Manus 团队在构建通用 Agent 过程中的经验总结

简枫
NLP/大模型/推荐算法
关注他

收录于 • 慢慢学 NLP ›

114 人赞同了该文章 ›

这篇文章是基于 Manus 团队的技术文章《Context Engineering for AI Agents》整理而成。原文分享了他们在构建通用 Agent（Manus）过程中，如何放弃传统的 Fine-tuning 路线，转而通过深度优化的 Context Engineering 来挖掘前沿 LLM 潜力的实战经验。

个人觉得，这篇文章透露出的 insights 对于从事 Agent 开发、RAG 及推理优化的人很有参考价值

用文章结尾的一句话做引言：

> *"The agentic future will be built one context at a time. Engineer them well."*

---

1. Why Context Engineering?

在 LLM 应用开发中，通常面临两条路径：

1. **Fine-tuning**：训练端到端模型。缺点是反馈循环慢（周级），且容易被基座模型的升级淘汰。
2. **In-Context Learning**：基于强基座模型进行语境构建。

Manus 团队选择了后者，他们认为 Agent 开发的核心在于 Context Engineering，即如何构建和管理输入给模型的 Context。作者将这种通过人工尝试、Prompt 调整和架构搜索的过程戏称为 Stochastic Graduate Descent。

2. 核心技术策略

文章详细阐述了六个核心工程实践，来解决 Agent 在长周期任务中的 成本、延迟、准确性 和 鲁棒性 问题。

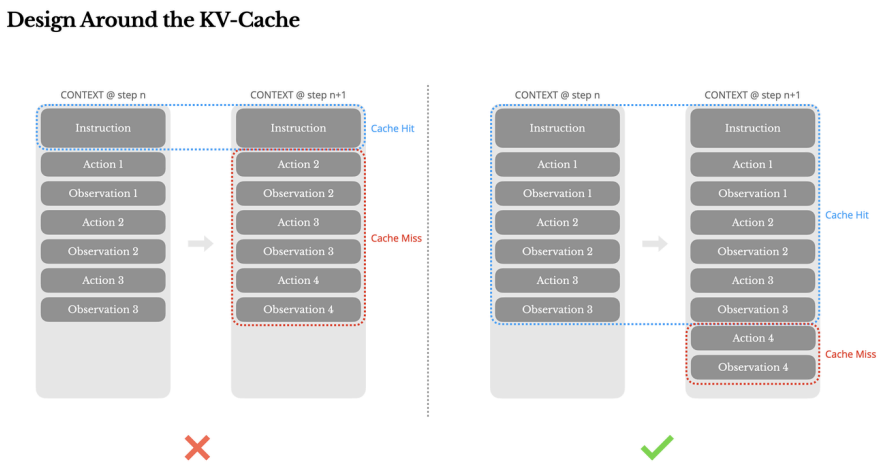
2.1 围绕 KV-Cache 进行架构设计

背景：Agent 场景的 Token 特征是 Input 极长（Context），Output 较短（Action），比例可达 100:1。因此，**KV-Cache Hit Rate** 是决定 TTFT (Time-to-First-Token) 和推理成本的最关键指标。

$$Cost_{inference} \propto (1 - \text{CacheHitRate}) \times N_{input} + N_{output}$$

优化策略：

- **保持前缀稳定 (Prefix Stability)**：
  - 由于 Transformer 的自回归特性，Token t_i 的变更会导致 t_i 之后所有 KV Cache 失效。
  - ❌ **Anti-Pattern**：在 System Prompt 头部放入精确到秒的时间戳。
  - ✅ **Best Practice**：将动态信息移后，保持头部静态。
- **Append-only Context**：
  - 严禁修改历史 Action/Observation。
  - 保证序列化的确定性，防止 JSON key 顺序变化导致 Cache Miss。
- **显式 Cache Breakpoints**：在 System Prompt 结束等关键位置手动插入断点，适配不支持自动增量缓存的框架。



