上下文工程

从“PROMPT”到“CONTEXT”

一份面向AI工程师的权威指南，旨在构建可靠、高效、可扩展的智能体。

技术三部-杨晨辉

为什么我们需要关心“上下文”?

您是否曾感觉AI像一个“健忘”的同事？聊了十几轮后，它忘记了最初的目标；处理长文档时，它遗漏了关键的约束条件。

这些问题的元凶，并非模型不够聪明，而是其核心限制——上下文窗口 (Context Window) 的管理不善。

上下文窗口 (Context Window) 是大语言模型能够一次性处理的最大信息量。它的大小直接影响模型的表现，过小会导致信息丢失，过大则会增加计算成本和延迟。

AI的“健忘症”正在耗费我们的时间和金钱。

核心定义：到底什么是“上下文工程”？

> *“上下文工程 (Context Engineering) 是一门‘艺术’与‘科学’，其核心目标是在AI智能体执行任务的每一步中，都有策略地、系统性地构建和管理提供给大语言模型（LLM）的信息输入流（即‘上下文’），以最大化其性能、效率和可靠性。”*

它涉及四个相互关联的阶段：

一个更生动的比喻：智能的“内存管理器”

我们的任务就是扮演高效的“操作系统”，智能地管理信息进出LLM这个有限的“内存”。

上一页 下一页

忽视上下文工程的昂贵代价

超出窗口限制

Token总数超限，API报错，任务中断。例如：分析长篇工程招标文件的AI在处理到第80页时，因上下文被占满而无法继续。

成本与延迟飙升

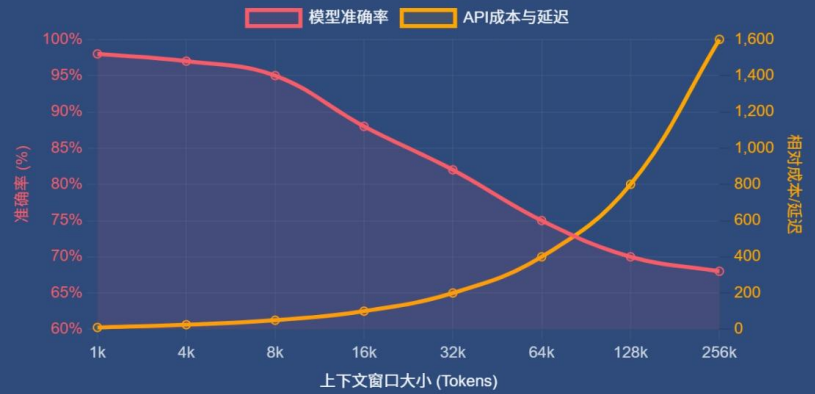
上下文越大，处理时间越长，API费用越高。例如：在线客服AI每次都发送50轮完整对话历史，导致延迟和费用剧增。

性能严重下降

“中间遗忘”现象导致AI忽略关键信息。例如：通过AI订票时“必须靠窗”的约束被淹没，AI最终预订了过道座位。

“中间遗忘”现象

研究表明，随着上下文长度增加，模型在处理位于信息“中间”部分时的准确率会显著下降。



图表显示：随着上下文窗口（Tokens）的增大，模型准确率下降，同时API成本和延迟急剧上升。

AI智能体的“双刃剑”：自主性与上下文危机

AI智能体通过一个“思考 → 行动 → 观察”的ReAct循环来主动完成任务。

每一次循环，新的“思考”、“行动”和“观察结果”都会被追加到上下文末尾，导致上下文不可逆地增长，我们称之为“上下文爆炸”。

上下文危机的四种典型症状

如果不加管理，不断膨胀的上下文很快就会变成一场灾难。

中毒 (Poisoning)

错误信息污染决策，导致任务走向错误方向。

干扰 (Distraction)

无关噪音使模型分心，忽略关键约束。

混淆 (Confusion)

技术细节导致输出机械，缺乏人性化。

冲突 (Clash)

