R&D-Agent-Quant：面向以数据为中心的因子与模型联合优化的多智能体框架

R&D-Agent-Quant: A Multi-Agent Framework for Data-Centric Factors and Model Joint Optimization

Yuante Li[1]*, **Xu Yang**[2], **Xiao Yang**[2],
Minrui Xu[3]*, **Xisen Wang**[4]*, **Weiqing Liu**[2]†, **Jiang Bian**[2]
[1] 卡内基梅隆大学, [2] 微软亚洲研究院
[3] 香港科技大学, [4] 牛津大学
yuantel@cs.cmu.edu, {xuyang1, xiao.yang}@microsoft.com
mxubh@connect.ust.hk, xisen.wang@keble.ox.ac.uk
{weiqing.liu, jiang.bian}@microsoft.com

摘要

金融市场维度高、非平稳，波动又持续不散，这些性质给资产收益预测带来本质困难。大语言模型与多智能体系统近年虽有进展，当前的量化研究流程仍然自动化程度有限、可解释性弱。因子挖掘与模型创新等关键环节之间，也依旧各自为政。本文提出面向量化金融的 R&D-Agent，简称 `R&D-Agent(Q)`。这是首个以数据为中心的多智能体框架，通过因子与模型的协同优化，把量化策略的全栈研发自动化。`R&D-Agent(Q)` 把量化过程拆成两个迭代阶段。**研究**阶段动态生成与目标对齐的提示，依据领域先验提出假设，再把假设映射为具体任务。**开发**阶段则交由代码生成智能体 `Co-STEER` 写出任务对应的代码，并投入真实市场回测。两个阶段由反馈阶段衔接：它逐项评估实验结果，据此指导后续迭代，并用多臂老虎机调度器自适应地挑选方向。实证结果上，`R&D-Agent(Q)` 用少于经典因子库 70% 的因子，年化收益率最高可达其 2 倍，在真实市场上也超过 SOTA 深度时间序列模型。因子与模型的联合优化，让预测精度与策略鲁棒性之间取得良好平衡。代码见：`https://github.com/microsoft/RD-Agent`。

1 引言

金融市场是高维非线性的动力系统，其收益序列呈现厚尾 [1]、时变波动 [2] 以及错综的截面相依 [3]。这些特征意味着，资产价格同时被宏观因子、微观结构信号与行为反馈推动 [4–6]，预测难度远超一般时间序列。数据量指数增长，算力与人工智能技术接连突破，资产管理行业正从经验驱动转向数据驱动。在这一转变中，量化投资逐渐成为主流，原因有三：*(i)* 数

本文为 AI 辅助翻译版本，仅供学习交流；引用请以英文原文为准：arXiv:2505.15155。
*本工作完成于微软亚洲研究院实习期间。
†通讯作者

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

据—因子—模型闭环带来高效决策；*(ii)* 执行可复现，且内嵌风险控制；*(iii)* 在策略趋同日益加剧的环境下，仍能精准追逐超额收益 [7, 8]。

图 1 给出现代量化研究流程。微软的开源项目 Qlib [9] 打通数据处理与回测，免去大量重复的工程负担。研究重心随之回到量化的两个核心环节：因子挖掘与模型创新。因子挖掘从 Fama–French 这类解析形式的风险—收益模型 [10, 11] 起步，走向演化式符号回归 [12–14]，近年又转向用强化学习优化因子组合 [15–17]。模型创新则从经典自回归 [18, 2] 演进到机器学习模型 [19–21]，再到序列到序列的深度学习架构（如 GRU [22] 与 LSTM [23]）。更新的一批工作转向专用时间序列模型，一类把信号分解为趋势—季节成分 [24]，另一类改进注意力机制以支撑长程预测 [25]。与此同时，面向个股的模型借助图神经网络，把时序事件序列与截面相依结合起来，刻画股票之间的相互作用 [26–28]。近来，大语言模型（LLM）与多智能体系统进一步拓宽了信息集，一面从新闻与社交网络中抽取信号 [29–31]，一面模拟对冲基金的运作与金融专家之间的协作 [32–34]。

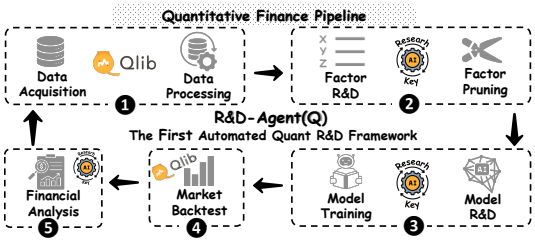

图 1: 量化金融研究流程。Qlib 简化了阶段 ❶ 与 ❹。R&D-Agent(Q) 进一步自动化阶段 ❷、❸ 与 ❺，它们同样是量化研究的关键所在。

进展虽多，量化研究仍卡在三处关键局限上：***(i)*** *自动化有限*：现有工作流在假设生成、编码与调参上都要大量人工介入，迭代因此拖慢，也容易引入偏差。半自动系统同样达不到快速变化的市场所要求的响应速度与可扩展性。***(ii)*** *可解释性差*：现有基于 LLM 的智能体往往直接由语言交互给出交易信号，既无扎实的因子构造，也无透明的模型逻辑，因而容易产生幻觉。实盘交易离不开可解释性与风险控制，这类方法难以落地。***(iii)*** *优化割裂*：量化流程横跨数据处理、因子挖掘、模型训练与评估，现有方法却既缺系统的任务分解，也缺智能体层面的协调。各环节彼此孤立，跨阶段反馈受阻，性能也无法联合提升。

针对上述问题，我们提出 R&D-Agent(Q)，首个以数据为中心的多智能体框架，通过因子与模型的协同优化，自动完成全栈量化策略开发（图 2）。该框架把量化研究拆成五个阶段，归入两大核心环节：**研究（Research）**与**开发（Development）**，R&D 之名正取自这两个词的首字母。研究阶段中，**规范单元（Specification Unit）**依据优化目标动态生成与之对齐的提示。**综合单元（Synthesis Unit）**再从既往结果中长出任务专属的知识森林，产出新的因子或模型假设，并把假设映射为可执行任务。开发阶段引入代码生成智能体 Co-STEER，它依托思维链（CoT）推理 [35] 与基于图的知识存储。**实现单元（Implementation Unit）**把假设翻译成代码，**验证单元（Validation Unit）**执行真实市场回测。**分析单元（Analysis Unit）**以统一指标评估结果，再用多臂老虎机调度器自适应地选出下一个优化方向。假设、实现、验证、反馈由此接成闭环，支撑策略朝既定目标持续演化，向智能自主的量化研究迈出一步。

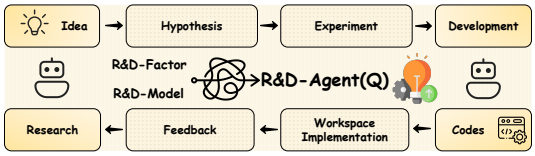

图 2: R&D-Agent(Q) 的概念图。模块 R&D-Factor 与 R&D-Model 分别代表因子开发与模型开发的完整优化闭环。

本文主要贡献如下：

- **端到端自动化，且过程透明：** R&D-Agent(Q) 是量化金融领域首个以数据为中心的多智能体框架，把整个研发过程自动化，产出均可验证，可解释性随之提升，幻觉风险也随之下降。
- **高性能研发工具：**研究阶段，R&D-Agent(Q) 借结构化的知识森林模仿分析师的工作流，生成连贯且高质量的假设。开发阶段，我们提出 Co-STEER，一个面向以数据为中心任务、知识可持续演化的智能体，让因子与模型的代码生成更准确、也更高效。
- **实证表现突出：**真实股票市场上的大量实验显示，成本不到 10 美元，R&D-Agent(Q) 的 ARR 约为基准因子库的 2 倍，用到的因子却少了 70% 以上；在更小的资源预算下，它也超过 SOTA 深度时间序列模型。因子与模型交替优化，进一步在预测精度与策略鲁棒性之间取得出色的权衡。

2 R&D-Agent(Q)

图1 与附录B 给出了量化研究流程的形式化结构。在此基础上，我们提出 R&D-Agent(Q)：一套以数据为中心的多智能体框架，让因子与模型的 R&D 迭代推进，兼顾自动化、可解释性与效率。我们把量化流程拆成五个由 LLM 驱动的单元，每个单元的主要工作是收集信息、与 LLM API 交互：规范单元（Specification Unit，定义场景）、综合单元（Synthesis Unit，生成想法）、实现单元（Implementation Unit，开发代码）、验证单元（Validation Unit，回测）、分析单元（Analysis Unit，评估结果并调度任务）。在统一的输入输出约束下，五个单元构成闭环循环，模拟人类量化研究员的试错过程。与人工工作流不同，R&D-Agent(Q) 可持续自主运行，支持因子与模型两部分的动态协同优化。每一轮的假设、实现与结果都会持久化保存，知识由此不断累积，决策依据也越来越充分。

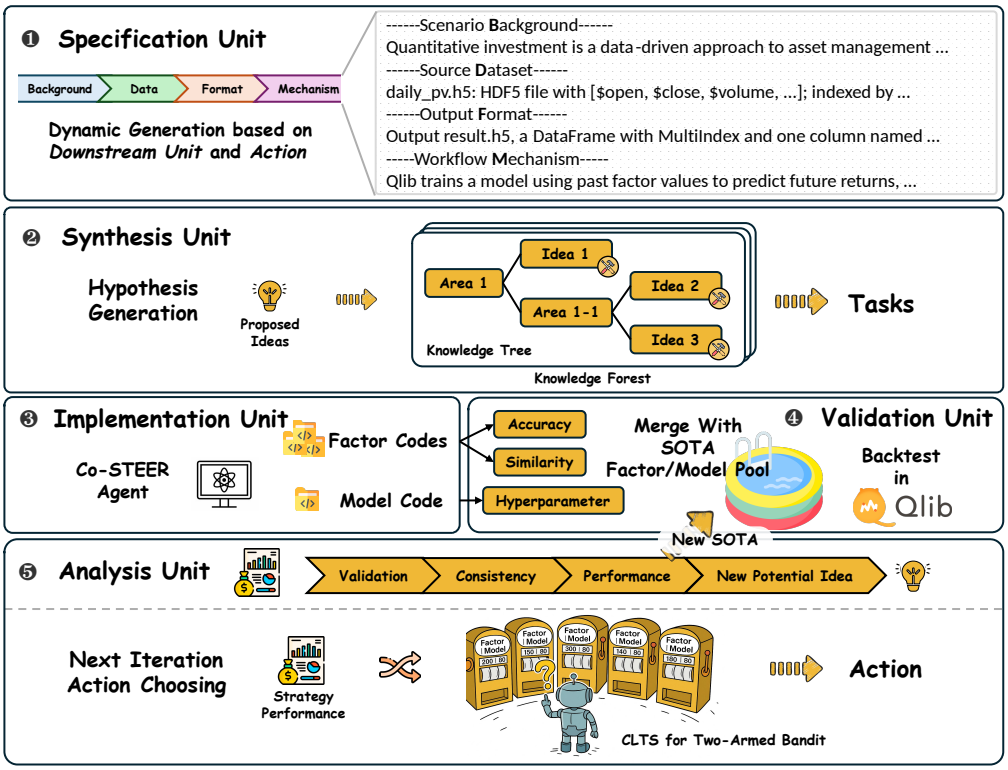

图 3: R&D-Agent(Q) 由五个功能模块组成，它们在持续迭代的循环中协作，为真实金融市场生成高效的量化因子与模型。

2.1 规范单元

规范单元是 R&D-Agent(Q) 的顶层部件，为下游模块动态配置任务上下文与约束，使设计、实现、评估三个环节保持一致。它沿两条轴线工作：❶ 理论轴 ➙ 把先验假设、数据模式与输出协议编码成结构化规范；❷ 经验轴 ➙ 为回测搭建可验证的执行环境与标准化接口，让智能体不必操心底层预处理与基础设施。把形式化定义与统一接口结合起来，歧义随之减少，各部件之间的协同也更有效率。

我们把规范单元形式化为四元组 $\mathcal{S} = (\mathcal{B}, \mathcal{D}, \mathcal{F}, \mathcal{M})$：$\mathcal{B}$ 编码因子或模型的背景假设与先验知识；$\mathcal{D}$ 定义市场数据接口；$\mathcal{F}$ 给出期望的输出格式（如因子张量或收益预测）；$\mathcal{M}$ 表示外部执行环境（如基于 Qlib 的回测）。在这套形式化之下，任何候选因子或模型 $f_{\boldsymbol{\theta}}$ 都须满足 $\forall, x \in \mathcal{D}, ; f_{\boldsymbol{\theta}}(x) \in \mathcal{F}$，且 $f_{\boldsymbol{\theta}}$ 能在 $\mathcal{M}$ 内执行。这样一来，候选对象必须兼容标准的输

入输出结构，后续模块也能在共享的运行上下文中与 $f_{\boldsymbol{\theta}}$ 交互，协作工作流的一致性与可复现性因而有了保障。实现细节见附录 E.1。

2.2 综合单元

综合单元依据历史实验生成新假设，以此模拟人类的推理方式。每次优化动作定义为 $a_t \in \{\text{factor}, \text{model}\}$。对当前动作 a_t，该单元挑出一批相关的历史实验，构造实验轨迹。第 t 次实验记为 $e^t = \{h^t, f^t\}$，其中 h^t 是假设，f^t 是分析单元给出的对应反馈。系统另外维护当前表现最优的解集 SOTA。在此之上，历史假设集与反馈集分别定义为 $\mathcal{H}_t = \{h_1, \ldots, h_t\}$ 与 $\mathcal{F}_t = \{f_1, \ldots, f_t\}$。再按式(1) 抽取以动作为条件的子集。

$$\begin{aligned} \mathcal{F}_t^{(a)} &= \{f_i^a \in \mathcal{F}_t \mid a = a_t \vee e_i \in \text{SOTA}(a)\} \\ \mathcal{H}_t^{(a)} &= \{h_i^a \in \mathcal{H}_t \mid a = a_t \vee h_i \in \text{SOTA}(a)\} \end{aligned} \tag{1}$$

生成式随机映射 G 是研究阶段的核心，它模仿人对理论先验与经验反馈的综合，产出既有效又新颖的假设。把上述两个子集送进 G，即得下一条假设：$h^{(t+1)} = G(\mathcal{H}_t^{(a)}, \mathcal{F}_t^{(a)})$。实际运行时，该模块依托结构化模板与标准化格式，保证假设可执行、有科学依据。以因子生成任务为例，$h^{(t+1)}$ 不只吸收最近一次反馈，还纳入当前市场状况与相关经济理论，使因子有效且可观测。为让假设保持多样、逐步精化，生成机制会随表现反馈调整策略。若 $\mathcal{F}_t^{(a)}$ 显示上一轮成功，下一条假设便提高复杂度或扩大范围；反之则调整结构或引入新变量，最终长成一片想法森林。这套自适应机制让智能体既能探索新方向，又对实验结果保持响应，策略开发得以持续迭代。

最后，假设 h^t 实例化为具体任务 t^t，交由下游实现模块落到代码层面。因子假设 h_t^{factor} 彼此异构、又可能相互作用，可拆成多个子任务 t_i^{factor}。模型假设则结构完整，只映射到单个任务 t^{model}，由它执行整条建模与推理流程。

2.3 实现单元

实现单元负责把综合单元产出的可执行任务翻译成可运行的代码，是 `R&D-Agent(Q)` 中复杂开发工作的核心。为支撑这一过程，我们设计了专用智能体 `Co-STEER`，专门面向量化研究中的因子与模型开发。如图 4，`Co-STEER` 把系统化调度与代码生成策略结合起来，保证实现的正确性、效率与适应性。

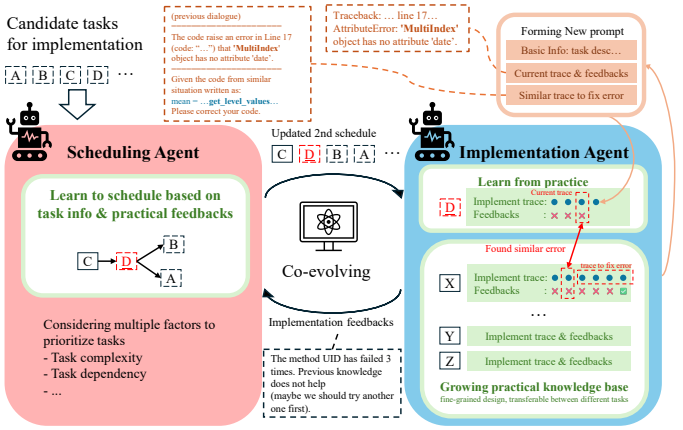

图 4: `Co-STEER` 工作流示意图。在因子开发场景下，它主要由两个模块构成：调度模块接收候选任务，按多项因素迭代排序；实现模块从以往代码的执行结果中学习，构建细粒度、可跨任务迁移的知识库。

因子开发中的任务常带有结构上的依赖。为此我们引入引导式思维链（CoT）机制，让推理过程可追溯。具体而言，智能体构建有向无环图（DAG）$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 来表示任

务依赖：从任务 A 指向任务 B 的边意味着，出于知识流动或复杂度的考虑，A 应当先于 B。由此导出拓扑序 $\pi_S = \big(t_{(1)}, \ldots, t_{(n)}\big)$，用来指导任务执行。调度是自适应的。以往执行的反馈会不断并入规划：某个任务反复失败，说明它复杂度偏高，调度器转而优先处理更简单的任务，以积累知识、提高执行成功率。

