, M., Jiang, Z., Xiang, R., Zhang, X., Chen, Z., He, Y., Han, W., Yang, Y., et al. "Open-FinLLMs: Open Multimodal Large Language Models for Financial Applications". In: *arXiv e-prints* (2024), arXiv–2408.

[57] Lin, S., Wang, K., and Liu, X.-Y. "Analyzing Cascading Outbreak of GameStop Event: A Practical Approach Using Network Analysis and Large Language Models". In: *Proceedings of the 5th ACM International Conference on AI in Finance*. 2024, pp. 428–436.

[58] Gao, Z. and Xiao, Y. "Enhancing Startup Success Predictions in Venture Capital: A GraphRAG Augmented Multivariate Time Series Method". In: *arXiv preprint arXiv:2408.09420* (2024).

[59] Choi, C., Lopez-Lira, A., Lee, Y., Kwon, J., Kim, M., Hwang, J., Ha, M., Kim, C., Ha, J., Yun, S., et al. "Structuring the Unstructured: A Multi-Agent System for Extracting and Querying Financial KPIs and Guidance". In: *arXiv preprint arXiv:2505.19197* (2025).

[60] Arun, A., Dimino, F., Agarwal, T. P., Sarmah, B., and Pasquali, S. "FinReflectKG: Agentic Construction and Evaluation of Financial Knowledge Graphs". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 283–290.

[61] Brach, W., Košt'ál, K., and Ries, M. "The Effectiveness of Large Language Models in Transforming Unstructured Text to Standardized Formats". In: *IEEE Access* (2025).

[62] Gondhalekar, C., Patel, U., and Yeh, F.-C. "MultiFinRAG: An Optimized Multimodal Retrieval-Augmented Generation Framework for Financial Question Answering". In: *2025 IEEE International Conference on Big Data (BigData)*. IEEE. 2025, pp. 7163–7172.

[63] Choi, C., Kwon, J., Lopez-Lira, A., Kim, C., Kim, M., Hwang, J., Ha, J., Choi, H., Yun, S., Kim, Y., et al. "FinAgentBench: A Benchmark Dataset for Agentic Retrieval in Financial Question Answering". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 632–637.

[64] Zhao, H., Liu, Z., Wu, Z., Li, Y., Yang, T., Shu, P., Xu, S., Dai, H., Zhao, L., Jiang, H., et al. "Revolutionizing Finance with LLMs: An Overview of Applications and Insights". In: *arXiv preprint arXiv:2401.11641* (2024).

[65] Sun, Y., Girón, O., and Ahmed, R. "Decoding the Beige Book: LLM-Powered Sentiment Analysis for Real-Time Recession Forecasting". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 317–325.

[66] Hansen, A. L. and Kazinnik, S. "Can ChatGPT Decipher Fedspeak?" In: *Available at SSRN 4399406* (2023).

[67] Assis, G., Dutra, H., Vianna, D., Meira, W., Silva, A. S. da, and Paes, A. "Language Models for Automated Market Commentary from Corporate Disclosures". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 727–735.

[68] Fatouros, G., Metaxas, K., Soldatos, J., and Karathanassis, M. "MarketSenseAI 2.0: Enhancing Stock Analysis through LLM Agents". In: *arXiv preprint arXiv:2502.00415* (2025).

[69] Sun, Z., Xiao, C., Harit, A., and Yu, J. "Quantifying Semantic Shift in Financial NLP: Robust Metrics for Market Prediction Stability". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 177–184.

[70] Cao, Y., Chen, Z., Kumar, P., Pei, Q., Yu, Y., Li, H., Dimino, F., Ausiello, L., Subbalakshmi, K., and Ndiaye, P. M. "RiskLabs: Predicting Financial Risk Using Large Language Model Based on Multimodal and Multi-Sources Data". In: *2025 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2025, pp. 901–909.

[71] Tang, Y. and Yang, Y. "Mind the Shift: Decoding Monetary Policy Stance from FOMC Statements with Large Language Models". In: *arXiv preprint arXiv:2603.14313* (2026).

[72] Jiang, Z., Zhao, L., Sun, R., Sun, R., Li, Z., Li, J., Jiang, D., Bai, Z., and Hua, C. "Alpha-R1: Alpha Screening with LLM Reasoning via Reinforcement Learning". In: *arXiv preprint arXiv:2512.23515* (2025).

[73] Zhang, Y., Yang, W., Wang, J., Ma, Q., and Xiong, J. "CAMEF: Causal-Augmented Multi-Modality Event-Driven Financial Forecasting by Integrating Time Series Patterns and Salient Macroeconomic Announcements". In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 2025, pp. 3867–3878.

