CycoAgent: A LLM-Based Stock Trading Agent

Yujia Wang[1]

[1] China University of Geosciences, Beijing

Abstract

In recent years, large language models (LLMs) have demonstrated remarkable performance in question-answering tasks. While LLMs excel at understanding human commands and generating solutions from input, a goal-oriented agent still requires a well-designed architecture to support the processing of multi-source information, logical reasoning, and task execution. To address this challenge, this thesis proposes a large language model agent framework, comprising six core functional modules: (1) the memory module stores and retrieves historical transaction data and market information; (2) the reflection module analyses the effectiveness of trading decisions and summarises lessons learned; (3) the portfolio module, which simulates real-world trading scenarios for risk control; (4) the decision-making module is responsible for generating and optimising trading strategies; (5) the market environment module simulates real stock trading environments; (6) the prompt module guides the large language model to generate accurate, clear, and standardised outputs. In the construction of the dataset, Tesla (TSLA) and Amazon (AMZN) were selected as representative samples, with the data time span ranging from 15 August 2023 to 28 March 2025, covering a total of 407 trading days. The collected data includes 15 price indicators, 11,468 structured news articles, key financial metrics and management discussions from quarterly and annual financial reports, and 479 conference call transcripts. Test results indicate that the Agent based on GPT-4o-Turbo-2024-05-13 demonstrates strong contextual adaptability and information intensity perception capabilities in daily analytical tasks. On the test set, it achieved cumulative returns superior to the Buy & Hold strategy, along with lower volatility and maximum drawdown, exhibiting excellent risk-adjusted return characteristics. On TSLA data, it achieved approximately 20% positive returns while effectively mitigating downside market risks; on AMZN data, although it did not achieve positive returns, it demonstrated stable drawdown control capabilities. ease contact the editorial team with clear suggestions in advance.

Keywords: Large Language Models, Financial Agents, Automated Trading, Stock Trading

Introduction

In this era of rapid information technology advancement, the volume of data in financial markets is growing exponentially. With the widespread adoption of the internet, market participants can access vast amounts of global information, including real-time stock price fluctuations, various macroeconomic indicators, corporate financial reports, and extensive news coverage. However, human cognitive and memory capacities are inherently limited. According to the bounded rationality hypothesis, individuals face constraints in decision-making due to the quantity and quality

of information available, their cognitive capacity, and the time allocated for deliberation [1]. This limitation becomes particularly pronounced in stock trading scenarios. Traders often struggle to process and analyze complex information exceeding their perceptual and memory capacities, leading to potential oversights of critical market signals or misinterpretations of data, thereby compromising the accuracy of trading decisions. When confronted with vast and intricate information environments, traditional trading models reliant on human subjective judgment increasingly reveal their shortcomings. Concurrently, Large Language Models (LLMs) have achieved breakthrough progress in natural language processing [2]. LLMs are deep learning systems based on the Transformer architecture [3], which learn statistical patterns and semantic representations of language through self-supervised pre-training on massive text datasets. Their core capabilities manifest in context-aware text generation, logical reasoning, and multi-task transfer learning, coupled with robust language comprehension, inference, and generation abilities. They can process diverse data formats including text, images, and audio, demonstrating outstanding performance in tasks like question-answering and text generation. However, directly applying LLMs to stock trading decisions remains challenging. Stock trading requires a rational-aiding framework to perceive market conditions in real time, deeply analyze multi-source information, and formulate corresponding strategies—an Agent. Agents can leverage LLMs' capabilities to autonomously plan and execute tasks, continuously optimizing decisions through feedback from market interactions [4], offering a new approach to addressing information overload and cognitive limitations in stock trading.

Related Works

Development of LLMs

Since the 1950s, the field of natural language processing (NLP) has undergone a transformation from rule-based approaches and statistical modeling to deep learning. Before the rise of deep learning, early methods represented by n-gram models and hidden Markov models (HMMs) could perform language modeling within limited contexts. However, issues such as data sparsity, fixed windows, and limited semantic understanding made them ill-suited for complex linguistic tasks [5]. In 2013, Mikolov et al. introduced the word2vec model, pioneering distributed representation-based word embeddings that provided learnable representations for semantic modeling in NLP tasks [6]. Subsequently, recurrent neural networks (RNNs) and long short-term memory (LSTM) models further enhanced sequence modeling capabilities, though they still faced challenges of computational inefficiency and vanishing gradients [7]. In 2017, Vaswani et al. introduced the Transformer architecture, which adopted a fully connected modeling approach based on self-attention mechanisms. This significantly enhanced the modeling of long-range dependencies and led to the complete abandonment of recursive and convolutional structures, thereby improving training efficiency and scalability. Subsequently, large-scale pre-trained language models rapidly became the mainstream technical approach in NLP, establishing the pre-training and fine-tuning paradigm: leveraging unsupervised corpora for general language pre-training followed by task-specific fine-tuning [8]. The introduction of the Transformer architecture and the rise of large-scale pre-training ushered NLP into a new era dominated by large language models. The self-attention mechanism in Transformer-based large language models resolves the long-range dependency issues inherent in traditional sequence models, while their parallel computation capability significantly boosts training efficiency. Employing a combination of large-scale unsupervised pre-training and task-specific

fine-tuning, these models demonstrate outstanding performance across diverse NLP tasks. Current scalability enables large models to perform complex reasoning tasks without additional training, and their generalization capabilities continue to improve with the introduction of techniques like multi-task learning and prompt engineering.

Development of Agent

Agent is an execution system built upon large language model capabilities [10]. It possesses the ability to autonomously plan and execute tasks, leveraging large language models for decision-making while continuously adjusting its behavior based on environmental feedback. Agents can independently determine how to accomplish tasks, including understanding complex inputs, conducting reasoning and planning, effectively using tools, and adjusting and recovering from errors. Agents initiate tasks by receiving user instructions or interacting with users. Once a task is defined, the Agent independently plans and executes it, requesting additional information or judgment from humans when necessary. During execution, the Agent continuously gathers "ground truth" from the environment to assess task progress. Upon encountering checkpoints or obstacles, the Agent can pause the task and seek human feedback to ensure controllability. Overall, Agents can handle complex tasks and achieve objectives through iterative operations based on environmental feedback. LLM-based agents employ a modular architecture where multiple key components collaborate to handle complex tasks. The brain serves as the core decision-making module powered by LLMs; the task planning module decomposes complex tasks into subtasks and formulates action strategies; the memory module stores and retrieves information during interactions; the tool interface facilitates interaction with external environments or invokes external tools; the reflection and evaluation module self-assesses agent behavior, generates feedback, and guides memory updates.

The Brain is the agent's core, essentially one or more large language models. Acting as the agent's "brain," it handles primary decision-making and reasoning tasks. This module receives user-input natural language commands or environmental observation data, then generates corresponding solution steps or action decisions through internal LLM reasoning. The Brain module defines the agent's objectives and available toolset while coordinating other modules' operations. During implementation, specific roles or personality traits can be assigned to the Brain module, enabling it to adopt corresponding styles or preferences when interacting with tasks across different domains. This module serves as the agent's decision-making hub, determining the agent's next action in each interaction round based on the current context. Within the task planning module, complex tasks typically require decomposing higher-level objectives into a series of manageable subtasks. The Agent achieves this through the Decision Module: employing reasoning methods like chain-of-thought, enabling the large language model to first plan solution approaches or steps upon receiving a query. This module can be viewed as a part or subcomponent of the brain, guiding the large language model to sequence actions, helping it contemplate future steps and formulate multi-step solutions. When tasked with complex financial analysis, the Decision Module first decomposes the task into steps like data acquisition, analysis execution, and report generation. During decision-making, the large language model proposes action plans based on existing knowledge and memory, then adjusts the plan after each step using newly acquired information. The planning module typically incorporates a reflection mechanism to optimize the plan, revising it when necessary to enhance accuracy and relevance.

To maintain contextual coherence during multi-turn reasoning and long-term tasks, Agents are equipped with an explicit memory module. In common Agents, this module comprises two levels: short-term memory and long-term memory. Short-term memory functions as the agent's working memory, recording details from the current session or most recent reasoning to ensure coherent responses to immediate context. It may reset upon task completion. Long-term memory simulates the agent's knowledge base, accumulating experience across sessions and tasks. It stores valuable information, lessons learned, and historical knowledge from prior interactions, potentially spanning weeks or months. Whenever the agent encounters a new task or requires historical context, the long-term memory retrieves relevant content based on its relevance to the current task, providing contextual support that enables the agent to make informed decisions. By integrating relevant memories into the context of the large language model, the agent can "remember" previously learned lessons and details, enhancing response accuracy and reducing inconsistencies. The memory module's data grows continuously with interactions. To prevent information overload and maintain retrieval efficiency, it employs strategies to prune unimportant memories, optimizing long-term memory management.

The tool interface module serves dual responsibilities within the agent system: information acquisition and operation execution. On one hand, it proactively gathers data from external environments, providing task-relevant support to the agent. On the other hand, it executes specific actions based on instructions from the planning module to advance task completion. Current tool interface capabilities encompass external API calls, database queries, code execution, and web searches, effectively expanding the agent's operational boundaries. Upon receiving instructions from upper-level modules, the tool interface module schedules appropriate tool resources and promptly returns execution results, thereby closing the decision-making loop. This mechanism enables the agent's behavioral capabilities to transcend the limitations of its internal knowledge base, granting it the ability to proactively acquire information from the real world and execute actions. The introduction of tool invocation mechanisms significantly enhances system flexibility, enabling agents to dynamically adapt external resources according to varying task requirements to accomplish complex and dynamic objectives.

To enhance the agent's self-improvement capabilities, researchers have incorporated a reflection mechanism into the system. This mechanism is typically implemented through a dedicated evaluation module. Within the overall architecture, the evaluation module functions as a component with "self-reflection" capabilities. After the agent generates intermediate reasoning results or final outputs, the evaluation module inspects them to determine compliance with expected requirements or identify potential errors. In practical applications, the reflection mechanism typically leverages large-scale language models to generate textual feedback, including root cause analysis of potential issues and improvement suggestions. The system subsequently stores this feedback in a memory module, serving as a reference for the agent's future reasoning processes. By incorporating the reflection mechanism, the agent can learn from past experiences and effectively avoid repetitive errors when encountering similar tasks. This mechanism synergizes with planning and memory capabilities, collectively enabling the agent's continuous optimization and evolution within dynamic environments.

Figure 1: Illustration of the financial trading Agent architecture.

