Language Model Guided Reinforcement Learning in Quantitative Trading

Adam Darmanin
University of Malta
Msida, Malta
adam.darmanin.03@um.edu.mt

Vince Vella
University of Malta
Msida, Malta
vvell04@um.edu.mt

***Abstract*—Algorithmic trading requires short-term tactical decisions consistent with long-term financial objectives. Reinforcement Learning (RL) has been applied to such problems, but adoption is limited by myopic behaviour and opaque policies. Large Language Models (LLMs) offer complementary strategic reasoning and multi-modal signal interpretation when guided by well-structured prompts.**

This paper proposes a hybrid framework in which LLMs generate high-level trading strategies to guide RL agents. We evaluate (i) the economic rationale of LLM-generated strategies through expert review, and (ii) the performance of LLM-guided agents against unguided RL baselines using Sharpe Ratio (SR) and Maximum Drawdown (MDD).

Empirical results indicate that LLM guidance improves both return and risk metrics relative to standard RL.

***Index Terms*—Large Language Models, Reinforcement Learning, Algorithmic Trading, Prompt Engineering, Agents**

I. Introduction

Algorithmic trading requires short-term execution aligned with long-term financial objectives, and accounts for over 60% of U.S. equity volume, particularly in high-frequency trading (HFT) [1]. Long-horizon strategies often build on econometric models such as the Fama–French five-factor framework [2], while short-horizon approaches exploit transient inefficiencies through momentum and mean reversion [3]. The most extreme form is HFT, where firms execute thousands of transactions per second by exploiting order book imbalances [1].

Modern trading systems leverage machine learning (ML) to process structured and unstructured data streams. Growth is driven by advances in electronic infrastructure, compute power, and the proliferation of large high-resolution financial data [4].

RL formalises trading as sequential decision making [5] and has shown promise through methods such as Deep Q-Networks (DQN) and actor–critic variants [3], [6]. Yet practical adoption is hindered by sparse rewards, credit assignment issues, and opaque policies, which reduce trust in high-stakes financial settings [7].

LLMs offer complementary strengths. They can understand heterogeneous signals and generate rationale explanations [8], yet remain limited by fixed knowledge cut-offs and an inability to adapt to changing environments in real time [9]. They are also fragile to prompt design and may produce plausible but invalid outputs [10], [11]. Advances in prompt engineering and human-in-the-loop (HITL) methods partially mitigate these risks [12].

This paper addresses these limitations by proposing a hybrid architecture that integrates LLM strategic guidance with RL execution, and sets the following objectives.

a) Objectives: (1) Design an approach for using LLMs to generate trading strategies that are economically grounded, assessed by expert review; (2) evaluate whether LLM guidance improves RL performance, measured by SR and MDD, without altering the base RL architecture.

These objectives directly inform the main contributions of the paper.

b) Contributions: (1) A structured prompting framework that produces domain-grounded strategies assessed by expert review and financial metrics; (2) a modular LLM+RL design in which the LLM contributes a single uncertainty-weighted scalar appended to the RL observation space, improving out-of-sample performance without modifying the RL algorithm.

A. Related Work

A Trading DQN (TDQN) for stock trading is introduced in [3]. The authors leverage a DDQN algorithm to mitigate overestimation bias and stabilize learning in stochastic market environments. A key aspect of their RL approach was the discretization of actions and the enforcement of capital constraints, which helped prevent infeasible or overleveraged actions.

The FinRL framework [6] introduced benchmark environments and unified APIs for financial RL research that features realistic data simulation. It includes a wide array of backtests using standard RL algorithms and focuses on two primary objectives in algorithmic trading: maximizing return (measured by cumulative return and SR) and minimizing risk (measured by MDD and return variance). The framework supports experiments in single-stock trading, multi-stock trading, and portfolio allocation.

In the survey [13], the authors identified key limitations in deep reinforcement learning (DRL), including Bellman backup instability and credit assignment failures. The authors recommend hierarchical reinforcement learning (HRL) or recurrent extensions to address the lack of long-range temporal dependencies.

For LLMs in finance, [14] showed that ChatGPT outperforms traditional sentiment lexicons on forward-looking financial news, but lacks temporal awareness and numerical reasoning capabilities.

The FINMEM framework in [12] combines structured memory with LLM-based decision modules. FINMEM's layered memory integrates recent news, financial reports, and long-term statements to inform trade recommendations, leveraging retrieval-augmented generation (RAG). Their architecture stores experiences in a vector database, which are retrieved and ranked using a decay mechanism that emulates a human's memory decay.

Prompting practices have been extensively surveyed in [11], which categorizes strategies into instruction-based, example-based, reasoning-based, and critique-based families. The study highlights self-refinement and constraint enforcement as key mechanisms for improving robustness. It also shows that minor variations in prompt wording can systematically influence model behavior.

In [15], Chain-of-Thought (CoT) prompting was shown to significantly enhance LLM reasoning. Self-improvement frameworks iteratively refine rationale quality, while problem decomposition and model fine-tuning help address complex tasks. Without structured prompting techniques, however, LLMs continue to struggle with planning problems.