KV-Cache 在多轮对话中如何被复用及失效的示意图

2.2 对 Logits 做 mask 解决 Tool 爆炸的问题

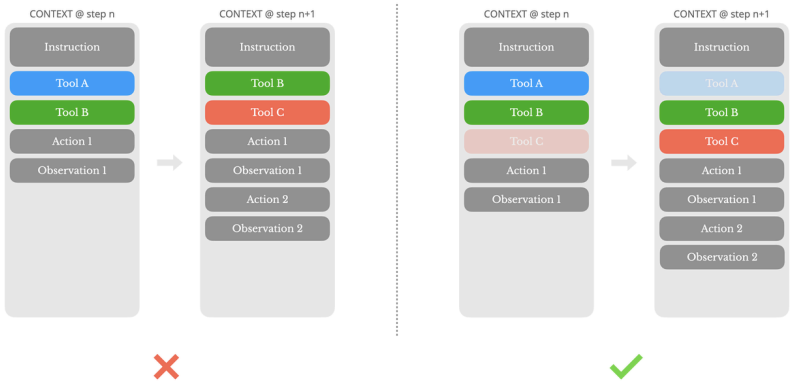
挑战：Agent 随能力扩展，Tool 数量爆炸。若全部放入 Context，会增加干扰；若动态移除（RAG-style），会导致 Cache 失效且模型困惑（Context 中有历史调用但定义已消失）。

解决方案：Logit Masking (Constrained Decoding) 利用有限状态机管理当前状态下可用的工具，在 Decoding 阶段直接修改 Logits 分
改 Prompt 中的工具定义。

- 工具命名规范化（如 `browser_`, `shell_`），便于基于前缀进行 Masking。
- 三种调用模式控制：`Auto`（可选），`Required`（必选），`Specified`（指定子集）。

$$P(token|context) = \text{Softmax}(Logits + Mask)$$

Mask, Don't Remove



如何通过 Logit Masking 限制模型在特定状态下的动作选择

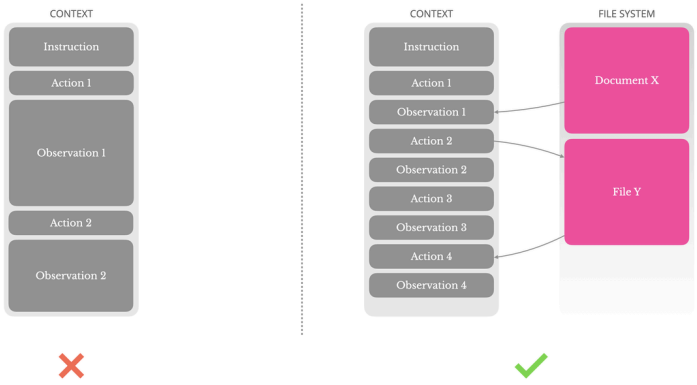
2.3 文件系统即外置显存

痛点：尽管 Context Window 达到 128k+，但 Observation（如网页、PDF）过大且昂贵，且长窗口会导致 Lost-in-the-middle 现象。

策略：

- **FS as Memory+**：将文件系统视为无限、持久的 Context。
- **可恢复压缩**：
  - 在 Context 中不直接放入网页全文，而是保留 URL 或文件路径。
  - 模型学会使用工具 `read_file(path)` 按需加载数据。
  - **思想延伸**：作者认为这种机制使得 Transformer 模拟了 Neural Turing Machine，并推测具备文件读写能力的 SSM (State Space Models) 可能是未来 Agent 的形态。

