Manus项目经验：上下文工程精华内容

2025年7月21日　1 分钟阅读

#Context Engineer　#Manus　#项目经验

> *本文介绍了Manus项目经验的上下文工程精华内容,总结为6个问题和解决方案。*

以下是基于Manus项目经验的上下文工程精华内容,总结为6个问题和解决方案。 参考来自Manus的blog (July 19, 2025)： Context Engineering for AI Agents: Lessons from Building Manus

问题1：如何提升AI代理在生产环境下的KV缓存命中率?

解决的方法论：
围绕上下文的稳定性和可预测性进行设计，最大化前缀复用，减少缓存失效，从而降低延迟与推理成本。

具体解决方案：

保持Prompt前缀高度稳定：系统提示（system prompt）必须避免动态内容（如精确到秒的时间戳），哪怕一个token的差异也会导致缓存失效。

上下文采用追加模式（Append-only）：所有历史动作和观察都只追加，不做修改，保证上下文序列化的确定性（如JSON序列化时保证key顺序一致）。

显式标记缓存断点：对于不支持自动增量缓存的框架，需手动在上下文中插入断点，至少覆盖系统提示的结尾，防止缓存过期导致全量失效。

分布式一致性：自托管模型时（如vLLM），确保开启前缀缓存，并用session ID等手段保证请求路由一致，提升缓存利用率。

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#Context Engineer　#Manus

问题2：如何应对代理能力扩展带来的动作空间爆炸和工具管理混乱？

解决的方法论：
通过“掩蔽”而非“移除”机制，利用上下文感知状态机动态约束模型的动作选择，保证系统稳定和缓存效率。

具体解决方案：

避免动态增删工具：工具定义一般位于上下文前部，变动会导致后续全部缓存失效，并让模型混淆（引用已不存在的工具）。

引入上下文感知状态机：不移除工具，而是在解码阶段通过掩蔽token logits，动态限制或强制模型只能选择某些动作。

动作命名规范化：统一工具前缀（如browser_、shell_），便于状态机掩蔽和分组管理。

利用响应预填充：结合推理框架的response prefill能力，三种模式（自动、强制、指定）灵活约束函数调用，提升动作选择的准确性和效率。

问题3：如何突破上下文窗口限制，实现大规模信息的高效管理与持久记忆？

解决的方法论：
将文件系统作为外部化、结构化的代理记忆体，实现信息的无限扩展、可恢复压缩与高效读写。

具体解决方案：

文件系统即上下文：让模型学会按需读写文件，将关键数据存储在文件系统中，释放上下文窗口压力。

可恢复压缩策略：如网页内容可只保留URL，文档内容可通过路径索引，既缩短输入长度，又可随时恢复详细信息。

结构化外部记忆：文件系统不仅是存储，更是代理可以主动操作和调用的长期记忆，实现高效信息管理。

问题4：如何防止LLM在长任务中遗忘目标或偏离主题？

解决的方法论：
通过“朗读”机制，将全局计划不断推送到上下文末尾，利用自然语言主动操纵模型注意力，防止“中间丢失”。

具体解决方案：

动态维护todo清单：代理在执行复杂任务时持续更新如todo.md的文件，每步都将目标和进展写入上下文最新部分。

自然语言自我提醒：通过不断“朗读”目标和计划，把全局意图推送到模型的短时记忆范围内，减少目标漂移和遗忘。

问题5：如何让代理从错误中自我修正，提升鲁棒性？

解决的方法论：
保留错误路径和失败信息，让模型能观察并自适应，减少重复犯错的概率。

具体解决方案：

完整记录失败与异常：每次失败的动作和返回的错误、堆栈追踪都保留在上下文中，不做清理。

利用错误反馈自我更新：模型通过看到历史失败案例，调整行为倾向，逐步降低同类错误的发生率。

问题6：如何避免上下文“少样本陷阱”，提升模型决策多样性？

解决的方法论：
引入结构化多样性和微量随机性，打破单一模式，防止模型陷入机械模仿。

具体解决方案：

多样化动作与观察模板：不同的序列化格式、措辞、顺序和微小格式噪声，避免上下文高度同质化。

受控随机性：通过有意识地制造小幅变化，激发模型关注点的转移，提升整体任务表现和鲁棒性。

分享这篇文章