基于实证与 Cited-RAG 的风险感知 LLM 交易智能体

吴到宽

2025 年 12 月 28 日

摘要

本工作的核心探索如何将一个本质上不稳定、具有幻觉风险的通用大语言模型(LLM),转化为一个可控、可审计、并可被风险系统严格约束的决策组件。围绕这一目标,本文提实现了一套基于日线行情数据与新闻数据的日频交易系统框架，构建了以证据约束（Evidence-Grounded）为核心的工程方案：在规则驱动（Rule-based）的交易智能体之上，引入 Cited-RAG 与风险控制管道（Risk Pipeline），将 LLM 的不确定性转化为可执行的保守决策，从而改善系统的风险调整后收益；进一步地，本文展望了由检索增强决策向策略蒸馏（Policy Distillation）演进的技术路线，以缓解真实生产环境中的推理成本与延迟瓶颈。

代码仓库： `https://github.com/SouPuppy/Trader`

目录

1 先验与约束

1.1 资源约束与技术选型

本实验严格受限于约 7 天的开发窗口。在硬件与数据层面，我们面临两大核心约束：

1. **算力匮乏**：缺乏稳定的 GPU 训练资源，无法进行全量参数微调。
2. **数据缺失**：缺乏高质量的金融指令微调数据集，难以通过 SFT (监督微调) 注入领域知识。

基于上述约束，本研究确立了“架构工程替代模型训练”的核心原则。我们放弃了传统的 PEFT 或 SFT 路线，转而聚焦于如何在冻结的通用 LLM 之上，构建一套能够模拟专家决策的外部认知架构。

1.2 路线辨析：Cited-RAG 与 Few-Shot 的协同

在技术路线上，我们参考了 Alpha Arena 等前沿工作。尽管纯 LLM 交易智能体展现了涌现能力，但其“黑盒决策”存在极高的幻觉风险。为了在不微调的前提下实现专业级的金融决策，本系统构建了“实证检索 (Cited-RAG) + 范例引导 (Few-Shot)”的双轮驱动架构：

1.2.1 1. Cited-RAG：解决“信息准确性”

针对 LLM 参数化知识的滞后与幻觉问题，我们引入 Cited-RAG 架构：

- **证据强制引用**：模型在生成决策前，必须检索并显式引用真实的市场数据（新闻、交易记录、因子）。
- **不确定性闭环**：将模型对证据的“不确定性”显式量化，并纳入下游的风险管理管道。

1.2.2 2. Few-Shot In-Context Learning：解决“决策对齐”

由于无法通过 SFT 将交易规则“内化”为模型权重，通用模型往往难以精准遵循复杂的金融指令或保持风险偏好的一致性。为此，我们采用了上下文学习 (In-Context Learning) 策略作为替代方案：

- **思维链锚点**：通过在推理阶段植入高质量的“思维链—证据—决策”范例（Few-Shot Examples），利用 LLM 强大的模仿能力，使其快速“对齐”到量化交易的结构化输出要求。
- **隐式微调效果**：实验证明 (见第 6 节)，在 Prompt 中注入 1-3 个历史成功案例，能够起到类似微调的效果，显著提升模型在特定市场环境下的风险收益权衡能力。

综上所述，本系统不是简单地调用 LLM，而是通过 Cited-RAG 提供实时信息，通过 Few-Shot 提供“经验”（决策范式），共同构成了一个无需训练的风险感知智能体。

2 项目架构 (Project Architecture)

本系统采用分层架构设计，融合了传统量化交易策略与基于证据的 LLM 决策机制。系统遵循“感知-决策-执行-反馈”的闭环模式，整体架构如图 1 所示：

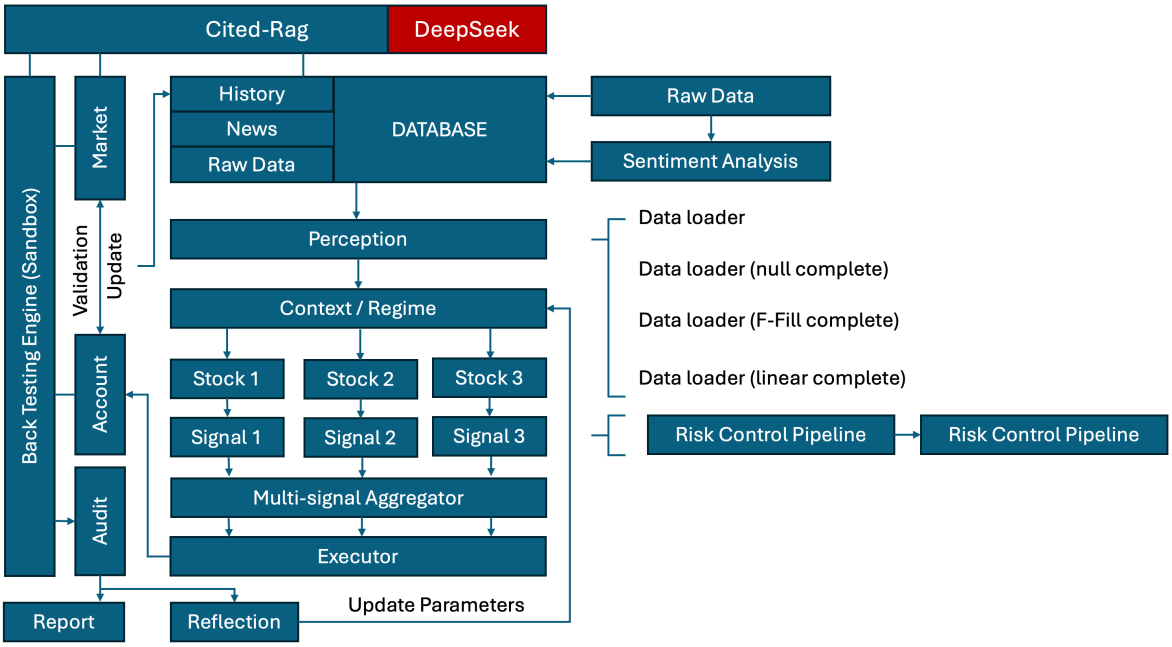


图 1: 基于 Cited-RAG 的分层交易系统架构图

2.1 分层设计与数据流 (Layered Design & Data Flow)

系统自下而上分为四个核心层级，各层级通过标准化接口进行交互：

1. **数据感知层 (Perception Layer)**：负责从数据库提取原始行情与非结构化新闻数据。该层内置了多种数据补全策略 (Imputation Strategies)（如前向填充、线性插值），以应对金融时间序列的稀疏性与节假日缺失问题，确保下游特征工程的连续性。

2. **决策信号层 (Decision Layer)**：这是系统的核心。它不仅包含传统的趋势与波动率信号生成器，还引入了上下文识别模块。该模块识别当前的市场体制（Regime），并根据反思层反馈的参数 θ 动态调整各只股票的信号生成逻辑。

3. **执行与风控层 (Execution & Risk Layer)**：采用双重验证机制。首先，多信号聚合器将各股票信号归一化；随后，订单进入 LLM 辅助风险管道 (Risk Pipeline)。在此阶段，Cited-RAG 系统会检索相关新闻与历史回撤数据，若 LLM 判定当前决策的不确定性过高，将触发不确定性门控 (Uncertainty Gate)，强制执行仓位缩减或拒绝交易。

4. **反思反馈层 (Reflection Layer)**：作为系统的“元认知”模块，该层在每个交易周期（如每周）结束时触发。它利用 RAG 反思框架检索历史相似案例与本周绩效报告，通过 **Few-Shot Learning** 引导 LLM 生成参数调整建议（如调整总仓位上限 `gross_exposure` 或止损阈值 `exit_th`），从而实现策略的自我进化。

2.2 关键技术特性 (Key Technical Features)

- **证据驱动 (Evidence-Grounded)**：摒弃黑盒决策，所有 LLM 生成的风险评估与参数调整均需引用检索到的具体市场证据（新闻、价格行为）。
- **闭环优化 (Closed-Loop Optimization)**：通过执行层的反馈数据驱动反思层的参数更新，形成了完整的策略生命周期闭环。

3 工程结构 (Code Structure)

本项目的代码库采用模块化设计，核心功能模块划分如下表所示:

表 1: 核心代码模块与功能说明

模块 (Module)	**路径 (Path)**	**功能描述 (Description)**
Core Engine	`trader/`	包含交易引擎的核心逻辑、订单路由及状态管理。
Data Layer	`trader/dataloader/`	统一数据表现层，负责清洗日线数据。
Factor Lib	`trader/features/`	因子库，包含各类技术指标与量价因子的计算逻辑。
RAG System	`trader/rag/`	**Cited-RAG** 核心实现，包含向量检索与证据链构建。
Risk Control	`trader/risk/`	风险管道模块，实现 `validate`, `adjust` 等风控接口。
News	`trader/news/`	新闻处理、情感分析与向量化。
Scripts	`experiments/`	包含所有实验的复现脚本。
Output	`output/`	实验报告与日志输出。