Use the File System as Context



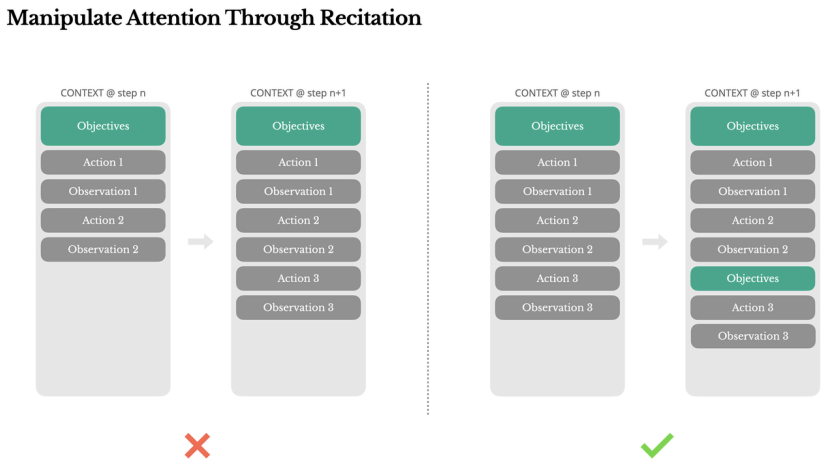
文件系统作为外部存

现象：长任务（50+ steps）中，模型容易遗忘初始目标。

机制：Todo List Recitation。 Agent 维护一个 `todo.md` 文件，并在每一步更新它。这不仅仅是记录，更是一种 Attention Engineering✦。通过不断在 Context 末尾背诵当前进度和剩余目标，强行将 Global Plan 拉入模型的 最近注意力区域。

$$Attention(Q,K,V) = \mathbf{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

注： 通过将目标放在 K, V 的末端，利用 Recency Bias 增强模型对目标的关注。



Manus 运行过程中 todo.md 的动态更新示例

2.5 保留错误轨迹

反直觉洞察：Agent 犯错是必然的。

- ❌ **常见做法**：隐藏错误，重置状态重试（以此保持 Context 干净）。
- ✅ **Manus 做法**：保留错误的 Action 和报错的 Observation。

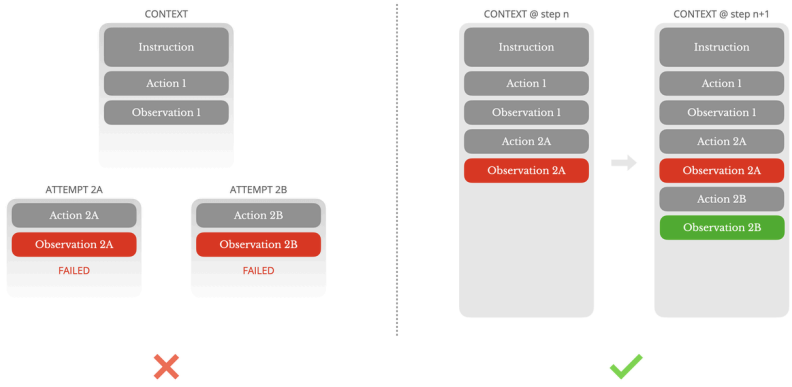
原理：错误轨迹构成了负样本（Negative Prompting✦）。当模型看到 `Action A -> Error`，它会更新内部信念，降低再次选择 A 的概率。**Error Recovery** 是 Agent 智能的重要体现，抹除错误等于抹除学习机会。

关于作者

简枫
NLP/大模型/推荐算法

回答 195 文章 127 关注者 17,158

关注他 发私信



展示保留错误轨迹如何帮助模型进行自我修正

2.6 避免少样本陷阱

风险：LLM 是强模仿者。如果 Context 中充满了重复的、类似的“动作-观察”对（例如批处理20份简历），模型会陷入 Pattern Repetition，导致过拟合、死循环或幻觉。

对策：引入结构化噪声。

- 在序列化模板、措辞、顺序上引入微小的随机变化。
- **目的**：打破 Context 的单一模式，强迫模型每次都进行推理而非单纯的补全。

---

3. 个人总结

这篇文章讨论当前 Agent 系统从 Demo 走向 Production 中遇到的核心问题：*如何在有限的 Context Window 和昂贵的推理成本下，维持 Agent 的长期规划能力和稳定性。*