[74] Yu, Y., Yao, Z., Li, H., Deng, Z., Jiang, Y., Cao, Y., Chen, Z., Suchow Jordan W andCui, Z., Liu, R., et al. "FinCon: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 137010–137045.

[75] Guo, Y., Lin, J., Wang, H., Han, Y., Hu, S., Ni, Z., Wang, L., and Chen, M. "SE-Agent: Self-Evolution Trajectory Optimization in Multi-Step Reasoning with LLM-based Agents". In: *Advances in Neural Information Processing Systems* 38 (2025), pp. 116314–116341.

[76] Li, Y., Yang, X., Yang, X., Wang, X., Liu, W., and Bian, J. "R&D-Agent-Quant: A Multi-Agent Framework for Data-Centric Factors and Model Joint Optimization". In: *Advances in Neural Information Processing Systems* 38 (2025).

[77] Li, J., Zhang, J., Li, H., and Shen, Y. "An Agent Framework for Real-Time Financial Information Searching with Large Language Models". In: *arXiv preprint arXiv:2502.15684* (2024).

[78] Cheng, J. and Chin, P. "SocioDojo: Building Lifelong Analytical Agents with Real-world Text and Time Series". In: *The Twelfth International Conference on Learning Representations*. 2024.

[79] Sinha, A., Agarwal, C., and Malo, P. "FinBloom: Knowledge-Grounding Large Language Model with Real-Time Financial Data". In: *Knowledge-Based Systems* (2026). DOI: 10.1016/j.knosys.2026.115559.

[80] Lee, Y., Kim, Y., Kim, J., Kim, S., and Lee, Y. "LLM-Enhanced Black-Litterman Portfolio Optimization". In: *arXiv preprint arXiv:2504.14345* (2025).

[81] Cheng, J. and Chin, P. "Empirical Asset Pricing with Large Language Model Agents". In: *arXiv preprint arXiv:2409.17266* (2025).

[82] Guo, T., Shen, H., Huang, J., Mao, Z., Luo, J., Chen, B., Chen, Z., Liu, L., Xia, b., Liu, X., et al. "MASS: Muli-Agent Simulation Scaling for Portfolio Construction". In: *arXiv preprint arXiv:2505.10278* (2025).

[83] Xiao, Y., Sun, E., Luo, D., and Wang, W. "TradingAgents: Multi-Agents LLM Financial Trading Framework". In: *arXiv preprint arXiv:2412.20138* (2024).

[84] Voronina, A., Romanko, O., Cao, R., Kwon, R. H., and Mendoza-Arriaga, R. "Generative AI-enhanced Sector-based Investment Portfolio Construction". In: *arXiv preprint arXiv:2512.24526* (2025).

[85] Yuan, H., Wang, S., and Guo, J. "Alpha-GPT 2.0: Human-in-the-Loop AI for Quantitative Investment". In: *arXiv preprint arXiv:2402.09746* (2024).

[86] Kou, Z., Yu, H., Luo, J., Peng, J., Li, X., Liu, C., Dai, J., Chen, L., Han, S., and Guo, Y. "Automate Strategy Finding with LLM in Quant Investment". In: *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics, 2025, pp. 18517–18533. DOI: 10.18653/v1/2025.findings-emnlp.1005. URL: https://aclanthology.org/2025.findings-emnlp.1005/.

[87] Wang, M., Izumi, K., and Sakaji, H. "LLMFactor: Extracting Profitable Factors Through Prompts for Explainable Stock Movement Prediction". In: *Findings Of The Association For Computational Linguistics: ACL 2024*. 2024, pp. 3120–3131.

[88] Duan, Y., c. zhang chuheng, and Li, J. "FactorMAD: A Multi-Agent Debate Framework Based on Large Language Models for Interpretable Stock Alpha Factor Mining". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 605–613.

[89] Han, J., Zhang, S., Li, W., Dong, Y., Hu, T., Zhu, Y., Yu, X., Guo, X., Liu, Z., Wang, K., et al. "QuantaAlpha: An Evolutionary Framework for LLM-Driven Alpha Mining". In: *arXiv preprint arXiv:2602.07085* (2026).

[90] Wang, Y., Xu, J., Zhang, H., Huang, S.-L., Sun, D. D., and Zhang, X.-P. "FactorMiner: A Self-Evolving Agent with Skills and Experience Memory for Financial Alpha Discovery". In: *arXiv preprint arXiv:2602.14670* (2026).

[91] Okpala, I., Golgoon, A., and Kannan, A. R. "Agentic AI Systems Applied to Tasks in Financial Services: Modeling and Model Risk Management Crews". In: *arXiv preprint arXiv:2502.05439* (2025).