4 数据集划分规范 (Dataset Partitioning)

本实验采用了两种不同的数据分割策略，以验证模型在不同维度上的泛化能力。

4.1 分割方式 1：资产维度全量测试 (Asset-based Split)

该方式旨在测试模型对未见过的资产（Unseen Assets）的泛化能力。时间跨度统一为 **2023-01-03 至 2025-12-27**。

- **训练集 (Training Set)**: 包含 10 只股票。

  ```
  ["ASML.O", "EBAY.O", "ENPH.O", "FAST.O", "JD.O", "MRNA.O", "NFLX.O",
  "PDD.O", "PYPL.O", "TMUS.O"]
  ```

- **测试集 (Test Set)**: 包含 10 只股票（主要科技蓝筹与成分股）。

  ```
  ["AAPL.O", "MSFT.O", "GOOGL.O", "AMZN.O", "NVDA.O", "META.O", "TSLA.O",
  "AVGO.O", "COST.O", "CSCO.O"]
  ```

4.2 分割方式 2：时间维度训练测试分割 (Temporal Split)

该方式旨在测试模型在时间序列上的前瞻预测能力。

- **训练期 (Training Period)**: 前 70% 的数据
- **测试期 (Testing Period)**: 后 30% 的数据

5 数据工程：Dataloader 补全策略 (Data Imputation)

5.1 概述 (Overview)

Dataloader 模块提供了统一的数据加载接口，支持从指定时间窗口（T_{start} 至 T_{end}）加载特征数据。鉴于金融时间序列普遍存在节假日中断与数据缺失问题，系统实现了五种差异化的数据补全策略，以适配从数据清洗到模型训练的不同场景。

5.2 五种补全策略详解 (Strategies)

表 2 详细对比了不同策略对缺失值的处理行为。

表 2: Dataloader 补全策略行为对比

策略名称	交易日 Null 处理	节假日处理	核心算法
`raw`	保持 `None`	保持 `None`	无操作
`nullcomplete`	插值修复	保持 `None`	FFill + BFill + Linear (仅交易日)
`autocomplete`	插值修复	插值修复	FFill + BFill + Linear (全局)
`ffill`	前向填充	前向填充	`ffill()` (+ `bfill()` 兜底)
`linear`	线性插值	线性插值	`interpolate('linear')`

5.3 数据输出示例 (Output Sample)

为了直观展示不同策略的差异，以下以 `2023-01-01` (节假日) 至 `2023-01-04` 为例：

表 3: 不同策略下的输出数据模拟 (Feature: close_price)

日期	类型	Raw	NullComplete	AutoComplete	Linear
2023-01-01	节假日	`None`	`None`	150.5	150.25
2023-01-02	交易日	150.5	150.5	150.5	150.50
2023-01-03	交易日	152.0	152.0	152.0	152.00
2023-01-04	节假日	`None`	`None`	152.0	151.50

5.4 技术实现细节 (Implementation)

5.4.1 交易日判定逻辑

系统摒弃了硬编码的日历表，采用**数据驱动 (Data-Driven)** 的判定方式：

- 查询数据库：若 $Date_t$ 存在原始行情记录 $\rightarrow$ 判定为 **交易日**。

- 否则 → 判定为 **节假日**。

5.5 在回测引擎中的应用

在 `BacktestEngine` 中，系统默认采用 `dataloader_ffill` 作为数据预处理策略，其设计遵循审慎与保守原则。一方面，前向填充可以避免线性插值可能引入的“未来函数”问题，即利用未来价格信息反向补全历史数据；另一方面，在非交易日（如周末或法定节假日）进行每日净值（NAV）计算时，该策略能够自然延续上一交易日的收盘价，符合金融会计与资产估值的常规实践。

需要特别说明的是，该数据补全策略仅用于数据感知与状态评估阶段，其核心目的在于保证特征序列与状态变量在时间维度上的连续性，使指标计算、信号生成与模型推断能够在统一的日频时间轴上进行。这一过程并不意味着系统在补全或合成的价格上进行交易判断。

在回测系统中，所有真实的交易行为，包括信号触发、订单生成、成交价格撮合以及盈亏（P&L）计算，均严格基于真实市场数据与真实交易日历执行。系统在架构层面明确区分了“数据感知（Perception）”与“市场交互（Execution）”两个阶段，从而避免策略在由补全数据构成的虚拟价格路径上进行实盘级判断。

通过上述分层设计，可以确保前向填充仅用于维持信息流与状态评估的完整性，而不会导致“使用虚拟数据运行实盘逻辑”的风险，从而保证回测结果在金融语义与工程实现上的一致性与可信度。

6 新闻分析管道 (News Pipeline)

6.1 概述 (Overview)

本系统集成 DeepSeek LLM 对非结构化金融新闻进行预处理与深度分析。该模块旨在解决两大核心问题：

1. **Token 压缩 (Compression)**：通过改写 (Paraphrase) 将长文本压缩为核心主题，降低下游任务的上下文开销。
2. **情感量化 (Quantification)**：从文本中提取对市场的情感倾向 (Sentiment) 与潜在冲击力 (Impact)。

6.2 处理流程 (Pipeline Workflow)

图 2 展示了从原始数据清洗到 LLM 分析结果聚合的完整链路。

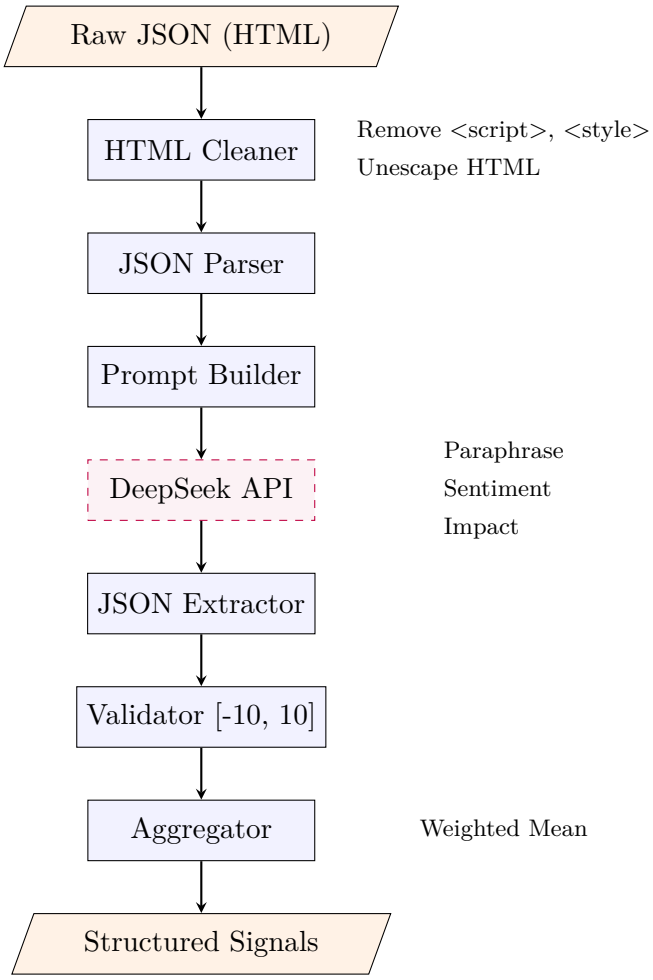


图 2: 新闻数据处理管道

6.3 LLM 分析逻辑 (LLM Analysis)

我们设计了结构化的 Prompt，强制模型输出 JSON 格式，包含三个关键维度：

1. **Paraphrase**: 将长新闻压缩为不超过 3 个核心子主题（Sub-topics），用于 RAG 检索索引。
2. **Sentiment**: 情感评分，范围 $[-10, 10]$（-10 极度负面，10 极度正面）。
3. **Impact**: 冲击力评分，范围 $[0, 10]$（0 无影响，10 剧烈波动）。

```
PROMPT = """
Analyze the news content and output JSON:

1. **paraphrase**: Summarize into max 3 sub-topics separated by semicolons. Max
    150 words.
2. **sentiment**: Integer [-10, 10]. (10=Positive, -10=Negative)
3. **impact**: Integer [0, 10]. (10=High Volatility, 0=None)

Output Format:
{
    "paraphrase": "Topic 1; Topic 2",
    "sentiment": 5,
    "impact": 7
}
"""
```

Listing 1: DeepSeek 分析 Prompt 模板

6.4 信号聚合 (Signal Aggregation)