II. Methodology

This section outlines the methodology developed to evaluate the integration of LLMs into RL agents. The proposed hybrid framework mirrors the top-down decision-making structures common in financial institutions.

Two experiments were conducted to address the research objectives. All LLM strategies were validated through historical backtesting and expert review prior to their integration with RL agents.

A. Benchmark Environment

For our benchmark, we utilized the trading system introduced in [3]. This benchmark includes a clearly defined environment, consistent state and reward functions, and extensive empirical results.

We replicated the core experimental settings, including the asset universe, data preprocessing, and evaluation metrics. Our reproduction yielded comparable statistically significant SR and MDD metrics.

We note minor discrepancies that affect financial interpretability: their cumulative returns are arithmetically summed rather than geometrically compounded, and their SR assumes a zero risk-free rate. For consistency, we preserve these conventions throughout our experiments.

B. Experiment 1: LLM Trading Strategy Generation

This experiment addressed Objective 1 by introducing two LLM agents: the **Strategist Agent** and the **Analyst Agent**. The Strategist Agent generates global trading policies using a financial dataset. The Analyst Agent processes news and distills it into signals to inform the Strategist Agent. This experiment serves as the foundation for Experiment 2.

A strategy defines a directional action ($\mathrm{dir}(\pi^g)$, where 1 = `LONG` and 0 = `SHORT`) and an associated confidence score (μ_{conf}, from 1 to 3). Each strategy is accompanied by an explanation and a weighted set of features. Strategies are generated on a monthly basis using time-aligned data.

1) Data and Feature Engineering: To support strategy generation, the LLM agents consumed a multi-modal dataset spanning 2012–2020, aligning with the benchmark's dataset dates in [3]. The dataset includes traditional Open, High, Low, Close, and Volume (OHLCV) price data, which we augmented with four additional categories of financial signals: market data, fundamentals, technical analytics, and alternative data [16]. These collectively define the LLM's context.

Market data was sourced from Interactive Brokers[1] and iVolatility[2], including OHLCV time series, SPX and NDX index returns, the VIX index, and Options implied volatility (IV). Fundamental data, comprising firm-level financial ratios and macroeconomic indicators (e.g., GDP, PMI, interest rates), were retrieved via SEC-API[3] and the FRED API[4]. Analytics features were computed using TA-Lib[5], applying rolling windows to extract indicators. Alternative data consisting of news headlines were collected from Alpaca[6] and processed into explanatory factors using few-shot LLM prompting, following the LLMFactor framework from [10]. The data was aligned by timestamp.

2) LLM Model: We used OpenAI's GPT-4o Mini for its strong performance in financial reasoning and cost-efficiency [12], [14], [15]. The model supports a 128k token context window with a 16k maximum prompt size, enabling the use of detailed prompts with embedded context memory, reasoning chains, and previous reflection results.

3) Prompt Engineering Methodology: The objective was to construct a prompt that generalized across equities and regimes while remaining interpretable. We proceeded in three stages: (i) baseline specification, (ii) incorporation of expert exemplars with feature pruning, and (iii) iterative refinement comprising two distinct processes: a *Writer–Trainer* process (feature and instruction selection) and a *Writer–Judge* process (prompt quality and rationale critique) with regret minimization.

To manage computational cost, we did not tune on full eight-year history and instead randomly sampled non-overlapping one-year intervals from the dataset per instrument, repeated five times, and used these subsets for creating candidate prompts.

a) Baseline: We began with a minimal prompt that exposed only raw OHLCV data and a small set of technical indicators: Simple Moving Averages (SMA; 20/50/200 periods), Relative Strength Index (RSI), and Moving Av-

[1]https://www.interactivebrokers.com/api
[2]https://www.ivolatility.com/data-cloud-api/
[3]https://sec-api.io/
[4]https://fred.stlouisfed.org/docs/api/fred/
[5]https://ta-lib.org/
[6]https://alpaca.markets/docs/api-documentation/