[92] Bhattacharyya, A., Yu, Y., Yang, H., Singh, R., Joshi, T., Chen, J., and Yalavarthy, K. "Model Risk Management for Generative AI In Financial Institutions". In: *arXiv preprint arXiv:2503.15668* (2025).

[93] Bi, Y., Wang, Z., Zhao, L., and Ventre, C. "Long-Term Financial Forecasting and Trading via Multi-Agent Reinforcement Learning". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 788–796.

[94] Wang, S., Ji, T., Wang, L., Sun, Y., Liu, S.-C., Kumar, A., and Lu, C.-T. "Stocktime: A Time Series Specialized Large Language Model Architecture for Stock Price Prediction". In: *arXiv preprint arXiv:2409.08281* (2024).

[95] Fan, T., Yang, Y., Jiang, Y., Zhang, Y., Chen, Y., and Huang, C. "AI-Trader: Benchmarking Autonomous Agents in Real-Time Financial Markets". In: *arXiv preprint arXiv:2512.10971* (2025).

[96] Wang, Z., Ventre, C., and Polukarov, M. "Multi-Agent Reinforcement Learning for Market Making: Competition without Collusion". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 814–822.

[97] Qian, L., Zhou, W., Wang, Y., Peng, X., Yi, H., Zhao, Y., Huang, J., Xie, Q., and Nie, J.-Y. "Fino1: On the Transferability of Reasoning-Enhanced LLMs and Reinforcement Learning to Finance". In: *arXiv preprint arXiv:2502.08127* (2025).

[98] Xiao, Y., Sun, E., Chen, T., Wu, F., Luo, D., and Wang, W. "Trading-R1: Financial Trading with LLM Reasoning via Reinforcement Learning". In: *arXiv preprint arXiv:2509.11420* (2025).

[99] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: *ACM Transactions on Information Systems* 43.2 (2025), pp. 1–55.

[100] Zhao, T., Lyu, J., Jones, S., Garber, H., Pasquali, S., and Mehta, D. "AlphaAgents: Large Language Model Based Multi-Agents for Equity Portfolio Constructions". In: *arXiv preprint arXiv:2508.11152* (2025).

[101] Wei, J., Karina, N., Chung, H. W., Jiao, Y. J., Papay, S., Glaese, A., Schulman, J., and Fedus, W. "Measuring Short-Form Factuality in Large Language Models". In: *arXiv preprint arXiv:2411.04368* (2024).

[102] Xu, C., Guan, S., Greene, D., Kechadi, M., et al. "Benchmark Data Contamination of Large Language Models: A Survey". In: *arXiv preprint arXiv:2406.04244* (2024).

[103] Liu, M., Xu, Z., Zhang, X., An, H., Qadir, S., Zhang, Q., Wisniewski, P. J., Cho, J.-H., Lee, S. W., Jia, R., et al. "LLM Can be a Dangerous Persuader: Empirical Study of Persuasion Safety in Large Language Models". In: *arXiv preprint arXiv:2504.10430* (2025).

[104] Hu, X., Peng, W., Lu, X., Liu, D., Huang, X.-J., and Shao, J. "LLMs Deceive Unintentionally: Emergent Misalignment in Dishonesty from Misaligned Samples to Biased Human-AI Interactions". In: *Findings of the Association for Computational Linguistics: ACL 2026*. 2026, pp. 9348–9372.

[105] Danry, V., Pataranutaporn, P., Groh, M., Epstein, Z., and Maes, P. "Deceptive AI Systems that Give Explanations are More Convincing than Honest AI Systems and Can Amplify Belief in Misinformation". In: *arXiv preprint arXiv:2408.00024* (2024).

[106] Pandey, A. K. and Rajwal, S. "Evaluating the Ethical Judgment of Large Language Models in Financial Market Abuse Cases". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 132–140.

[107] Byrd, D. "The Accidental Pump and Dump: When Agentic AI Meets Autonomous Trading". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 88–95.

[108] Hughes, J., Price, S., Lynch, A., Schaeffer, R., Barez, F., Somani, A., Koyejo, S., Sleight, H., Jones, E., Perez, E., et al. "Best-of-N Jailbreaking". In: *Advances in Neural Information Processing Systems* 38 (2026), pp. 73137–73221.

[109] Doumbouya, M. K. B., Nandi, A., Poesia, G., Ghilardi, D., Goldie, A., Bianchi, F., Jurafsky, D., and Manning, C. "H4RM3L: A Language for Composable Jailbreak Attack Synthesis". In: *International Conference on Learning Representations*. Vol. 2025. 2025, pp. 57253–57286.