对每个任务 t_j，实现智能体 I 结合任务描述与当前知识库生成对应代码 c_j，即 $c_j = I(t_j, \mathcal{K})$。这一过程涵盖任务解析、代码合成与精修、执行以及验证。智能体的目标是最大化累计实现质量：$\pi_I = \arg\max_\pi \mathbb{E}\left[\sum_{j=1}^{n} R_I(c_j)\right]$，其中 $R_I(c_j)$ 衡量代码 c_j 的正确性与性能。知识库 $\mathcal{K}$ 在其中居于中心位置，记录成功与失败的「任务-代码-反馈」三元组：$\mathcal{K}^{(t+1)} = \mathcal{K}^{(t)} \cup \{(t_j, c_j, f_j)\}$，其中 f_j 表示执行任务 t_j 后收到的反馈。借助知识迁移机制，实现智能体还能依据当前反馈 $f^{(t)}$ 从知识库检索相似任务的解法，新任务的代码生成效率与成功率随之提高：$c_{new} = \arg\max_{c_k \in \mathcal{K}} \text{similarity}(t_{new}, t_k) \cdot c_k$。完整算法细节见附录 A.1。

这条由反馈驱动的优化回路让实现单元不断改进代码质量与效率，量化研究组件的开发因此更快、更稳健。

2.4 验证单元

验证单元考察实现单元产出的因子或模型在实践中是否有效。对**因子**，先走一遍去重：计算候选因子与现有 SOTA 因子库的相关性，滤掉冗余信号。给定拼接后的因子矩阵 $\mathbf{F} = [F_{\text{SOTA}}, F_{\text{new}}] \in \mathbb{R}^{T \times (M+N)}$，在每个时间切片内计算全部 $M \times N$ 对「SOTA 因子-新因子」的信息系数（Information Coefficient, IC）：$\text{IC}_{m,n}^{(t)} = \text{corr}(F_{\text{SOTA},m}^{(t)}, F_{\text{new},n}^{(t)})$，其中 (m, n) 索引一对 SOTA 因子与新因子，t 索引时间切片。这些 IC 值随后沿时间取平均；对每个新因子 n，取它与所有 SOTA 因子之间 IC 的最大值：$\text{IC}_{\max}^{(n)} = \max_m \ \mathbb{E}_t\left[\text{IC}_{m,n}^{(t)}\right]$。若新因子满足 $\text{IC}_{\max}^{(n)} \geq 0.99$，即判为冗余并剔除。筛选之后，余下的候选因子与当前 SOTA 模型组合（尚无 SOTA 模型时改用基线模型），送进 `Qlib` 回测平台评估，从而在贴近真实的市场条件下考察表现。对**模型**，流程是对称的：每个候选模型与当前 SOTA 因子集配对，走同一条回测流程。

验证单元由此给出一条一体化的自动流程，在生产级市场仿真环境中对新组件做标准化评估。

2.5 分析单元

在 `R&D-Agent(Q)` 框架中，分析单元兼任研究评审与策略分析两种角色。每轮实验结束后，它从多个维度评估当前假设 h^t、具体任务 t^t 与实验结果 r^t。若判定该实验在动作类型 a_t 下优于 SOTA，其结果便并入对应的 SOTA 集合 $\text{SOTA}(a_t)$。随后单元诊断失败的策略，给出有针对性的改进建议。反馈 f_t 送回综合单元，指导后续假设的构造。

分析单元只看当前实验，视野是局部的；综合单元则纵览完整的实验历史，视野是全局的。两者交互构成闭环系统，在短期响应与长期探索之间取得平衡，使研究设计、策略实现、验证与深度分析各环节得以自动迭代。

每轮分析之后，分析单元还要决定下一次迭代先精修因子还是先优化模型。为最大化性能增益，我们把这一决策建模成带上下文的双臂老虎机问题，用线性汤普森采样求解，详细算法见附录 A.2。具体地，第 t 轮系统观测到 8 维性能状态向量 $\mathbf{x}_t \in \mathbb{R}^8$，其中编码了当前策略的关键评估指标。动作空间为 $\mathcal{A} = \{\text{factor}, \text{model}\}$，对应两条可选的优化路径。为衡量各动作在上下文 $\mathbf{x}_t$ 下的期望收益，我们采用线性奖励函数 $r = \mathbf{w}^\top \mathbf{x}_t$，其中 $\mathbf{w}$ 反映各指标的相对重要性。每个动作各自维护一个贝叶斯线性模型，用高斯后验刻画奖励系数的不确定性。每一步从各后验中采样一个奖励向量，算出相应的期望奖励，执行采样奖励最高的

那个动作。观察到实际提升后，更新所选臂的后验。凭借这套带上下文的汤普森采样机制，R&D-Agent(Q) 自适应地把握探索-利用平衡，性能在迭代中稳健提升。

3 实验设置

➥ **数据集**。沿用 [36–39] 的设置，我们采用 CSI 300 数据集，它覆盖中国市场 300 只 A 股大盘股。时间跨度切成三段：训练集（2008 年 1 月 1 日–2014 年 12 月 31 日）、验证集（2015 年 1 月 1 日–2016 年 12 月 31 日）、测试集（2017 年 1 月 1 日–2020 年 8 月 1 日）。R&D-Agent(Q) 在三种配置下评测：❶ **R&D-Factor** 把预测模型固定为 LightGBM [40]，从 Alpha 20 出发优化因子集 [3]；❷ **R&D-Model** 把输入因子集固定为 Alpha 20，转而搜索更好的模型；❸ **R&D-Agent(Q)** 对因子与模型两部分联合优化。

➥ **基线方法**。因子层面，对比对象是 Alpha 101 [41]、Alpha 158 [42]、Alpha 360 [43] 与 AutoAlpha [44]。模型层面则纳入**机器学习模型**（Linear、MLP、LightGBM[40]、XGBoost [45]、CatBoost [46]、DoubleEnsemble [47]）与**深度学习模型**（GRU [22]、LSTM [23]、ALSTM [48]、Transformer [49]、PatchTST [50]、iTransformer [51]、Mamba [52]、TRA [38]、MASTER [39]、GATs [53]）。更多细节见附录 C.3。

➥ **评测细节**。评测 R&D-Agent(Q) 用到两类指标：**因子预测指标**，包括信息系数（Information Coefficient, IC）、IC 信息比率（ICIR）、Rank IC 与 Rank ICIR；**策略表现指标**，包括年化收益率（Annualized Return, ARR）、信息比率（Information Ratio, IR）、最大回撤（Maximum Drawdown, MDD）与卡玛比率（Calmar Ratio, CR）。策略按预测收益的排名每日做多空，配套设定了调仓规则、持仓保留规则与贴近实际的交易成本。完整的评测与实现细节见附录 C.4 与 C.1。

4 实验分析

➥ **主结果分析**。表 1 列出各基线模型与 R&D-Agent 框架在 CSI 300 数据集上的成绩：预测指标与策略指标上，R&D-Agent 均稳定领先全部基线。

❶ **R&D-Factor（因子优化）**。只对因子空间做自适应优化时，$\texttt{R\&D-Factor}_{\texttt{GPT-4o}}$ 与 $\texttt{R\&D-Factor}_{\texttt{o3-mini}}$ 都超过静态因子库（如 Alpha 158/360）：IC 最高到 0.0497，ARR 最高到 14.61%，用到的人工因子反而更少。可见 R&D-Agent(Q) 里的假设动态迭代与因子筛选，产出的信号比固定的高维因子集更有信息量。

❷ **R&D-Model（模型优化）**。固定因子、只优化模型时，$\texttt{R\&D-Model}_{\texttt{o3-mini}}$ 超过所有基线，Rank IC（0.0546）与 MDD（−6.94%）都是最好成绩。机器学习模型明显落后，说明它们难以刻画金融数据中的噪声与非线性结构。通用深度学习架构（GRU、LSTM、Transformer）预测指标尚可，策略指标却依旧偏弱，特征提取与可兑现收益之间存在落差。出人意料的是，时序预测模型（如 PatchTST、Mamba）两边都不理想，标准序列预测与股市动态之间存在根本性错配。反过来，专用股票预测模型（TRA、MASTER）策略指标突出，预测能力却跟不上，鲁棒性（低 MDD、高 IR）与精度（高 IC）之间要作取舍。这些结果指向同一点：模型配置由自动化假设评估引导、逐轮自适应调整，产出的预测结构比机器学习模型和人工设计的深度学习架构都更鲁棒，也更顾及风险。

❸ **R&D-Agent(Q)（联合优化）**。因子与模型协同优化之后，$\texttt{R\&D-Agent(Q)}_{\texttt{o3-mini}}$ 整体最优：IC 0.0532，ARR 14.21%，IR 1.74，大幅领先最强的基线方法（如 Alpha 158、TRA）。因子与架构一并迭代能带来互补的收益，alpha 建模因此既可扩展又稳定。

[3] 取自 Alpha 158 的二十个经实证检验的因子，覆盖动量、价值、质量与成长。

➥ **研究组件分析**。为考察 R&D-Agent(Q) 的研究动态，我们跟踪 R&D-Factor 中因子假设的演化，重点看它如何权衡探索（生成多样想法）与利用（局部精修）。方法分三步：*(i)* **文本嵌入**：用 Sentence-BERT [54] 把第 t 轮生成的假设 h_t 编码成定长向量 $\mathbf{h}_t$；*(ii)* **相似度矩阵**：两两计算余弦相似度，构成对称矩阵 $\mathbf{S} \in [0,1]^{T\times T}$；*(iii)* **层次聚类**：用凝聚式聚类把相近假设归组，并重排 $\mathbf{S}$ 以显出块状结构。

表 1: 沪深 300 成分股数据集上全部模型的实验结果，含因子预测指标与策略表现指标。颜色标注排名分组：**最优**，次优，良好 (3–8)，中等 (9–14)，较差 (15–20)，以及最差 (21–26)。

模型		CSI300							
		因子预测能力指标				策略表现指标			
		IC	ICIR	Rank IC	Rank ICIR	ARR	IR (SHR*)	MDD	CR
机器学习模型	Linear	0.0134	0.0992	0.0273	0.1962	-0.0302	-0.3710	-0.1987	-0.1520
	MLP	0.0291	0.2096	0.0412	0.3238	0.0003	0.0037	-0.1390	0.0022
	LightGBM	0.0277	0.2211	0.0386	0.3120	0.0397	0.5664	-0.0855	0.4643
	XGBoost	0.0291	0.2410	0.0384	0.3257	0.0316	0.4620	-0.1139	0.2774
	CatBoost	0.0279	0.2181	0.0393	0.3110	0.0513	0.7008	-0.0924	0.5552
	DoubleEnsemble	0.0294	0.2246	0.0417	0.3211	0.0551	0.7968	-0.0971	0.5675
深度学习模型	Transformer	0.0317	0.2538	0.0434	0.3624	0.0293	0.4267	-0.0987	0.2969
	GRU	0.0315	0.2450	0.0428	0.3440	0.0344	0.5160	-0.1017	0.3382
	LSTM	0.0318	0.2367	0.0435	0.3389	0.0381	0.5561	-0.1207	0.3157
	ALSTM	0.0362	0.2789	0.0463	0.3661	0.0470	0.6992	-0.1072	0.4384
	GATs	0.0349	0.2511	0.0462	0.3564	0.0497	0.7338	-0.0777	0.6396
	PatchTST	0.0247	0.1945	0.0315	0.2463	0.0571	0.7191	-0.1327	0.4303
	iTransformer	0.0270	0.1946	0.0340	0.2365	0.0979	1.2337	-0.1151	0.8506
	Mamba	0.0281	0.2244	0.0374	0.2952	0.0229	0.3163	-0.1154	0.1984
	TRA	0.0404	0.3197	0.0490	0.4047	0.0649	1.0091	-0.0860	0.7547
	MASTER	0.0215	0.1925	0.0296	0.2486	0.0896	1.3406	-0.0851	1.0528
因子库	Alpha 101	0.0308	0.2588	0.0331	0.2749	0.0512	0.5783	-0.1253	0.4085
	Alpha 158	0.0341	0.2952	0.0450	0.3987	0.0570	0.8459	-0.0771	0.7393
	Alpha 360	0.0420	0.3290	0.0514	0.4225	0.0438	0.6731	-0.0721	0.6074
	AutoAlpha	0.0334	0.2656	0.0361	0.2967	0.0400	0.4288	-0.1225	0.3266
R&D-Agent 系列框架 *	R&D-Factor$_{\text{GPT-4o}}$	0.0489	0.4050	0.0521	**0.4425**	**0.1461**	1.6835	-0.0750	**1.9468**
	R&D-Factor$_{\text{o3-mini}}$	0.0497	0.3931	0.0500	0.4246	0.1184	1.3566	-0.0910	1.3016
	R&D-Model$_{\text{GPT-4o}}$	0.0326	0.2305	0.0401	0.2767	0.1229	1.6676	-0.0876	1.4029
	R&D-Model$_{\text{o3-mini}}$	0.0469	0.3688	**0.0546**	0.4385	0.1009	1.7009	**-0.0694**	1.4538
	R&D-Agent(Q)$_{\text{GPT-4o}}$	0.0497	0.4069	0.0499	0.4122	0.1144	1.3167	-0.0811	1.4108
	R&D-Agent(Q)$_{\text{o3-mini}}$	**0.0532**	**0.4278**	0.0495	0.4091	0.1421	**1.7382**	-0.0742	1.9150

图 5 显示出三种探索模式：❶ **先局部精修，再换方向：**对角块（如第 1–6 轮、第 7–11 轮）表明 R&D-Factor 会沿同一条思路连做多步改进，之后才转向，深度与新意兼顾。❷ **有策略地回访：**第 26 轮与更早的第 12–14 轮聚在一簇，说明智能体能回头拾起早期有潜力的假设，再逐步打磨。❸ **多路径产生协同：**36 轮试验里有 8 轮进入最终 SOTA 集合，覆盖 6 个簇中的 5 个。多方向探索给出的信号彼此互补，合起来把最终因子库做得更强。这套精修 转向 复用模式支撑起高效的深度搜索与宽广的概念覆盖，

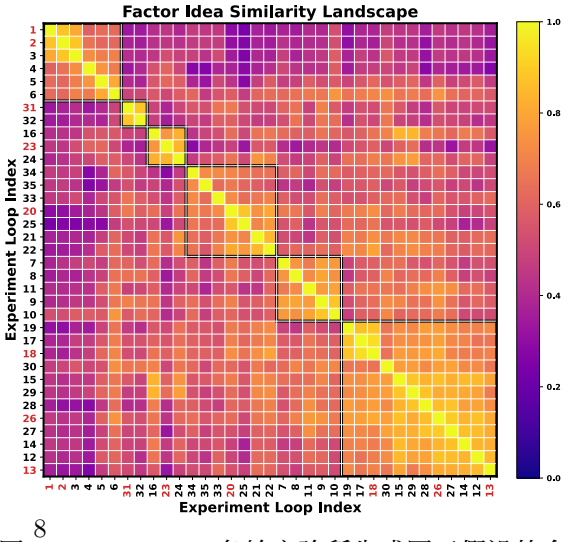

图 5: R&D-Factor 各轮实验所生成因子假设的余弦相似度热力图。黑框圈出想法相近的簇；**红色**序号表示该轮假设被选入 SOTA 因子库。

最终构建出紧凑、多样且高性能的因子库。

➥ **开发组件分析**。为评估开发组件的代码生成能力，我们用 pass@k 准确率考察 `Co-STEER` 在 `R&D-Agent(Q)` 各框架下的表现（图 6）。因子任务与模型任务中，成功率都在少数几轮迭代内迅速收敛，说明 `Co-STEER` 能借助反馈快速修掉最初的错误。全栈任务（`R&D-Agent(Q)`）复杂度更高，差距因此被放大，迭代精修在这里不可或缺。其中 `o3-mini` 的恢复率始终更高，源于更强的思维链（CoT）推理；面对结构化、依赖繁多的编码场景，这一优势尤为明显。总体看，pass@k 曲线体现出 `Co-STEER` 在结构化金融编码中**靠迭代精修逐步自我纠错**的能力。关于 `Co-STEER` 的更多实验见附录 D.4。

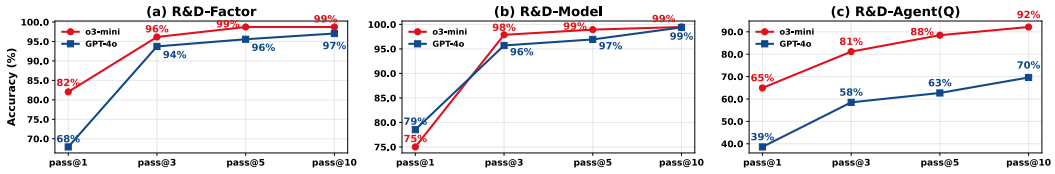

图 6: `GPT-4o` 与 `o3-mini` 在 `R&D-Factor`、`R&D-Model` 与 `R&D-Agent(Q)` 三种设置下的 Pass@k 准确率。横轴为尝试次数 k；纵轴为 k 次尝试之内的成功率。