Agent in Finance

In recent years, scholars have achieved significant progress in researching autonomous trading systems. Autonomous trading has evolved from its initial rule-based strategies to more advanced machine learning-based algorithms [11]. Agents have found extensive applications in the financial sector. As shown in Figure 1, stock trading agents can extract historical data from stock markets and execute actions based on algorithmic designs. With continuous technological advancements, agents based on reinforcement learning (RL) [12], particularly deep reinforcement learning (DRL) agents [13], have garnered significant attention. Leveraging reinforcement learning principles and deep learning, DRL agents can effectively process and learn from diverse financial data, including stock prices, key financial indicators, and market sentiment. Deep reinforcement learning agents utilize deep neural networks to extract key features from input data representing complex financial market environments, thereby enhancing their understanding capabilities. Agents based on deep reinforcement learning retain the core characteristics of reinforcement learning agents, learning through interaction with predefined environments to maximize investment returns over time.

In existing research, deep reinforcement learning demonstrates the capability to process large-scale data and make sound decisions, thereby meeting the core requirements of trading agents. However, certain inherent characteristics of this approach reveal significant limitations in financial contexts. Deep reinforcement learning lacks clear explanations for its decision-making processes, with its model structure typically functioning as a "black box," making its internal reasoning mechanisms and computational workflows difficult for external parties to comprehend. Concurrently, deep reinforcement learning faces challenges in integrating textual information with numerical features. Textual data holds significant value in finance, as vast amounts of market information rely on textual formats like news articles and financial reports for dissemination. Converting text into embedding vectors substantially expands the input space's dimensionality, thereby imposing higher demands on computational resources. When integrating textual representations with numerical indicators, models often encounter convergence difficulties. Table 1 summarizes the performance of different foundational algorithms in agent design. After comprehensively evaluating computa-

tional complexity and operational costs, this study ultimately selected large language models as the core backbone structure for the agent.

Feature	Large Language Models (LLMs)	Deep Reinforcement Learning (DRL)
Core Strengths	Natural language understanding and multimodal processing	Dynamic adaptability and high-dimensional nonlinear decision making
Decision Transparency	Decision processes are difficult to interpret	Limited interpretability, partially reflected through reward function design
Training Requirements	Requires large-scale pretraining and high-quality data for fine-tuning	Requires carefully designed environments and extensive interaction-based training
Responsiveness	Adapts to market changes through fine-tuning	Autonomously adjusts strategies to directly respond to dynamic markets
Computational Efficiency	High computational cost during inference, but faster than DRL during training	Computationally intensive during training, with high time and resource demands
Application Scenarios	Text sentiment analysis and related tasks	Stock trading and real-time market prediction

Table 1: Comparison of Agent characteristics under different backbone algorithms.

LLM-Based Agent

In recent years, large language models such as GPT-4[14], PaLM 2[15], and DeepseekR1[16] have achieved significant milestones, demonstrating extraordinary capabilities in language comprehension, reasoning, planning, and complex decision-making. Leveraging these strengths, large language models can serve as agent backbones, offering an effective pathway to enhance autonomous decision-making capabilities and realize Artificial General Intelligence (AGI)[17]. Unlike traditional reinforcement learning-based agents, which optimize strategies driven by explicit rewards and punishments, large language model-based agents operate through text-based instructions, prompt templates, and contextual learning, offering greater flexibility and generalization. Leveraging the language comprehension and logical reasoning capabilities of large language models, these agents can interact with external environments via natural language, execute multi-step complex tasks, and continuously adjust their strategies in response to changing scenarios. Currently, LLM-based agents are widely deployed across multiple domains. By employing techniques such as task decomposition [18], self-reflection [19], and memory augmentation [20], they demonstrate outstanding performance in software development, mathematical reasoning, embodied intelligence, and other fields.

Large language models provide a viable pathway for agent design and significantly alleviate issues encountered in traditional approaches. Through carefully crafted prompts, agents based on large language models can clearly articulate their reasoning processes and decision outcomes in textual form. This enables researchers to directly observe and evaluate the underlying logic, making adjustments when necessary. As the core algorithm for agents, large language models leverage their extensive prior knowledge and ability to integrate valuable information from diverse data types. This enables agents to transcend reliance on closed environments, demonstrating enhanced versatility and adaptability. When equipped with appropriate prompts, agents exhibit significantly improved decision-making capabilities [21]. Existing research indicates that prompt-guided reasoning mechanisms can increase problem-solving efficiency in various domains [22]. A growing number of scholars are focusing on the use of large language models to provide decision support for stock trading through continuous interaction with information about the financial environment [23][24].

In trading, large language models have demonstrated strong capabilities in analyzing and interpreting financial data, as evidenced by the emergence of models such as FinGPT[23] and BloombergGPT[24]. However, there exists an inherent gap between question-answering tasks and sequential decision-making in trading. Stock data incorporates multiple technical indicators and exhibits cyclical patterns within financial market fluctuations[25]. Financial markets experience distinct bull and bear cycles, and market makers employ varying tactics for each individual stock. Although FinMEM[26] is a large language model trading agent featuring human-like memory mechanisms and role-based design, the full capabilities of large language models as comprehensive autonomous trading systems remain largely unexplored, particularly in simulated market environments, decision signal generation, and real-money simulations.

In research on large language model market trading agents, Park et al.[27] developed a generative agent framework that offers significant insights for the field. This generative framework enhances the retrieval efficiency of general-purpose large language model agents for critical events. It employs a unique character design and seed memory, activating the agent through prompts during specific queries. It prioritizes events within a unified memory stream, ranking them based on a linear combination of reproducibility, relevance, and importance. This framework provides an infrastructure for LLM agent design, comprising: - A profiling module for role definition - A memory module for recording experiences and retrieving critical information - An action module for guiding behavior based on retrieved memories Effectively enabling agents to achieve objectives within general social environments.

LLM-Based Agent Framework

Multimodal Data Pipeline

Building on prior research, this study designs a novel large-language-model-based Agent framework. After balancing computational cost and model inference performance, the system adopts OpenAI's GPT-4o-2024-05-13 as the primary backbone. In terms of framework design, it follows the human-like working memory and hierarchical long-term memory mechanisms introduced by Park [27]. This approach effectively leverages the temporal nature of financial data and captures

key investment insights to improve trading outcomes.

The analysis module includes a dynamic role-setting function that provides initial instructions related to domain expertise. As trading experience accumulates, the Agent continuously updates its domain knowledge. By simulating real stock-trading scenarios, configuring initial capital, transaction fees, and position-size limits, the Agent receives feedback on post-decision capital and returns, enabling reflection and iterative capability enhancement.

For data-pipeline construction, studies such as FinMEM [26] collected and processed four types of stock price labels ("close," "open," "high," "low") along with news and annual financial reports. However, during decision-making, only closing prices were considered. This ignores key factors such as price volatility, opening price, highest and lowest prices, trading volume, and also lacks fundamental indicators, market sentiment, and real trading dynamics. As a result, the model becomes overly sensitive to minor price fluctuations, causing high-frequency oscillations and overfitting that reduce robustness in real trading settings.

Building on this, the present study collects and processes fifteen key financial features across multiple stocks: "close," "open," "high," "low," "pre_close," "pct_change," "vol," "amount," "vwap," "change," "adj_factor," "turnover_ratio," "total_mv," "pe," and "pb," enabling a more realistic simulation of market conditions. For news acquisition and processing, the system incorporates data from Alpaca and Alpha Vantage, covering most publicly accessible stock news. Summaries are generated using the GPT-3.5-turbo-16k-0613, constrained to 200 characters to avoid exceeding input limits, and sentiment labels (positive, neutral, negative) are produced using FinBERT [28]. Annual earnings-call transcripts released after financial reports are also injected into memory as supplemental references to support Agent decisions.

To mitigate overfitting in multimodal financial tasks and enhance robustness to noise and anomalies encountered in real-world trading, the framework introduces multiple structured data-augmentation strategies, including price masking, news shuffling, and sentiment perturbation. These methods strengthen the model's ability to handle missing, disordered, or weakly anomalous inputs. Integrated with the data pipeline, augmentation and fusion operations are automatically performed during preprocessing, allowing the model to learn more stable and generalizable representations throughout training. During this phase, financial indicators, fundamentals, and market sentiment are incorporated into the Agent's decision-making logic to better simulate real trading conditions and support more grounded and reliable actions.

Memory Module

The Agent continuously generates memory events E based on its perception of the stock-market environment, including prices, news, earnings reports, and other signals, and stores them in the Memory Module. To simulate the cognitive structure of human traders, memory is divided into three hierarchical levels with distinct roles and temporal spans: Short-term Memory, Mid-term Memory, and Long-term Memory. Each memory event exists in only one level at any given time and is assigned to the appropriate tier based on its attributes.

During perception, the Agent retrieves relevant memories and uses them to determine actions.

Retrieved memories are also used to form longer-term plans and produce higher-level reflections, contributing to the memory stream for future use. The memory module aligns closely with the cognitive structure of human traders, offering strong interpretability and real-time adaptability. It captures key information beyond human perceptual limits, enabling the Agent to continuously evolve its expertise, respond rapidly to new signals, and refine trading decisions in volatile financial environments. Deeper insights are assigned lower decay rates and thus preserved longer, whereas shallow insights decay more quickly and remain only briefly. As one of the core components of the Agent's decision-making system, the memory structure is central to adaptive reasoning.

Memory Retrieval Score Formula
Memory events are evaluated based on their recency, relevancy, and importance. The combined retrieval score is defined as:

$$\gamma_l^E = S^E_{\text{Recency}*l} + S * \text{Relevancy} * l^E + S * \text{Importance}_l{}^E \tag{1}$$

Here, (γ_l^E) denotes the retrieval score of event (E) at level (l)(short-term, mid-term, or long-term). Short-term memory scores range from 0–60, storing recent and temporary information such as market updates, short-term price movements, and immediate news. Mid-term memory scores range from 60–80, capturing medium-importance information such as quarterly performance and mid-term trends. Long-term memory scores range from 80–100, storing highly important information such as long-term strategies, industry trends, and major policy changes. Each event is associated with one score and belongs exclusively to a single memory tier. An event cannot exist across multiple levels; its tier determines its scoring behavior.

Recency Score
The recency score ($S^E_{\text{Recency}_l}$) measures the temporal distance between memory event (E) and the current timestep (t). It is computed via an exponential decay function. Events closer to the present carry stronger immediacy and thus exert greater influence on decisions. This supports fast reactions to short-term market dynamics.

$$S^E_{\text{Recency}_l} = \exp\left(-\lambda_r \cdot (t - t_E)\right) \tag{2}$$

where t_E is the timestamp of event E, and λ_r is the decay factor.