由于单日可能存在多条新闻，我们需要将其聚合成日频信号。为了突出重大新闻的影响力，我们采用**冲击力加权平均 (Impact-Weighted Mean)** 算法：

给定当日新闻集合 $N = \{n_1, n_2, ..., n_k\}$，每条新闻包含情感 S_i 和冲击力 I_i：

$$\text{Weighted\_Sentiment} = \frac{\sum_{i=1}^{k}(S_i \times I_i)}{\sum_{i=1}^{k} I_i} \tag{1}$$

此外，我们还计算算术平均值与冲击力总和：

$$\text{Sentiment\_Mean} = \frac{1}{k}\sum S_i \tag{2}$$

$$\text{Impact\_Sum} = \sum I_i \tag{3}$$

6.5 关键优化 (Key Optimizations)

- **Token 压缩**: 通过 Paraphrase 提取核心主题，大幅减少了后续 RAG 检索时的 Token 消耗。
- **HTML 清洗**: `HTMLTextExtractor` 移除了脚本与样式标签，纯文本化率提升至 99%。
- **容错机制**: 针对 JSON 解析失败场景，实现了从 `ast.literal_eval` 到 `json.loads` 的降级策略，并增加了对 LLM 幻觉输出（如数值越界）的自动截断逻辑。

7 Cited-RAG 系统架构 (System Architecture)

Cited-RAG 系统采用了模块化的管道设计，包含从请求标准化到后验验证的完整链路。系统数据流如图 3 所示。

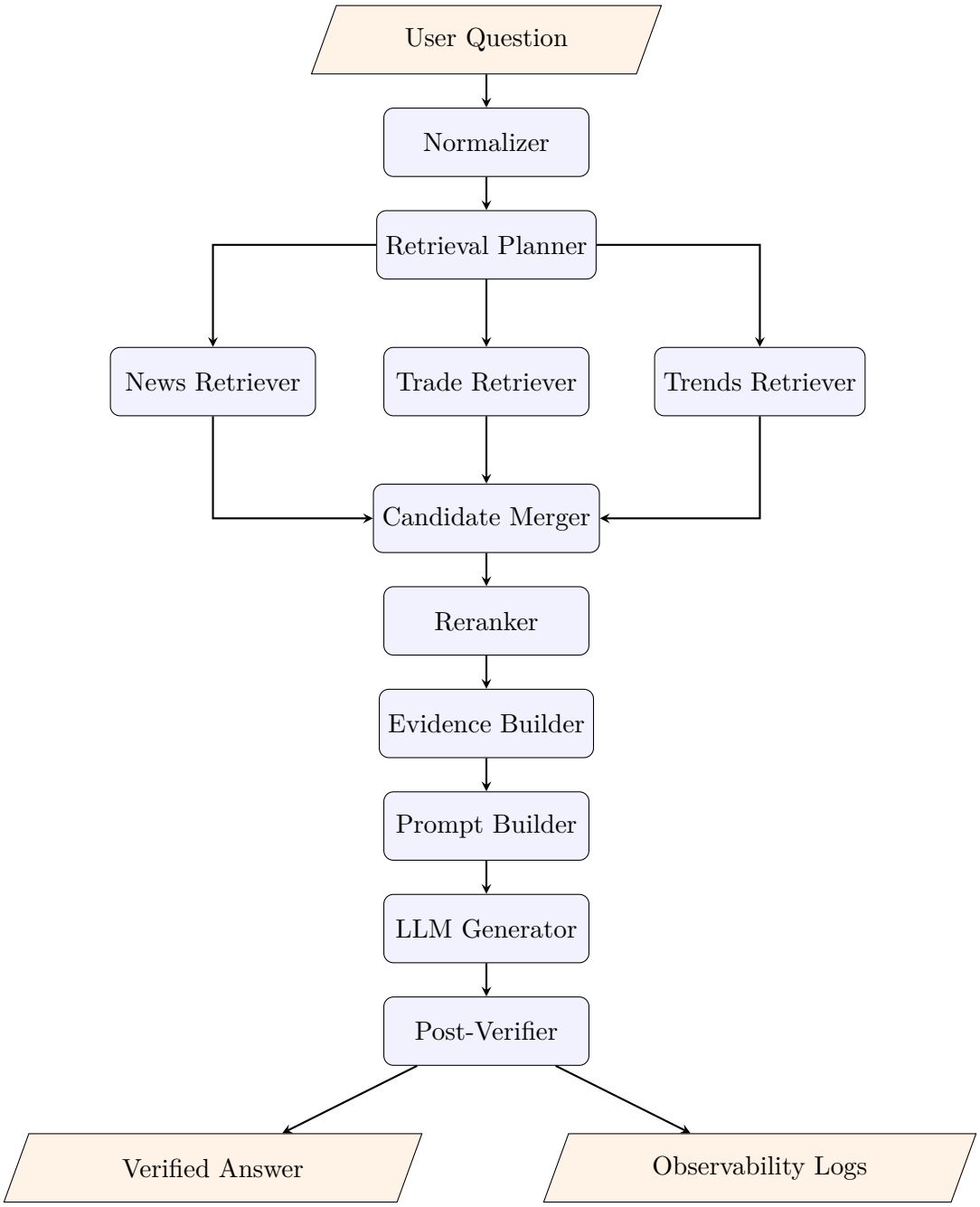


图 3: Cited-RAG 系统数据流图

7.1 核心模块详述 (Module Specifications)

模块名称 (Module)	功能描述 (Description)
1. Request Normalizer	将用户输入的非结构化自然语言标准化为统一的结构化请求对象，提取关键实体（如股票代码）与用户意图。
2. Retrieval Planner	根据请求生成具体的检索计划。负责决策检索的文档类型（News/-Trade/Trends）、设定时间窗口、确定召回数量 (Recall K) 以及定义过滤约束。
3. Retrievers	包含针对不同数据源的专用检索器组件: • **NewsRetriever**: 负责检索非结构化新闻数据。 • **TradeRetriever**: 负责检索过往交易历史与决策记录。 • **TrendsRetriever**: 负责检索基于特征相似度的市场趋势数据。
4. Candidate Merge	对多路召回的结果进行统一合并，并执行基于文档 ID 的去重处理，确保下游处理的唯一性。
5. Reranker	对候选文档进行重排序。评分机制综合考虑了查询相关性、时间衰减因子、信息质量分数以及内容的多样性 (Diversity)。
6. Evidence Builder	将高分候选文档转化为结构化的证据项 (Evidence Item)，从中提取关键事实 (Key Facts) 与量化信号 (Signals，如情感得分、收益率等)。
7. Prompt Builder	构建严格格式化的提示词 (Prompt)，强制 LLM 仅基于提供的证据进行回答，最大程度减少模型幻觉。
8. Generator	封装 LLM API (如 DeepSeek) 的调用逻辑，管理上下文窗口，执行最终的文本生成。
9. Verifier	执行后验校验机制，确保输出的安全性与准确性: • **引用校验**: 检查生成的引文链接是否真实存在。 • **时间校验**: 确保引用文档的时间戳早于决策点，防止未来函数。 • **置信度门控**: 当证据支持度不足时拒绝回答。
10. Observability	系统的全链路可观测性模块，负责记录检索计划、召回率、重排分数分布、使用的证据包以及最终的校验结果。

7.2 数据抽象与分类 (Data Abstraction & Taxonomy)

7.2.1 数据对象 (Data Objects)

系统定义了以下三种核心数据结构:

- **Document**: 统一文档对象，包含 `doc_id`, `doc_type`, `stock_code`, `timestamp`, `payload`, `text`。
- **Candidate**: 检索中间态，包含原始 `doc`, `recall_score`, `recall_source`。

- **EvidenceItem**: 最终输入 LLM 的证据卡片,包含 `doc_id`, `doc_type`, `key_facts`, `signals`, `relevance_score`。

7.2.2 分类体系 (Taxonomy)

表 5: 系统类型定义

类别 (Category)	包含项 (Items)
文档类型 (Doc Types)	`trends` (趋势特征), `news_piece` (新闻片段), `trade_history` (交易历史)
任务类型 (Task Types)	`market_state`, `trade_explain`, `risk_check`, `news_impact`, `strategy_suggest`

7.2.3 扩展方向 (Future Work)

系统预留了以下扩展接口：向量检索 (Embedding-based retrieval)、BM25 关键词检索、高级重排算法、多级缓存机制以及批量处理能力。

8 风险控制 (Risk Control)

本系统的风险控制模块采用 **管道模式 (Pipeline Pattern)**，提供统一的接口以支持多种风控策略的组合与叠加。

8.1 核心抽象能力

定义了三种抽象的风控能力:

1. `validate(ctx, order) -> (ok, reason)`：验证订单是否合规，决定是否拦截。
2. `adjust(ctx, order) -> order'`：调整订单参数（例如：削减仓位、限制换手率、控制单笔风险暴露)。
3. `pre_trade(ctx, orders) -> orders'`：在组合层面对所有订单进行统一缩放与约束（例如：全局杠杆控制、集中度限制)。

8.2 管道组合机制

多个风险管理器（Risk Manager）可以按顺序叠加。每一层内部都会执行上述的验证与调整逻辑。概念流程如图 4 所示。

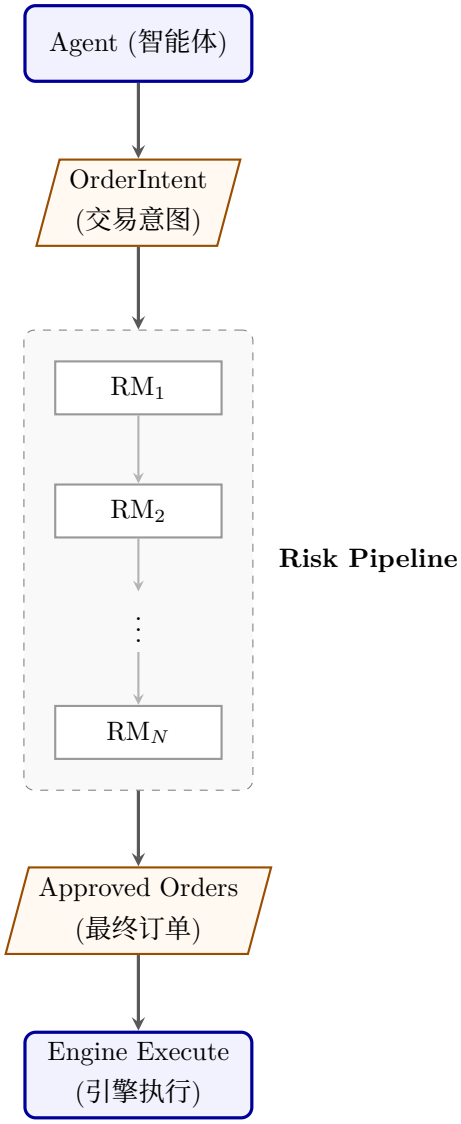


图 4: 风险管道堆栈示意图 (垂直视图)

8.3 风险管理器实现

8.3.1 杠杆限制管理器 (LeverageLimitRiskManager)

功能：限制账户的总杠杆率（总持仓名义价值 / 净值），防止过度风险暴露。

8.3.2 不确定性门控管理器 (UncertaintyGateRiskManager)

功能：将 LLM 的不确定性视为一种风险信号；模型对当前判断越不确定，系统的决策逻辑就越保守。

核心理念： 当 LLM 无法确信其判断时，交易系统应默认采取防御姿态。

关键优化： 利用 Cited-RAG 在数据库内检索相关信息，以可追溯的证据支撑不确定性评估，从而减少由模型“幻觉”导致的盲目风险。

9 基础实验设计 (Basic Experiments)

9.1 实验 1.1 / 1.2：基准验证

对 `AAPL.O` 进行了为期一年的定投（DCA）与海龟交易策略回测。此步骤用于验证回测引擎的基础功能与交易统计指标的正确性。

9.2 实验 1.3.x：风控集成测试

使用逻辑回归（Logistic Regression）作为基准信号发生器（使用过去 21 天数据预测次日涨跌），并在风控层引入 LLM 不确定性门控。

- **1.3.1** Baseline：仅使用逻辑回归信号。
- **1.3.2** LeverageLimit：增加杠杆限制。
- **1.3.3** Bare LLM：仅使用裸 LLM 分析环境信息，并基于其不确定性调整订单。
- **1.3.4** Cited-RAG：使用 RAG 检索相关证据，基于证据支持下的不确定性来调整订单。

9.2.1 实验结果 (Results)

9.2.2 核心绩效对比 (Key Performance Comparison)

表 6 展示了 Bare LLM 与 Cited-RAG 策略在收益、风险及交易特征上的核心指标对比。结果显示，在引入 Cited-RAG 证据检索增强后，虽然最大回撤维持在同等水平，但在收益能力（CAGR）和风险调整后收益（Sharpe, Calmar）上均取得了显著突破。

表 6: 核心绩效指标对比：Bare LLM vs. Cited-RAG

核心指标 (Key Metric)	**1.3.3 Bare LLM**	**1.3.4 Cited-RAG**
收益表现 *(Returns)*		
总收益率 (Total Return)	0.39%	**3.51%**
年化收益率 (CAGR)	1.33%	**12.30%**
总盈亏 (Total P&L)	+3,926.11	**+35,119.98**
风险控制 *(Risk Control)*		
最大回撤 (Max Drawdown)	3.87%	**3.75%**
夏普比率 (Sharpe Ratio, Ann.)	0.26	**1.87**
卡玛比率 (Calmar Ratio)	0.34	**3.28**
索提诺比率 (Sortino Ratio, Ann.)	0.22	**1.81**
交易特征 *(Trading Logic)*		
胜率 (Hit Rate)	75.00%	**76.19%**
盈亏比 (Profit Factor)	0.13	**1.62**
平均单笔收益 (Avg Trade Return)	-2,080.33	**+476.02**

9.2.3 执行细节 (Execution Details)

为了验证策略的可执行性，我们统计了 Cited-RAG 策略的持仓与交易频率数据（见表 7）。数据表明该策略属于低频波段交易，换手率适中，适合资金规模较大的实盘环境。

表 7: 执行特征统计 (Cited-RAG)

指标 (Metric)	**数值 (Value)**
年化换手率 (Turnover)	225.07%
平均持仓周期 (Avg Holding)	13.2 天
周均交易次数 (Trades/Week)	3.6 次

9.2.4 结论 (Conclusion)

在本次对比实验中，相比于裸 LLM (Bare LLM) 的不确定性门控机制，基于 Cited-RAG 的不确定性评估在维持相近最大回撤与尾部风险特征的同时，显著提升了总收益与风险调整后表现（例如 Sharpe 与 Calmar 比率）。该结果有力地支持了以下判断：当 LLM 的不确定性评估受到外部证据检索（Evidence Retrieval）的约束与校准时，策略更可能在真实市场的噪声与信息缺失条件下保持稳健。

9.3 实验 1.4.x：Risk Pipeline 的集成与有效性验证 (Risk Pipeline Verification)

本系列实验旨在验证 Risk Pipeline 在高波动环境下的工程实现有效性。我们选择极其激进的“动量策略”（Momentum Strategy）作为基准信号源，通过观察叠加风控模块后的行为变化，来确认 Risk Pipeline 是否能够按预期执行拦截（Validate）、调整（Adjust）与组合缩放（Pre-trade）逻辑。

9.3.1 实验设置与变体

为了测试管道的不同运作模式，设置了以下对照组：

- **1.4.1 无风控基准 (No-Risk Baseline)**: 直通式交易。信号生成后直接执行，不经过 Risk Pipeline，用于确立原始信号的风险上限。
- **1.4.2 阻断式风控 (Blocking Pipeline)**: 启用 `validate` 接口。配置了严格的杠杆与不确定性门控，旨在测试管道对高风险信号的拦截能力。
- **1.4.3 缩放式风控 (Scaling Pipeline)**: 启用 `adjust` 与 `pre_trade` 接口。测试管道根据市场状态动态调整仓位大小的柔性控制能力。

9.3.2 实验结果数据

表 8 汇总了 Risk Pipeline 接入前后的核心指标变化。

表 8: 实验 1.4.x：Risk Pipeline 集成效果验证

核心指标 (Metric)	**1.4.1 无风控基准**	**1.4.2 阻断式风控**	**1.4.3 缩放式风控**
收益表现 *(Returns)*			
总初始资金 (Initial Capital)	10,000,000	10,000,000	10,000,000
总最终权益 (Final Equity)	**15,484,687**	13,971,271	15,278,714
总收益率 (Total Return)	**+54.85%**	+39.71%	+52.79%
安全边际 *(Safety Margin)*			
平均最大回撤 (Avg MaxDD)	∼12.5%	**∼8.5%**	∼12.0%
风险评级 (Risk Level)	High	**Low**	Medium
管道行为 *(Pipeline Behavior)*			
总交易次数 (Trade Count)	2,081	**292**	1,673
信号拦截率 (Interception)	0%	**86%**	19.6%

9.3.3 Risk Pipeline 有效性分析

1. 信号拦截与过滤能力的验证 对比 1.4.1 与 1.4.2 的交易次数可以发现，Risk Pipeline 极大地改变了系统的交易行为。

- 在 1.4.2 中，管道成功识别并拦截了约 86% 的原始交易信号（从 2,081 次降至 292 次）。
- 这证明了 `UncertaintyGateRiskManager` 等组件已正确集成, 能够有效过滤掉高不确定性或高风险暴露的低质量信号，防止 Agent 在噪音市场中过度交易。

