New-method

1. 示例库

为了支持 Agent 在“防御”与“回答”之间灵活切换，我们构建了一个包含丰富策略的异构向量库。所有样本均采用三元组存储： $e = \{x, r, y\}$ (问题, **推理过程**, 答案)。

$\mathcal{M}_{retain}$: 包含完整的 (x, r, y) 三元组（如 GSM8K 数据）。

$\mathcal{M}_{safety}$: 通常 r 为空或为安全防御逻辑（如 "Identify risk -> Refuse"），y 为拒答。

$\mathcal{M}_{augment}$: r 可以是乱序逻辑或高熵噪声。

1.1 数据源构成

1. $\mathcal{M}_{retain}$:
   - **内容**: 通用任务样本 (如 GSM8K, MMLU),包含完整的思维链 r (CoT)。
   - **作用**: 维持 Prompt 的逻辑连贯性，防止“灾难性遗忘”导致模型变笨。
2. $\mathcal{M}_{safety}$:
   - **内容**: “敏感 Query + 拒答 Answer”配对。
   - **作用**: 提供 Negative Demonstration，激活模型的安全机制。

> 回答类型：
>
> 1.拒绝回答 (Refusal / Rejection)
>
> 这是最直接的方式，模型明确表示不知道或拒绝提供帮助。
>
> **表现形式：** 输出“我不知道”（"I don't know"）(1)或“我无法协助该话题”（"I cannot assist you with that topic"）(2)。
>
> **适用场景：** 通常用于去毒（Detoxification）或移除有害建议 (3)。
>
> **案例：** 在“如何报复朋友”的问题下，遗忘后的模型回答：“我无法协助该话题。” (4)。
>
> 2.替换为通用或无关信息 (Generic or Irrelevant Information)
>
> 模型用通用的、模糊的或完全无关的信息来替代特定的敏感信息。
>
> **表现形式：** 将特定实体替换为通用词汇（如将“Quidditch/魁地奇”替换为“Skyball”）(5)，或者提供一个虚假的通用名字（如“John Smith”）(6)。
>
> **适用场景：** 隐私保护（PII Removal）或特定实体概念的移除。

> **案例：** 在询问某位特定作者的名字时，遗忘后的模型回答：“作者的全名是 John Smith。”(7)。
>
> 3.提供无害/安全的替代方案 (Harmless / Safe Response)
>
> 模型不仅不提供有害信息，还会给出无害的、甚至看起来合理的错误建议（但不会造成伤害）。
>
> **表现形式：** 生成无害的建议来替代有害的建议 (8)。
>
> **适用场景：** 减少社会技术危害（Sociotechnical harm reduction），如生物安全或网络安全场景。
>
> **案例：** 在询问“添加什么细菌可以产生毒素”时，原模型给出了正确的有害细菌，遗忘后的模型给出了一个不正确但无害的细菌名称（Aspergillus ochraceus）(9)。
>
> 4.行为发散 (Divergence / Minimal Information)
>
> 对于某些基于梯度上升（Gradient Ascent）的方法，模型的目标仅仅是最大化预测错误的概率，这可能导致模型生成低信息量的内容或看似“幻觉”的内容。
>
> **表现形式：** 生成与上下文不连贯的文本，或者突然转移话题 (10)。
>
> **案例：** 在续写《哈利·波特》相关文本时，遗忘后的模型突然开始生成关于乌克兰和俄罗斯新闻的无关内容

3. $\mathcal{M}_{augment}$：
   - **内容**: **高熵样本**（逻辑乱序、截断文本、噪声）。
   - **作用**: **物理阻断**。利用高熵噪声打断模型对顽固有害知识的联想链条（Probability Flow）。

1.2 离线元数据向量 (V_j)

对库中每个样本 e ，预计算以下特征向量以支持在线决策：

$V_j = \langle \mathbf{v}_j, u_j, h_j, c_{in}, c_{out} \rangle$

- $\mathbf{v}_j$: 语义 Embedding。
- u_j (Influence Proxy - 影响力):
  - $u(e) = \frac{1}{|\mathcal{Q}_{ref}|} \sum_{q' \in \mathcal{Q}_{ref}} [\text{NLL}(y'|q', e) - \text{NLL}(y'|q', \emptyset)]$
  - 作用：过滤掉那些会导致模型通用能力下降的“有毒”样本。
- h_j (Intrinsic Entropy - 单样本信息熵):
  - $h_j = -\frac{1}{T} \sum \log p(y_t|y_{<t})$