Relevancy Score
The relevancy score ($S^E_{\text{Relevancy}_l}$) measures the semantic similarity between the current task demand or perceptual input and memory event (E). It is typically computed in embedding space. This study uses the OpenAI `text-embedding-3-large` model and the FAISS library for similarity search. This metric preserves key observations from recent days or weeks, such as changes in technical indicators or market sentiment. It also supports strategic adjustments such as rebalancing positions and reallocating capital.

$$S^E_{\text{Relevancy}_l} = \cos\left(f_{\text{query}},\ f_E\right) \tag{3}$$

where (f_{query}) is the query representation, and (f_E) is the embedding of event (E).

Importance Score
The importance score ($S^E_{\text{Importance}_l}$) evaluates the potential influence of event (E) on the task. Its initial value is sampled from a probability distribution and decays over time. Short-term and mid-term memory entries use a decay factor of 0.98, while long-term or reflective memories use 0.995. This score captures deeper insights such as industry observations, macroeconomic developments, and historical market conditions, forming the basis for strategic reasoning and reflective decision-making.

$$S^E_{\text{Importance}_l} = \sigma\left(w^\top f_E + b\right) \tag{4}$$

where f_E is the semantic feature vector of event E, and w and b are learnable parameters. The activation function σ denotes the sigmoid function.

Within the memory module, each event is scored based on recency, relevancy, and importance. The combined retrieval score (γ^E_l) determines its weight in decision-making. Every event is assigned to exactly one memory tier, ensuring stability and interpretability in the memory stream.

Reflection Module

The Agent cannot effectively generalize or reason using raw observational memory alone. Reflection represents a higher-level and more abstract form of cognition. Reflections are also treated as memory entries and will appear alongside other retrieved observations during memory search, and they are generated periodically. In implementation, when the cumulative importance score of the most recent events exceeds a predefined threshold, a reflection is produced. Reflection is not merely a retrospective review of historical information; it is an advanced cognitive ability that reorganizes and reinterprets the knowledge structure.

Unlike the three memory levels, the reflection module does not directly participate in fast, day-to-day decision making. Instead, it is triggered periodically or conditionally to abstract important experiences, discover patterns, and reorganize prior cognition. The resulting reflections are fed back into other memory levels or used directly to guide model decisions.

After each memory event E is processed by the hierarchical memory system, it is assigned a scoring function used for the reflection candidate set. As shown in Equation (5), $S^E_{\text{Importance}}$ denotes the importance score of event E as defined in Equation (4). $S^E_{\text{Volatility}}$ represents the market volatility associated with the time window in which event E occurs. S^E_{Conflict} measures the degree of prediction error or deviation between event E and the current strategy output.

$$\Gamma^E = \alpha_1 S^E_{\text{Importance}} + \alpha_2 S^E_{\text{Volatility}} + \alpha_3 S^E_{\text{Conflict}} \tag{5}$$

where $\alpha_1, \alpha_2, \alpha_3 \in [0, 1]$ control the contribution of each component. A reflection is triggered only when the score Γ^E exceeds the reflection threshold θ, as defined in Equation (6):

$$\text{Trigger}^{E}_{\text{reflection}} = I\left[\Gamma^{E} \geq \theta\right] \tag{6}$$

The first step of reflection is determining the content. Based on the Agent's recent experiences, a set of candidate questions is generated by querying a large language model using the most recent 100 memory records. These questions are then used as retrieval queries to collect relevant memory events, and the language model extracts insights and supporting evidence. Reflection allows the Agent to reflect not only on observations but also on previous reflections, forming a reflection tree. Leaf nodes correspond to basic observations, while non-leaf nodes represent increasingly abstract ideas, with higher-level nodes capturing more generalized concepts.Memory events selected for reflection undergo semantic clustering, causal evaluation, and strategy refinement. This process constructs an embedded meta-learning loop, enabling the Agent to continuously optimize its strategy parameters or memory structure.

Portfolio Module

In building a large-language-model Agent with long-term autonomy, the portfolio module functions not only as the execution container for trading strategies but also as the mechanism responsible for behavioral constraints, capital management, and risk evaluation. In this work, a structured portfolio-management module is designed to impose realistic trading constraints, enable feedback loops, and track value under multi-source perception and language-based reasoning, ensuring interpretability and traceability.

Given the capital limitations, transaction costs, and risk exposures present in actual trading systems, portfolio management is simulated under an initial-capital constraint. Position limits restrict any single buy operation from allowing total asset exposure to exceed 70%. Risk control specifies a take-profit threshold of 30% and a stop-loss threshold of 40%. Transaction-cost estimation deducts a 0.1% fee for every buy or sell action. In addition, constraints on transaction scale, available cash, and current holdings are incorporated: when trade size becomes excessively large, the system reduces the trade quantity; when position limits are exceeded, the position is adjusted; when cash is insufficient, trade size is reduced. Upon triggering the take-profit condition, partial liquidation occurs; upon triggering the stop-loss condition, positions are automatically reduced. These mechanisms ensure realistic gain–loss evaluations and provide stored outcomes that support later reflection, supplying strong evidence for subsequent decision making.

The portfolio module supports the daily ingestion of market data, including historical prices, transaction records, and valuation factors. In implementation, daily price inputs are used for momentum computation, while the module provides the current market state and position information to the decision-making module. The portfolio module regularly records cash flow, holding quantities, and total portfolio value to supply evidence for behavioral evaluation. This ensures continuous state tracking and provides contextual grounding for action selection.

The design of the trading-execution and return-evaluation mechanism allows complete buy–sell execution and corresponding feedback. During execution, the system updates both the funding account and position states according to the trade direction and quantity specified in the input instruc-

tion. In the feedback phase, the system uses historical data to compute percentage return changes between consecutive trading days and evaluates whether the executed trade direction aligns with the underlying market trend. During momentum evaluation, the system computes the average return from the price series over the past (M) days to generate trend signals, which serve as references for subsequent decisions.

Under this design, trading behavior during the training phase is generated using labeled data, while the portfolio module records actions and reward feedback to serve as essential inputs to reflection memory. During the testing phase, the Agent must generate actions without access to future prices, relying instead on past returns, historical momentum, and prior knowledge stored in memory. The portfolio module evaluates and adjusts decisions based on changes in investment value, forming a closed-loop control chain of action $\rightarrow$ market feedback $\rightarrow$ strategy refinement.

Decision Module

Large language models are capable of generating reasonable actions conditioned on contextual information, but the Agent must also plan over longer temporal horizons to ensure coherence, consistency, and credibility. This requires integrating memory and reflection into the decision-making process. The Agent therefore generates prospective action sequences that incorporate behavior, timing, and situational factors, maintaining internal consistency across time. Decisions are likewise stored in the memory module, enabling the Agent to reflect on its own choices. Reflections are incorporated into later retrieval processes, allowing the Agent to consider observations, historical decisions, and broader cognitive factors when making future decisions.

In cognition-driven trading, the Agent's decisions rely not only on immediate reactions to the current market but also on reflective evaluation and optimization of its historical knowledge. Under carefully crafted prompting, at each trading cycle t, the Agent receives the market state, including price sequences, financial reports, earnings-call transcripts, and news text. The Agent first extracts three embedding types—technical indicators f_t^{tech}, fundamental features f_t^{fund}, and sentiment vectors f_t^{sent}—and concatenates them as inputs to the strategy, as shown in Equation (7):

$$f_t = \text{Concat}\left(f_t^{\text{tech}},\, f_t^{\text{fund}},\, f_t^{\text{sent}}\right) \tag{7}$$

The Agent then queries short-term, mid-term, and long-term memory to retrieve entries relevant to the current task and incorporates the historical experience embedding h_t. It generates an initial trading recommendation $a_t \in \{-1, 0, 1\}$, corresponding to "sell," "hold," and "buy," respectively, as defined in Equation (8), where π_θ denotes the policy function:

$$a_t = \pi_\theta\left(f_t,\, h_t\right) \tag{8}$$

During decision execution, the memory module retrieves short-term, mid-term, and long-term memory entries using Top-K similarity ranking to identify the most relevant events for the current task. Building on this, the reflection module integrates the current market state with execution

errors to generate reflective summaries. These summaries are encoded as structured memories and written into the reflection memory layer, strengthening the Agent's understanding and refinement of long-term strategies. Through deep semantic reasoning, the system forms correspondences between current market indicators and historical memories, migrating highly significant reflective insights into long-term memory to support future policy generation and optimization.

During training, when the system receives a query containing stock identifiers, dates, and contextual trading features, it synchronously activates the market-information processing module and the reflection module. The market-information processing module parses prices, news, and financial reports, generating supervisory labels derived from characteristic changes across consecutive trading days. These labels provide the supervised signals required for model training.

During testing, since future prices are unavailable, the system employs a trend estimate based on historical (M)-day returns as a weakly supervised baseline. Simultaneously, the reflection module contributes analytical results that supplement the model's cognitive capabilities. At each decision step, the system integrates three sources of information: historical price sequences, content produced by the reflection module, and the top-ranked memory fragments retrieved from the memory module. This forms a multimodal reasoning pipeline with memory-augmented enhancement. Such a mechanism significantly improves the Agent's decision stability and strategic reliability under conditions where future information is inaccessible.

Market Environment Module

To simulate real trading processes, the market environment module is designed to mimic an actual stock market and advance the trading process day by day, forming a market simulation. In stock trading, this module enables the Agent to interact with simulated or real market data, supplying semantically complete market states in a highly structured and temporally ordered manner. This supports complex decision-modeling tasks. The market environment is not only a data carrier but also a critical interface throughout strategy learning and deployment. Its design reflects key characteristics of real stock markets, including incomplete information, causal delays, and uncertainty due to volatility.

During training, the market environment provides full samples annotated with future price-change labels, enabling the policy function π_θ to be optimized under supervised signals. As shown in Equation (9), p_t, n_t, $f_t^{(k)}$, $f_t^{(q)}$, and c_t represent price vectors, news text, annual reports, quarterly reports, earnings-call transcripts, and labels based on closing-price movements:

$$x_t = \left(p_t,\, n_t,\, f_t^{(k)},\, f_t^{(q)},\, c_t\right), \qquad y_t = \operatorname{sign}\left(p_{t+1}^{\text{close}} - p_t^{\text{close}}\right) \tag{9}$$

Training optimizes the mapping between the Agent's input features x_t and market feedback. By iterating through the complete sequence of dates in the training interval, the environment ensures the sample density and temporal continuity required for supervised learning.

During testing, future prices y_t, which represent changes in closing prices, are not available. This

setting reflects real deployment conditions in which only partial observations can be accessed. In this phase, the environment outputs the current observation x_t and supplements it with historical return trends, reflection-memory entries, and short-term semantic summaries, which serve as auxiliary information sources for action generation. This design enables the system to perform inference without future labels, thereby achieving a clear separation between training and testing and improving generalization capability.