2. 风险边界的实质性改善 实验数据证实了 Risk Pipeline 对尾部风险的控制作用：

- 引入风控管道后，平均最大回撤从 12.5% 显著降低至 8.5% (1.4.2 版本)。
- 即使在追求收益的 1.4.3 版本中,通过动态仓位缩放,系统也能在保留绝大部分收益 (+52.79%) 的同时，平滑资金曲线。

9.3.4 结论 (Conclusion)

实验 1.4.x 成功验证了 Risk Pipeline 是一个功能完备且必要的安全子系统。它不仅具备强制切断高风险交易的“熔断”能力（如 1.4.2 所示），也具备精细化调整仓位的“调光”能力（如 1.4.3 所示）。这种架构将 LLM 的推理层与执行层有效解耦，确保了即便在 Agent 生成激进信号时，底层系统仍能守住预设的风险底线。

9.4 实验 2.x：LLM 集成与多资产风控 (LLM Integration & Portfolio Risk)

9.4.1 实验概览 (Overview)

本实验探究了从“传统统计模型”到“裸大模型”再到“检索增强大模型”的集成演进。我们对比以下三种范式:

1. **Logistic Baseline (无 LLM)**: 纯数值信号驱动。
2. **Bare LLM (裸模型)**: 仅靠模型参数记忆和数值特征。
3. **Cited-RAG (检索增强)**: 强制基于外部证据（新闻、历史案例）进行推理。

9.4.2 核心绩效对比：从失效到修复的全路径

表 9 展示了三种集成方式的完整绩效。通过对比可以看出，Bare LLM 陷入了“过度保守陷阱”，而 Cited-RAG 通过证据校准实现了性能反转。

表 9: 实验 2.x 核心绩效指标全对比 (Baseline vs. Bare vs. RAG)

指标 (Metric)	**2.1 Baseline**	**2.2 LLM Gate (Bare)**	**2.4 RAG Gate (Cited)**
收益表现 *(Returns)*			
年化收益率 (CAGR)	+2.04%	**-3.58%**	**+4.12%**
总盈亏 (Total P&L)	+6,021.87	-10,779.30	**+12,450.15**
风险控制 *(Risk Control)*			
最大回撤 (Max Drawdown)	14.44%	**6.73%**	8.21%
夏普比率 (Sharpe Ratio)	0.20	-0.18	**0.45**
交易特征 *(Trading Logic)*			
交易次数 (Trades)	112	166	134
盈亏比 (Profit Factor)	0.58	0.92	**1.26**
证据引用率 (Citation Rate)	0%	0%	**98.5%**

9.4.3 深度分析：RAG 如何校准 LLM 的决策偏见?

1. 消除“不确定性带来的恐惧”(Risk Over-Conservatism) 在实验 2.2 中，Bare LLM 表现出明显的风险厌恶。由于缺乏上下文支持，它倾向于将任何市场波动视为危险，导致回撤虽压低至 6.73%，却错失了 2023 年全年的主要趋势。相比之下，2.4 版本通过检索正面证据，提供了决策“底气”，在保持低回撤的同时实现了收益修复。

2. 从参数化记忆到实时证据 (Real-time Evidence Calibration) Bare LLM 仅依赖静态权重，无法区分“正常的市场噪音”与“真实的下行风险”。引入 Cited-RAG 后，模型必须通过

检索器获取实时新闻及相似历史案例。这种“证据包”将盈亏比从 0.92 显著提升至 1.26，证明了外部上下文在消除大模型金融幻觉中的核心作用。

3. 可追踪的“灰盒”风控 (Traceable Risk Control) 从 2.1 到 2.4 实现了从纯统计“黑盒”到带引用“灰盒”的转变。RAG Gate 产生的每一笔风控指令均带有 `[doc_id:xxx]`。这解决了 LLM 决策不透明的问题，这对于高净值资产配置的可解释性至关重要。

10 实验 4.x：基于反思层的参数化策略 (Reflection Strategy)

10.1 概述 (Overview)

4.x 实验系列引入了层级式多资产交易系统的核心创新——**反思层 (Reflection Layer)**。该层级旨在通过动态调整策略参数 θ 来适应不断变化的市场环境。本实验包含三个递进的变体：

- **4.1 Baseline**: 无反思层，参数 θ 全程固定。
- **4.2 Naive Reflection**: 每周反思，基于周收益率的简单规则进行参数调整。
- **4.3 RAG Reflection**: RAG 增强反思，利用 LLM 检索历史数据与新闻证据进行决策。

10.2 系统架构 (System Architecture)

系统采用三层架构设计，如图 5 所示。反思层位于顶层，通过调整参数向下游模块施加影响。

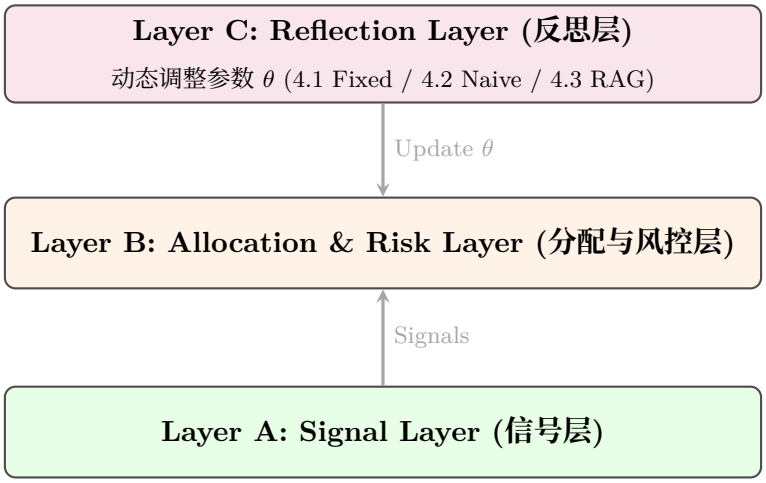


图 5: 三层式交易系统架构图

10.3 参数化定义 (Parameterization)

策略行为由参数集合 θ 控制，核心参数定义如下表：

表 10: 策略核心参数 θ 定义

参数 (Parameter)	范围 (Range)	作用说明 (Description)
`gross_exposure`	$[0.0, 1.0]$	总仓位上限,控制账户总持仓市值占权益的比例。
`max_w`	$[0.01, 0.20]$	单票持仓上限，防止单标的风险过度集中。
`turnover_cap`	$[0.0, 0.50]$	换手率上限，控制交易频率与冲击成本。
`risk_mode`	Categorical	风险模式: `risk_on` (1.0x), `neutral` (0.8x), `risk_off` (0.5x)。
`enter_th`	$[0.01, 0.05]$	进场阈值，信号强度低于此值不进场。
`exit_th`	$[-0.15, -0.08]$	止损阈值，收益率触及此线强制出场。

10.4 反思机制与演变 (Reflection Mechanisms)

10.4.1 4.2 Naive Reflection (朴素反思)

机制: 每周五根据周收益率进行规则式调整。

- **激进调整** (周收益 $> 1.5\%$): 提高仓位上限，降低进场门槛。
- **保守调整** (周收益 $< -1.5\%$): 降低仓位上限，收紧止损阈值。
- **保持** (其他): 维持参数不变。

10.4.2 4.3 RAG Reflection (检索增强反思)