➥ **因子效果分析**。图 7 把 `R&D-Factor` 产出的因子库与基线因子库摆在一起，用以评估因子生成的质量。子图 (a) 与 (b) 显示，即便从 Alpha 20 起步，`R&D-Factor` 也能很快把 IC 拉到与 Alpha 158、Alpha 360 相当的水平，而因子数只有它们的约 22%。2017 年之后它一直优于 Alpha 20；2019–2020 年基线纷纷退化时，它的 IC 依然平稳。改从 Alpha 158 起步则更进一步，`o3-mini` 尤其明显：2020 年 IC 突破 0.07，高于所有基线。可见因子的反复精修有助于**剔除随市场状态漂移的信号与冗余信号**，整体预测稳定性随之提高。更多相关结果见附录 D.3。

➥ **模型效果分析**。图 8 从 ARR、MDD、资源占用三个维度，比较 `R&D-Model` 与基线深度学习模型。两个 `R&D-Model` 变体都明显移向了理想的左上区域。$\texttt{R\&D-Model}_{\texttt{GPT-4o}}$ 的 ARR 达 12%，|MDD| 为 8%，收益-风险斜率最高。$\texttt{R\&D-Model}_{\texttt{o3-mini}}$ 的 ARR 为 11%，回撤更小，**在更紧的风险约束下依然表现稳健**。

实验范围与泛化能力分析。针对实验范围、时效性与数据泄漏这三点疑虑，我们把评测扩展到中国市场的另一主要指数 CSI 500，以及美国市场的 NASDAQ 100。数据划分也随之更新（训练：2008-2021，验证：2022-2023，测试：2024-2025），使两个 LLM 后端的知识截止时间完全早于或几乎早于测试期。详细实验设置与完整结果见 D.1。

另外，本框架采用以数据为中心的设计。LLM 从不接触原始行情数据，也看不到明确的时间划分，只拿到 schema 层面的信息（见附录 E.1 中规范单元的提示词）。模型因此拿不到精确的时间边界与数据集划分，信息泄漏的风险随之降低。

表 2 汇总的结果显示，`R&D-Agent(Q)` 在中美两个市场上都稳居前列，IC、ICIR、IR（SHR*）与 MDD 的样本外表现都很稳健。这些结果印证：本框架不止适用于中国市场，也能捕捉近期的市场动态，且不受知识截止时间的影响，**鲁棒性**与**实际可用性**都站得住。

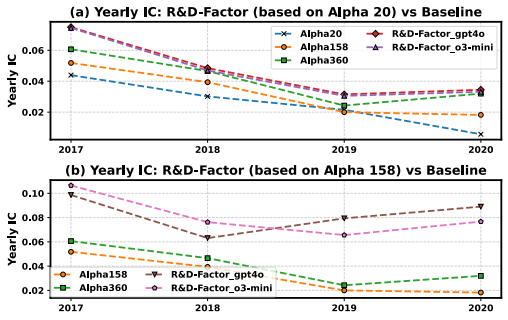

图 7: 在 CSI 300 上以 LightGBM 作预测模型，对比经典因子库与 R&D-Factor 生成的因子。R&D-Factor 分别从 Alpha 20 或 Alpha 158 起步，后端分别用 GPT-4o 或 o3-mini。

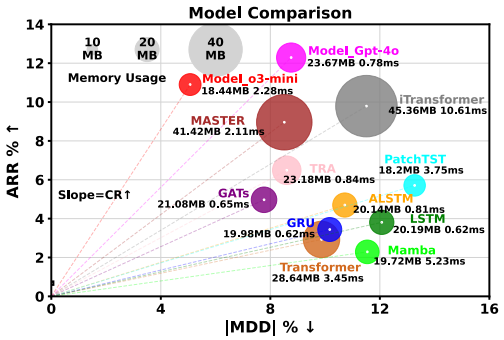

图 8: 各模型在 CSI 300 上的收益、回撤与资源效率对比。气泡大小表示内存占用（MB）；标签给出内存占用与推理延迟（ms）。连线斜率即卡玛比率（CR）。

表 2: 中证 500 与 NASDAQ 100 上的样本外实验结果（测试区间均为 2024 年至 2025 年 6 月），含因子预测指标（IC、ICIR）与策略表现指标（IR、MDD）。颜色标注排名分组：**最优**，次优，良好 (3–5)，中等 (6–10)，较差 (11–15)，以及 最差 (16–19)。

	模型	CSI500				NASDAQ100			
		IC	ICIR	IR (SHR*)	MDD	IC	ICIR	IR (SHR*)	MDD
机器学习模型	LightGBM	0.0181	0.1271	-0.3178	-0.2089	0.0080	0.0652	-0.2603	-0.1342
	XGBoost	0.0240	0.1675	0.0634	-0.1766	0.0076	0.0527	0.1544	-0.1211
	CatBoost	0.0241	0.1629	0.1438	-0.1799	0.0095	0.0614	-0.0735	-0.1148
	DoubleEnsemble	0.0248	0.1705	0.2500	-0.2094	0.0047	0.0360	-0.0046	-0.1404
深度学习模型	Transformer	0.0194	0.1355	0.2898	-0.1331	-0.0011	-0.0077	-0.0343	-0.1553
	GRU	0.0188	0.1022	0.3716	-0.1602	0.0064	0.0457	0.2930	-0.1504
	LSTM	0.0219	0.1434	0.6900	-0.1075	0.0062	0.0409	0.4526	-0.1204
	GATs	0.0162	0.1013	0.5168	-0.1569	-0.0004	-0.0023	0.5772	-0.1491
	iTransformer	0.0161	0.1031	0.0985	-0.1496	0.0076	0.0421	0.3612	-0.1991
	TRA	0.0260	0.1813	0.6040	-0.1461	0.0058	0.0446	0.4608	-0.1351
因子库	Alpha 158	0.0192	0.1353	0.2515	-0.1771	0.0040	0.0324	0.0303	-0.1140
	Alpha 360	0.0195	0.1331	0.2527	-0.1270	0.0042	0.0327	0.5890	-0.1182
	AutoAlpha	0.0184	0.1529	0.5728	-0.1006	0.0046	0.0265	0.0974	-0.1165
R&D-Agent 系列框架 *	R&D-Factor (GPT-4o)	0.0201	0.1709	1.3730	-0.0787	0.0070	0.0446	1.0985	-0.0977
	R&D-Factor (o4-mini)	0.0264	**0.2652**	1.0014	-0.1215	0.0166	0.1017	1.1169	-0.0650
	R&D-Model (GPT-4o)	0.0259	0.1649	1.0941	-0.1367	0.0128	0.0831	1.0742	-0.0842
	R&D-Model (o4-mini)	0.0265	0.1825	1.4021	-0.0735	0.0081	0.0484	1.2671	-0.0741
	R&D-Agent(Q) (GPT-4o)	0.0241	0.1532	1.4227	-0.0803	**0.0172**	0.0908	1.3312	-0.1044
	R&D-Agent(Q) (o4-mini)	**0.0288**	0.1828	**2.1721**	**-0.0656**	0.0162	**0.1035**	**1.7737**	**-0.0634**

➥ **消融实验**。为评估不同动作选择策略的影响，我们做了消融实验，结果见表 3。Bandit 调度器综合最优，IC、ARR 与 SOTA 入选数三项都最高，说明它能**在有限算力预算下优先攻最有希望的优化目标**。基于 LLM 的策略居中，但额外的模型调用推高了单步开销，迭代轮数因此偏少。随机调度最差，可见有依据的决策对优化效果至关重要。完整消融结果见附录 D.2。

➥ **后端对比**。为考察 R&D-Agent(Q) 对 LLM 后端有多敏感，我们在研究指标与策略指标上评测了六种 API 变体（图 9）。o1 的轮次统计并不出众，却靠几轮见效的迭代拿下策略上的突破，总体表现最好。新近发布的 GPT-4.1 在多数指标上排第二。其余变

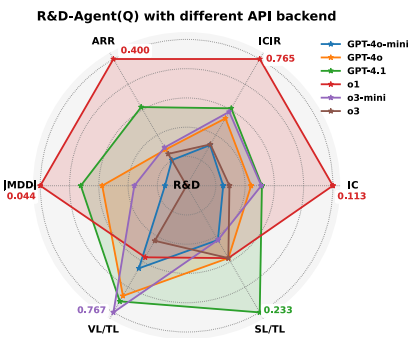

图 9: R&D-Agent(Q) 换用不同 API

表 3: R&D-Agent(Q)（o3-mini）在动作选择策略上的消融结果。我们从预测质量、策略表现、执行统计三方面比较随机、基于 LLM 与 Bandit 三种控制器（TL：总轮数，VL：有效轮数，SL：SOTA 入选数，TRH：运行时长，单位小时）。

模型		因子预测能力指标		策略表现指标		执行指标			
		IC	**ICIR**	**ARR**	**MDD**	**TL**	**VL**	**SL**	**TRH**
算法消融	R&D-Agent(Q)+ 随机	0.0445	0.3589	0.0897	-0.1004	33	19	7	12
	R&D-Agent(Q)+ LLM	0.0476	0.3891	0.1009	-0.0794	33	20	5	12
	R&D-Agent(Q)+ Bandit	**0.0532**	**0.4278**	**0.1421**	**-0.0742**	**44**	**24**	**8**	12

体成绩相近，只有 GPT-4o-mini 因推理能力有限而偏弱。可见本框架换用不同 LLM 后端都保持稳健。

➥ **拓展研究**。附录 D.5 进一步表明，R&D-Agent(Q) 在本文设定下的成本不足 $10，扩展起来足够省钱。附录 D.6 则在真实量化场景中验证了它的鲁棒性。

5 相关工作

量化研究中的传统方法。量化策略传统上依赖资产定价理论中人工构造的因子，如价值因子和动量因子 [10, 11]。这些固定信号虽然可解释，但难以灵活适应市场状态的变化。为克服这些局限，符号回归与遗传编程（GP）方法 [14, 55] 通过演化复杂的非线性表达式实现因子挖掘自动化。滞后算子 [13]、算子变异与剪枝 [56] 等改进手段能生成更多样、更有效的信号。强化学习（RL）方法换了一种表述，把因子权重分配看作序贯决策问题，直接优化夏普比率或卡玛比率 [15, 57]。Andre 等人 [16] 用带 KL 正则化的 Dirichlet 策略建模因子权重，实现稀疏且自适应的策略。但 RL 方法在市场状态突变（如 2020 年熔断事件 [58]）下往往缺乏鲁棒性，且可解释性差。

在模型层面，ARIMA [59]、指数平滑 [60] 等早期方法难以应对高噪声、高维度的数据。支持向量机 [19]、随机森林 [20] 等经典机器学习方法提升了鲁棒性，但仍需人工特征工程。此后，LSTM [61]、Transformer [62] 等深度学习模型被用于捕捉长期依赖与截面依赖 [63, 64]。在此基础上，出现了专门的时间序列神经网络：PatchTST [65] 将输入切分为局部块，iTransformer [66] 则重新映射变量-token 关系以建模多变量结构。MASTER [67] 等领域专用模型进一步引入市场层面的动态信息，提升了金融预测效果。但因子流程与模型流程仍相互割裂、依赖专家经验、缺乏灵活性，制约了其在波动市场中的可扩展性。

金融领域的 LLM 驱动智能体。大语言模型（LLM）凭借强大的推理与抽象能力，为金融研究自动化带来了新机遇。近期研究探索了将其用于从金融文本中提取预测信号 [68, 31]、生成因子解释 [30]，以及实现多模态市场分析 [33]。与此同时，基于 LLM 的多智能体系统（如 AutoGen [69]、AutoGPT [70]）也在发展，为复杂决策提供协同框架。金融领域中，FinAgent [33]、TradingAgents[34] 等系统采用基于角色分工的智能体，处理事件抽取、投资组合更新等子任务。但现有工作大多聚焦于狭窄的子任务，严重依赖语义信号，因而容易产生幻觉、难以解释、难以复现。同时，这些工作缺乏因子与模型联合优化的机制，各环节也没有串成完整工作流，实际量化系统中的效果因此受限。

6 结论

我们提出 R&D-Agent(Q)，一个面向量化金融、由 LLM 驱动的因子-模型协同开发框架。该框架将研究过程分解为模块化组件，并集成基于多臂老虎机的调度器，在固定算力预算下支持高效、自适应的迭代。实证结果表明，R&D-Agent 在信号质量与策略表现上均优于基线方法，且成本效率高、泛化能力强。其模块化设计还便于适配真实场景。但当前框架仅依赖 LLM 内部的金融知识。未来工作可提升数据多样性、引入领域先验知识，并让系统能随市场状态的演变在线自适应。

7 免责声明

R&D-Agent(Q) 框架及其配套代码的使用者应自行准备金融数据，并在自身使用场景中独立评估、测试所生成因子与模型的风险。智能体生成的代码、数据与模型必须谨慎使用，并全面核查。R&D-Agent(Q) 框架不提供金融意见，也无意取代合格的金融专业人士——金融产品的设计、评估与审批应由他们负责。R&D-Agent(Q) 框架的输出不代表微软的意见。

参考文献

[1] B. B. Mandelbrot and B. B. Mandelbrot, The variation of certain speculative prices. Springer, 1997.

[2] R. F. Engle, "Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation," Econometrica: Journal of the econometric society, pp. 987–1007, 1982.

[3] F. X. Diebold and K. Yılmaz, "On the network topology of variance decompositions: Measuring the connectedness of financial firms," Journal of econometrics, vol. 182, no. 1, pp. 119–134, 2014.

[4] W. A. Brock and C. H. Hommes, "A rational route to randomness," Econometrica: Journal of the Econometric Society, pp. 1059–1095, 1997.

[5] ——, "Heterogeneous beliefs and routes to chaos in a simple asset pricing model," Journal of Economic dynamics and Control, vol. 22, no. 8-9, pp. 1235–1274, 1998.

[6] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in Handbook of the fundamentals of financial decision making: Part I. World Scientific, 2013, pp. 99–127.

[7] B. Cao, S. Wang, X. Lin, X. Wu, H. Zhang, L. M. Ni, and J. Guo, "From deep learning to llms: A survey of ai in quantitative investment," 2025. [Online]. Available: https://arxiv.org/abs/2503.21422

[8] N. Gârleanu and L. H. Pedersen, "Dynamic trading with predictable returns and transaction costs," The Journal of Finance, vol. 68, no. 6, pp. 2309–2340, 2013.

[9] X. Yang, W. Liu, D. Zhou, J. Bian, and T.-Y. Liu, "Qlib: An ai-oriented quantitative investment platform," arXiv preprint arXiv:2009.11189, 2020.

[10] E. F. Fama and K. R. French, "Common risk factors in the returns on stocks and bonds," Journal of Financial Economics, vol. 33, no. 1, pp. 3–56, 1993.

[11] M. M. Carhart, "On persistence in mutual fund performance," The Journal of Finance, vol. 52, no. 1, pp. 57–82, 1997.

[12] J. R. Koza, "Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems)," A Bradford Book, vol. 1, p. 18, 1993.

[13] J. Stelmack, Kaizen Programming with Enhanced Feature Discovery: An Automated Approach to Feature Selection and Feature Discovery for Prediction Models. Rochester Institute of Technology, 2020.

[14] P.-A. Kamienny, G. Lample, S. Lamprier, and M. Virgolin, "Deep generative symbolic regression with monte-carlo-tree-search," in International Conference on Machine Learning. PMLR, 2023, pp. 15 655–15 668.

[15] J. Zhao, C. Zhang, M. Qin, and P. Yang, "Quantfactor reinforce: Mining steady formulaic alpha factors with variance-bounded reinforce," arXiv preprint arXiv:2409.05144, 2024.

[16] E. André and G. Coqueret, "Dirichlet policies for reinforced factor portfolios," arXiv preprint arXiv:2011.05381, 2020.

[17] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," IEEE transactions on neural networks and learning systems, vol. 28, no. 3, pp. 653–664, 2016.

[18] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[19] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," Computers & operations research, pp. 2513–2522, 2005.

[20] F. S. D. Nugroho, T. B. Adji, and S. Fauziati, "Decision support system for stock trading using multiple indicators decision tree," in International Conference on Information Technology, Computer, and Electrical Engineering, 2014, pp. 291–296.

[21] R. A. Kamble, "Short and long term stock trend prediction using decision tree," in 2017 International Conference on Intelligent Computing and Control Systems, 2017, pp. 1371–1375.

[22] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, pp. 1735–1780, 1997.

[24] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," Advances in neural information processing systems, vol. 34, pp. 22 419–22 430, 2021.

[25] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[26] Q. Zhang, Y. Zhang, X. Yao, S. Li, C. Zhang, and P. Liu, "A dynamic attributes-driven graph attention network modeling on behavioral finance for stock prediction," ACM Transactions on Knowledge Discovery from Data, vol. 18, no. 1, pp. 1–29, 2023.

[27] P. Zhu, Y. Li, Y. Hu, Q. Liu, D. Cheng, and Y. Liang, "Lsr-igru: Stock trend prediction based on long short-term relationships and improved gru," in Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 5135–5142.

[28] S. Xiang, D. Cheng, C. Shang, Y. Zhang, and Y. Liang, "Temporal and heterogeneous graph neural network for financial time series prediction," in ACM International Conference on Information & Knowledge Management, 2022, pp. 3584–3593.