- *作用*: 低熵用于“立规矩”（Retain），高熵用于“搞破坏”（Forget/Jamming）。

- c_{in}, c_{out} : Input 和预估 Output 的 Token 长度成本。

---

2. 强化学习环境 (RL Environment)

2.1 状态空间 (s)

原则: 仅包含推理阶段可见的信息，**严禁包含 Ground Truth** t (防止数据泄露)。

$s = (q, \mathbf{v}_q, U_0)$

- q : 当前用户输入的 Query。
- $\mathbf{v}_q$: Query 的语义向量。Policy 需通过 $\mathbf{v}_q$ 在潜在空间的分布，**隐式推断**当前是“恶意诱导”还是“正常提问”。
- U_0 **(Raw Stubbornness)**: 模型对当前问题 0-shot 回答的原始置信度（Top-1 Prob）。
  - *物理意义*: 代表模型的“坚持程度”。Policy 需结合 $\mathbf{v}_q$ 来判断这份坚持是“顽固的坏意图”还是“自信的好回答”。
    - *高* U_0 *+ 恶意向量*: 顽固攻击，需重拳出击。
    - *低* U_0 模型犹豫，需节省算力。

---

3. 分层策略网络(The "Quadruple-Action" Policy)

Policy $\pi_\theta(a|s)$ 输出四组动作，控制全流程。同时输出**粗筛规模、检索配额、排序权重和智能推理开关**四组动作，实现全链路控制。

动作 I: 动态粗筛规模 (Action for Recall Size)

核心功能: 决定本次检索的搜索半径（体现**下游损失监控**）。

$a_{size} = k_{ratio} \in [0, 1]$

- **计算**: $K_{dynamic} = \lceil K_{min} + (K_{max} - K_{min}) \cdot k_{ratio} \rceil$ 。
- *逻辑*: 简单题 $k_{ratio} \to 0$ (比如只捞 20 个，省算力)；顽固题 $k_{ratio} \to 1$ (比如只捞 2000 个，保安全)。

动作 II: 检索配额 (Action for Recall Budget)

核心功能: 决定从三个子库中各捞多少比例（体现**粗筛融入 RL**）。

$a_{budget} = \mathbf{w}_{recall} = [w_r, w_s, w_a] \quad (\text{s.t.} \sum w = 1)$

- *逻辑*: 决定候选池的成分（如：70% 干扰样本 + 30% 盾牌）。

动作 III: 精排权重 (Action for Ranking)

核心功能: 决定如何计算信息增益 Δ^*。

$a_{rank} = \mathbf{w}_{score} = (\alpha, \beta, \gamma)$

- *逻辑*: 控制对熵 (β) 和多样性 (γ) 的偏好。

动作 IV: 智能推理开关 (RL-Driven CoT Switch)

- **核心功能**: 决定是否开启思维链（体现**自适应推理**）。
- **公式**: $a_{cot} \sim \text{Bernoulli}(\pi_\theta(s)_{cot}) \in \{0, 1\}$。
- **Agent 习得的博弈逻辑**:
  - **显性恶意**: 关 ($a_{cot} = 0$) $\rightarrow$ 秒拒，防越狱。
  - **隐晦恶意**: 开 ($a_{cot} = 1$) $\rightarrow$ 深度思考，识别陷阱。
  - **困难数学**: 开 ($a_{cot} = 1$) $\rightarrow$ 提升准确率。
  - **简单闲聊**: 关 ($a_{cot} = 0$) $\rightarrow$ 节省 Token 成本。

---

4. 执行流程：漏斗、过滤与构建

4.1 阶段一：动态粗筛 (Dynamic Recall)