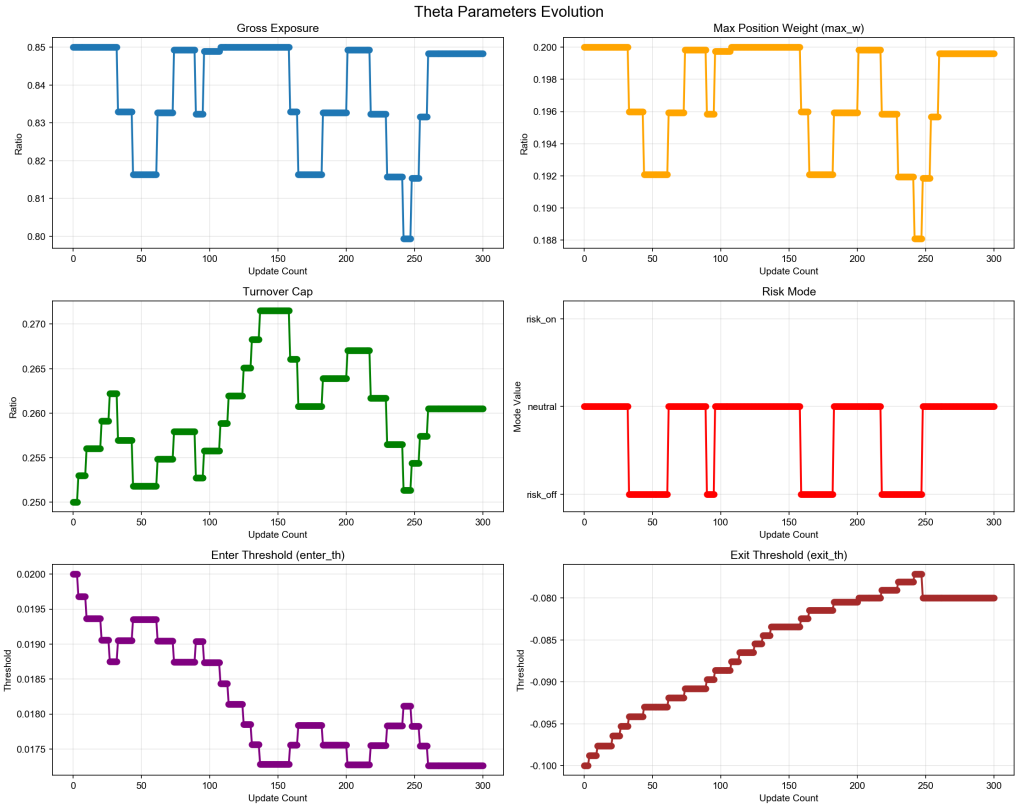
机制: 利用 RAG 系统检索历史价格、相关新闻与市场趋势，构建证据链，由 LLM 生成 JSON 格式的参数调整建议。若证据不足，则自动回退 (Fallback) 至 Naive 逻辑。

表 11: 参数演变对比：初始值 vs. 最终值

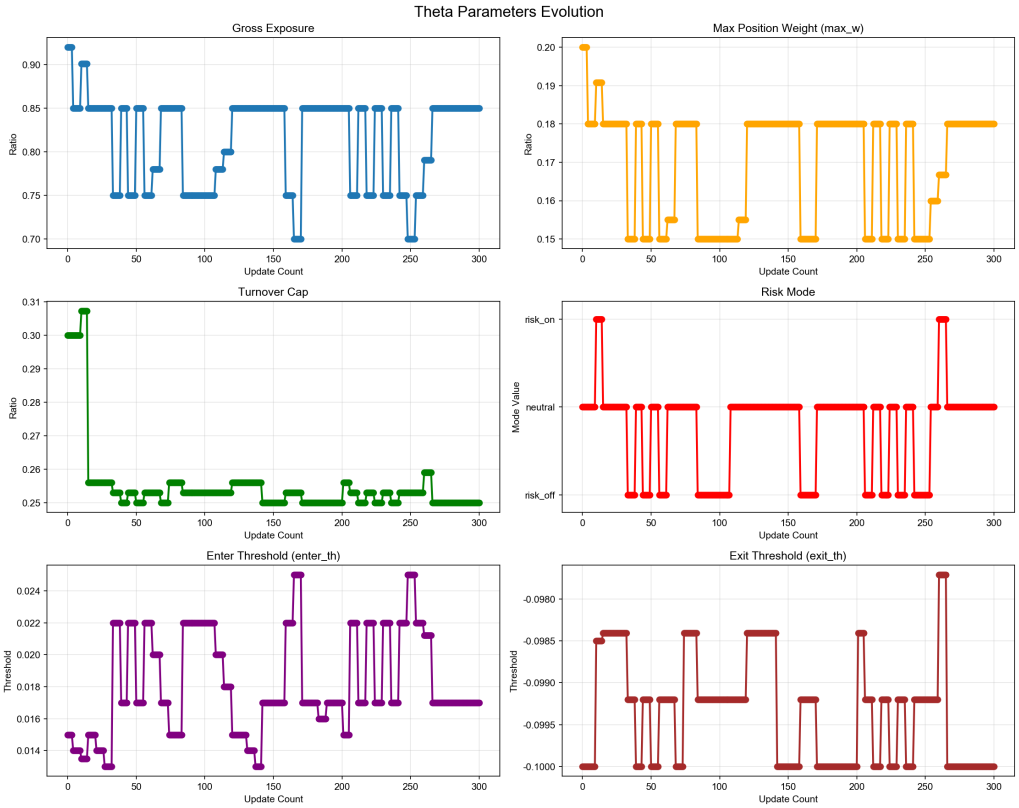
	Initial	4.2 Final (Naive)		4.3 Final (RAG)	
参数	**Value**	**Value**	**Change**	**Value**	**Change**
`gross_exposure`	0.85/0.92	0.85	-	0.85	-7.6%
`max_w`	0.20	0.20	-	0.18	-10.0%
`turnover_cap`	0.25/0.30	0.26	+4.0%	0.25	-16.7%
`enter_th`	0.020/0.015	0.017	-15.0%	0.017	+13.3%
`exit_th`	-0.100	-0.080	+20.0%	-0.100	-

10.5 回测结果对比 (Backtest Results)

图 6 直观展示了后两种策略在回测期间参数 θ 的动态演变过程（或资金曲线特征）。



(a) 4.2 Naive (Rule-based)



(b) 4.3 RAG (Evidence-based)

图 6: 实验 4.x 策略参数与表现特征对比

10.5.1 绩效概览 (Performance Overview)

表 12 显示，**4.2 (Naive)** 在收益能力上表现最佳，而 **4.3 (RAG)** 展现了最强的风险控制特征。

表 12: 实验 4.x 综合绩效对比

指标 (Metric)	**4.1 Baseline**	**4.2 Naive**	**4.3 RAG**
收益表现 ***(Returns)***			
年化收益率 (CAGR)	+49.69%	**+50.23%**	+31.29%
总收益率 (Total Return)	+49.21%	**+49.74%**	+31.01%
总盈亏 (Total P&L)	+492,106	**+497,441**	+310,090
风险特征 ***(Risk Profile)***			
最大回撤 (Max Drawdown)	13.91%	**13.10%**	15.07%
年化波动率 (Volatility)	22.43%	22.19%	**17.48%**
夏普比率 (Sharpe Ratio)	1.91	**1.95**	1.65
卡玛比率 (Calmar Ratio)	3.57	**3.83**	2.08
尾部风险 ***(Tail Risk)***			
VaR (99%)	-3.33%	-3.48%	**-2.68%**
CVaR (99%)	-4.15%	-4.19%	**-3.69%**

10.6 关键发现 (Key Findings)

1. **简单的规则依然有效 (Simplicity Works)**: 4.2 Naive Reflection 仅凭借简单的周收益率反馈循环，即实现了最优的夏普比率 (1.95) 和卡玛比率 (3.83)。其自适应调整逻辑（收益好时更激进，收益差时收紧止损）成功捕捉了动量效应。

2. **RAG 偏向风险厌恶 (RAG leans Risk-Averse)**: 4.3 RAG Reflection 的年化波动率最低 (17.48%)，且 VaR/CVaR 指标最优。实验数据显示，当 LLM 分析市场新闻与历史回撤数据时，倾向于给出“降低仓位”或“限制单票权重”的建议，导致其在牛市中略显保守，但在极端风险下提供了更厚的安全垫。

3. **参数漂移的方向 (Direction of Parameter Drift)**:

   - **Naive**: 倾向于收紧 `exit_th` (-0.10 → -0.08)，即更早止损。
   - **RAG**: 倾向于降低 `gross_exposure` (0.92 → 0.85)，即系统性降低杠杆。

10.7 技术实现：RAG 反思流程 (RAG Process)

图 7 展示了 RAG Reflection Engine 的具体执行链路。系统在每周五触发反思，若检索到的证据不足，将自动降级为 Naive 逻辑。

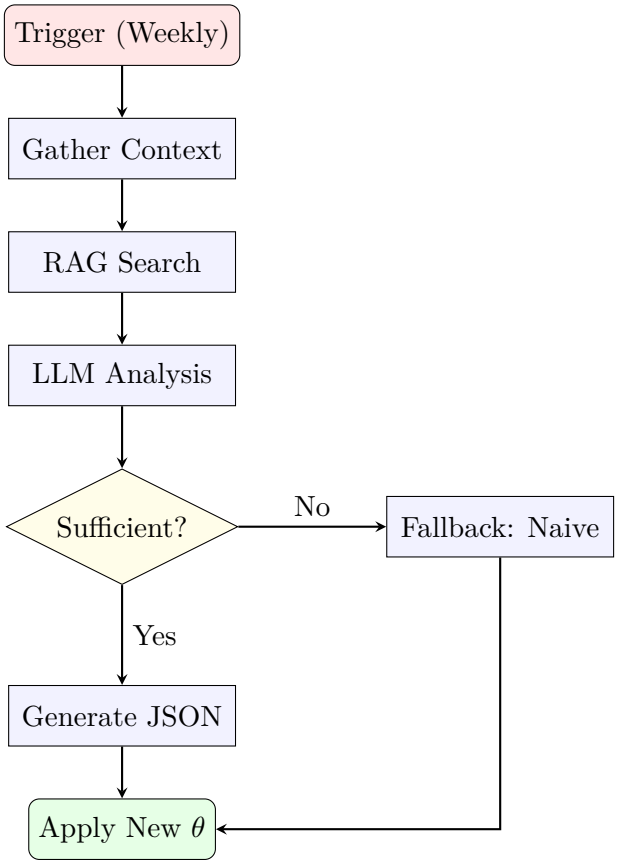


图 7: RAG 反思层执行流程图

10.8 反思：LLM 推理与统计优化的鸿沟 (LLM Inference vs. Statistical Optimization)