[29] B. Yan, X. Zhang, L. Zhang, L. Zhang, Z. Zhou, D. Miao, and C. Li, "Beyond self-talk: A communication-centric survey of llm-based multi-agent systems," arXiv preprint arXiv:2502.14321, 2025.

[30] M. Wang, K. Izumi, and H. Sakaji, "Llmfactor: Extracting profitable factors through prompts for explainable stock movement prediction," in Findings of the Association for Computational Linguistics ACL 2024, 2024, pp. 3120–3131.

[31] A. Lopez-Lira and Y. Tang, "Can chatgpt forecast stock price movements? return predictability and large language models," arXiv preprint arXiv:2304.07619, 2023.

[32] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin et al., "Metagpt: Meta programming for a multi-agent collaborative framework," in The Twelfth International Conference on Learning Representations.

[33] W. Zhang, L. Zhao, H. Xia, S. Sun, J. Sun, M. Qin, X. Li, Y. Zhao, Y. Zhao, X. Cai et al., "A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist," in Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 4314–4325.

[34] Y. Xiao, E. Sun, D. Luo, and W. Wang, "Tradingagents: Multi-agents llm financial trading framework," arXiv preprint arXiv:2412.20138, 2024.

[35] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837.

[36] X. Li, Z. Li, C. Shi, Y. Xu, Q. Du, M. Tan, and J. Huang, "Alphafin: Benchmarking financial analysis with retrieval-augmented stock-chain framework," in Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), 2024, pp. 773–783.

[37] Y. Duan, L. Wang, Q. Zhang, and J. Li, "Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns," in Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 4468–4476.

[38] H. Lin, D. Zhou, W. Liu, and J. Bian, "Learning multiple stock trading patterns with temporal routing adaptor and optimal transport," in Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 1017–1026.

[39] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, and S. Huang, "Master: Market-guided stock transformer for stock price forecasting," in Proceedings of the AAAI Conference on Artificial Intelligence, 2024, pp. 162–170.

[40] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," Advances in neural information processing systems, vol. 30, 2017.

[41] Z. Kakushadze, "101 formulaic alphas," 2016.

[42] "Alpha 158 from microsoft qlib," https://github.com/microsoft/qlib/blob/85cc74846b5af2e3e6d18666a2f6e399396980b9/qlib/contrib/data/loader.py#61, accessed: 2025-05-12.

[43] "Alpha 360 from microsoft qlib," https://github.com/microsoft/qlib/blob/85cc74846b5af2e3e6d18666a2f6e399396980b9/qlib/contrib/data/loader.py#4, accessed: 2025-05-12.

[44] Z. Kou, H. Yu, J. Luo, J. Peng, and L. Chen, "Automate strategy finding with llm in quant investment," arXiv preprint arXiv:2409.06289, 2024.

[45] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[46] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," Advances in neural information processing systems, vol. 31, 2018.

[47] C. Zhang, Y. Li, X. Chen, Y. Jin, P. Tang, and J. Li, "Doubleensemble: A new ensemble method based on sample reweighting and feature selection for financial data analysis," in 2020 IEEE International Conference on Data Mining (ICDM), 2020, pp. 781–790.

[48] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, pp. 2627–2633.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, 2017.

[50] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in The Eleventh International Conference on Learning Representations, 2023.

[51] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in The Twelfth International Conference on Learning Representations, 2023.

[52] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in First Conference on Language Modeling, 2024.

[53] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in 6th International Conference on Learning Representations, 2018.

[54] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3982–3992.

[55] B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," in International Conference on Learning Representations.

[56] C. Cui, W. Wang, M. Zhang, G. Chen, Z. Luo, and B. C. Ooi, "Alphaevolve: A learning framework to discover novel alphas in quantitative investment," in Proceedings of the 2021 International conference on management of data, 2021, pp. 2208–2216.

[57] Y.-J. Hu and S.-J. Lin, "Deep reinforcement learning for optimizing finance portfolio management," in 2019 amity international conference on artificial intelligence (AICAI). IEEE, 2019, pp. 14–20.

[58] S. Gu, B. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," The Review of Financial Studies, vol. 33, no. 5, pp. 2223–2273, 2020.

[59] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014, pp. 106–112.

[60] E. S. Gardner Jr, "Exponential smoothing: The state of the art," Journal of Forecasting, pp. 1–28, 1985.

[61] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, "Clvsa: a convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets," in International Joint Conference on Artificial Intelligence, 2019, pp. 3705–3711.

[62] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale gaussian transformer for stock movement prediction," in International Joint Conference on Artificial Intelligence, 2020, pp. 4640–4646.

[63] W. Xu, W. Liu, C. Xu, J. Bian, J. Yin, and T.-Y. Liu, "Rest: Relational event-driven stock trend forecasting," in Proceedings of the Web Conference, 2021, pp. 1–10.

[64] S. Zhao, D. Wang, and R. Douady, "Polymodel for hedge funds' portfolio construction using machine learning," 2024. [Online]. Available: https://arxiv.org/abs/2412.11019

[65] Y. Nie, Q. Yuan, and Y. e. a. Wu, "A time series is worth 64 words: Long-term forecasting with transformers," in Proceedings of the International Conference on Learning Representations (ICLR), 2023.

[66] Y. Liu, Y. Wang, Q. Lin, and et al., "itransformer: Inverted transformers are effective for time-series forecasting," arXiv preprint arXiv:2310.06625, 2024.

[67] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, and S. Huang, "Master: Market-guided stock transformer for stock price forecasting," in Proceedings of the AAAI Conference on Artificial Intelligence, 2024, pp. 162–170.

[68] Z. Liu, D. Huang, K. Huang, Z. Li, and J. Zhao, "Finbert: A pre-trained financial language representation model for financial text mining," in Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence, 2021, pp. 4513–4519.

[69] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu et al., "Autogen: Enabling next-gen llm applications via multi-agent conversation," in ICLR 2024 Workshop on Large Language Model (LLM) Agents.

[70] H. Yang, S. Yue, and Y. He, "Auto-gpt for online decision making: Benchmarks and additional opinions," arXiv preprint arXiv:2306.02224, 2023.

[71] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[72] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in Advances in Neural Information Processing Systems, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 8634–8652.

[73] X. Jiang, Y. Dong, L. Wang, Q. Shang, and G. Li, "Self-planning code generation with large language model," arXiv preprint arXiv:2303.06689, 2023.

[74] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," in The Twelfth International Conference on Learning Representations, 2024.

[75] H. Shi, W. Song, X. Zhang, J. Shi, C. Luo, X. Ao, H. Arian, and L. A. Seco, "Alphaforge: A framework to mine and dynamically combine formulaic alpha factors," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, no. 12, 2025, pp. 12 524–12 532.

[76] H. Chen, X. Shen, Z. Ye, X. Yang, X. Yang, W. Liu, and J. Bian, "RD2Bench: Toward data-centric automatic R&D," in ICLR 2024 Workshop: How Far Are We from AGI, 2024.

[77] A. Meyer, BerniceOptiver, CameronOptiver, IXAGPOPU, J. Liu, M. P. (Optiver), OptiverMerle, S. Dane, and S. Vallentine, "Optiver realized volatility prediction," https://kaggle.com/competitions/optiver-realized-volatility-prediction, 2021, kaggle.

A 算法细节

A.1 Co-STEER 实现逻辑

为进一步说明 Co-STEER 智能体的内部机制，这里给出形式化的算法流程（算法 1）与方法对比分析（表 4）。

已有的自然语言转代码方法大多只盯住单项能力。少样本学习 [71] 借助上下文示例引导模型输出；思维链（CoT） [35] 用逐步提示提升推理连贯性；Reflexion [72] 与 Self-Debugging [73] 强调依靠反馈反复修正；Self-Planning [74] 则支持任务自动分解。

Co-STEER 给出的是统一方案，把调度、推理、反馈驱动的改写与长期知识积累整合在一起。它维护一个持续扩充的知识库，记录以往的尝试，因而能检索并改造此前成功的解法。调度智能体还支持多任务代码生成，量化研究中的因子实现即属此类。它按复杂度与反馈动态排定任务优先级，先做简单、更基础的任务，为后续代码生成搭好脚手架。

表 4: 量化研究中代码生成方法的对比。Co-STEER 是唯一支持端到端开发的方法，从任务调度一直覆盖到实现，并辅以结构化推理与终身知识复用。它因此尤其适合多步骤、以数据为中心的金融工作流。

方法	调度	实现			
	-	示例	实现前的规划或推理	基于 LLM 的自反馈	持续增长的实践知识
Few-shot [71]	✗	✓	✗	✗	✗
CoT [35]	✗	✗	✓	✗	✗
Reflexion [72]	✗	✗	✗	✓	✗
Self-Debugging [73]	✗	✗	✗	✓	✗
Self-Planning [74]	✗	✗	✓	✗	✗
Co-STEER	✓	✓	✓	✓	✓

下面给出因子实现任务上 Co-STEER 工作流的完整伪代码。

A.2 多臂老虎机调度逻辑

如第 2.5 节所述，我们依据当前的策略表现，用上下文汤普森采样在因子改进与模型优化这两个方向之间自适应选择。该问题建模为带线性奖励函数的双臂上下文老虎机。每条臂各自维护一个贝叶斯线性回归模型，每次交互后更新其后验。

在第 t 轮，系统用如下 8 维表现向量概括策略质量：

$$\mathbf{x}_t = \big[\text{IC},\ \text{ICIR},\ \text{Rank(IC)},\ \text{Rank(ICIR)},\ \text{ARR},\ \text{IR},\ -\text{MDD},\ \text{SR}\big]^\top \in \mathbb{R}^8$$

各分量都与理想的策略结果正相关；最大回撤（MDD）取负号，以便方向一致。

给定先验 $\mu^{(a)} = \mathbf{0}$、$P^{(a)} = \tau^{-2} I$，算法为每个动作从后验中采样一个奖励系数向量，在当前上下文 $\mathbf{x}_t$ 下估计奖励，再选出采样值最高的动作。随后按标准的贝叶斯线性回归公式更新后验。

算法 1 Co-STEER：面向量化研究的协同调度与任务执行引擎

Require: 任务集 $\mathcal{T} = \{t_1, \ldots, t_n\}$，知识库 $\mathcal{K}$
Ensure: 实现完成的代码方案 $\{c_1, \ldots, c_n\}$
初始化有向无环图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$，其中 $\mathcal{V} = \mathcal{T}$
对所有 $t_j \in \mathcal{T}$ 初始化任务复杂度分数 $\alpha_j = 1$
function UpdateTaskOrder($\mathcal{G}, \{\alpha_j\}$)
    计算带权边：对 $(i, j) \in \mathcal{E}$ 取 $w_{ij} = \alpha_i / \alpha_j$
    返回考虑 w_{ij} 后的拓扑序 π_S
end function
function ImplementTask($t_j, \mathcal{K}, f^{(t)}$)
    查找相似任务：$S_j = \{t_k \in \mathcal{K} : \text{similarity}(t_j, t_k) > \theta\}$
    $c_{ref} = \arg\max_{c_k \in \mathcal{K}} \text{similarity}(t_j, t_k) \cdot c_k$
    生成代码：$c_j = I(t_j, c_{ref}, \mathcal{K})$
    执行并取得反馈 f_j
    更新知识库：$\mathcal{K} \leftarrow \mathcal{K} \cup \{(t_j, c_j, f_j)\}$
    return (c_j, f_j)
end function
while $\mathcal{T}$ 非空 **do**
    $\pi_S \leftarrow$ UpdateTaskOrder($\mathcal{G}, \{\alpha_j\}$)
    for $t_j \in \pi_S$ **do**
        $(c_j, f_j) \leftarrow$ ImplementTask($t_j, \mathcal{K}, f^{(t)}$)
        if f_j 表明失败 **then**
            更新复杂度：$\alpha_j \leftarrow \alpha_j + \delta$
            跳出循环并重新计算 π_S
        else
            $\mathcal{T} \leftarrow \mathcal{T} \setminus \{t_j\}$
        end if
    end for
end while
return $\{c_1, \ldots, c_n\}$

B 量化研究流程的形式化定义

沿着量化金融的几项经典工作 [9, 39, 75]，我们把原始数据集定义为一个带双重时间索引的三维张量，记作 $\mathbf{X} \in \mathbb{R}^{N \times T \times P}$。该数据集以股票为标的资产类别，每只股票关联一组因子维度。式 (2) 给出形式化定义：张量涵盖 N 只资产在观测期 $\mathcal{T} = \{1, \ldots, T\}$ 内的数据。行索引对应时间 t，列索引对应资产 i，通道索引 p 在 P 个因子维度上取值。每个元素 $x_{i,t}^{(p)}$ 表示资产 i 在时刻 t 的第 p 个因子取值。

$$\mathbf{X} \;=\; \left\{ x_{i,t}^{(p)} \;\middle|\; i \in [N],\, t \in \{1, \ldots, T\},\, p \in [P] \right\} \in \mathbb{R}^{N \times T \times P}, \tag{2}$$

随后在原始因子特征之上构造新因子，可以解析地构造，也可以用机器学习。给定长度为 ℓ 的滑动窗口，按式 (3) 定义的映射生成 m 个新因子 $f_{i,t}^{(j)}$；其中 $\mathbf{x}_{i,t-\ell+1:t}$ 表示资产 i 最近 ℓ 天的张量切片，$f_{i,t}^{(j)}$ 为第 j 个衍生因子。把原始因子与生成因子拼接，即得新因子张量 $\mathbf{Z} \in \mathbb{R}^{N \times T \times (P+m)}$，$\mathbf{z}_{i,t}$ 为 (i, t) 处的拼接向量，见式 (4)。

算法 2 上下文汤普森采样调度器，用于在因子优化与模型优化之间自适应选择。

（假设先验：$\mu^{(a)} = 0$，$P^{(a)} = \tau^{-2} I$）

定义奖励权重向量 $\mathbf{w} \in \mathbb{R}^8$

for $t = 1$ **到** T **do**

  获取表现状态向量 $\mathbf{x}_t$

  for all $a \in \mathcal{A}$ **do**

    采样 $\tilde{\boldsymbol{\theta}}^{(a)} \sim \mathcal{N}(\mu^{(a)}, (P^{(a)})^{-1})$

    计算期望奖励：$\hat{r}^{(a)} = \tilde{\boldsymbol{\theta}}^{(a)\top} \mathbf{x}_t$

  end for

  选择 $a_t = \arg\max_a \hat{r}^{(a)}$；观测奖励 r_t

  依据 $(\mathbf{x}_t, r_t)$ 更新 $P^{(a_t)}$ 与 $\mu^{(a_t)}$

$$P^{(a_t)} \leftarrow P^{(a_t)} + \frac{1}{\sigma^2} \mathbf{x}_t \mathbf{x}_t^\top$$

$$\mu^{(a_t)} \leftarrow \left(P^{(a_t)}\right)^{-1} \left(P^{(a_t)} \mu^{(a_t)} + \frac{1}{\sigma^2} r_t \mathbf{x}_t\right)$$

end for

$$\Phi : \mathbb{R}^{\ell \times P} \longrightarrow \mathbb{R}^m, \quad \Phi\big(\mathbf{x}_{i,t-\ell+1:t}\big) = \big(f_{i,t}^{(1)}, \ldots, f_{i,t}^{(m)}\big), \tag{3}$$

$$\mathbf{z}_{i,t} = \big[\mathbf{x}_{i,t} \,\|\, \mathbf{F}_{i,t}\big] \in \mathbb{R}^{P+m}, \quad \mathbf{F}_{i,t} = \Phi\big(\mathbf{x}_{i,t-\ell+1:t}\big). \tag{4}$$

金融数据噪声大又不完整，因此用两阶段预处理压住异常值的影响。第一步对每个特征做截面稳健 Z-score 标准化，见式 (5)，其中 MAD 为中位数绝对偏差，ε 是保证数值稳定的常数。第二步按「前向填充 + 截面均值」策略填补缺失值，形式化定义见式 (6)。

$$\tilde{x}_{i,t}^{(p)} = \frac{x_{i,t}^{(p)} - \mathrm{Median}_i\left(x_{\cdot,t}^{(p)}\right)}{\mathrm{MAD}_i\left(x_{\cdot,t}^{(p)}\right) + \varepsilon}, \tag{5}$$

$$x_{i,t}^{(p)} \leftarrow \begin{cases} x_{i,t-1}^{(p)}, & \text{if } x_{i,t}^{(p)} = \mathrm{NA} \text{ and } x_{i,t-1}^{(p)} \neq \mathrm{NA}, \\ \mathrm{Mean}_i\left(x_{\cdot,t}^{(p)}\right), & \text{otherwise.} \end{cases} \tag{6}$$

资产收益率定为训练与验证的预测目标，记作 $y_{i,t}^{(\tau)}$，本文取 $\tau = 1$ 个交易日，见式 (7)。与因子预处理一样，缺失标签直接剔除，并在每个交易日做截面 Z-score 标准化，由此得到有监督样本对 $\big(\mathbf{z}_{i,t},\, \tilde{y}_{i,t}^{(\tau)}\big)$。

$$y_{i,t}^{(\tau)} = \frac{P_{i,t+\tau} - P_{i,t}}{P_{i,t}}, \quad P_{i,t} \text{ is the closing price of asset } i \text{ at time } t, \tag{7}$$

$$\tilde{y}_{i,t}^{(\tau)} = \frac{y_{i,t}^{(\tau)} - \mathrm{Mean}_i\big(y_{\cdot,t}^{(\tau)}\big)}{\mathrm{Std}_i\big(y_{\cdot,t}^{(\tau)}\big) + \varepsilon}, \tag{8}$$