[110] Wang, J. J. and Wang, V. X. "Assessing Consistency and Reproducibility in the Outputs of Large Language Models: Evidence across Diverse Finance and Accounting Tasks". In: *arXiv preprint arXiv:2503.16974* (2025).

[111] Lee, H., Seo, J., Park, S., Lee, J., Ahn, W., Choi, C., Lopez-Lira, A., and Lee, Y. "Your AI, not your View: The Bias of LLMs in Investment Analysis". In: *Proceedings of the 6th ACM International Conference on AI in Finance*. 2025, pp. 150–158.

[112] Xie, J., Zhang, K., Chen, J., Lou, R., and Su, Y. "Adaptive Chameleon or Stubborn Sloth: Revealing the Behavior of Large Language Models in Knowledge Conflicts". In: *International Conference on Learning Representations*. Vol. 2024. 2024, pp. 35623–35646.

[113] Nie, Y., Kong, Y., Dong, X., Mulvey, J. M., Poor, H. V., Wen, Q., and Zohren, S. "A Survey of Large Language Models for Financial Applications: Progress, Prospects and Challenges". In: *arXiv preprint arXiv:2406.11903* (2024).

[114] Yuan, J., Li, H., Ding, X., Xie, W., Li, Y.-J., Zhao, W., Wan, K., Shi, J., Hu, X., and Liu, Z. "Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference". In: *Advances in Neural Information Processing Systems* 38 (2026), pp. 169819–169851.

[115] Fu, J., Zhao, X., Yao, C., Wang, H., Han, Q., and Xiao, Y. "Reward Shaping to Mitigate Reward Hacking in RLHF". In: *arXiv preprint arXiv:2502.18770* (2025).

[116] Wang, X., Joshi, N., Plank, B., Angell, R., and He, H. *Is It Thinking or Cheating? Detecting Implicit Reward Hacking by Measuring Reasoning Effort*. 2025. DOI: 10.48550/arXiv.2510.01367.

[117] Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. "An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning". In: *IEEE Transactions on Audio, Speech and Language Processing* (2025).

[118] Abdi, I., Gupta, A., Mok, M., Lu, A., Lee, N., and Anumanchipalli, G. "Evolutionary Strategies lead to Catastrophic Forgetting in LLMs". In: *arXiv preprint arXiv:2601.20861* (2026).

[119] Staniszewski, K. and Łańcucki, A. *KV Cache Transform Coding for Compact Storage in LLM Inference*. 2026. eprint: 2511.01815. URL: https://arxiv.org/abs/2511.01815.

[120] Zhao, Y., Lin, C.-Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., and Kasikci, B. "Atom: Low-bit Quantization for Efficient and Accurate LLM Serving". In: *Proceedings of Machine Learning and Systems* 6 (2024), pp. 196–209.

[121] Zandieh, A., Daliri, M., Hadian, M., and Mirrokni, V. "TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate". In: *arXiv preprint arXiv:2504.19874* (2025).

[122] Kim, J., Shin, B., Chung, J., and Rhu, M. "The Cost of Dynamic Reasoning: Demystifying AI Agents and Test-Time Scaling from an AI Infrastructure Perspective". In: *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE. 2026, pp. 1–16.

[123] Sajith, A. and Kathala, K. C. R. "Is Training Data Quality or Quantity More Impactful to Small Language Model Performance". In: *International Conference on Neural Computing for Advanced Applications*. Springer. 2025, pp. 75–87.

[124] Mehta, S. "Beyond Accuracy: A Multi-Dimensional Framework for Evaluating Enterprise Agentic AI Systems". In: *arXiv preprint arXiv:2511.14136* (2025).

[125] Modarressi, A., Deilamsalehy, H., Dernoncourt, F., Bui, T., Rossi, R. A., Yoon, S., and Schütze, H. "Nolima: Long-Context Evaluation Beyond Literal Matching". In: *arXiv preprint arXiv:2502.05167* (2025).

[126] Nelson, E., Kollias, G., Das, P., Chaudhury, S., and Dan, S. "Needle in the Haystack for Memory Based Large Language Models". In: *arXiv preprint arXiv:2407.01437* (2024).

[127] Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., et al. "Lima: Less is More for Alignment". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 55006–55021.

Chief Editor

Monica DEFEND
Head of Amundi Investment Institute

Editors

Marie BRIÈRE
Head of Investors' Intelligence & Academic Partnership

Thierry RONCALLI
Head of Quant Portfolio Strategy

Investment Institute

WORKING PAPER 192 | **AUGUST 2026**

Find out more about Amundi Investment Institute Publications

Visit our Research Center

SCAN ME

Amundi
Investment Solutions

Trust must be earned