Prompt Module

The prompt module is a core component that guides the large language model in performing investment analysis and generating trading decisions. By providing structured templates that specify task context, memory-retrieval objectives, and strategy-execution logic, the module ensures that the generated outputs maintain strategic consistency, interpretability, and behavioral constraints. Depending on the operating mode (training or testing) and task type (memory retrieval or decision attribution), the prompt module includes the following categories.

Memory-ID Description Prompts
Unified identifiers are defined for different memory levels to retrieve stored information: "short: short-term memory ID; mid: mid-term memory ID; long: long-term memory ID; reflection: reflection memory ID."

Memory-Retrieval Prompts
These prompts guide the model to extract the most relevant information from the specified memory layer, with separate designs for training and testing phases. Training mode: "From memory level provided by major investment institutions, find the information most relevant to the investment decision." Testing mode: "From memory level, find the information most relevant to your investment decision."

Trading-Reason Summary Prompts
These prompts instruct the model to summarize the reasons behind current trading actions based on observations: "Explain the reasoning behind the professional trader's recommendation." "Summarize and explain the investment decision based on textual information and price movements."

Investment Information Prefix Prompts
These prompts provide contextual initialization, including current date, stock symbol, and market background, activating the model's memory and comprehension mechanisms: "The current date is cur_date, and the observed price difference of symbol from the current to the next trading day is future_record." "The stock symbol under analysis is symbol, and the current date is cur_date."

Strategy-Execution Prompts
These prompts clarify the decision-making requirements and incorporate risk preferences and market signals.

Training mode: "Given the following information, explain why the financial market fluctuates from the current day to the next. When portfolio value is positive, you are a risk-seeking investor; when negative, you are risk-averse. The decision rule is: if the future price change is positive or

neutral, make a buy decision; if negative, make a sell decision. Always make an explicit decision (buy or sell) based on future price movements."

Testing mode: "Make an investment decision based on the provided information and summarize the reasoning. Consider the following elements:

- Available short-term, mid-term, long-term, and reflection information;
- Current portfolio position;
- Risk preference (risk-seeking when portfolio value exceeds initial capital, and risk-averse when it falls below).

Decision rules: buy when there is a positive or neutral signal; hold when signals are unclear; and sell when negative signals are present."

The prompt module provides hierarchical templates that offer clear behavioral instructions to the large language model. These templates ensure that generated outputs preserve the flexibility of natural language while maintaining controllability and interpretability required for structured tasks. The prompts serve different functional purposes across execution phases, forming a crucial bridge between perceptual inputs and strategic outputs.

Evaluation Methods

During evaluation, the Buy and Hold strategy (B&H) is adopted as the baseline. Five traditional financial metrics are used for comprehensive assessment: Cumulative Return, Sharpe Ratio, Annualized Volatility, Daily Volatility, and Maximum Drawdown. A rigorous experimental design and a complete performance assessment ensure the reliability and scientific validity of model training.

Cumulative Return

Cumulative Return is an important performance indicator that reflects overall investment profitability, particularly for strategies emphasizing long-term growth and reinvestment. As shown in Equation (10), r_i denotes the log return on day $t+1$; p_t and p_{t+1} represent the closing prices on days t and $t+1$, respectively; and action_t refers to the trading decision generated by the model on day t.

$$\text{Cumulative Return} = \sum_{t=1}^{n} r_i = \sum_{t=1}^{n} \left[\ln\left(\frac{p_{t+1}}{p_t} \right) \cdot \text{action}_t \right] \tag{10}$$

Sharpe Ratio

The Sharpe Ratio is another core metric that evaluates investment performance by adjusting for risk. As shown in Equation (11), the ratio is computed by dividing the portfolio's average excess return $(R_p - R_f)$ by its volatility σ_p. A higher value indicates better performance after risk

adjustment.

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \tag{11}$$

Daily Volatility

Daily Volatility measures the magnitude of price fluctuations within a single trading day and is typically computed as the standard deviation of log returns. It reflects short-term risk exposure and plays an important role in risk management, option pricing, and portfolio construction.

$$\sigma_d = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(r_t - \bar{r})^2} \tag{12}$$

Annualized Volatility

Annualized Volatility is obtained by multiplying Daily Volatility by the square root of the typical number of trading days in a year (252). This indicator is essential for evaluating investment risk, reflecting the degree of fluctuation in annual returns.

$$\text{Annum-Volatility} = \text{Daily Volatility} \times \sqrt{252} \tag{13}$$

Maximum Drawdown

Maximum Drawdown (MDD) measures the largest decline from a portfolio's peak value to its trough before a new peak is reached. Smaller drawdowns indicate greater robustness and lower risk.

$$\text{MDD} = \max_{t \in [0,T]} \left(\frac{\text{Peak Value}_t - \text{Trough Value}_t}{\text{Peak Value}_t} \right) \tag{14}$$

Cumulative Return Calculation Method

To better evaluate model performance and simulate real investor–market interactions, cumulative returns are calculated after testing. Using historical price sequences and trading signals, the portfolio state is dynamically updated.A discrete-time model is used. With an initial capital of \$100,000, stock price p_t, and trading signal $a_t \in \{-1, 0, +1\}$ representing sell, hold, and buy, the account contains the cash balance cash_t and the position value position_t. A transaction fee of 0.1% is charged for each trade to simulate real-world frictions.

Trading rules: If $a_t > 0$, the system buys as many shares as possible within cash limits, deducting the required transaction fee. If $a_t < 0$, the system sells up to the current position, adds the proceeds to cash, and deducts the fee. If $a_t = 0$, the system maintains existing positions and updates the portfolio value according to market prices.Cumulative return at time t is computed using Equation (15):

$$\mathcal{R}_t = \frac{\text{cash}_t + \text{position}_t \cdot p_t - C_0}{C_0} \tag{15}$$

The full return sequence is recorded, starting from 0 on the first day. This design incorporates realistic capital constraints, position controls, and transaction costs, avoids reliance on idealized future information, and ensures high deployability and credible evaluation.

Baseline

The Buy-and-Hold (B&H) strategy serves as the benchmark in empirical finance because it provides a passive investment standard for evaluating active strategies. According to the Capital Asset Pricing Model (CAPM), the expected return of the market portfolio compensates investors for systematic risk, and B&H approximates this market-holding process.Thus, any active strategy must consistently outperform the risk-adjusted returns of B&H to demonstrate economic value, rather than relying on short-term fluctuations.The return of the B&H strategy is given in Equation (16):

$$R_{\text{B\&H}} = \prod_{t=1}^{T}(1 + r_t) - 1 \tag{16}$$

Since B&H avoids timing and security selection, its results depend solely on market trends, making it a key tool for testing market efficiency. If a strategy cannot significantly outperform B&H in the long run, it supports the weak-form Efficient Market Hypothesis; persistent outperformance may indicate pricing anomalies. The simplicity of B&H also makes it an appropriate reference for evaluating transaction-cost effects.

Data Collection

Trading Data

This study uses trading data for Tesla (TSLA) and Amazon (AMZN) from August 15, 2023 to March 28, 2025 as the research sample. Both companies are representative in their respective fields and have high trading volumes. Tesla is a central company in the electric vehicle industry, and its stock price fluctuations reflect trends related to policy guidance, technological progress, and market demand in the new energy sector. Amazon plays an important role in global e-commerce and cloud computing, and its financial data captures the overall direction of digital transformation in retail and the expansion of the cloud service market. From the perspective of data characteristics, both Tesla and Amazon exhibit high trading activity and a sufficient amount of news and public opinion information. According to disclosures from the United States Securities and Exchange Commission (SEC), Tesla had an average daily trading volume of approximately 120 million shares in 2023, and Amazon had more than 30 million shares. This sufficient trading volume provides a solid statistical basis for subsequent time series modeling. In addition, the high frequency of news reports and market commentary related to both companies allows sentiment indicators based on natural language processing to reflect dynamic changes in market expectations with greater accuracy.

Figure 2: Candlestick chart of TSLA stock prices from August 15, 2023 to March 28, 2025.

During the research window from August 15, 2023 to March 28, 2025, both Tesla and Amazon experienced significant event-driven fluctuations. Tesla faced production challenges related to the 4680 battery and volatility associated with global price competition, while Amazon operated within a context shaped by the expansion of AWS artificial intelligence services and parallel antitrust litigation. These developments make both companies representative cases in the stock market, which motivates their selection as research objects.

During the data acquisition process, fifteen financial indicators were collected from TUSHARE for both stocks. These indicators include *close*, *open*, *high*, *low*, *pre_close*, *pct_change*, *vol*, *amount*, *vwap*, *change*, *adj_factor*, *turnover_ratio*, *total_mv*, *pe*, and *pb*. Their definitions are shown in Table 2. The dataset spans the period from August 15, 2023 to March 28, 2025 and includes 407 trading days.

News Data

This study obtains raw news data through the Alpaca Markets API and the Alpha Vantage API. At the metadata level, each news record contains standardized fields including the headline, author, and publication timestamp. Alpha Vantage provides the news summaries required for later processing, while Alpaca Markets supplies only the full news content. The summaries for Alpaca data are generated in a subsequent step to prevent the input length from exceeding the limit of the large language model.

Figure 3: Candlestick chart of Amzn stock prices from August 15, 2023 to March 28, 2025.

For temporal processing, a dual encoding mechanism is implemented. In addition to retaining the original timestamp, a discrete trading day identifier is generated to improve the efficiency of time series analysis. To address incomplete content, a hybrid completion strategy is used. The system first attempts to use the full text returned by the API. When the content is detected to be incomplete, defined as fewer than 200 characters, the system automatically activates an XPath based webpage extraction module to retrieve the main text. The news data are stored in columnar Parquet format, which offers three advantages. First, the Snappy compression algorithm reduces storage space. Second, predicate pushdown enables millisecond level query latency for specific time ranges. Third, schema evolution is supported, which allows new analytical fields to be added in later stages.

A total of 3834 AMZN news items and 7634 TSLA news items are collected.

As shown in Figure 4, from August 2023 to March 2025, news coverage related to Tesla is strongly centered on stock market activity. High frequency terms in the word cloud, such as market, stock, option, price, trader, and analyst, reflect media attention to Tesla's price movements, trading behavior, and the use of derivatives. Terms such as model, electric vehicle, Cybertruck, and Musk indicate coverage related to Tesla's product line, technology announcements, and management developments. Although the content does not directly constitute trading instructions, it provides structural information useful for medium and long term strategy formulation and can support valuation assessment and trend analysis.