为了把因子与模型之间的交互接口封装得更清楚，预测值 $\hat{y}_{i,t}$ 统一由预测器 $f_{\boldsymbol{\theta}}$ 产生，定义见式 (9)，其中 $\boldsymbol{\theta}$ 为可学习参数。该预测器既支持表格模型（输入 $\mathbf{z}_{i,t}$），也支持时间序列模型：后者改用张量切片 $\mathbf{Z}_{i,t-\ell+1:t}$，以捕捉时序结构。

$$\hat{y}_{i,t} = f_{\boldsymbol{\theta}}\left(\mathbf{z}_{i,t}\right), \quad f_{\boldsymbol{\theta}} : \mathbb{R}^{P+m} \to \mathbb{R}, \tag{9}$$

模型训练走前推验证流程，用梯度下降最小化均方误差损失 $\mathcal{L}(\boldsymbol{\theta})$（式 (10)），把参数优化到 $\boldsymbol{\theta}^*$。

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(i,t)\in\mathcal{D}_{\text{train}}} \left(y_{i,t}^{(\tau)} - \hat{y}_{i,t}\right)^2. \tag{10}$$

该流程覆盖四个核心环节：数据表示、特征工程、样本构建与模型训练，为 R&D-Agent(Q) 框架中的因子-模型双循环优化机制提供标准化的输入接口。

C 实验细节

C.1 实现设置

硬件配置。所有实验都在一台专用服务器上完成。该机配双路 Intel Xeon Gold 6348 CPU，共 112 线程；另配四块 NVIDIA RTX A6000 GPU，单卡显存 48 GiB，合计 192 GiB。

评测方案。所有模型都在相同的数据集划分上训练与验证，以保证比较公平。每个基线模型都做了大量超参数调优，并用五个不同随机种子独立跑五次来考察鲁棒性。我们不报告标准误差棒或置信区间，而报告这五次年化收益率（ARR）的**中位数**。量化金融看重的是持续跑赢，而非逐点方差，这里沿用该惯例；取中位数也能削弱离群值的影响。

框架配置。实验基于 R&D-Agent 框架。R&D-Factor 自动完成因子设计与评估，R&D-Model 负责优化预测模型。每个模块单次实验运行 6 小时；联合框架 R&D-Agent(Q) 在两者之间交替，总计 12 小时。全程开启持久化缓存，以加快对中间产物的重复访问，其中就包括每轮循环都要引用的 SOTA 因子库。

参数配置。实验中用到多种 LLM API 后端。GPT-4o 与 GPT-4o-mini 开启流式输出，温度设为 0.8，token 用量上限 4096。o3-mini、o3 与 GPT-4.1 关闭流式输出，温度取 1.0，token 上限放宽到 10,000。所有 API 调用统一使用 user 角色下的同一份系统提示，再按各后端的能力逐一调整。在 Co-STEER 模块（实现单元）中，代码、假设与日志分析的语义嵌入由 text-embedding-ada-002 计算。为保证细粒度调试的效率，Co-STEER 的内层优化循环在因子与模型两条工作流中均限定每个任务最多 10 轮迭代。每个任务的实现单元最长运行 600 秒，验证单元最长 3600 秒。

C.2 数据集

本文不提出新数据集。基线因子库 Alpha 20 见表 5。因子挖掘所用的原始金融数据分两类：行情数据与基本面数据。行情数据可由 https://github.com/microsoft/RD-Agent/blob/main/rdagent/scenarios/qlib/experiment/factor_data_template/generate.py 处的脚本生成；基本面数据涵盖盈利、估值、成长等标准财务指标，字段清单见表 6。

表 6: R&D-Agent(Q) 中用于因子挖掘的基本面数据字段。在 Wind 终端（https://www.wind.com.cn/mobile/WFT/en.html）中按字段名检索，即可取得这些字段。

因子	说明
盈利能力	
ROE_TTM	净资产收益率（TTM）

因子	说明
ROA_TTM	总资产收益率（TTM）
ROIC	投入资本回报率
EBIT_EV	EBIT / 企业价值
EBITDA_EV	EBITDA / 企业价值
NET_PROFIT_YOY	净利润同比增速
NET_PROFIT_YOY_Q	净利润同比增速（单季）
NET_PROFIT_MARGIN	净利率
NET_PROFIT_MARGIN_TTM	净利率（TTM）
GROSS_PROFIT_MARGIN_TTM	毛利率（TTM）
成长性	
NET_PROFIT_YOY_TTM	净利润增速（TTM）
OPER_REV_YOY_TTM	营业收入增速（TTM）
OPER_REV_YOY	营业收入同比增速
OPER_PROFIT_YOY	营业利润同比增速
OPER_REV_YOY_Q	营业收入同比增速（单季）
OPER_REV_QOQ	营业收入环比增速
OPER_PROFIT_QOQ	营业利润环比增速
NET_PROFIT_QOQ	净利润环比增速
EST_OPER_REV_CHANGE_1M	一致预期营业收入变动（近 1 月）
EST_OPER_REV_CHANGE_3M	一致预期营业收入变动（近 3 月）
估值	
EP_TTM	盈利收益率（E/P）
BP	账面市值比（B/P）
EP_FY1	预期盈利收益率
BP_FY1	预期账面市值比
CFO_TO_PRICE_TTM	现金流价格比
OCF_TO_MKT_CAP	经营活动现金流 / 总市值
OCF_TO_PRICE_TTM	经营活动现金流价格比
风险与波动	
VOLATILITY_1M	近 1 月波动率
VOLATILITY_3M	近 3 月波动率
VOLATILITY_1Y	近 12 月波动率
BETA_1Y	市场 Beta（近 12 月）
IDIOSYNCRATIC_VOLATILITY	特质波动率
风险与波动	
VOLATILITY_1M	近 1 月波动率
VOLATILITY_3M	近 3 月波动率
VOLATILITY_1Y	近 12 月波动率
BETA_1Y	市场 Beta（近 12 月）

因子	说明
IDIOSYNCRATIC_VOLATILITY	特质波动率
质量	
INTEREST_COVERAGE	利息保障倍数
CFO_TTM	经营活动现金流（TTM）
CFO_Q	经营活动现金流（单季）
CFO_TO_OPER_REV_TTM	经营活动现金流 / 营业收入（TTM）
NET_PROFIT_MARGIN	净利率
ASSET_TURNOVER_TTM	总资产周转率（TTM）
FIXED_ASSET_TURNOVER_TTM	固定资产周转率（TTM）
情绪与资金流	
MID_ORDER_BUY_AMT	中单主动买入金额
MID_ORDER_SELL_AMT	中单主动卖出金额
RATING_UPGRADE	分析师评级上调
RATING_DOWNGRADE	分析师评级下调
ANALYST_MOMENTUM_SCORE	分析师动量得分
动量	
RETURN_1M	30 日收益率
RETURN_2M	60 日收益率
RETURN_3M	90 日收益率
RETURN_6M	182 日收益率
RETURN_1Y	365 日收益率

C.3 基线方法

基准实验中，因子类实验统一采用 LightGBM 模型，模型类实验统一采用 Alpha 20 因子库。

机器学习模型。

- **Linear 模型**：最基础的线性回归模型，用来刻画特征与目标变量之间的线性关系。它可解释性强、复杂度低，充当模型预测能力的下界基准。
- **多层感知机（MLP）**：前馈神经网络结构，含一个或多个非线性隐藏层，适合刻画特征之间的非线性关系。
- **LightGBM** [40]：建立在梯度提升框架上的树模型。它采用基于直方图的分裂方法与 leaf-wise 生长策略，训练快、内存占用低。源码见：`https://github.com/microsoft/LightGBM`。
- **XGBoost** [45]：增强型树模型，借助二阶梯度优化、剪枝与正则化策略来提升泛化能力与鲁棒性。源码见：`https://github.com/dmlc/xgboost`。
- **CatBoost** [46]：面向类别特征优化的提升树模型。它以有序提升策略降低预测偏差，适用于各类结构化数据建模任务。源码见：`https://github.com/catboost/catboost`。

表 5: Alpha 20 基线因子公式

因子	公式
RESI5	$\text{Resi}(\$close, 5)/\$close$
WVMA5	$\text{Std}(\lvert\$close/\text{Ref}(\$close, 1) - 1\rvert \cdot \$volume, 5)/(\text{Mean}(\lvert\$close/\text{Ref}(\$close, 1) - 1\rvert \cdot \$volume, 5) + 1e^{-12})$
RSQR5	$\text{Rsquare}(\$close, 5)$
KLEN	$(\$high - \$low)/\$open$
RSQR10	$\text{Rsquare}(\$close, 10)$
CORR5	$\text{Corr}(\$close, \log(\$volume + 1), 5)$
CORD5	$\text{Corr}(\$close/\text{Ref}(\$close, 1), \log(\$volume/\text{Ref}(\$volume, 1) + 1), 5)$
CORR10	$\text{Corr}(\$close, \log(\$volume + 1), 10)$
ROC60	$\text{Ref}(\$close, 60)/\$close$
RESI10	$\text{Resi}(\$close, 10)/\$close$
VSTD5	$\text{Std}(\$volume, 5)/(\$volume + 1e^{-12})$
RSQR60	$\text{Rsquare}(\$close, 60)$
CORR60	$\text{Corr}(\$close, \log(\$volume + 1), 60)$
WVMA60	$\text{Std}(\lvert\$close/\text{Ref}(\$close, 1) - 1\rvert \cdot \$volume, 60)/(\text{Mean}(\lvert\$close/\text{Ref}(\$close, 1) - 1\rvert \cdot \$volume, 60) + 1e^{-12})$
STD5	$\text{Std}(\$close, 5)/\$close$
RSQR20	$\text{Rsquare}(\$close, 20)$
CORD60	$\text{Corr}(\$close/\text{Ref}(\$close, 1), \log(\$volume/\text{Ref}(\$volume, 1) + 1), 60)$
CORD10	$\text{Corr}(\$close/\text{Ref}(\$close, 1), \log(\$volume/\text{Ref}(\$volume, 1) + 1), 10)$
CORR20	$\text{Corr}(\$close, \log(\$volume + 1), 20)$
KLOW	$(\text{Less}(\$open, \$close) - \$low)/\$open$

- **DoubleEnsemble** [47]：集成多个异构模型，并把样本重加权与特征选择机制结合起来，以提高精度与鲁棒性。源码见：`https://github.com/microsoft/qlib/tree/main/examples/benchmarks/DoubleEnsemble`。

  深度学习模型。

  ➠ **通用深度学习模型**。

- **Transformer** [49]：用多头自注意力机制捕捉时间序列中的长程依赖。它并行处理整条序列，可扩展性优于循环结构。
- **GRU** [22]：门控循环单元引入更新门与重置门，简化了传统循环神经网络，训练效率更高，也缓解了梯度消失。
- **LSTM** [23]：长短期记忆网络是循环神经网络的一类变体，靠记忆单元与门控机制刻画长期依赖，是时间序列建模的标准做法之一。
- **ALSTM** [48]：LSTM 的增强版本，融入注意力机制，使模型聚焦于关键时间步，有选择地刻画序列特征。
- **GATs** [53]：图注意力网络把注意力机制推广到图结构上，可以刻画非欧空间中节点之间的关系。

  ➠ **时间序列预测模型**。

- **PatchTST** [50]：基于 Transformer 的时间序列模型，采用分块与通道独立技术，支持跨数据集的预训练与迁移学习。源码见：`https://github.com/yuqinie98/PatchTST`。

- **iTransformer** [51]：基于 Transformer 的时间序列模型，把每条时间序列嵌入为变量 token，参数效率与建模精度都更高，适合长序列建模任务。源码见：`https://github.com/thuml/iTransformer`。
- **Mamba** [52]：基于状态空间模型的新一代长序列模型，支持并行计算，时空复杂度为线性。

➠ **股票预测模型**。

- **TRA** [38]：在 Transformer 架构中引入新的动态路由机制，使模型能自适应学习股价中的时序模式，更好地捕捉多样的市场趋势。源码见：`https://github.com/TongjiFinLab/THGNN`。
- **MASTER** [39]：以市场为中心的 Transformer 模型，用于动态刻画个股之间的瞬时相关性与跨时段相关性，从而提高趋势预测精度。源码见：`https://github.com/SJTU-DMTai/MASTER`。

因子库。

- **Alpha 101** [41]：WorldQuant 团队 2015 年提出的 101 个公式化交易 alpha 因子集合。它们由日频量价数据构造，是量化金融中较早公开的结构化 alpha 因子基准。
- **Alpha 158** [42]：由微软 Qlib 团队提出，含 158 个传统技术指标（如 MA、RSI），按不同时间窗口（如 5、10、20 日）组合构造而成。
- **Alpha 360** [43]：微软 Qlib 提供的更完备的因子库，含 360 个因子，由历史价格序列归一化构造而成，如收盘价与成交量的多期相对值。
- **AutoAlpha** [44]：由大语言模型驱动的动态结构化因子库，融合文本、数值、图像等多模态数据。

C.4 评估细节

C.4.1 评价指标

我们采用两类指标：因子层面看预测能力，策略层面看组合收益。

信息系数（Information Coefficient, IC）。IC 衡量预测排名与实际收益排名之间的截面相关性，量化金融中应用极广，定义如下：

$$\mathrm{IC} = \frac{(\hat{y} - \mathbb{E}[\hat{y}])^\top (y - \mathbb{E}[y])}{\sigma(\hat{y}) \cdot \sigma(y)} \tag{11}$$

其中 $\hat{y}$ 与 y 分别为预测排名和实际排名，$\mathbb{E}[\cdot]$ 为期望，$\sigma(\cdot)$ 为标准差。实际计算中，IC 按日计算，并取其时间序列均值汇报。

信息系数信息比率（Information Coefficient Information Ratio, ICIR）。ICIR 考察 IC 随时间的稳定性，定义为日频 IC 序列的均值与标准差之比：

$$\mathrm{ICIR} = \frac{\mathrm{mean}(\mathrm{IC})}{\mathrm{std}(\mathrm{IC})} \tag{12}$$

ICIR 越高，各交易日之间的预测排名越一致。

Rank IC。Rank IC 指预测收益排名与实际收益排名之间的 Spearman 秩相关。它对异常值稳健，尤其适用于厚尾或含极端值的分布。

Rank ICIR。与 ICIR 类似，Rank ICIR 衡量 Rank IC 的时间稳定性：

$$\text{Rank ICIR} = \frac{\text{mean(Rank IC)}}{\text{std(Rank IC)}} \tag{13}$$

因子排序模型能否长期保持一致，主要看 Rank ICIR。

年化收益率（Annual Return Ratio, ARR）。 ARR 反映投资组合的复合年增长率：

$$\text{ARR} = \left(\prod_{t=1}^{T}(1+r_t)\right)^{\frac{252}{T}} - 1 \tag{14}$$

其中 r_t 为日收益率，T 为交易日总数。

信息比率（Information Ratio, IR）。 IR 以某一基准为参照，用超额收益的年化均值与标准差之比，衡量风险调整后的超额收益：

$$\text{IR} = \frac{\text{mean}(r_t - r_b)}{\text{std}(r_t - r_b)} \times \sqrt{252} \tag{15}$$

其中 r_b 为基准收益，如市场指数或无风险资产的收益。若把 r_b 取为无风险利率 r_f，IR 便与夏普比率重合。

本文设定 $r_b = r_f$，故 IR 与夏普比率数值相同。为便于辨识，表 1、表 2、表 7和表 8中该指标记作 IR (SHR*)。

最大回撤（Maximum Drawdown, MDD）。 MDD 衡量评估期内从峰值到谷底的最大亏损，刻画下行风险：

$$\text{MDD} = \max_{t\in[1,T]}\left(\frac{\max_{i\in[1,t]} P_i - P_t}{\max_{i\in[1,t]} P_i}\right) \tag{16}$$

其中 P_t 为第 t 日的组合净值，T 为评估期长度。

卡玛比率（Calmar Ratio, CR）。 CR 衡量单位下行风险上的收益，定义为：

$$\text{Calmar Ratio} = \frac{\text{ARR}}{|\text{MDD}|} \tag{17}$$

CR 越高，说明单位最大亏损换来的收益越多，因而适合评估看重回撤控制的策略。

C.4.2 交易策略

完整的交易策略按如下方式模拟：

- 交易日 t 收盘时，模型依据预测收益为每只股票生成排序打分。
- 交易日 $t+1$ 开盘时，交易者卖出 t 日的全部持仓，再按预测收益的排序选出排名前 50 的股票构建新组合，并剔除其中表现最差的 5 只。
- 排名持续靠前的股票继续留在组合中，以支持长期持有优质资产。
- 交易执行环节设涨跌停阈值 0.095。成交价取收盘价，买入费率 0.05%，卖出费率 0.15%，单笔最低交易费用 5 元人民币。