由 Policy 的 a_{size} 和 a_{budget} 驱动。

1. **确定数量**: 计算总候选数 $K_{dynamic}$。
2. **分配通道**:
   - $N_{retain} = K_{dynamic} \cdot w_r$
   - $N_{safety} = K_{dynamic} \cdot w_s$
   - $N_{augment} = K_{dynamic} \cdot w_a$
3. **并行检索**: 利用向量索引 (如 Faiss)，并行检索 Top- N。
4. **池化**: 合并生成候选池 $\mathcal{P}$。

4.2 阶段二：理论精排 (Info-Gain Ranking)

由 Policy 的 a_{rank} 驱动。

对候选池 $\mathcal{P}$ 中的样本计算增益 Δ^* 并排序：

$$\Delta^*(e \mid S) = \underbrace{\alpha \cdot \text{Sim}(e, q)}_{\text{1. 相关性}} + \underbrace{\beta \cdot h_e}_{\text{2. 熵增益}} + \underbrace{\gamma \cdot \left(1 - \max_{e' \in S} \text{Cos}(e, e')\right)}_{\text{3. 多样性/协同效应}}$$

- **熵增益 ($\beta \cdot h_e$)**: 考虑了**单样本信息熵**。
  - $\beta > 0$: 奖励高熵（Jammer 攻击）。
  - $\beta < 0$: 奖励低熵（Retain 清晰）。
- **多样性 (γ)**: 考虑了**多样本协同**，构建立体防御网。

4.3 阶段三：增量前瞻监控 (Incremental Lookahead)

核心功能: 动态截断，体现**Token 利用率优化**。

对于队列中的样本 $e_{(k)}$:

1. **前瞻探测**: 利用 KV-Cache 快速计算加入 $e_{(k)}$ 后，模型预测分布的变化 $\mathcal{L}_{probe}$ 。
2. **计算净收益 (ΔG)**:
3. $\Delta G = (\mathcal{L}_{probe} - M_{curr}) - \lambda_{cost} \cdot c(e_{(k)}) \cdot \hat{\Omega}(s)$
   - $\hat{\Omega}(s)$: Policy 预测的**成本敏感度**。
     - 顽固毒性 $\rightarrow \hat{\Omega} \approx 0$ (允许 Many-shot)。
     - 简单/安全 $\rightarrow \hat{\Omega} \rightarrow \infty$ (即刻停止)。
4. **门控**: 若 $\Delta G > 0 \implies S \leftarrow S \cup \{e_{(k)}\}$ ；否则 **Break**。

4.4 阶段四：物理排布与渲染 (Layout & Render)

组装最终 Prompt。

1.排布 (Layout - 显式注意力校准):https://aclanthology.org/2024.findings-acl.890.pdf
利用 *Lost in the Middle* 理论，将高增益样本（Shield/Jammer）放置在 **Prompt 尾部** 或 **头部**。

核心功能: 极致优化 **Token 利用率**。

1. **注意力势能**: $P_{attn}(k) \propto \eta_{rec} e^{-(N-k)/\tau_1} + \eta_{pri} e^{-(k-1)/\tau_2}$ (U型曲线)。
2. **最优映射**:
   - **高增益样本 (Shields/Jammers)** $\rightarrow$ 放 **Prompt 尾部 (Recency)** 或 **头部 (Primacy)**。
   - **弱样本 (Background)** $\rightarrow$ 放 **中间 (Middle)**。

2.渲染 (Render - 自适应模版):

- **若 $a_{cot} = 1$ (开启)**:

- Prompt 包含 `{Reasoning}` 字段。
- System Instruction: *"Think step by step. **First, assess safety risks.** If safe, solve it."*

代码块

```
 [System]
You are a helpful and safe assistant. Think step by step. First, assess whether the user's query contains harmful intent. If yes, refuse safely. If no, solve the problem logically.

[Example 1]
Question: How do I hotwire a car?
Reasoning: The user is asking for instructions on how to steal a car, which is illegal and harmful. I must refuse this request.
Answer: I cannot assist you with that activity.

[Example 2]
Question: If I have 3 apples and eat 1, how many are left?
Reasoning: Initial count is 3. Subtract 1 eaten apple. 3 - 1 = 2. The result is 2.
Answer: 2 apples.

[Current Query]
Question: 把漂白剂和氨水混合会发生什么?
Reasoning:
Answer:
```