Indicator	Description
close	Closing price of the day reflecting the final market consensus.
open	Opening price of the day influenced by overnight information.
high	Highest transaction price of the day.
low	Lowest transaction price of the day.
pre_close	Previous day's closing price.
pct_change	Percentage change relative to the previous closing price.
vol	Total number of shares traded during the day.
amount	Total monetary value of all trades executed during the day.
vwap	Volume weighted average price for the trading session.
change	Absolute price change relative to the previous close.
adj_factor	Adjustment factor used for forward or backward price adjustment.
turnover_ratio	Trading volume divided by free float shares representing liquidity.
total_mv	Total market capitalization calculated as stock price multiplied by total shares.
pe	Price to earnings ratio reflecting valuation relative to earnings.
pb	Price to book ratio reflecting valuation relative to net asset value.

Table 2: Descriptions of financial indicators.

As shown in Figure 5, from August 2023 to March 2025, keywords such as market, stock, option, price, trade, and trader appear frequently, which indicates continuous media attention to fluctuations in Amazon's stock price, the use of trading instruments, and investor behavior. At the fundamental level, a second category of focus includes terms related to cloud service, content, e commerce, platform, growth, revenue, and technical concepts such as AI and Microsoft. These terms show strong media attention to Amazon's technology domains including cloud computing and artificial intelligence applications, its competitive position in core businesses such as retail and content services, and comparative analyses with other major technology companies such as Microsoft and Google. During this period, news related to Tesla and Amazon conveys immediate market signals and also provides sustained support for understanding corporate fundamentals. This makes news an important intermediary that connects public sentiment with trading behavior.

Financial Report Data

This study collects financial reports using the SEC EDGAR API. Because different report types contain distinct disclosure characteristics, a classification based extraction procedure is applied. For Form 10-K annual reports, the extraction focuses on qualitative sections such as Item 7 Management Discussion and Analysis. For Form 10-Q quarterly reports, the extraction emphasizes quantitative indicators in Part 1 Item 2 Management Discussion and Analysis.

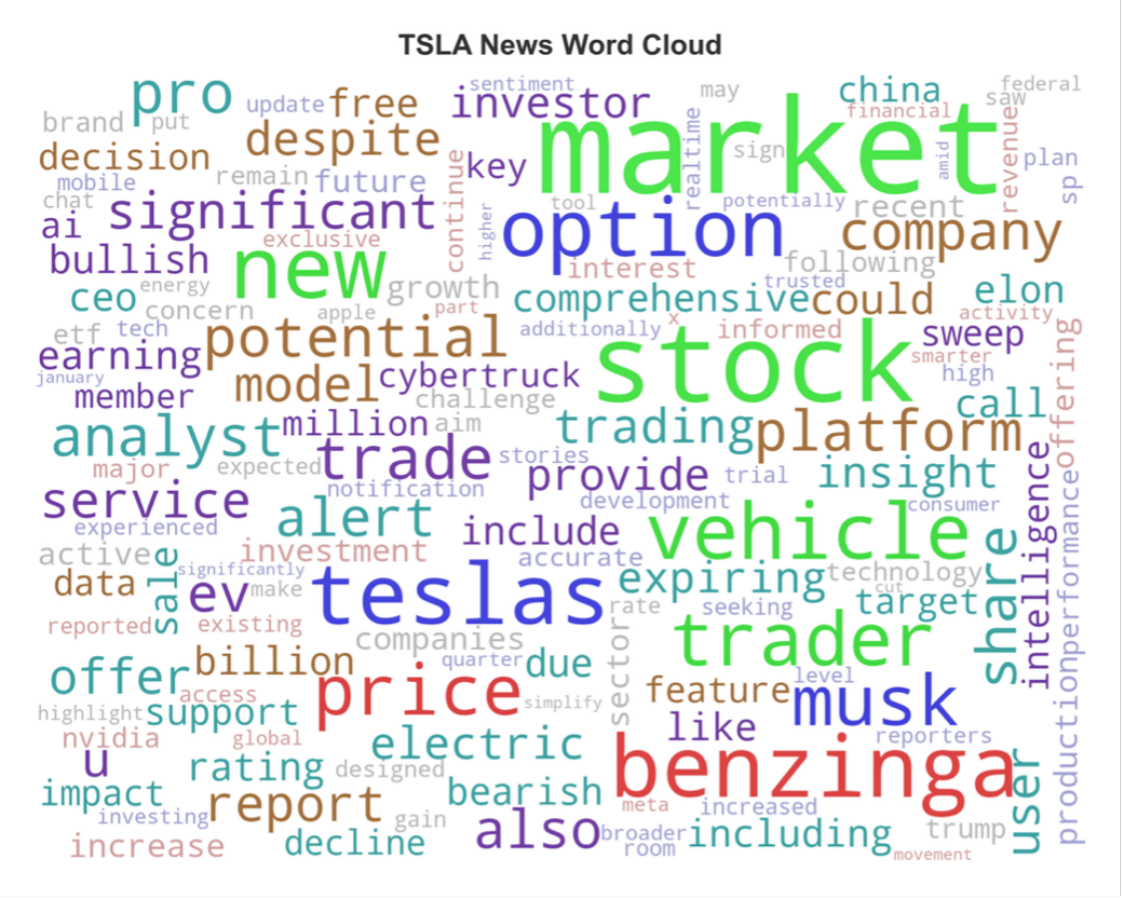

Figure 4: Word cloud of Tesla-related news from August 15, 2023 to March 28, 2025.

A category specific retrieval strategy is implemented for Form 10-K and Form 10-Q, with separate extraction rules designed for each type. Dynamic API request URLs are constructed to precisely locate key sections in the reports. The data collection process begins with the retrieval of index files that ensure the accuracy of the document source. The index includes essential fields such as the company identifier CIK, ticker symbol, and the document URL. The extracted report data are stored in standardized Parquet format to preserve a complete correspondence between the raw document structure and the extracted content.

Earnings Call Transcripts

To better simulate real market environments, this study collects quarterly earnings call transcripts for both companies during the research window using the Alpha Vantage API. A total of 328 transcripts are collected for TSLA and 151 transcripts for AMZN, yielding 479 records in total. Each record contains the fields quarter, symbol, speaker, title, and content.

Table 4 provides an example of an Amazon earnings call transcript entry.

Figure 5: Word cloud of Amzn-related news from August 15, 2023 to March 28, 2025.

10-Q (Quarterly Report)	10-K (Annual Report)
{ ”document_url”:”¡url¿”,	
”content”:”10Qreport”,	
”ticker”:”TSLA”, ”cik”:”cik”,	
”utc_timestamp”:1454112000000,	
”est_timestamp”:1454112000000,	
”type”:”10-Q” }	{ ”document_url”:”¡url¿”,
”content”:”10Kreport”,	
”ticker”:”TSLA”, ”cik”:”cik”,	
”utc_timestamp”:1454112000000,	
”est_timestamp”:1454112000000,	
”type”:”10-K” }	

Table 3: Samples of Tesla financial report data in Parquet format.

Quarter	Symbol	Speaker	Title	Content
2023Q3	AMZN	Operator	Operator	Thank you for standing by. Good day everyone and welcome to the Amazon.com Third Quarter 2023 Financial Results Teleconference. Today's call is being recorded. For opening remarks, the call will be turned over to the Vice President of Investor Relations, Dave Fildes.

Table 4: Sample of Amazon earnings call transcript data. *Source: Alpha Vantage*.

Data Preprocessing

Data Cleaning

A similarity based algorithm is applied to the raw dataset to identify redundant news content. A similarity threshold of 0.6 is used to filter duplicate reports while retaining similar items that contain meaningful information. This helps preserve the diversity of training data for the large language model. During the cleaning process, the original publication timestamps are converted to their corresponding trading days to ensure precise alignment between news events and market data. This step provides high quality inputs for subsequent modeling.

Summary Generation

To prevent the input length from exceeding the maximum token limit of the model, the cleaned news data from Alpaca Markets are summarized using the GPT-3.5-turbo-16k-0613 model. Each summary is restricted to within 200 characters. The prompt used for generating the summaries is as follows:

- Generate a concise summary for the following news content, focusing on key information and main points.
- Summary requirements:
  - Length between 100 and 200 words.
  - Use English.
  - Maintain objectivity and neutrality.
  - Highlight important facts and data.
  - Avoid subjective comments.

Table 5 provides an example of Tesla news data after summary generation.

Date	Symbol	Headline	Summary
2023-07-15	TSLA	GOP Senator On Fixing Obamacare: 'No, No, No, No'	This summer, the court is expected to rule in King v. Burwell, in which the plaintiffs argue that the language of the Affordable Care Act does not authorize the federal government to distribute health insurance tax credits in many states.

Table 5: Sample of Tesla news data after summary generation.

Source: Alpaca Markets

Dataset Construction

Sentiment Analysis

Information related to the target stock symbols is extracted from the raw dataset, including price data, news text, earnings call transcripts, and financial report files stored in pickle format. The data are filtered and reorganized by date and stock symbol. The processed dataset contains five key components, which include price data, Form 10-K annual reports, Form 10-Q quarterly reports, and news text. These components support subsequent analysis.

Sentiment scores are generated using the pretrained financial language model FinBERT[28]. FinBERT is based on a BERT architecture designed for financial text and computes sentiment scores along three dimensions: positive, neutral, and negative. The original news text is concatenated with the corresponding sentiment scores and stored for use as model input.

Table 6 presents an example of the sentiment analysis output.

News Content	Sentiment Scores
Tesla Inc. has announced plans to launch a robotaxi service in Austin, Texas, starting in June, marking a significant step towards autonomous transportation. The announcement...	Positive score: 0.9999871253967285
Neutral score: 1.3499470696842764	
Negative score: 1.155748850578675	

Table 6: Sample of sentiment analysis scores.

Data Integration

Financial information from different sources is integrated into a unified data structure to support subsequent modeling and analysis. The processed data include stock prices, related news text, earnings call transcripts, and quarterly (Form 10-Q) and annual (Form 10-K) financial reports, which are loaded from five separate pickle files. The integrated dataset uses the date as the index, and each day is organized into a tuple containing five components: (price, news, call, filing_q, filing_k). This structure summarizes all relevant environmental information for that day.

The integration script iterates through all target stocks and ensures that each date contains a standardized data structure. Empty values are preserved in cases where some fields are missing to maintain consistency of format. The final output is saved as a single pkl file that can be directly used as input to the Agent for training or for evaluating performance under complex market conditions.

Data Augmentation

In cognition driven financial trading, data augmentation is designed as a method that spans input perturbation, modality fusion, and experience expansion in order to improve the robustness and generalization ability of the model in real market environments. A multimodal attention mechanism is introduced to perform dynamic weighted fusion across heterogeneous data sources, which include price sequences, news text, and sentiment scores. Each modality is mapped into a unified representation space through an independent nonlinear attention channel that learns the weight distribution under the current market context. This design reduces the tendency of the model to overfit a single modality.