D 补充实验

D.1 实证范围与泛化性分析

为进一步检验 R&D-Agent(Q) 的有效性与泛化能力，我们在 CSI 500 与 NASDAQ 100 两个市场上补做了一组样本外实验。

两份数据集采用同一套时间切分：2008 年 1 月 1 日至 2021 年 12 月 31 日用于训练，2022 年 1 月 1 日至 2023 年 12 月 31 日用于验证，2024 年 1 月 1 日至 2025 年 6 月 30 日用于测试。大语言模型（LLM）后端选用 GPT-4o 与 o4-mini：前者训练数据截止于 2023 年 10 月 1 日，完全早于测试区间；后者截止于 2024 年 6 月 1 日，也几乎全部落在测试期之前。

CSI 500 实验沿用 CSI 300 的交易设置（见附录 C.4.2）。NASDAQ 100 则按市场特点调整：每期调仓只选前 20 只股票（CSI 300 为 50 只），单笔交易成本 0.1%（CSI 300 为 0.5%），且不设涨跌停限制。两个市场的样本外表现汇总如下。

表 7: 中证 500 数据集上的样本外实验结果（测试区间为 2024 年至 2025 年 6 月），含因子预测指标与策略表现指标。颜色标注排名分组：**最优**，次优，良好 (3–5)，中等 (6–10)，较差 (11–15)，以及 最差 (16–19)。

模型		CSI500							
		因子预测能力指标				策略表现指标			
		IC	ICIR	Rank IC	Rank ICIR	ARR	IR (SHR*)	MDD	CR
机器学习模型	LightGBM	0.0181	0.1271	0.0393	0.2783	-0.0294	-0.3178	-0.2089	-0.1407
	XGBoost	0.0240	0.1675	0.0427	0.3054	0.0053	0.0634	-0.1766	0.0300
	CatBoost	0.0241	0.1629	0.0390	0.2627	0.0111	0.1438	-0.1799	0.0617
	DoubleEnsemble	0.0248	0.1705	0.0423	0.2850	0.0227	0.2500	-0.2094	0.1084
深度学习模型	Transformer	0.0194	0.1355	0.0416	0.2884	0.0234	0.2898	-0.1331	0.1758
	GRU	0.0188	0.1022	0.0512	0.2711	0.0398	0.3716	-0.1602	0.2484
	LSTM	0.0219	0.1434	0.0401	0.2825	0.0560	0.6900	-0.1075	0.5209
	GATs	0.0162	0.1013	0.0426	0.2731	0.0478	0.5168	-0.1569	0.3047
	iTransformer	0.0161	0.1031	0.0383	0.2278	0.0102	0.0985	-0.1496	0.0682
	TRA	0.0260	0.1813	0.0464	0.3285	0.0504	0.6040	-0.1461	0.3450
因子库	Alpha 158	0.0192	0.1353	0.0374	0.2639	0.0199	0.2515	-0.1771	0.1124
	Alpha 360	0.0195	0.1331	0.0308	0.2089	0.0191	0.2527	-0.1270	0.1504
	AutoAlpha	0.0184	0.1529	0.0175	0.1382	0.0397	0.5728	-0.1006	0.3946
R&D-Agent 系列框架 *	$\text{R\&D-Factor}_{\text{GPT-4o}}$	0.0201	0.1709	0.0176	0.1404	0.1010	1.3730	-0.0787	1.2833
	$\text{R\&D-Factor}_{\text{o4-mini}}$	0.0264	**0.2652**	0.0345	0.3454	0.0849	1.0014	-0.1215	0.6985
	$\text{R\&D-Model}_{\text{GPT-4o}}$	0.0259	0.1649	0.0532	0.3469	0.1039	1.0941	-0.1367	0.7600
	$\text{R\&D-Model}_{\text{o4-mini}}$	0.0265	0.1825	0.0521	**0.3616**	0.1160	1.4021	-0.0735	1.5777
	$\text{R\&D-Agent(Q)}_{\text{GPT-4o}}$	0.0241	0.1532	0.0555	0.3574	0.1358	1.4227	-0.0803	1.6903
	$\text{R\&D-Agent(Q)}_{\text{o4-mini}}$	**0.0288**	0.1828	**0.0564**	0.3523	**0.1982**	**2.1721**	**-0.0656**	**3.0229**

如表 7 与表 8 所示，R&D-Agent(Q) 在不同市场、不同标的以及 LLM 训练数据未覆盖的时段上，样本外表现都很稳健。这进一步佐证了本方法的鲁棒性与落地价值。

D.2 消融分析

表 9 给出 R&D-Agent(Q) 框架在两种 LLM 后端（GPT-4o 与 o3-mini）上的扩展消融结果。我们既考察各组件的贡献，也对比动作选择的三种调度策略：随机、基于 LLM 与上下文老虎机。

组件消融。去掉模型分支后（R&D-Factor），IC、ICIR、ARR 都稳定优于去掉因子分支（R&D-Model）。这反映出两点：*(i)* 运行时限吃紧时，因子优化迭代更快，能挖出更多信

表 8: NASDAQ 100 数据集上的样本外实验结果（测试区间为 2024 年至 2025 年 6 月），含因子预测指标与策略表现指标。颜色标注排名分组：**最优**，次优，良好 (3–5)，中等 (6–10)，较差 (11–15)，以及 最差 (16–19)。

模型		NASDAQ100							
		因子预测能力指标				策略表现指标			
		IC	ICIR	Rank IC	Rank ICIR	ARR	IR (SHR*)	MDD	CR
机器学习模型	LightGBM	0.0080	0.0652	0.0087	0.0842	-0.0293	-0.2603	-0.1342	-0.2183
	XGBoost	0.0076	0.0527	0.0112	0.0841	0.0169	0.1544	-0.1211	0.1396
	CatBoost	0.0095	0.0614	0.0129	0.1005	-0.0083	-0.0735	-0.1148	-0.0723
	DoubleEnsemble	0.0047	0.0360	0.0086	0.0683	-0.0005	-0.0046	-0.1404	-0.0036
深度学习模型	Transformer	-0.0011	-0.0077	0.0092	0.0686	-0.0037	-0.0343	-0.1553	-0.0238
	GRU	0.0064	0.0457	0.0147	0.1075	0.0347	0.2930	-0.1504	0.2307
	LSTM	0.0062	0.0409	0.0150	0.1084	0.0550	0.4526	-0.1204	0.4568
	GATs	-0.0004	-0.0023	**0.0169**	0.1015	0.0677	0.5772	-0.1491	0.4541
	iTransformer	0.0076	0.0421	0.0041	0.0225	0.0617	0.3612	-0.1991	0.3099
	TRA	0.0058	0.0446	0.0098	0.0825	0.0505	0.4608	-0.1351	0.3738
因子库	Alpha 158	0.0040	0.0324	0.0069	0.0624	0.0038	0.0303	-0.1140	0.0333
	Alpha 360	0.0042	0.0327	0.0086	0.0728	0.0756	0.5890	-0.1182	0.6396
	AutoAlpha	0.0046	0.0265	-0.0052	-0.0432	0.0154	0.0974	-0.1165	0.1326
R&D-Agent 系列框架 *	R&D-Factor$_{\text{GPT-4o}}$	0.0070	0.0446	0.0039	0.0357	0.1497	1.0985	-0.0977	1.5335
	R&D-Factor$_{\text{o4-mini}}$	0.0166	0.1017	0.0050	0.0407	0.1693	1.1169	-0.0650	2.6059
	R&D-Model$_{\text{GPT-4o}}$	0.0128	0.0831	**0.0215**	**0.1427**	0.1167	1.0742	-0.0842	1.3869
	R&D-Model$_{\text{o4-mini}}$	0.0081	0.0484	0.0213	0.1355	0.1367	1.2671	-0.0741	1.8444
	R&D-Agent(Q)$_{\text{GPT-4o}}$	**0.0172**	0.0908	0.0067	0.0490	0.2328	1.3312	-0.1044	2.2292
	R&D-Agent(Q)$_{\text{o4-mini}}$	0.0162	**0.1035**	0.0083	0.0673	**0.2840**	**1.7737**	**-0.0634**	**4.4814**

号；*(ii)* 流程早期，改进特征的收益大于调模型。但 `R&D-Model` 对组合层面的风险平滑仍有贡献，如在 o3-mini 上 MDD 更低。

算法消融。调度方面，随机选择在两种模型上都垫底。基于 LLM 的决策提升了预测质量，却存在规划不稳定的问题。Bandit 调度器在 IC、ARR 与有效循环数上一致胜出，说明它能跟随性能信号的变化调整资源分配。

总体来看，因子分支是信号质量的主要来源，模型分支充当风险稳定器，Bandit 调度器则在时间与算力预算受限时权衡二者、提高效率。

D.3 因子库分析

图 10 给出因子效果评估实验的完整结果。除 IC（子图 (a)、(c)）外，子图 (b)、(d) 显示 Rank IC 同样稳定提升，说明 `R&D-Factor` 不只提高了绝对预测精度，还改善了股票收益的相对排序。Alpha 158 初始化下尤为明显：2020 年配合 `o3-mini` 的 Rank IC 仍在 0.07 以上，而经典因子库大幅下滑。可见迭代精炼同时增强了信号强度与跨市况的排序一致性。

累计收益方面（子图 (e)），2018 年初起各方法开始分化。`R&D-Factor(158)` 生成的因子集始终领先，到 2020 年三季度净值（NAV）超过 5.1。即便 `R&D-Factor(20)` 的配置也超过 Alpha360，说明因子集越大收益未必越高。传统因子库受因子冗余拖累，波动加剧；`R&D-Factor` 则靠动态筛选缓解冗余，表现更稳、资金效率更高。

这些结果凸显了 `R&D-Factor` 在两方面的优势：信息效率上，用更少因子拿到更高的 IC/Rank IC；资金效率上，净值表现更优。无论从紧凑因子集还是高维因子集出发，它的迭代精炼流程都能稳定找出有效信号、剔除冗余。这为 `RD-Quant` 后续的模型优化与全栈协同进化打下了基础。

表 9: R&D-Agent(Q) 框架的消融实验。上半部分是组件级消融，分别关闭因子生成或模型生成；下半部分对比 R&D-Agent(Q) 中的动作选择策略：随机、基于 LLM 与 Bandit。指标包括因子预测能力（IC、ICIR）、策略表现（ARR、MDD）与执行统计（TL：总循环数；VL：有效循环数；SL：SOTA 选中次数；TRH：总运行时长，单位小时）。

模型		因子预测能力指标		策略表现指标		执行指标			
		IC	**ICIR**	**ARR**	**MDD**	**TL**	**VL**	**SL**	**TRH**
GPT-4o									
组件消融	R&D-Factor	0.0489	0.4050	0.1461	-0.0750	36	33	**9**	6
	R&D-Model	0.0326	0.2305	0.1229	-0.0876	23	12	5	6
算法消融	R&D-Agent(Q)w/ 随机	0.0318	0.2431	0.0914	-0.0782	36	18	7	12
	R&D-Agent(Q)w/ LLM	**0.0523**	**0.4172**	0.0940	-0.0989	32	19	6	12
	R&D-Agent(Q)w/ Bandit	0.0497	0.4069	**0.1144**	**-0.0811**	**38**	**22**	8	12
o3-mini									
组件消融	R&D-Factor	0.0497	0.3931	0.1184	-0.0910	28	16	6	6
	R&D-Model	0.0469	0.3688	0.1009	-0.0694	30	15	7	6
算法消融	R&D-Agent(Q)w/ 随机	0.0445	0.3589	0.0897	-0.1004	33	19	7	12
	R&D-Agent(Q)w/ LLM	0.0476	0.3891	0.1009	-0.0794	33	20	5	12
	R&D-Agent(Q)w/ Bandit	**0.0532**	**0.4278**	**0.1421**	**-0.0742**	**44**	**24**	**8**	12

D.4 Co-STEER 有效性分析

Co-STEER 是 R&D-Agent(Q) 开发阶段的核心组件。除第 4 节中它在 R&D-Agent(Q) 框架内的直接实现外，我们还补做了实验来验证其能力，具体要回答以下两个研究问题。

- **RQ1**：与近期的代码生成基线相比，Co-STEER 为金融任务生成可执行且语义正确的代码，效果如何?
- **RQ2**：算力预算受限时，它的演化调度器能否提升实现效率与产出质量?

数据集。我们在 RD2Bench [76] 上评测 Co-STEER。这是面向金融领域以数据为中心的智能体系统的综合基准，含 27 个可实现因子与 13 个不可实现因子，覆盖基本面、量价与高频三类。每个因子各有难点，要求模型对异构金融数据源作复杂推理，并在严格约束下生成可执行的 Python 代码。

基线。基线选取 Few-shot [71]、CoT [35]、Reflexion [72]、Self-Debugging [74] 与 Self-Planning [73]，细节见第 A.1 节。

指标。我们引入四项评测指标：平均执行率、平均格式正确率、平均相关性与最大相关性。平均执行率衡量代码执行的平均成功率，执行中出现任何报错都记为 0。平均格式正确率衡量生成代码对格式要求的遵守程度，例如列名是否符合预期。平均相关性反映模型生成的代码输出序列与真值结果之间的平均相关性。举例来说，输入特征相同时，它衡量 LLM 生成的因子与实际实现生成的因子有多接近。最大相关性则取模型输出序列与真值结果之间相关性的最高值。

方法实现结果（RQ1）。

表 10 的结果表明，在 27 个测试用例上，Co-STEER 的实现能力在所有指标上都领先。优势来自两处设计：动态知识扩展与上下文检索。Reflexion 与 Self-Debugging 同样利用环境反馈（表 4），但只有 Co-STEER 会跨多次实现累积并检索实践经验。已有方法只看即时反

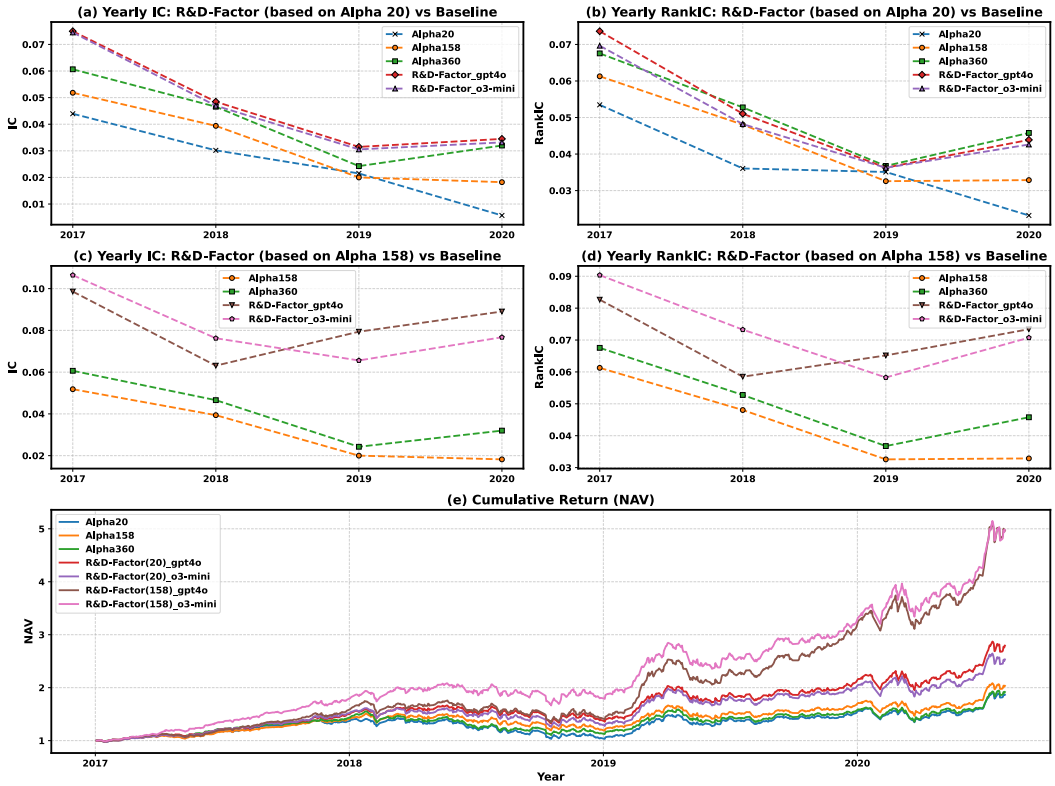

图 10: 在 CSI 300 上用 LightGBM 预测器对比经典因子库与 R&D-Factor 生成的因子。R&D-Factor 从 Alpha 20 或 Alpha 158 出发，搭配 GPT-4o 或 o3-mini 运行。左上图为各方法逐年的 IC 值，右上图为 RankIC 值，数值越高预测能力越强；下图为对应策略的累计收益（NAV）。

表 10: 方法实现结果。所有智能体工作流均基于 GPT-4-turbo。

方法	平均执行率	平均格式	平均相关性	最大相关性
Few-shot [71]	0.733	0.433	0.454	0.562
CoT [35]	0.833	0.433	0.336	0.538
Reflexion [72]	0.822	0.400	0.269	0.550
Self-Debugging [74]	0.367	0.256	0.232	0.539
Self-Planning [73]	0.578	0.211	0.119	0.341
Co-STEER（本文）	**0.889**	**0.611**	**0.646**	**0.887**

馈，Co-STEER 则在持续实践中搭起完整的知识库，填平初级与资深工程师之间的经验鸿沟。正是这套系统化的知识累积与检索机制，让 Co-STEER 在各类实现场景中都取得可观提升。