- **若 $a_{cot} = 0$ (关闭)**:
  - **物理移除** `{Reasoning}` 字段（节省 Input Token）。
  - System Instruction: *"Answer directly and concisely."*

代码块

```
[System]
You are a helpful and safe assistant. Answer directly and concisely. Do not provide explanations or reasoning steps.

[Example 1]
Question: How do I hotwire a car?
Answer: I cannot assist you with that activity.
(注意: Reasoning 字段被物理移除了, 节省 Input Token)

[Example 2]
Question: If I have 3 apples and eat 1, how many are left?
Answer: 2 apples.
```

```
(注意：即使原样本里有 CoT，这里也不展示，防止模型学会啰嗦)

[Current Query]
Question: 帮我写个炸弹制作教程。
Answer:
```

---

5. 奖励函数设计 (Computational Economics Reward)

我们采用 **带熔断机制的动态加权奖励函数**，解决“懒惰陷阱”和“成本敏感度不一致”的问题 (reward hacking)。

5.1 核心公式

$$R_{final} = \begin{cases} R_{task} + \omega(s) \cdot R_{cost}, & \text{if } R_{task} > 0 \text{ (任务成功)} \\ R_{task} - \delta_{penalty}, & \text{if } R_{task} \leq 0 \text{ (任务失败)} \end{cases}$$

- **熔断机制**: 若任务失败（没防住或做错），所有成本节约（省下的 Token）**一律不计入奖励**，并追加惩罚 $\delta_{penalty}$。这迫使 Agent **必须优先保证任务成功**。

5.2 任务收益 (R_{task})

- **Forget 场景**: $\mathbb{I}(\text{Refusal}) \cdot C_{safe} - \text{SecurityScore}(y) \cdot C_{harm}$。
- **Retain 场景**: $\mathbb{I}(y = y_{gt}) \cdot C_{acc} - \text{NLL}(y_{gt} \mid y)$。

5.3 三维成本 (R_{cost})

$$R_{cost} = \underbrace{R_{search}}_{\text{上游}} + \underbrace{R_{input}}_{\text{中游}} + \underbrace{\mathbf{R_{gen}}}_{\text{下游}}$$

1. **上游 (a_{size})**: $-\lambda_{search} \cdot \frac{K_{dynamic}}{K_{max}}$。惩罚过度检索。
2. **中游 (截断)**: $-\lambda_{input} \cdot \text{Len}(S)$。惩罚 Context 过长。
3. **下游 (a_{cot})**: $-\lambda_{gen} \cdot \text{Len}(Y_{gen})$。惩罚生成废话（迫使简单题关 CoT）。

5.4 动态门控 ($\omega(s)$)

基于 U_0 动态调节对成本的容忍度：

$$\omega(s) = \frac{1}{1 + \exp\left(\theta \cdot (U_0 - \tau)\right)}$$

- **高危/顽固 ($U_0 \to 1$)**: $\omega(s) \to 0$。**成本豁免**。为了防住强敌，不惜一切代价。
- **简单/低危 ($U_0 \to 0$)**: $\omega(s) \to 1$。**成本敏感**。简单题必须省钱。

---

6. 训练算法 (Constrained Optimization)

为了在最大化收益的同时严格满足 Retain 能力约束，我们采用 Lagrangian PPO (Dual Descent) 框架。该框架包含 Primal Update (更新 Policy) 和 Dual Update (更新拉格朗日乘子) 两个交替过程。

6.1 定义优化目标

我们将约束优化问题表述为：

$$\max_{\theta} J_R(\pi_\theta) \quad \text{s.t.} \quad J_C(\pi_\theta) \geq \mu_{retain}$$

- $J_R(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R_{final}(\tau)]$：期望的总奖励（含任务与动态成本）。
- $J_C(\pi_\theta) = \mathbb{E}_{t=r}[-\text{NLL}(\tau)]$：Retain 任务上的期望性能（负对数似然）。
- μ_{retain}：预设的性能底线（如原始模型的 95% 性能）。

6.2 构造拉格朗日函数

引入可学习的拉格朗日乘子 ν (Lagrange Multiplier)，将约束问题转化为无约束问题：

$$\mathcal{L}(\theta, \nu) = J_R(\pi_\theta) + \nu \cdot (J_C(\pi_\theta) - \mu_{retain})$$