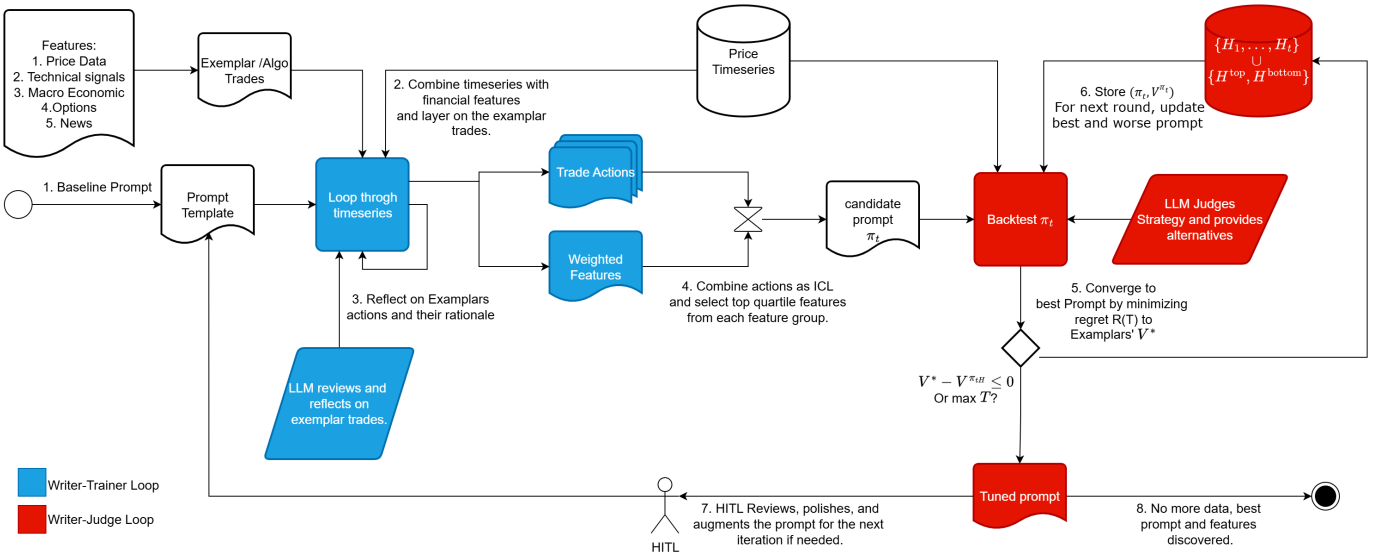

Fig. 1. Prompt Tuning Workflow.

erage Convergence Divergence (MACD). This configuration reflected common trading heuristics in both algorithmic and retail practice [17], [18] and served to set the candidate prompt and V^* target.

b) Writer–Trainer, and Writer–Judge Loops: To refine the information set, we introduced a *Writer–Trainer* process that reflected on expert trade exemplars derived from HITL feedback and a heuristic algorithm approximating these. Candidate features were selected by top quartile through ranking importance on a three-point Likert scale (low/medium/high), and rationales were clustered with only the ten unique ones being selected to become instructions for the candidate prompt.

Prompt refinement was then conducted through a heuristic regret–minimization loop, inspired by [19], with pruning and rationale discovery as the inner stage and backtesting as the outer stage. At each iteration t, a *writer* generated a candidate prompt π_t conditioned on the KB and the retained features and rationales. The candidate π_t was evaluated through backtesting to obtain its SR, denoted V^{π_t}. A *judge* then assessed the prompt for its rationale and suggested alternative instructions or feature combinations for the next iteration, and stored in the Knowledge Base (KB)

We adopted a regret heuristic to guide exploration:

$$\mathcal{R}(T) \;=\; \mathbb{E}\left[\sum_{t=1}^{T}\left(V^* - V^{\pi_t}\right)\,\middle|\,\mathcal{H}_t\right], \tag{1}$$

where V^* denotes the best SR defined by the baseline prompt or the market's SR for the whole dataset (initialized as $V^* = \max\{V_{\text{baseline}}, 0.8\}$), V^{π_t} is the SR of the current candidate, and $\mathcal{H}_t$ represents the KB at iteration t (features, rationales, and prior outcomes). To manage the LLM's finite context window, the loop retained only an *extremes memory* within $\mathcal{H}_t$, consisting of the best- and worst-performing prompts with their associated features and instructions. These extremes biased subsequent generations toward more promising candidates.

Iterations terminated when either (i) $\mathcal{R}(T) \leq 0$ (no expected improvement) or (ii) the maximum T iterations elapsed. The final augmented baseline prompt, generalized across equities and regimes through this process, was then subjected to three additional refinements.

c) Prompt Improvement 1 – In-Context Memory (ICM): Inspired by [12], we introduced a memory buffer that stores the most recent strategies π_t observed prior to time T. Each stored strategy π^g_{T-1} is represented by its directional action, weighted features, and rationale. Within the prompt, these prior strategies are recalled in-context and reflected on, enabling the current strategy π^g_T to be conditioned on past decisions. This reflection mechanism reduces the persistence of suboptimal strategies.

d) Prompt Improvement 2 – Instruction Decomposition: To enhance reasoning, instructions and their associated drivers were decomposed [11] into six feature groups: stock, technical, fundamental, macroeconomic, options, and prior strategy reflection [9]. Each group supplied few-shot examples and domain-specific heuristics to elicit CoT, requiring the model to reason sequentially across domains and prior strategies.

e) Prompt Improvement 3 – News Factors: Unstructured news data was introduced via the analyst agent, which applied instruction-decomposition factor extraction [10]. Entities and timestamps were anonymized, disabling the LLM's memory to prevent leakage [20]. Extracted news factors were ranked and integrated alongside numerical indicators.

The final system integrates selected numerical and textual signals into a global strategy policy π^g. All prompt iterations used in Experiment 1 are summarized in Table I.

4) Parameters and Evaluation: Prompt tuning used temperature 0.7 following prior work [12], [14], with frequency penalty 1.0 and presence penalty 0.25. These values discour-