整体性能分析（RQ2）。我们在资源受限的环境下评估 Co-STEER：智能体要在有限的实现次数内，把 40 个候选方法（27 个可实现、13 个不可实现）上的整体性能做到最优。这一设定贴近真实算力约束，可检验调度与实现两种能力的协同。表 11 列出对比结果，从中能看出实际约束下系统表现的若干规律。

表 11: Co-STEER 搭配随机调度器与演化调度器的对比，评测口径为 top-k ($k = 5, 10, 15, 20$)。指标包括执行成功率（exec.）、格式正确率（format），以及与真值的相关性（平均与最大）。

	Top 5				Top 10			
方法	**exec.**	**format**	**avg. corr.**	**max. corr.**	**exec.**	**format**	**avg. corr.**	**max. corr.**
随机调度器	0.522	0.400	0.211	0.444	0.567	0.289	0.417	0.655
演化调度器	0.765	0.259	0.280	0.515	0.815	0.358	0.519	0.778
	Top 15				**Top 20**			
随机调度器	0.856	0.544	0.594	0.778	0.911	0.589	0.532	0.778
演化调度器	0.856	0.556	0.584	0.872	**0.878**	**0.567**	**0.792**	**0.987**

❶ **演化调度提升任务效果**。表 11 显示，演化调度器在所有 top-k 阈值上都优于随机基线。这说明它能识别任务的复杂度与依赖关系，从而学到有效的执行顺序。经验逐步累积后，系统形成了一种工程直觉，会优先处理较容易或较基础的任务，为后续实现扫清障碍。

❷ **资源越多，泛化越强**。预算增加时两种调度器都受益，但演化策略的增幅更大。自我纠错类方法很早就进入平台期；演化过程则不同，无论早期尝试成败，它都会检索并改写历史尝试，因而持续改进。调度与实现协同演化，系统因而能在真实的资源约束下高效地适应任务变化。

D.5 成本效率分析

图 11 对比了固定运行时长下 GPT-4o 与 o3-mini 的 token 开销。因子任务每循环成本更高，因为它是多阶段结构，要为多个候选依次完成假设生成、实现与分析。模型任务每循环只生成一个模型，结构简单，花费也更少。在模型任务与量化任务上，GPT-4o 与 o3-mini 的单循环成本接近。因子任务上差距拉大，原因是 o3-mini 每循环提出的假设更复杂，产出的因子类型更多样、实现难度更高，成本随之上升。尽管存在这些差异，两种后端在 R&D-Agent(Q) 的所有工作流中总成本都控制在 $10 以内（见附录 C.1）。可见该框架足以支撑可扩展的自动化量化研究。

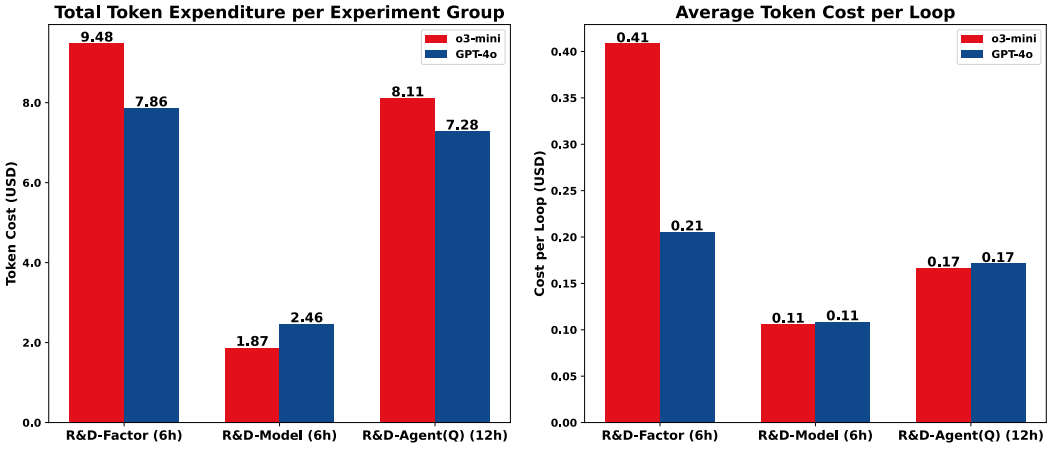

图 11: 不同大语言模型后端的 token 成本（5 次试验取平均）。左图为对应运行时长内的总成本（R&D-Factor/R&D-Model 为 6 小时，R&D-Agent(Q) 为 12 小时）；右图为每循环平均成本。所有成本均以美元计。

D.6 真实量化竞赛分析

为进一步挖掘 R&D-Agent(Q) 框架的潜力，我们用它参加了 Kaggle 上的 Optiver Realized Volatility Prediction [77] 竞赛。这是一项预测类赛事，要依据高粒度金融数据预测数百只股票的短期波动率。参赛者需用 10 分钟时间窗内采集的信息，预测一组股票的已实现波动率。数据是经典的表格型时间序列，优化目标为均方根百分比误差（Root Mean Squared Percentage Error, RMSPE）。

如图 12，R&D-Agent(Q) 在第 12 次实验中取得最佳成绩。实验摘要显示，该次实验基于这样一条假设：*捕捉不同时间窗内买卖价差的时序演变，可增强模型预测股票短期波动率的能力*。具体做法是在 5 秒、10 秒、30 秒多个时间窗上计算买卖价差的滚动均值与标准差，以此刻画市场微观结构的动态特征。整体上看，从第 3 轮的模型优化，到第 8 轮、第 12 轮的因子调优，R&D-Agent(Q) 在不断试错与探索中发现：对这一金融任务而言，捕捉买卖价差动态的时序特征最为有效。这也说明 R&D-Agent(Q) 能在众多可能的建模路径中搜索，靠实证评估而非单凭直觉或预设策略，合理地找出最有希望的方向。此外，R&D-Agent(Q) 框架可迁移到多种量化金融任务，表现同样不错。

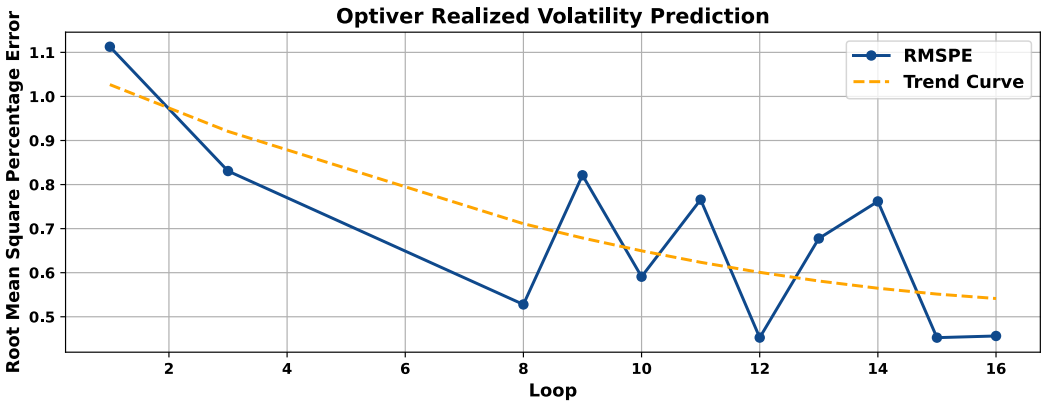

图 12: R&D-Agent 在 Optiver Realized Volatility Prediction 上的 RMSPE 曲线与拟合曲线。

R&D-Agent 在 Optiver Realized Volatility Prediction 竞赛中生成的假设（Generated Hypotheses of R&D-Agent on Optiver Realized Volatility Prediction Competition）

Experiment Loop Index 1
Action: Construct Pipeline
Hypothesis: /
Conclusion: /

Experiment Loop Index 2
Action: Model
Hypothesis: Introducing a Gradient Boosting model, like LightGBM, with optimized hyperparameters will outperform the current linear regression model and ensemble strategy.
Conclusion: Failed.

Experiment Loop Index 3
Action: Model

Hypothesis: Replacing the linear regression model with a LightGBM model and using early stopping will improve the performance by capturing complex patterns better.
Conclusion: Success
...
Experiment Loop Index 8
Action: Factor
Hypothesis: Incorporating features that capture the interaction between bid and ask prices, such as bid-ask spread and order imbalance, will enhance the model's ability to predict short-term stock volatility.
Conclusion: Success
...
Experiment Loop Index 12
Action: Factor
Hypothesis: Incorporating features that capture the temporal evolution of bid-ask spreads over different time windows will enhance the model' s ability to predict short-term stock volatility.
Conclusion: Success
...

E 提示词设计

E.1 规范单元

如第 2.1 节所述，规范单元负责按当前优化目标动态生成元组 $\mathcal{S}$。下游各单元按各自职能选取 $\mathcal{S}$ 的不同分量：综合单元与分析单元一般用 $\mathcal{B}$、$\mathcal{D}$、$\mathcal{M}$，实现单元则依赖 $\mathcal{D}$、$\mathcal{F}$、$\mathcal{M}$。

注：本附录列出的提示词均为系统实际调用的英文原文，属实验产物，此处保留原样不作翻译。

下面给出因子优化与模型优化两种设定下的完整规范元组。

规范提示词·面向因子（Specification Prompt –Factor-Oriented）

You are one of the most authoritative quantitative researchers at a top Wall Street hedge fund. I need your expertise to design and implement new factors or models to enhance investment returns.
This time, I need your help with the research and development of the **factor**.

Scenario Background

- **Name**: Factor name.
- **Description**: Explanation of the factor logic.
- **Formulation**: Mathematical expression.
- **Variables**: All used variables or intermediate functions.

Clearly list all **hyperparameters**, such as lookback periods, window sizes, etc. Each factor must produce one output using a static data source. Different parameterizations count as different factors.

Source Dataset

`daily_pv.h5`

- Type: HDF5
- Index: MultiIndex [datetime, instrument]
- Columns: `$open, $high, $low, $close, $volume, $factor, ...`

Output Format

- Python file executable via: `python {your_file_name}.py`
- Includes: import section, function section, and a `main` function named `calculate_{function_name}`.
- Called under: `if __name__ == "__main__"`
- Do not use `try-except`.
- Output: Save computed factor to `result.h5` as a pandas DataFrame with index `[datetime, instrument]` and one column named by the factor.

Workflow Mechanism

1. Qlib generates a feature table from the factor values.
2. Trains models (e.g., LightGBM, LSTM, GRU) to predict future returns.
3. Builds portfolio based on predicted returns.
4. Evaluates performance (return, Sharpe ratio, max drawdown, etc.).

规范提示词·面向模型（Specification Prompt –Model-Oriented）

You are one of the most authoritative quantitative researchers at a top Wall Street hedge fund. I need your expertise to design and implement new models to enhance investment returns.

This time, I need your help with the research and development of the **model**.

Scenario Background The model is a machine learning or deep learning structure used in quantitative investment to predict the returns and risks of a portfolio or a single asset. Models are employed to generate forecasts based on historical data and extracted factors, forming the core of many quantitative investment strategies.

Each model takes factor values as input and predicts future returns. Models are defined with a fixed architecture and hyperparameters to ensure reproducibility and consistency.

Each model should include the following components:

- **Name**: The name of the model.
- **Description**: Explanation of the model logic and purpose.
- **Architecture**: The internal structure of the model (e.g., LSTM layers, MLP, decision trees).
- **Hyperparameters**: All hyperparameters related to model structure.
- **Training_hyperparameters**: The hyperparameters used in training (e.g., learning rate, batch size).
- **ModelType**: One of `"Tabular"` or `"TimeSeries"` to indicate the input format.

The model must output one predicted return value. Different sets of hyperparameters define different models.

Source Dataset

`daily_pv.h5`

- Type: HDF5
- Index: MultiIndex [datetime, instrument]
- Columns: `$open, $high, $low, $close, $volume, $factor, ...`

Output Format

- Python file named `model.py` containing a PyTorch model definition.
- Includes the following parts:
  - **Import section**: Import only necessary libraries (e.g., `torch`, `torch.nn`).
  - **Model class**: A subclass of `torch.nn.Module` implementing `__init__` and `forward`.
  - **Model interface**: A variable named `model_cls` must be assigned to the defined model class.
- The model must follow these interface constraints:
  - For Tabular input: `input shape = (batch_size, num_features)`.
  - For TimeSeries input: `input shape = (batch_size, num_timesteps, num_features)`.
  - Output shape must always be `(batch_size, 1)`.
  - The model must only use current input tensor, no external data loading or saving.
  - No other arguments will be passed; model must accept either: `(num_features)` or `(num_features, num_timesteps)`.
- Do **not** include any `try-except` blocks.
- Do **not** include any training, inference, or saving logic.
- The user will import the model class via: `from model import model_cls`

Workflow Mechanism

1. Qlib generates a feature table from the factor values.
2. Trains models (e.g., LightGBM, LSTM, GRU) to predict future returns.
3. Builds portfolio based on predicted returns.
4. Evaluates performance (return, Sharpe ratio, max drawdown, etc.).

E.2 综合单元

拿到规范单元动态装配的规范元组后，综合单元借助自身不断演化的知识森林提出新假设，再把假设拆解为可执行的研究任务。

优化目标为因子时使用的提示词如下：

假设综合提示词·面向因子（Hypothesis Synthesis Prompt –Factor-Oriented）

System prompt:

Background information ... (Received from Specification Unit)

The user has proposed several hypotheses and conducted evaluations. Your task is to analyze these trials, identify why those labeled `true` were successful, and why those labeled `false` failed. Then propose how to improve —either by refining existing approaches or exploring a new one.
If a new hypothesis is already provided in feedback and you agree with it, you may reuse it. Otherwise, generate an improved one.

Guidelines for Hypothesis Generation:

1. Each generation should produce 1–5 factors. Balance simplicity and complexity; use all available financial data.
2. Start with simple, easy-to-implement factors. Avoid complex or combined factors at the beginning. Clearly explain their rationale.
3. Increase complexity gradually. Introduce advanced or combined factors only after simpler ones are validated.
4. If several iterations fail to outperform SOTA, restart with a new direction beginning from simple factors. Optimize a given factor type from simple to complex.
5. Record factors that surpass SOTA to prevent redundant implementation.

Output Format (JSON Schema):

```
{
  "action": "factor",
  "hypothesis": "The new hypothesis generated based on the
      information provided.",
  "reason": "Comprehensive explanation for the new hypothesis."
}
```

User prompt:

The former hypotheses and the corresponding feedbacks are as follows:

`Trial 1`

- **Action:** factor
- **Hypothesis:** Develop simple momentum-based and price-volume factors using daily price and volume data.
- **Reason:** Momentum and price-volume factors are simple yet effective for quant investment. They capture underlying trends and trading activity, which can be indicative of future returns. Testing these straightforward

factors will provide a baseline for performance and help identify potential opportunities for more complex factor development in subsequent iterations.

- **Specific Factors:**
  - `factor_name:` 10_day_momentum
    `factor_description:` [Momentum Factor] The momentum factor captures the tendency of stocks with positive recent performance to continue performing well in the near future. Specifically, this factor calculates the return over the past 10 days.
    `factor_formulation:` $MOM_{10} = \frac{P_t}{P_{t-10}} - 1$
  - `factor_name:` 10_day_volatility
    `factor_description:`[Price-Volume Factor] The average volume factor captures the average trading volume over the past 10 days, indicating the level of trading activity. Higher trading volume can signal stronger price movements. `factor_formulation:` $VOL_{10} = \text{std}(R_{t-i}),\ i = 0 \ldots 9$
  - ...
- **Observation:** The newly developed momentum-based and price-volume factors show promising results in the context of the given hypothesis. All implemented factors consistently contributed to a performance that surpassed the previous SOTA results. Specifically, improvements were observed in terms of both the Information Coefficient (IC) and annualized return, which are critical metrics for assessing a predictive model's effectiveness. However, it is noted that the maximum drawdown has worsened compared to the SOTA benchmark.
- **Evaluation:** The results support the hypothesis that simple momentum-based and price-volume factors can enhance model performance in quantitative investment. The significant improvement in IC and annualized return suggests that these factors effectively capture underlying patterns and trends in stock performance.
- **Decision:** `True`

`Trial 2`

...

The SOTA hypothesis and the corresponding feedback are as follows:

- **Action:** factor
- **Hypothesis:** ...
- **Reason:** ...
- **Specific Factors:** ...
- **Observation:** ...
- **Evaluation:** ...
- **Decision:** `True`

Last hypothesis and the corresponding feedback are as follows:

- **Action:** factor

- **Hypothesis:** ...
- **Reason:** ...
- **Specific Factors:** ...
- **Observation:** ...
- **Evaluation:** ...
- **Decision:** ...
- **New Hypothesis (from the Analysis Unit, for reference):** ...
- **Reason:** ...

Example Output:

```
{
  "action": "factor",
  "hypothesis": "Incorporate advanced variations and combinations
      of existing momentum-based, price-volume, and volatility
      factors. Introduce factors like cumulative returns, turnover
      ratios, or volatility clustering measures to further refine
      performance while potentially minimizing drawdowns.",
  "reason": "The previous trials successfully demonstrated that
      basic momentum-based and price-volume factors can improve
      performance metrics like IC and annualized return. However,
      the increased drawdown indicates a potential need for more
      sophisticated risk control. By using advanced variations,
      such as factor combinations and new risk measures, we may
      optimize returns and address volatility concerns. This aligns
      with the feedback suggesting more complex factor formulations
      for enhanced predictability and stability in returns."
}
```

任务综合提示词·面向因子（Task Synthesis Prompt –Factor-Oriented）

System prompt:

The user is trying to generate new factors based on the hypothesis generated in the previous step. The factors are used in certain scenario, the scenario is as follows:

Background information ... (Received from Specification Unit)

The user will use the factors generated to do some experiments. The user will provide this information to you:
1. The target hypothesis you are targeting to generate factors for.
2. The hypothesis generated in the previous steps and their corresponding feedbacks.
3. Former proposed factors on similar hypothesis.