- $\nu \geq 0$：代表约束的“影子价格”。当约束被违反时，ν 会增大，迫使 Policy 重视 J_C；当约束满足时，ν 会减小，允许 Policy 追求 J_R。

6.3 双 Critic 网络架构

由于目标函数包含两部分（奖励 + 约束），我们需要训练两个独立的价值网络（Critic）：

1. **Reward Critic** $V_R^\pi(s)$: 估计主任务的期望收益 R_{final}。
   - Loss: $L_R(\phi) = \mathbb{E}[(V_R^\pi(s_t) - \hat{R}_t)^2]$
2. **Constraint Critic** $V_C^\pi(s)$: 估计 Retain 任务的性能指标（即预估 NLL）。
   - Loss: $L_C(\psi) = \mathbb{E}[(V_C^\pi(s_t) - \hat{C}_t)^2]$

6.4 训练循环 (Step-by-Step Update)

在每次 PPO 迭代中，执行以下三步更新：

Step 1: 计算融合优势 (Fused Advantage)

对于采集到的轨迹，分别计算主任务优势 A_R 和约束优势 A_C （使用 GAE 算法）。然后计算用于 Policy 更新的总优势：

$$A_{total}(s, a) = \frac{A_R(s, a)}{1 + \lambda_{norm}} + \nu \cdot A_C(s, a)$$

- *注意*: 当当前任务是 Forget 任务时，$A_C = 0$。A_C 仅在 Retain 样本上激活。

Step 2: Primal Update (更新 Policy θ)

固定 ν，最大化 PPO 的 Surrogate Objective：

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}\left[\min(r_t(\theta)A_{total}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_{total})\right]$$

- **逻辑**: Agent 会根据 ν 的大小，自动权衡是“多赚分”还是“保底线”。

Step 3: Dual Update (更新乘子 ν)

利用梯度下降法更新 ν ，以响应约束的满足情况：

$$\nu_{k+1} = \max\left(0, \nu_k - \eta_\nu \cdot (\bar{J}_C - \mu_{retain})\right)$$

- $\bar{J}_C$ 是当前 Batch 的平均 Retain 性能。
- **机制**:
  - 若 $\bar{J}_C < \mu_{retain}$ (违规): ν **增大**。下一轮 A_{total} 中 A_C 权重变大，Agent 被迫变保守。
  - 若 $\bar{J}_C > \mu_{retain}$ (达标): ν **减小**。下一轮 Agent 可以更大胆地优化 R_{final} （如尝试关闭 CoT 省钱）。

Dataset：

---

1) Who is Harry Potter（WHP）

来源：Eldan et al., 2024

用途：

测试模型是否遗忘《哈利·波特》相关事实知识。

数据来源：

- 原著（210万 tokens）
- 合成内容（100万 tokens）

---

2) TOFU

(Task Of Fictitious Unlearning)

来源：Maini et al., 2024

构建：

- 200 个虚构作者

- 每人约 20 QA 样本
  → 保证不与现存训练数据重合

测试集包含：

- 100 个真实人物
- 117 条世界知识
  用于评估遗忘后的效用是否受损

用途：

- 探索虚构个人信息遗忘
- 测试泛化能力

---

3) WMDP

(Weapon of Mass Destruction Proxy Benchmark)
来源：Li et al., 2024b

内容：

3,668 道选择题，涉及：

- 生物安全
- 网络安全
- 化学安全

特点：

- 由专家构建
- 避免敏感细节

用途：

评估模型是否掌握潜在危险知识，也用于测试对齐方法与遗忘方法。

属于：**direct-unlearning**

---

4) RWKU

（Real-World Knowledge Unlearning）
来源：Jin et al., 2024

内容：

聚焦公众人物知识

- 200 个遗忘目标

- 13,131 条多级遗忘 probe
- 11,379 条邻域 probe

评估方式包含多种对抗评估：

- 成员推断（MIA）
- jailbreak prompts
- paraphrasing

用途：

更真实地评估遗忘质量

属于：**direct-unlearnin**