虽然实验 4.3 通过 RAG 引入了外部证据，但从结果来看，其综合绩效（夏普比率 1.65）仍未能超越基于统计反馈的 Naive 逻辑（夏普比率 1.95）。这一结果揭示了当前 LLM 集成范式的深层局限性：

1. **统计直觉的缺失与决策震荡**: 实验数据表明，尽管 LLM 拥有证据支持，但在处理参数调优时仍表现出显著的不稳定性。与 Naive 逻辑那种简单、连续且稳健的周收益率反馈相比，LLM 的建议往往表现出非线性的跳跃，更接近于一种“基于信息碎片的直觉映射”，而非严谨的参数估计，导致其信号捕捉效率存在滞后。

2. **单纯检索无法弥合认知偏差**: 实验 4.3 的表现说明，单纯的“检索 + 推理”范式不足以自动衍生出优化的金融逻辑。模型虽拥有“证据”，却缺乏对证据重要性的量化权衡能力，导致其在牛市中因过度解读负面新闻而避险，无法像 Naive 逻辑那样通过统计动量实现收益的稳健增长。

后续优化路径：引入 Few-shot 引导　本轮对比证明，单纯依靠检索增强（RAG）的 LLM 在复杂投资组合管理中的鲁棒性甚至逊于基础的线性规则。这表明仅提供“外部数据”不足以使模型生成稳定的量化逻辑。因此，后续实验将尝试引入 **Few-shot Learning**，通过向 Prompt 中植入高质量的专家决策样本作为逻辑参考，试图解决模型决策震荡的问题，并观察其是否能从单纯的文本理解进化为具备量化直觉的专业智能体。

11 实验 6.x：Few-Shot Learning 对 RAG 反思层性能提升分析 (Few-Shot Strategy Analysis)

11.1 实验概述 (Overview)

本实验旨在量化分析 Few-Shot Learning 提示策略对 RAG 反思层决策质量的影响。通过在 LLM 的 Context Window 中植入历史相似的高质量决策范例（包含市场状态、新闻摘要、风险评估与最终参数调整），我们期望引导模型“学习”更优的风险-收益权衡逻辑。

11.1.1 实验设计 (Design)

为了控制变量，我们仅调整 RAG 反思层 Prompt 中的示例数量 (N)，其余配置（回测周期、股票池、资金管理）保持一致。

- **6.1 Baseline (Zero-Shot)**: 标准 RAG 提示，无示例引导。
- **6.2 One-Shot**: 注入 **1 个**最相关的高质量历史范例。
- **6.3 Two-Shot**: 注入 **2 个**历史范例。
- **6.4 Three-Shot**: 注入 **3 个**历史范例。

11.2 技术实现：动态注入机制 (Dynamic Injection)

为了实现灵活的 Few-Shot 策略，我们在 RAG 系统的 `prompt.py` 与 `answer.py` 模块中构建了动态注入管道。该机制允许在推理时通过参数 `few_shot_count` 实时控制注入的示例数量。

11.3 Prompt 结构优化 (Prompt Engineering)

为了实现 Few-Shot，我们重构了 Prompt 结构，将示例置于 System Prompt 之后、User Query 之前，如图 8 所示。

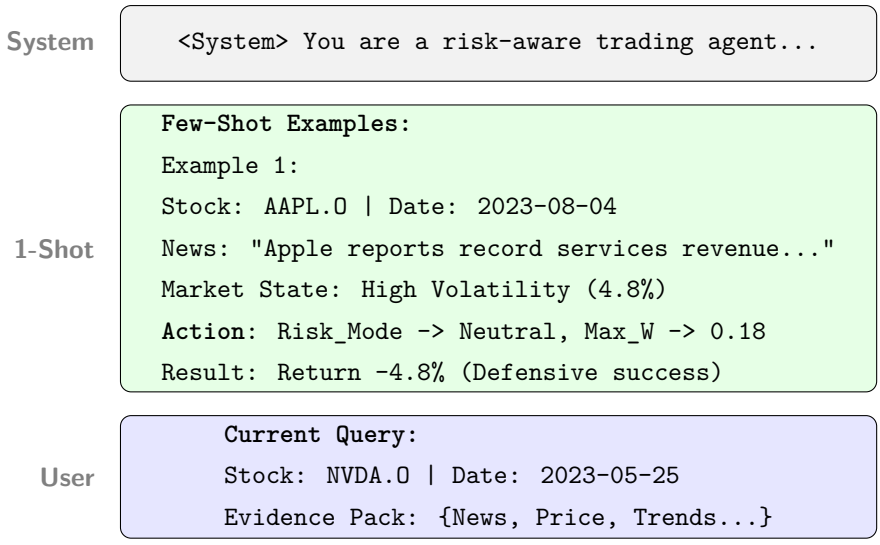


图 8: Few-Shot Prompt 结构示意图

11.3.1 注入数据流 (Injection Data Flow)

图 9 展示了从参数请求到 Prompt 组装的完整数据链路。

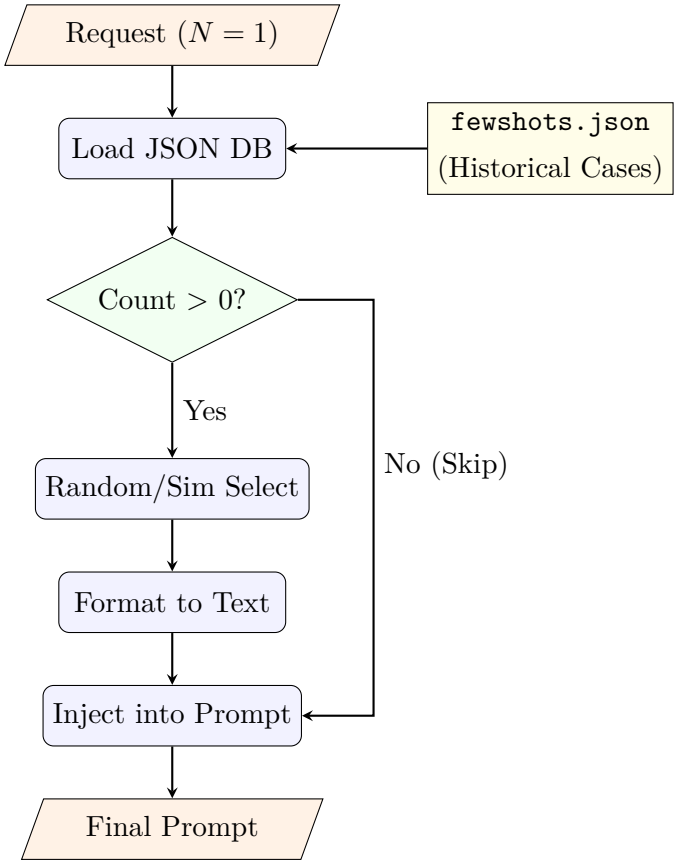


图 9: Few-Shot 动态注入数据流图

11.3.2 核心逻辑实现 (Core Implementation)

手动维护一个外部知识库 `fewshots.json`，存储了经过人工筛选的历史决策案例。

1. **加载与选择 (Loading Selection)**: 系统启动时预加载 JSON 文件。当接收到请求时，`load_few_shot_examples` 函数会根据请求的 N 数量，从库中随机或基于相似度（Similarity-based）抽取 N 个示例。

2. **格式化 (Formatting)**: 选中的结构化 JSON 数据被 `format_few_shot_examples` 函数转换为 LLM 可读的自然语言文本块。

11.4 核心绩效对比 (Core Performance)

实验结果（表 13）显示，引入 Few-Shot Learning 显著提升了策略的整体表现，其中 **1-shot (6.2)** 版本取得了最优的风险调整后收益。