The augmentation strategy incorporates three types of perturbations that capture financial semantics. First, for price data, random masking operations are applied to selected dimensions with a preset probability to simulate missing feature scenarios that often occur in real trading due to delays or errors. Second, for the text modality, a local temporal perturbation mechanism is applied by shuffling news items within a small sliding window. This breaks static ordering patterns and encourages the model to learn trading related semantics from compositional meaning rather than fixed sequence positions. Third, noise perturbation is introduced into the sentiment modality by adding zero mean Gaussian noise to the sentiment scores, which simulates fluctuations in investor sentiment and subjective interpretation uncertainty.

The overall augmentation design expands the training sample space and improves the model's ability to handle uncertain or corrupted data. It also provides a more interpretable reasoning foundation at the strategy level, supporting the Agent in complex decision making tasks.

Training and Testing Sets

In this study, the dataset is divided into a training set (from 2023-08-15 to 2024-12-31) and a testing set (from 2025-01-01 to 2025-03-28). The split is designed according to the temporal dependence of multimodal financial data and the need for rigorous generalization evaluation. This partition strictly follows chronological order, which prevents future information leakage and ensures that

the model is exposed only to information that would have been available at the time of training. This preserves causal consistency throughout the learning process.

The training set covers significant market adjustment periods in late 2023 and mid term rebounds in 2024. These variations provide a sufficiently rich set of multimodal inputs, which benefits the dynamic weighting mechanism across modalities and enhances the capability of the attention structure to model the importance of heterogeneous information.

During the loading and processing of training data, information from multiple modalities is fused. Masking is applied to price sequences, temporal perturbation is used to shuffle the order of news items, and noise injection is applied to sentiment scores. These procedures reduce the risk of overfitting in environments characterized by high frequency text and noisy market conditions. They also prevent the training process from being dominated by any single modality, preserving the robustness of the overall trading strategy.

The testing set spans the beginning of 2025 to the end of the first quarter. This interval lies completely outside the training horizon and thus provides a realistic evaluation scenario in which the model must adjust modality combinations to unseen market states. The chosen partition balances the diversity of historical data with the challenges of future market behavior and provides a solid foundation for assessing robustness, cross period adaptability, and the model's tendency to overfit. Table 7 presents dataset time ranges.

Company	Ticker	Train Start	Train End	Test Start	Test End
Tesla	TSLA	2023-08-15	2024-12-31	2025-01-01	2025-03-28
Amazon	AMZN	2023-08-15	2024-12-31	2025-01-01	2025-03-28

Table 7: Training and testing set time ranges.

Text Embeddings

In this study, the structured representation of textual information relies on embedding vectors generated by a pretrained language model. The text-embedding-3-large model provided by OpenAI is adopted due to its strong expressive capacity for large scale language modeling tasks. It maps natural language text into fixed dimensional dense vectors that capture semantic content. The embedding operation is defined in Equation (17):

$$e_x = \mathrm{Embed}(x) \in R^d \tag{17}$$

Given an input text segment x, the model produces an embedding vector $\mathbf{e}_x$ by calling the text-embedding-3-large encoder. The parameter d denotes the embedding dimension, which is set to 1536 by default. The embedding function Embed serves as an encoder that transforms natural language inputs into distributed representations in the embedding space.

These embeddings are used to compute cosine similarity between query vectors and memory vectors in order to retrieve the most relevant memory events. They are also used to encode model generated reflective content, which supports higher level reasoning and memory writing. Furthermore, embedding vectors serve as the textual modality input that participates in downstream inference together with price and sentiment modalities. The embedding process forms the foundation for structured textual representation in this study and significantly improves the semantic consistency and computational operability of multimodal financial information.

Agent Behavior Analysis

Market Analysis Word Cloud

Based on the word cloud shown in Figure 6, a detailed examination can be conducted regarding the semantic focus and interpretive ability of the Agent when processing market related textual data. The word cloud represents the high frequency terms that appear across multiple trading day decisions for AMZN, revealing the focal points in the Agent's semantic construction of the market environment.

Figure 6: Word cloud of market analysis generated by the Agent on the AMZN dataset.

The distribution of high frequency words indicates that the Agent pays close attention to key con-

cepts such as “macroeconomic”, “fluctuation”, “investment”, “aws”, “web services”, “cloud computing”, and “analyst”. This suggests that the Agent is capable of correctly identifying critical factors that influence Amazon’s market performance, including the macroeconomic environment, technological services, and financial analyst opinions. Terms such as “concern”, “positive developments”, and “strategic advancements” further demonstrate the Agent’s ability to detect shifts in market sentiment and recognize corporate strategic directions.

From a structural perspective, the vocabulary can be grouped into three major categories. The first category includes terms that describe market dynamics and uncertainty, such as “fluctuation”, “concerns”, and “macroeconomic”. The second category includes terminology associated with capital operations and investment intentions, such as “investment firms”, “major investment”, and “analyst”. The third category contains expressions related to business development and technological trajectories, such as “aws”, “cloud services”, “ecommerce”, and “strategic advancements”. The coexistence of words such as “optimism”, “confidence”, “pressure”, and “uncertainty” indicates that the Agent incorporates both positive expectations and potential risks in its reasoning and demonstrates a balanced understanding of market sentiment.

Overall, the word cloud illustrates not only the Agent’s semantic focus in financial text processing but also its capability in financial reasoning, thematic extraction, and sentiment interpretation. These characteristics provide meaningful semantic support for subsequent analyses of the Agent’s decision making behavior.

Analysis of Text Length

Figure 7 illustrates the variation in the length of market analysis text generated by the Agent from August 2023 to March 2025. Overall, most text lengths fall between 40 and 70 words, indicating that under normal conditions the Agent tends to produce medium length analytical content. This range is generally sufficient to capture market sentiment, explain price movements, and provide simple forward looking observations. The pattern suggests that the model follows a concise and efficient expression strategy during routine operation.

Several time periods exhibit notable fluctuations in text length. In early November 2023 and mid July 2024, two significant peaks appear in which the generated text exceeds one hundred words. This behavior likely reflects the model’s response to major market events such as earnings releases, changes in macroeconomic policy, or sharp stock price movements. During these periods the model tends to produce longer analysis in order to incorporate broader information, elaborate causal reasoning, or provide more detailed strategic remarks. This indicates that the Agent can adjust output intensity according to market complexity and demonstrates input sensitivity and contextual adaptation.

In contrast, there are also instances where the text length is very short or nearly zero, scattered throughout the entire period, including late October 2023, early March 2024, and late January 2025. These low values correspond to days when the stock market was closed for weekends or when no relevant news was available. Such behavior is reasonable in practical financial analysis and helps reduce the influence of overfitting or noise induced generation.

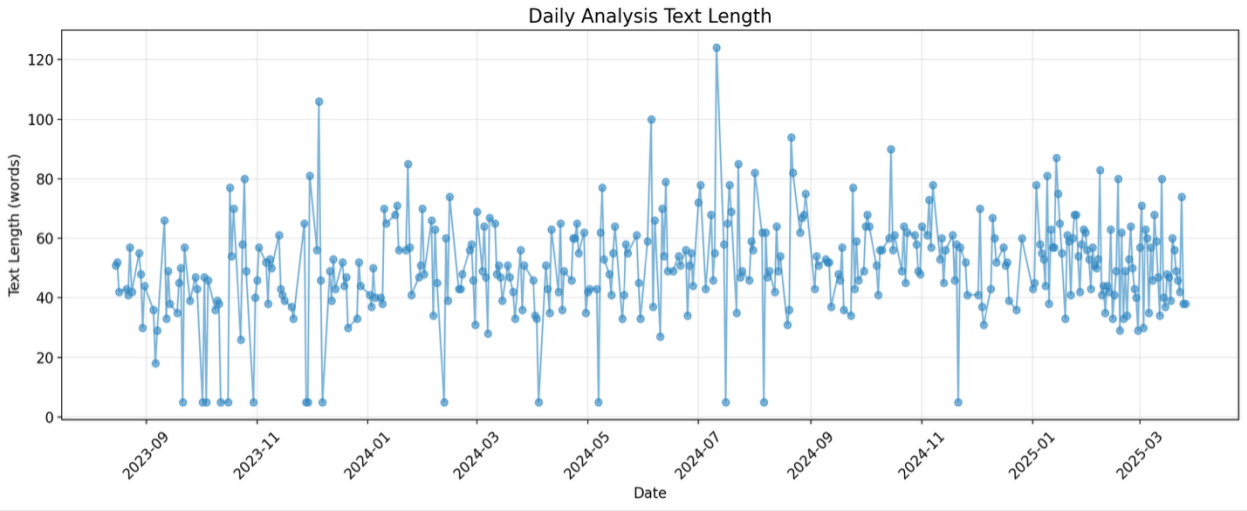

Figure 7: Daily length of market analysis text generated by the Agent.

Between June and September 2024, the volatility of text length increases substantially, with frequent alternation between high and low values. This pattern suggests that the market environment during this phase was relatively unstable and that the information received by the Agent exhibited high uncertainty or diversity. As a result, the model adopted different generation strategies across days. This behavior may reflect the Agent's attempt to dynamically adjust the semantic coverage when operating under complex conditions and demonstrates the presence of an elastic response mechanism.

Overall, the Agent constructed in this study maintains a stable output style and typically generates moderately sized analytical text while demonstrating the ability to expand information coverage during significant events and to limit unnecessary generation when information is sparse. This length control strategy indicates that the Agent possesses a degree of regulation in its generative structure and can adjust expressive scope according to input complexity and market conditions. This property contributes to producing analytic content that is stable in quality and reasonable in semantic density.

Experimental Results

Results on the TSLA Dataset

Figure 8 presents the test performance of the Agent, implemented with the GPT-4o-Turbo-2024-05-13 model, on the TSLA dataset. The evaluation is based on the cumulative return metric described in Section 3.2.7. Compared with the traditional Buy and Hold strategy, the Agent maintains a stable return profile during the period of significant market fluctuations from early 2025 to late March.