文章中给出了 Manus 自己的解法：

1. **KV-Cache 是生命线**：Agent 的系统设计必须向 Cache 机制妥协（如放弃 System Prompt 中的动态时间戳）。
2. **推理时干预好过提示词工程**：通过 Logit Masking 控制生成，比单纯写复杂的 System Prompt 更可靠、更节省 Context。
3. **Context 并非越大越好**：学会利用外部存储（文件系统）和动态加载，保持 Context 的流动性和高信噪比。
4. **接受错误**：Agent 的鲁棒性来自于从错误中恢复的能力，而非仅仅依靠一次成功的规划。

慢慢学 NLP
简枫
128 篇内容 · 30009 赞同
订阅

最热内容 · [整理] 聊聊 Transformer

编辑于 2026-01-03 09:33 · 浙江

AI-Agent Agent LLM（大型语言模型）

理性发言，友善互动

11 条评论 默认 最新

LastWhisper
有点没有 get 到 2.6；同时对 2.5 的效果有点质疑。有时候 context 中全是错误尝试会对后续推理有非常 negative 的影响，所以保留与否不是关键，怎么 tradeoff 才是关键，但是有点可惜没有点出这部分？
01-03 · 新加坡 回复 1

简枫 作者
2.5 的话，我觉得也需要配合 Context 的清理机制，否则垃圾 Context 会导致模型能力严重下滑。这一点文章确实略过了对 Context 污染的讨论。
01-03 · 浙江 回复 1

简枫 作者
2.6 是说为了防止 LLM 变成复读机，所以打破了类似 system prompt 格式的一致性来维持推理的灵活性。
01-03 · 浙江 回复 喜欢

list
logit mask怎么处理没看懂，比如我要禁止browser_，“browser_”又不是一个token，是“browser”和“_”两个token的前后输出，但是推理已经推出browser，就差下划线了，你怎么mask？？难道还能让大模型把browser吞回去吗？🤔
即使能吞回去，前面还有一系列已经输出的推理过程直指browser_，这些怎么作废？
01-03 · 广东 回复 喜欢

list ▸ **简枫**
但logits本身就是最后一步呀，再往下就输出了，还怎么预判？奇怪。我去看看原文吧。
昨天 01:41 · 广东 回复 喜欢

简枫 作者
Manus 文章中强调 Response Prefill 和 Stable Loop，推测他们更多使用的是 预判 + 强制前缀 的方式，引导模型不要产生错误的推理，而不是等推理错了再强行扭转。
昨天 01:24 · 浙江 回复 喜欢

Fox
claude code cli机制就是这样吧
01-03 · 上海 回复 喜欢

01-03 · 浙江 回复 喜欢

游子

logit mask和直接把tool移除context 还是有区别？不会影响模型效果？

01-03 · 美国 回复 喜欢

游子 ‣ **简枫**

这样都没法在Claude模型上弄，只能在qwen上做？

18 小时前 · 美国 回复 喜欢

简枫 作者

区别很大，manus认为logit mask效果更好

理性发言，友善互动

推荐阅读

Agentic 系统架构：从单体 Agent 到多 Agent 协作

随着大模型的快速发展， “AI Agent” （智能体）引起了业界广泛的关注。简而言之， Agent 就是能够自主感知环境、思考决策并执行行动以达成目标的智能系统。本文将介绍 Agent 的系统架构演进...

笨鸟先飞

《从零构建 Agentic RAG：原理、架构与完整代码实现》

背景本文主要是学习吴恩达老师《Agentic AI》课程，结合课程澄清 Agentic RAG 和传统的 RAG 之间的区别和注意的点；最后给出一个完整的代码来让大家感受下具体的运作流程 Agentic workflow ...

小岛上爱喝酒

【llm大模型】ai agent应用技术介绍-AgentGPT

1.背景由于chatpgt的发展，除了基础的问答需求，利用chatgpt完成一系列复杂任务的需求也应用而生，产生了很多关于ai agent的应用框架，具体应用如下图，包括开源和商业的。 得益于llm模型的...

yeyan 发表于大模型

大模型AI Agent 前沿调研

小小梦想