矛盾指令导致决策瘫痪或错误选择。

策略一：写入 (Write)

构建智能体的“外部大脑”，对抗遗忘。

暂存区 (Scratchpad)

当前单个任务的“状态追踪器”，记录“正在做什么”和“发现了什么”。

- 计划分解与当前步骤
- 中间思考与决策
- 关键工具调用结果

记忆 (Memory)

跨任务、跨会话的长期“知识库”，让智能体从经验中学习。

- 用户画像与偏好
- 可复用解决方案与代码片段
- 项目复盘与最佳实践

挑战： 全面性与数据管理（异构性、噪音）及隐私风险的平衡。分布式处理范式是克服这些挑战的关键。

策略二：选择 (Select)

精准调取信息，实现精准打击，对抗干扰。

从记忆中选择 (RAG)

使用智能体的当前需求作为查询，从庞大的记忆库中搜索最相关的信息片段，为LLM提供精准、即时的指导。

工具选择

当智能体拥有海量工具时，用RAG的方式，帮它“选择”出当前最可能用到的几个工具，降低模型被“混淆”的概率。

关键洞察： 建模方法存在“复杂性与表达性”的逆向关系，需选择“恰好足够复杂”的范式。建模与推理阶段深度交织，应协同设计。

策略三：压缩 (Compress)

为上下文“瘦身减负”，对抗成本和延迟。

上下文总结

使用LLM自身的能力，将一段长文本（如对话历史、网页内容）提炼成一段简短的摘要，用更少的Token承载同样多的核心含义。

上下文裁剪

通过更直接的规则或模型，过滤和“修剪”掉上下文中不那么重要的部分，如丢弃最旧的对话历史或无关的闲聊。

关键洞察： 推理技术存在“透明性与预测能力”的权衡。上下文是动态变化的，推理是一个持续的重新评估和完善过程，需要系统持续监控和主动适应。

策略四：隔离 (Isolate)

“分而治之”的架构智慧，对抗复杂性。

多智能体架构

将一个宏大的任务，分解成多个子任务，交给一个由“管理者”和多个“专家”组成的智能体团队来协同完成，从根本上避免“上下文干扰”和“上下文混淆”。

沙盒环境 (Sandboxing)

为智能体的某些操作（尤其是代码执行）提供一个隔离的“沙盒”环境。智能体可以把复杂的、状态繁多的工作“外包”给沙盒，自己只关心最终的简洁结果。

关键洞察：若控制不当，适应可能带来意外风险（如隐私泄露、操纵行为）。系统适应会产生新的上下文信息，形成连续反馈循环，使上下文工程成为动态迭代过程。

核心要点回顾：我们的“四件套”工具箱

这四大策略共同构成了上下文工程的强大工具箱。

策略	做什么	为什么
写入 (Write)	建立外部大脑	对抗遗忘
选择 (Select)	精准调取信息	对抗干扰
压缩 (Compress)	减少Token消耗	对抗成本和延迟
隔离 (Isolate)	拆分任务与环境	对抗复杂性

最终目标：将AI应用从“玩具”转变为可靠、可扩展、可维护的强大“工具”。

下一步行动与未来展望

下一步行动

1. **养成“上下文思维”习惯：**设计时主动用四大策略进行安全检查。
2. **从“小实验”开始：**为现有应用增加策略，获得成功经验。
3. **共建团队“模式库”：**分享技巧和架构，加速团队成长。

未来展望

超长上下文窗口并非“银弹”，它让成本、延迟和“中间遗忘”问题更突出，因此更需要上下文工程。

- **上下文工程的“自动化”：**元智能体动态管理上下文。
- **从“文本”到“结构化上下文”：**实现更精确可靠的控制。
- **预测与主动适应：**系统将预测未来状态并主动调整。
- **多学科领域融合：**与数据治理、AI伦理、安全协议深度结合。

谢谢大家！

接下来是演示环节。

上下文工程，是释放大语言模型全部潜力的钥匙。

让我们一起，构建出真正智能、强大的AI应用。