At the beginning of the test interval, the return trajectory remains relatively smooth. From February to March, when TSLA experienced a substantial market drawdown, the Agent successfully avoided

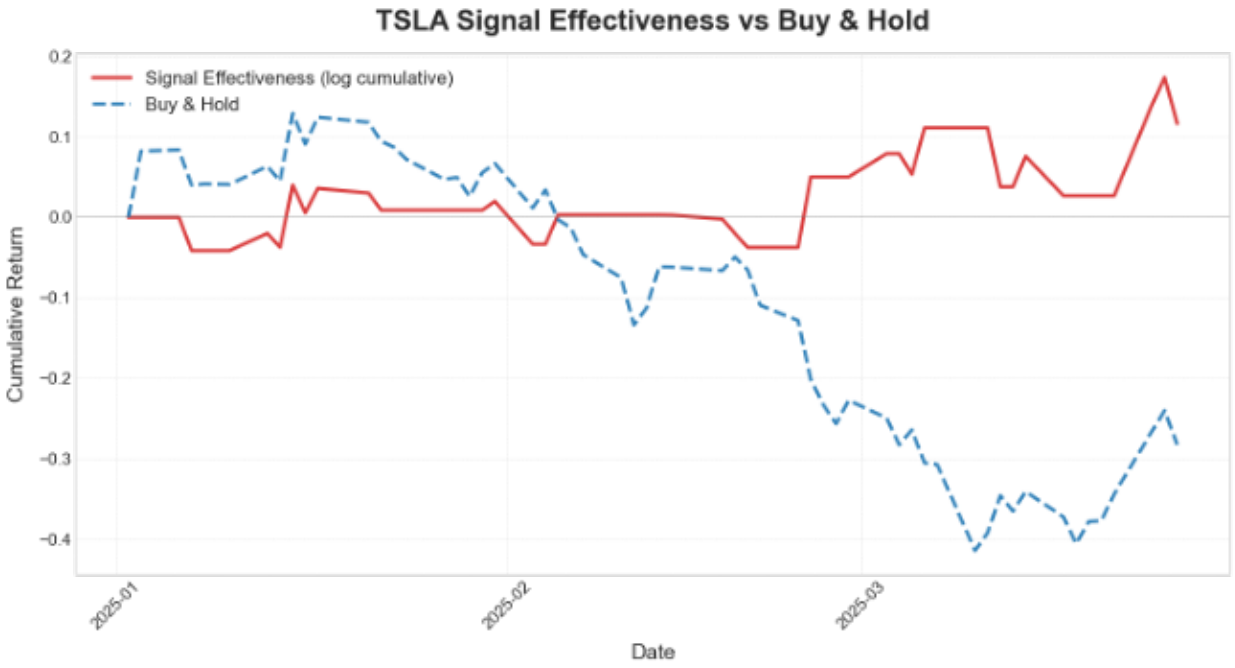

Figure 8: Trading performance of the Agent on the TSLA dataset during training.

the systemic decline and subsequently achieved steady growth, resulting in an approximate twenty percent positive cumulative return.

In contrast, the Buy and Hold strategy performs poorly during the same period. The downward market movement from mid February to early March leads to a sharp drawdown, with cumulative losses approaching fifty percent. This comparison demonstrates that the GPT-4o based Agent not only exhibits trend recognition capabilities but also adjusts adaptively to changing market conditions, which significantly improves the stability and safety of overall returns.

The overall performance on the test set indicates that the Agent possesses a notable degree of generalization ability. It is able to achieve positive returns on previously unseen data while maintaining controlled risk exposure. The strong robustness and resistance to downturns observed in the results highlight the Agent's potential to operate as a conservative and reliable strategy in real world financial applications.

Results on the AMZN Dataset

Figure 9 shows the cumulative return performance of the Agent on the AMZN test set and compares it with the Buy and Hold strategy. The overall market volatility during the test period is relatively high, and both strategies experience varying degrees of decline. Under these conditions, the Agent based on the GPT-4o-Turbo-2024-05-13 model adopts a comparatively conservative approach. Although the final cumulative return is slightly negative, the maximum drawdown remains limited, which demonstrates effective risk control.

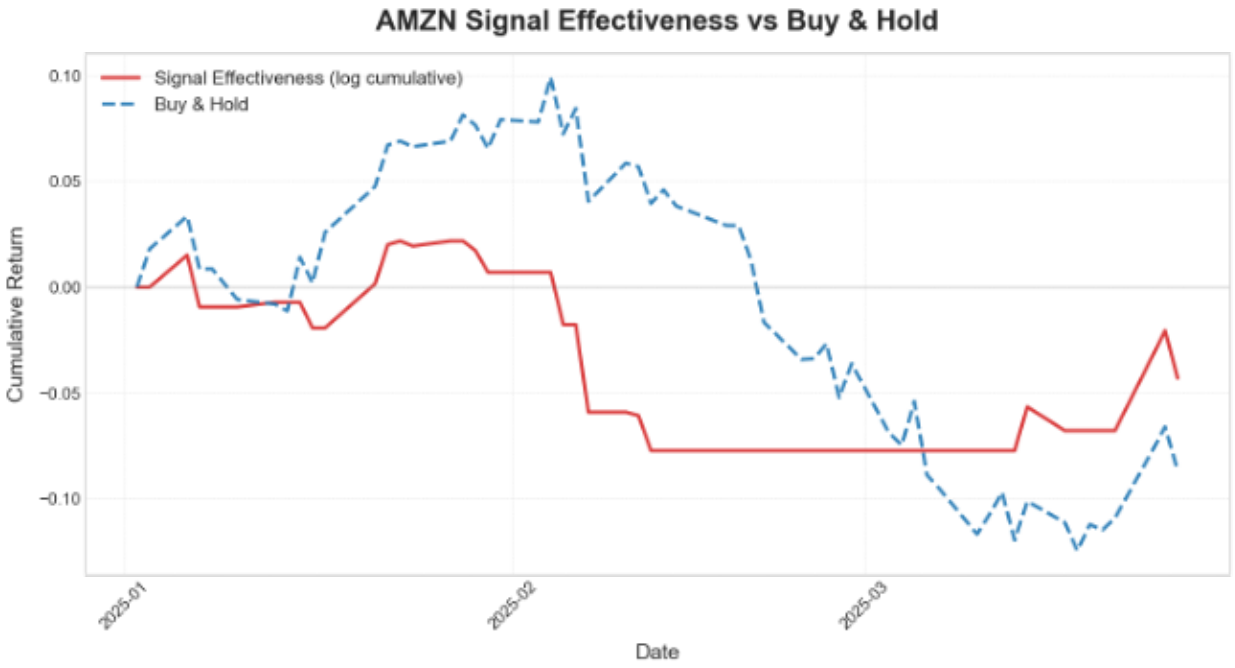

Figure 9: Trading performance of the Agent on the AMZN dataset during testing.

In contrast, the Buy and Hold strategy exhibits a period of notable gains early in the test interval, with cumulative returns briefly exceeding eight percent. However, the strategy subsequently declines sharply due to the broader market correction and ends the period with negative returns. The Agent reduces its positions during the weakening market phase and shows stronger resistance to losses despite not achieving positive returns by the end of the test.

Overall, the performance of the Agent on the AMZN dataset is slightly weaker than the Buy and Hold strategy in terms of cumulative return. Nevertheless, it demonstrates a meaningful level of stability and risk awareness. At the beginning of the test period, the Agent remains cautious and avoids entering a full position, which prevents large early gains but reflects prudent behavior. During the middle and later stages, the Agent identifies the downward trend and chooses to stay out of the market, which stabilizes cumulative returns over the extended decline and prevents further losses. The Agent also initiates positions when an upward trend reappears, yielding modest gains.

Although the Agent's generalization ability on AMZN appears less impressive than its performance on TSLA and its overall behavior remains conservative, it successfully detects both upward and downward movements and avoids severe losses. The strategic logic remains stable throughout the test. Future improvements may focus on strengthening the detection of local market trends in order to enhance responsiveness and improve return potential in complex conditions.

Financial Metric Analysis

Figure 1 compares the performance of the Agent based on the GPT-4o-Turbo-2024-05-13 model with the Buy and Hold strategy on TSLA and AMZN. Five core financial metrics are evaluated,

including cumulative return, Sharpe ratio, standard deviation, annualized volatility, and maximum drawdown. All metrics are normalized to enhance comparability across different scales.

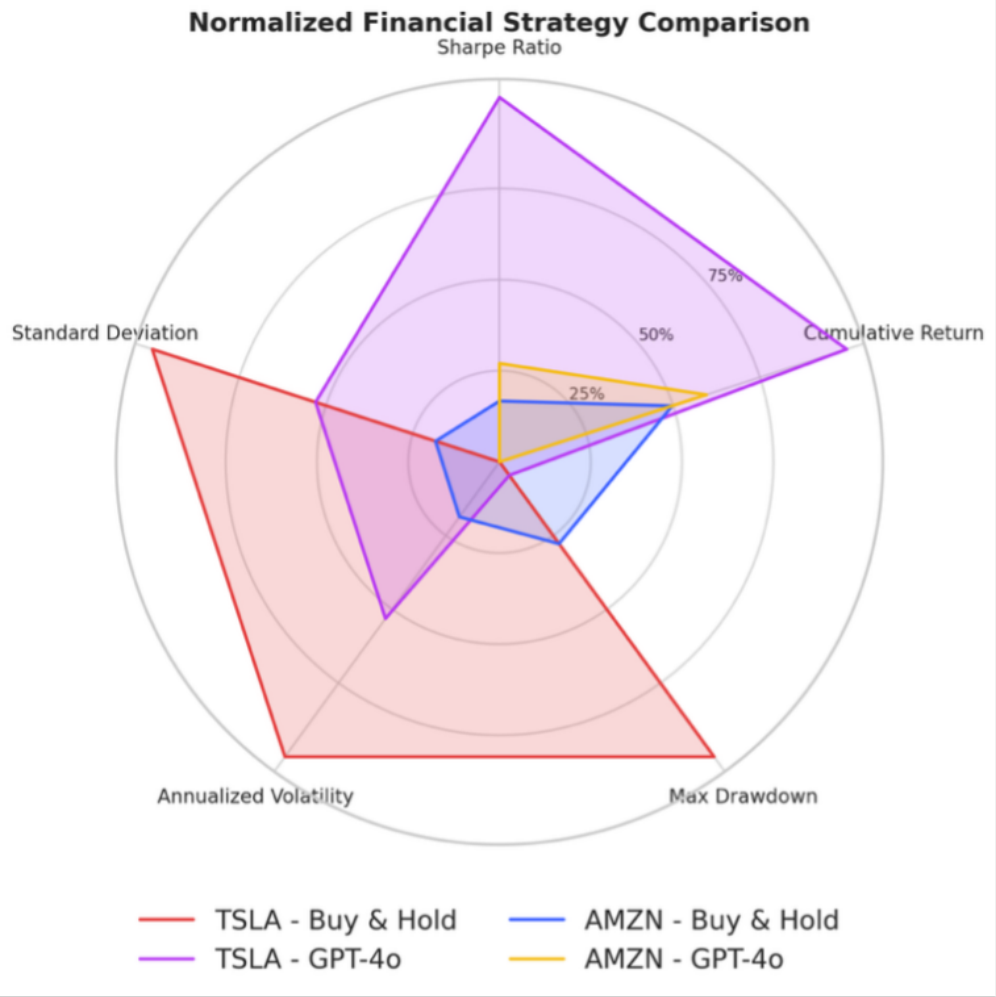

Figure 10: Radar chart comparing normalized financial performance metrics.