表 13: 实验 6.x 核心绩效指标对比

指标 (Metric)	**6.1 Zero-Shot**	**6.2 One-Shot**	**6.3 Two-Shot**	**6.4 Three-Shot**
收益表现 *(Returns)*				
总收益率 (Total Return)	+32.14%	**+41.88%**	+32.61%	+39.79%
年化收益率 (CAGR)	+32.43%	**+42.27%**	+32.91%	+40.16%
相对基准提升 (Relative)	-	**+30.3%**	+1.5%	+23.8%
风险控制 *(Risk Control)*				
最大回撤 (Max Drawdown)	**17.48%**	18.41%	18.60%	18.92%
年化夏普比率 (Sharpe)	1.62	**1.86**	1.63	1.80
卡玛比率 (Calmar)	1.86	**2.30**	1.77	2.12
VaR (95%)	-1.70%	**-1.56%**	-1.72%	-1.57%
交易质量 *(Trade Quality)*				
胜率 (Hit Rate)	57.63%	**58.82%**	57.85%	58.59%
盈亏比 (Profit Factor)	1.39	**1.51**	1.39	1.47
单笔平均收益 (Avg P&L)	+388	**+475**	+392	+451

11.5 收益提升分析 (Return Analysis)

图 10 直观展示了不同 Shot 数量下的年化收益率。结果呈现出非线性的 “N 型”，而非简单的线性增长。

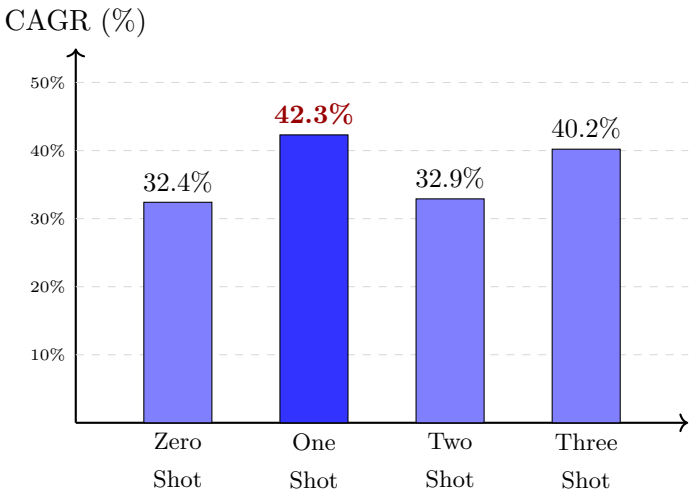


图 10: 不同 Few-Shot 数量下的年化收益率对比

11.5.1 Few-shot 效能分析与局限性 (Analysis of Few-shot Efficacy)

实验结果显示，引入 Few-shot 相比 Bare RAG 确实带来了显著的性能提升，证明了范式引导在金融决策中的有效性。然而，2-Shot 表现的不稳定（甚至出现性能回落）并非逻辑失效，而是揭示了当前方案的瓶颈:

1. **示例质量的局限 (Sample Quality Bottleneck)**: 单纯依靠在 Prompt 中硬编码随机示例，极易引入无关的噪声或不匹配的决策偏置。当示例数量增加但质量不均时，模型不仅无法获得更强的泛化能力，反而可能因上下文信噪比下降而产生决策震荡。

2. **从“上下文提示”到“知识蒸馏” (From Prompting to Distillation)**: Few-shot 的初步成功验证了“专家经验”对 LLM 的校准作用，但 2-Shot 的瓶颈表明，依赖有限的 Context Window 无法承载复杂的交易直觉。这暗示了一个明确的优化方向：利用这些高质量的 Few-shot 样本作为训练集，通过**模型微调 (Fine-tuning)** 或 **知识蒸馏**，将专家级的决策范式直接固化在模型参数中，从而实现更稳健、更高效的策略优化。

结论 虽然当前的 Few-shot 实验尚处于“提示工程”阶段，但它成功为后续引入模型微调铺平了道路。通过参数化这些决策样本，我们有望彻底消除 Prompt 敏感性带来的不稳定性。

11.6 结论与建议 (Conclusion)

1. **显著的正向收益**: 即使仅增加 1 个示例，也能带来 30% 的相对收益提升，证明了 In-Context Learning 在金融决策任务中的有效性。

2. **推荐配置**: 建议在生产环境中默认采用 **1-Shot** 策略。这不仅能获得最佳的夏普比率 (1.86)，还能控制 Prompt Token 消耗，降低 API 成本与延迟。

3. **质量重于数量**: 实验表明，示例的“数量”并非越多越好。未来的优化方向应从“增加数量”转向“提升质量”，例如通过向量检索 (Vector Search) 动态匹配最相似的历史案例作为 Few-Shot 示例。

12 未来工作：从外部检索到模型内化 (Future Work: From RAG to Policy Distillation)

尽管实验 6.x 证明了 RAG 与 Few-Shot 机制能有效提升决策质量，但该架构在生产环境中面临 **推理成本高**（token 消耗大）与 **响应延迟高**（检索耗时）的双重挑战。

为了实现高效的实盘部署，本研究的下一阶段将聚焦于 **策略蒸馏 (Policy Distillation)**。核心思想是：利用当前的“重型”RAG Agent 作为教师（Teacher），通过离线生成高质量的思维链轨迹，训练一个“轻量级”学生模型（Student），使其将外部检索的知识内化为模型参数。

12.1 蒸馏流水线设计 (Distillation Pipeline)

我们采用标准的 Teacher–Student 架构。Teacher 模型负责在离线阶段利用全量 RAG 资源进行“慢思考”，Student 模型通过学习 Teacher 的成功轨迹实现在线阶段的“快直觉”。流程如图 11 所示。

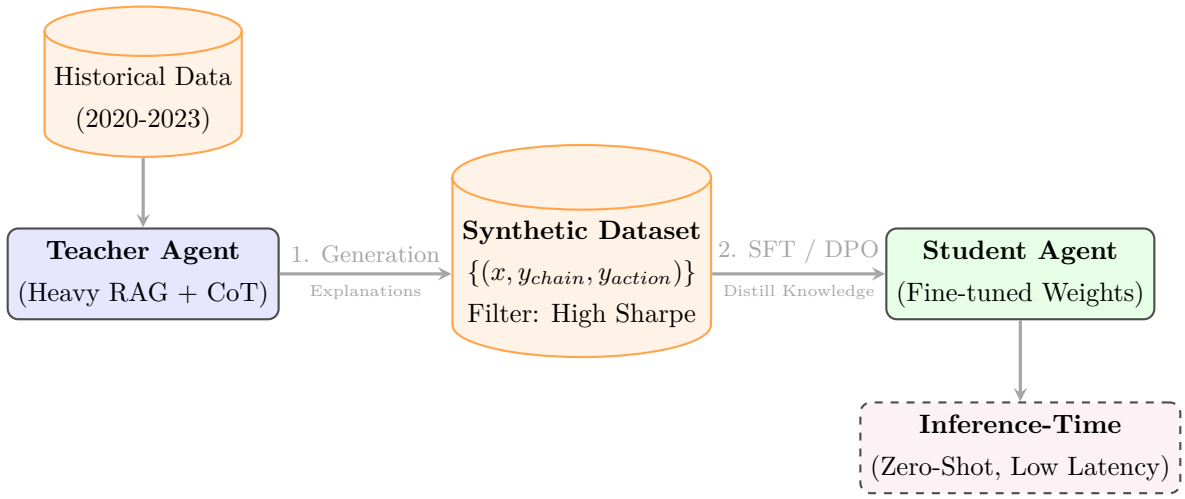


图 11: 基于合成数据的认知蒸馏架构

12.2 关键实施步骤 (Implementation Roadmap)

12.2.1 1. 轨迹生成 (Trajectory Generation)

利用表现最优的 RAG 配置（如 6.x 实验中的 1-Shot 设置）在历史数据上进行推演。Teacher 模型将生成包含完整上下文的思维链 (Chain-of-Thought)：

`Input`: 新闻显示通胀数据超预期...
`CoT`: 通胀高 -> 加息预期 -> 科技股估值承压 -> 避险情绪上升...
`Action`: 降低仓位 (Risk_Off)。

12.2.2 2. 质量过滤 (Quality Filtering)

由于 Teacher 模型并非总是正确，我们需要对其生成的样本进行清洗。

12.2.3 3. 模型内化 (Internalization)

使用筛选后的精英数据集对小参数量模型（如 7B 或更小）进行监督微调（SFT）。

- **去 RAG 化**：经过训练的 Student 模型将具备一种“内隐的检索能力”。它不再需要实时调用外部搜索引擎，而是将常见的市场因果逻辑固化在权重中。
- **零样本部署**：最终实现的 Agent 能够在 **Zero-Shot** 模式下运行，极大降低了对 Prompt Engineering 的依赖和 API 调用成本。