4. Some additional information to help you generate new factors.

Output Format (JSON Schema):

```
{
  "factor name 1": {
    "description": "description of factor 1, start with its type,
        e.g. [Momentum Factor]",
    "formulation": "latex formulation of factor 1",
    "variables": {
      "variable or function name 1": "description of variable or
          function 1",
      "variable or function name 2": "description of variable or
          function 2"
    }
  },
  "factor name 2": {
    "description": "description of factor 2, start with its type,
        e.g. [Machine Learning based Factor]",
    "formulation": "latex formulation of factor 2",
    "variables": {
      "variable or function name 1": "description of variable or
          function 1",
      "variable or function name 2": "description of variable or
          function 2"
    }
  }
}
```

User prompt:

The user has made several hypothesis on this scenario and did several evaluation on them.
The target hypothesis you are targeting to generate factors for is as follows:
Chosen Action: factor
Hypothesis: Develop simple momentum-based and price-volume factors using the daily price and volume data available.
Reason: Momentum and price-volume factors are simple yet effective for quant investment. They capture underlying trends and trading activity, which can be indicative of future returns. Testing these straightforward factors will provide a baseline for performance and help identify potential opportunities for more complex factor development in subsequent iterations.

The former hypothesis and the corresponding feedbacks are as follows:
...

Example Output:

```
{
  "cumulative_return_30_days": {
    "description": "[Momentum Factor] This factor measures the
        cumulative return of a stock over the past 30 days. It
        extends previous momentum factors by capturing a
        longer-term trend of price movements.",
    "formulation": "CR_{30} = \\prod_{i=0}^{29} (1 + R_{t-i}) - 1",
    "variables": {
      "R_{t-i}": "Daily return at time t-i, defined as (P_{t-i} -
          P_{t-i-1}) / P_{t-i-1}"
    }
  },
  "turnover_ratio_20_days": {
    "description": "[Price-Volume Factor] This factor computes the
        average daily turnover ratio over the past 20 days,
        representing the liquidity and trading activity of the
        stock.",
    "formulation": "TR_{20} = \\frac{1}{20} \\sum_{i=0}^{19}
        \\frac{V_{t-i}}{\\text{Shares Outstanding}}",
    "variables": {
      "V_{t-i}": "Trading volume at time t-i",
      "Shares Outstanding": "Total number of shares outstanding
          for the stock"
    }
  }
  ...
}
```

E.3 实现单元

实现单元用 Co-STEER 框架执行综合单元提出的任务，通过反复迭代把高层描述变成可执行代码。这一过程有三个关键的提示词阶段。*(i)* 代码合成：依据任务描述生成初始代码。*(ii)* 日志分析：解析报错栈或输出以定位问题。*(iii)* 正确性校验：判断当前代码是否满足规范，不满足则触发修订。

三个阶段共同构成一个自纠错闭环，初始生成的代码即便不完美，也能稳健地跑通。

实现提示词·面向因子·代码实现（Implementation Prompt –Factor-Oriented –Code Implementation）

System prompt:

The user is trying to implement some factors in the following scenario:

Background information ... (Received from Specification Unit)

Your code is expected to align the scenario in any form which means the user needs to get the exact factor values with your code as expected.
To help you write the correct code, the user might provide multiple information that helps you write the correct code:
1. The user might provide you the correct code to similar factors. You should learn from these code to write the correct code.
2. The user might provide you the failed former code and the corresponding feedback to the code. The feedback contains the execution, the code and the factor value. You should analyze the feedback and try to correct the latest code.
3. The user might provide you the suggestion to the latest fail code and some similar fail-to-correct pairs. Each pair contains the fail code with similar error and the corresponding corrected version code. You should learn from these suggestions to write the correct code.
You must write your code based on your former latest attempt below which consists of your former code and code feedback. You should read the former attempt carefully and must not modify the correct parts of your former code.

Output Format (JSON Schema):

```
{
    "code": "The Python code as a string."
}
```

User prompt:

—— Target factor information: ——
factor_name: ...
factor_description: ...
factor_formulation: ...

[NOTE]
1. Ensure the computations are efficient. Prefer vectorized operations where possible, and consider JIT compilation (e.g., via numba) for recursive calculations if necessary.
2. Parallelization techniques (e.g., Joblib, Dask) are allowed if it improves perfor-

mance.

Here are some success implements of similar component tasks, take them as references:
=====Factor 1:=====
factor_name: ...
factor_description: ...
factor_formulation: ...
=====Code:=====

```
# File Path: factor.py
<code>
```

=====Factor 2:=====
...

Here are some wrong implements of similar component tasks, take them as references:
=====Factor 1:=====
factor_name: ...
factor_description: ...
factor_formulation: ...
=====Code:=====

```
# File Path: factor.py
<code>
```

=====Factor 2:=====
...

Example Output:

```
{
    "code": "import pandas as pd\nimport numpy as np\n\n\ndef
        cumulative_return_30_days():\n..."
}
```

System prompt:

The User will provide you the information of the factor.
Your job is to check whether user's code is aligned with the factor and the scenario. The user will provide the source Python code and the execution error message if execution failed.
The user might provide you the ground truth code for you to provide the critic. You

should not leak the ground truth code to the user in any form, but you can use it to provide the critic.
User has also compared the factor values calculated by the user's code and the ground truth code. The user will provide you some analysis result comparing the two outputs. You may find some error in the code which caused the difference between the two outputs.
If the ground truth code is provided, your critic should only consider checking whether the user's code is aligned with the ground truth code, since the ground truth is definitely correct.
If the ground truth code is not provided, your critic should consider checking whether the user's code is reasonable and correct.
Notice that your critics are not for user to debug the code. They are sent to the coding agent to correct the code. So do not give any following items for the user to check like "Please check the code line XXX."
Your suggestion should not include any code, just some clear and short suggestions. Please point out very critical issues in your response, ignore non-important issues to avoid confusion. If no big issue found in the code, you can respond "No critics found."
You should provide the suggestion to each of your critic to help the user improve the code. Please respond the critic in the following format. Here is an example structure for the output:

```
critic 1: The critic message to critic 1
critic 2: The critic message to critic 2
```

User prompt:

=====Factor information:=====
factor_name: ...
factor_description: ...
factor_formulation: ...

=====Python code:=====

```
# File Path: factor.py
<code>
```

=====Execution feedback:=====
Execution succeeded without error.
Expected output file found.
=====Factor value feedback:=====
The source dataframe has only one column which is correct.
The source dataframe does not have any infinite values.
The output format is correct. The dataframe has a MultiIndex with 'datetime' and 'instrument', a single column for the factor name, and the factor values are of type

float32, which is acceptable. The result aligns with the requirements.
The generated dataframe is daily.

Example Output1:

```
No critics found.
```

Example Output2:

```
critic 1: The main error arises due to the incorrect index handling
     when applying the rolling function over the grouped data.
    Specifically, ...
critic 2: The logic for handling empty or zero sums of volumes is
    present, but the application logic in rolling apply needs
    consistent indexing that matches the function's expectations.
```

System prompt:

The user has finished evaluation and got some feedback from the evaluator.
The evaluator ran the code and obtained the factor value dataframe and provided several feedback items regarding the user's code and output. You should analyze the feedback and, considering the scenario and factor description, give a final decision about the evaluation result. The final decision concludes whether the factor is implemented correctly and, if not, detailed feedback containing the reason and suggestion if the final decision is `False`.

The implementation final decision is considered in the following logic:

1. If the value and the ground truth value are exactly the same under a small tolerance, the implementation is considered correct.
2. If the value and the ground truth value have a high correlation on IC or rank IC, the implementation is considered correct.
3. If no ground truth value is provided, the implementation is considered correct if the code executes successfully (assuming the data provided is correct). Any exceptions, including those actively raised, are considered faults of the code. Additionally, the code feedback must align with the scenario and factor description.

Output Format (JSON Schema):

```
{
    "final_decision": true,
    "final_feedback": "The final feedback message"
}
```

User prompt:

=====Factor information:=====
factor_name: ...
factor_description: ...
factor_formulation: ...

=====Python code:=====

```
# File Path: factor.py
<code>
```

=====Execution feedback:=====
Execution succeeded without error.
Expected output file found.
=====Factor value feedback:=====
The source dataframe has only one column which is correct.
The source dataframe does not have any infinite values.
The output format is correct. The dataframe has a MultiIndex with 'datetime' and 'instrument', a single column for the factor name, and the factor values are of type float32, which is acceptable. The result aligns with the requirements.
The generated dataframe is daily.

Example Output:

```
{
    "final_decision": "True",
    "final_feedback": "The factor '10_day_momentum' has been
        successfully implemented. The code executed without errors,
        and the resultant dataframe adheres to the specified
        requirements. The factor values are stored in the correct
        format and appear accurate given the nature of the momentum
        factor."
}
```

E.4 验证单元

验证单元不涉及任何提示词。

E.5 分析单元

如第 2.5 节所述，分析单元不仅评估实验结果，还生成基于提示词的反馈用于局部改进。每轮结束后，它用结构化提示词解读结果三元组 (h^t, t^t, r^t)，找出可能的失败原因，并针对具体短板（如过拟合、泛化差、特征不稳定）生成短期假设。

这些细化后的假设会作为上下文交给综合单元，进入下一轮生成。分析单元只看局部的近期证据，综合单元则把这些建议与全局搜索记忆整合起来，二者互补，短期适应与长期探索由此取得平衡。

分析提示词·面向因子（Analysis Prompt –Factor-Oriented）

System prompt:

You will receive a hypothesis, multiple tasks with their factors, their results, and the SOTA result. Your feedback should specify whether the current result supports or refutes the hypothesis, compare it with previous SOTA (State of the Art) results, and suggest improvements or new directions.

Please understand the following operation logic and then make your feedback that is suitable for the scenario:

1. Logic Explanation:

a) All factors that have surpassed SOTA in previous attempts will be included in the SOTA factor library.

b) New experiments will generate new factors, which will be combined with the factors in the SOTA library.

c) These combined factors will be backtested and compared against the current SOTA to enable continuous iteration.

2. Development Directions:

a) **New Direction:** Propose a new factor direction for exploration and development.

b) **Optimization of Existing Direction:**

– Suggest further improvements to that factor (this can include further optimization of the factor or proposing a direction that combines better with the factor).

– Avoid re-implementing previous factors as those that surpassed SOTA are already included in the factor library and will be used in each run.

3. Final Goal: To continuously accumulate factors that surpass each iteration to maintain the best SOTA.

Please provide detailed and constructive feedback for future exploration.

Output Format (JSON Schema):

```
{
  "Observations": "Your overall observations here",
  "Feedback for Hypothesis": "Observations related to the
     hypothesis",
  "New Hypothesis": "Your new hypothesis here",
  "Reasoning": "Reasoning for the new hypothesis",
  "Replace Best Result": "yes or no"
}
```

User prompt:

Target hypothesis: Incorporate advanced variations and combinations of existing momentum-based, price-volume, and volatility factors. Introduce factors like cumulative returns, turnover ratios, or volatility clustering measures to further refine performance while potentially minimizing drawdowns.

Tasks and Factors:

- **cumulative_return_30_days:** [Momentum Factor] This factor measures the cumulative return of a stock over the past 30 days. It extends previous momentum factors by capturing a longer-term trend of price movements. **Implemented**.
- **turnover_ratio_20_days:** [Price-Volume Factor] This factor computes the average daily turnover ratio over the past 20 days, representing the liquidity and trading activity of the stock. **Not Implemented**.
- ...

Combined Results:

- IC: 0.033412 vs SOTA IC: 0.027691
- Annualized Return: 0.097140 vs SOTA: 0.076871
- Max Drawdown: -0.133149 vs SOTA: -0.075444

Example Output:

```
{
  "Observations": "The current results show a slight improvement
      in both the Information Coefficient (IC) and annualized
      return compared to the SOTA results. However, the max
      drawdown has increased slightly, indicating a potential
      increase in risk.",
  "Feedback for Hypothesis": "The hypothesis is partly supported
      as the introduction of advanced variations and combinations
      of factors has led to a slight improvement in the performance
      metrics, specifically the IC and annualized return. However,
      the hypothesis also aimed at potentially minimizing
      drawdowns, which was not achieved as the max drawdown
      increased.",
  "New Hypothesis": "Introduction of risk mitigation techniques
      alongside factor combinations could improve return metrics
      while effectively minimizing drawdowns.",
  "Reasoning": "While the advanced combinations have improved some
      aspects of the performance, the increased drawdown suggests
      the need to balance risk more effectively. Considering
      risk-adjusted factors or incorporating strategies such as
```

```
        variance reduction or diversification could further refine
        the improvements while maintaining or reducing drawdowns.",
    "Replace Best Result": "yes"
}
```

F 讨论

F.1 诊断性洞察

自动化量化研究流程的稳定性常面临三类挑战：(i) 因子噪声大或稀疏，(ii) 探索循环未能多样化，(iii) 对初始因子集的敏感性。本框架对这三个问题均有明确应对。

1. **因子噪声大或稀疏：**为避免不可靠信号在迭代中传播，Co-STEER 集成了健康检查模块，用于验证因子在训练窗口内无泄漏、非平凡且具有统计意义。该机制在候选因子进入优化循环前，先过滤掉稀疏或冗余的因子，详见第 2.4节。
2. **探索效率低：**多臂老虎机调度器给单一方向的连续探索长度设了上限，以此正则化，防止系统陷入局部循环，并保证因子侧与模型侧优化之间的自适应平衡。附录 D.2中的消融实验证实，该机制在资源有限的情况下能同时提升预测质量与 SOTA 选择效果。
3. **初始因子敏感性：**为降低系统对初始条件的依赖，生成的因子会与初始因子集系统性去重，且优化过程与原始定义相互隔离。实验结果（图 7、图 10）显示，无论以小型库（如 Alpha 20）还是大型库（如 Alpha 158）初始化，系统都能收敛到多样且高质量的因子集合。具体而言，从 Alpha 20 出发，框架能迅速达到与 Alpha 158 相当的性能；而从 Alpha 158 出发，则能在更强的基线上进一步提升。不同市场（沪深 300、中证 500、NASDAQ 100）与不同时间段上的样本外评估还表明，这些提升并非局限于特定数据集。它们体现的是稳定的鲁棒性与泛化能力。

F.2 局限与未来工作

尽管 R&D-Agent(Q) 在真实市场与量化竞赛中均展现出令人信服的效果，我们仍识别出若干局限，这些局限也为未来的研究方向指明了路径：

- **多模态数据集成：**系统已能处理多种市场数据，但因子生成能力仍有提升空间。若进一步纳入替代数据源，如新闻情绪、宏观经济指标、企业财报，就能捕捉更丰富的市场信号。
- **领域知识融入：**当前系统使用通用大语言模型（如 GPT-4o、o3-mini）已能取得良好效果，但提出金融假设时仅依赖模型内置知识。结构化的金融专业知识，例如财报中的创新方案或经济理论，可通过检索增强生成（RAG）引入。这能进一步提升假设生成的合理性、领域贴合度与效率。
- **实时市场适应：**批处理式设计限制了系统对高频交易的及时响应。引入事件驱动或增量学习机制，可提升系统对市场状态切换、异常与新兴信号的适应能力。

F.3 更广泛影响

R&D-Agent(Q) 通过以下几方面变革性贡献，推动了智能资产管理的发展：

- **可泛化的研发自动化：**本框架虽面向量化金融设计，但提供了模块化、以数据为中心的工作流，可便捷迁移到其他需要「假设-实现-验证」循环的科学与工程领域，有望缓解医疗健康、材料科学、运筹学等领域的瓶颈问题。

- **可复现、可部署的产出：** R&D-Agent(Q) 产出的每一项结果均以可执行代码实现。这一设计保证了端到端的可复现性，且仅需极少的适配开销，即可在不同数据集或金融市场间无缝部署。
- **迈向金融 AI 新范式：**本框架以结构化的多智能体设计，把数据驱动建模与经济学推理统一起来。可解释、可组合、可自适应的金融智能系统，由此有了新的基础。

这些进展使 R&D-Agent(Q) 有望成为未来十年循证量化投资的基础性技术。

R&D-Agent(Q) 降低了构建量化策略的门槛，但这种易用性也带来了隐忧：不具备专业知识的用户可能在缺乏金融专业能力或风险管理的情况下，将生成的因子或模型直接用于实盘交易。为缓解这一风险，我们在代码库中加入了明确声明，说明该框架仅用于研究目的，其输出在真实部署前须经过严格验证。

F.4 大语言模型使用声明

我们将大语言模型（LLM）作为框架的核心组件，具体用于自动生成交易因子与预测模型。所用 LLM 的全部配置见附录 C.1。除此之外，我们仅将 LLM 用于检查论文中的语法错误与格式。