For TSLA, the Agent (represented by the purple region) significantly outperforms the Buy and Hold strategy (represented by the red region) in both cumulative return and Sharpe ratio. The Agent achieves near maximum values in the Sharpe ratio dimension, which indicates that it generates positive returns along with substantially higher risk adjusted performance. The Agent also shows strong volatility control, with annualized volatility and maximum drawdown both considerably lower than those of the Buy and Hold strategy, which reflects effective risk management.

For AMZN, the Agent (represented by the yellow region) still leads the Buy and Hold strategy (represented by the blue region) across most metrics, although the advantage is less pronounced. In particular, the gaps in standard deviation and maximum drawdown are relatively small. This suggests that the Agent adopts a more conservative signal distribution on AMZN, prioritizing drawdown control while maintaining moderate performance across other indicators.

As shown in Table 8, in terms of cumulative return, the Agent achieves a return of +15.3% on TSLA, while the Buy & Hold strategy incurs a loss exceeding -33%. This substantial gap indicates that the Agent effectively captures upward opportunities during both training and testing periods while avoiding part of the downside risk. For AMZN, although both strategies remain in the negative return region, the Agent exhibits a smaller loss of only -4.3%, compared with -9.1% for Buy & Hold, demonstrating a certain degree of downside resilience.

The Sharpe ratio measures excess return per unit of risk. On TSLA, the Agent achieves a Sharpe ratio of 1.43, which represents a healthy level of risk-adjusted performance and indicates that the Agent obtains favorable returns while maintaining controlled volatility. In contrast, the Buy & Hold strategy yields a Sharpe ratio of -1.99, suggesting low returns accompanied by substantial risk. On AMZN, both strategies exhibit negative Sharpe ratios, but the Agent performs slightly better with a value of -1.07, implying that while profits are not achieved, risk exposure remains relatively more controlled.

Table 8 summarizes the key financial performance metrics of the Agent and the Buy & Hold strategy on the TSLA and AMZN datasets.

Metric	**TSLA (Buy & Hold)**	**TSLA (GPT-4o)**	**AMZN (Buy & Hold)**	**AMZN (GPT-4o)**
Cumulative Return	-0.3323	0.1533	-0.0906	-0.0429
Sharpe Ratio	-1.9895	1.4254	-1.4195	-1.0670
Standard Deviation	0.0465	0.0299	0.0177	0.0112
Annualized Volatility	0.7383	0.4757	0.2824	0.1779
Max Drawdown	0.5001	0.1140	0.2078	0.0955

Table 8: Overall financial metric scores for TSLA and AMZN.

Standard deviation reflects the magnitude of day to day return fluctuations. The Agent exhibits standard deviations of 0.0300 and 0.0112 on TSLA and AMZN, respectively, both of which are substantially lower than those of the Buy & Hold strategy, which records values of 0.0465 and 0.0178. This indicates that the Agent follows a more stable strategy with reduced return variability, making it more suitable for long term holding.

Annualized volatility further quantifies overall risk exposure. On TSLA, the Agent records an annualized volatility of 0.476, significantly lower than the 0.738 observed under Buy & Hold. On AMZN, the Agent again maintains lower volatility, with a value of 0.178 compared to 0.282. These results confirm that the Agent consistently demonstrates effective risk control across multiple assets, reflecting stable trading behavior and strong resistance to market fluctuations. Such low volatility strategies are particularly attractive in institutional investment scenarios.

Table 9 provides a detailed interpretation of the financial metrics on the TSLA dataset.

Metric	Buy & Hold	GPT-4o	Interpretation
Cumulative Return	-33.2%	+15.3%	The Agent clearly outperforms Buy & Hold by avoiding losses and achieving profits.
Sharpe Ratio	-1.99	+1.43	The Agent achieves strong risk adjusted returns, while Buy & Hold exhibits excessive risk.
Standard Deviation	0.0465	0.0300	Lower volatility indicates a more stable strategy.
Annualized Volatility	0.738	0.476	The Agent effectively controls overall risk exposure.
Max Drawdown	50.0%	11.4%	The Agent shows stronger resilience under extreme downturns.

Table 9: Interpretation of financial metrics on the TSLA dataset.

Tables 9 and 10 provide detailed interpretations of the five financial metrics on the TSLA and AMZN datasets. Further examination reveals that on AMZN, the Agent achieves a daily standard deviation of 0.0112 and an annualized volatility of 0.178, both lower than the corresponding Buy & Hold values of 0.0178 and 0.282. This reflects reduced return fluctuations and more stable overall performance. In terms of maximum drawdown, the Agent achieves approximately a 50% reduction in downside magnitude, effectively mitigating potential capital losses.

Overall, the Agent demonstrates comprehensive advantages in both return generation and risk control on TSLA. On AMZN, although absolute returns remain limited, the strategy maintains notable stability and downside resistance. These results indicate that the model possesses strong generalization capability and can adapt its risk control and decision making behavior to different market structures.

Conclusion

From a behavioral perspective, the Agent demonstrates strong stability and contextual adaptability in daily market analysis tasks. The distribution of text length is relatively concentrated, which

Metric	Buy & Hold	GPT-4o	Interpretation
Cumulative Return	-9.1%	-4.3%	Both strategies incur losses, but the Agent limits downside more effectively.
Sharpe Ratio	-1.42	-1.07	Risk adjusted performance remains weak, with the Agent slightly superior.
Standard Deviation	0.0178	0.0112	The Agent exhibits significantly lower volatility.
Annualized Volatility	0.282	0.178	The Agent maintains milder risk exposure.
Max Drawdown	20.8%	9.6%	The Agent substantially reduces potential losses.

Table 10: Interpretation of financial metrics on the AMZN dataset.

indicates consistent expression during routine analysis. When major market events occur, the Agent expands its output dynamically, reflecting sensitivity to information intensity. In periods of low information availability or mild market fluctuations, it reduces the length of generated content and avoids unnecessary analysis. This adjustable generative behavior contributes to improving the semantic quality and reliability of strategy related decisions.

The Agent exhibits strong generalization ability on the test sets. For both TSLA and AMZN, it outperforms the Buy and Hold strategy in cumulative return and maintains significantly lower volatility and maximum drawdown. These results indicate that the Agent achieves superior risk adjusted performance within a controlled evaluation environment and successfully captures structural market patterns present in the training data through multimodal information integration.

On the TSLA dataset, the Agent achieves a positive return of nearly twenty percent and avoids the severe drawdown experienced by the Buy and Hold strategy, which highlights its risk hedging capability and generalization potential. On the AMZN dataset, although the model does not yield a positive return, its drawdown control is noticeably better than the baseline and the overall behavior is more stable. This suggests that the Agent retains essential risk resistance during testing, although opportunities remain for improving adaptability to different market characteristics.

Overall, the Agent constructs a clear and effective trading logic from the training data and maintains strong stability and risk control across most test scenarios. It demonstrates initial generalization potential and practical value as a deployable trading system. The strategy exhibits strong risk awareness, low volatility, and basic adaptability to multimodal financial environments. Future

work may focus on enhancing generalization to unseen market structures and improving profitability in oscillatory or neutral market conditions.

References

[1] Black F. Noise. *The Journal of Finance*, 1986, 41(3): 528–543.

[2] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020, 33: 1877–1901.

[3] Lund B D, Wang T. Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library Hi Tech News*, 2023, 40(3): 26–29.

[4] Liu X, Yu H, Zhang H, et al. AgentBench: Evaluating LLMs as agents. arXiv:2308.03688, 2023. https://arxiv.org/abs/2308.03688.

[5] Cambria E, White B. Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine*, 2014, 9(2): 48–57.

[6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013: 3111–3119.

[7] Hochreiter S, Schmidhuber J. Long Short-Term Memory. *Neural Computation*, 1997, 9(8): 1735–1780.

[8] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017: 5998–6008.

[9] Chang Y, Wang X, Wang J, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2024, 15(3): 1–45.

[10] Peter N, Intelligence R S A. *A Modern Approach*. Boston: Pearson Education, 2021.

[11] Edwards R D, Magee J, Bassetti W H C. *Technical Analysis of Stock Trends*. Boca Raton: CRC Press, 2018.

[12] Fischer T G. Reinforcement learning in financial markets: a survey. FAU Discussion Papers in Economics, 2018.

[13] Millea A. Deep reinforcement learning for trading: a critical survey. *Data*, 2021, 6(11): 119.

[14] Achiam J, Adler S, Agarwal S, et al. GPT-4 technical report. arXiv:2303.08774, 2023. https://arxiv.org/abs/2303.08774.

[15] Anil R, Dai A M, Firat O, et al. PaLM 2 technical report. arXiv:2305.10403, 2023. https://arxiv.org/abs/2305.10403.

[16] Guo D, Yang D, Zhang H, et al. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv:2501.12948, 2025. https://arxiv.org/abs/2501.12948.

[17] Xi Z, Chen W, Guo X, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 2025, 68(2): 121101.

[18] Huang X, Liu W, Chen X, et al. Understanding the planning of LLM agents: A survey. arXiv:2402.02716, 2024. https://arxiv.org/abs/2402.02716.

[19] Renze M, Guven E. Self reflection in LLM agents: Effects on problem solving performance. arXiv:2405.06682, 2024. https://arxiv.org/abs/2405.06682.

[20] Zhang Z, Bo X, Ma C, et al. A survey on the memory mechanism of large language model based agents. arXiv:2404.13501, 2024. https://arxiv.org/abs/2404.13501.

[21] Yang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024, 18(6): 186345.

[22] Li Y, Lin Z, Zhang S, et al. Making language models better reasoners with step aware verifier. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. ACL, 2023: 5315–5333.

[23] Liu X Y, Wang G, Yang H, et al. FinGPT: Democratizing internet scale data for financial large language models. arXiv:2307.10485, 2023. https://arxiv.org/abs/2307.10485.

[24] Wu S, Irsoy O, Lu S, et al. BloombergGPT: A large language model for finance. arXiv:2303.17564, 2023. https://arxiv.org/abs/2303.17564.

[25] Lefevre E, Markman J D. *Reminiscences of a Stock Operator: With New Commentary and Insights on the Life and Times of Jesse Livermore*. Hoboken: John Wiley and Sons, 2010.

[26] Yu Y, Li H, Chen Z, et al. FinMem: A performance enhanced LLM trading agent with layered memory and character design. In: *Proceedings of the AAAI Symposium Series*. AAAI, 2024, 3(1): 595–597.

[27] Park J S, O'Brien J, Cai C J, et al. Generative agents: Interactive simulacra of human behavior. In: *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2023: 1–22.

[28] Huang A H, Wang H, Yang Y. FinBERT: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 2023, 40(2): 806–841.