

Context Engineering

上下文工程 —— 从提示词工程到上下文装配

2026年10月11日 | AI Agent 知识分享 | 第73期

一、一句话概括

上下文工程 (Context Engineering) 不再问「这句话怎么写得更好」，而是问「这一次推理，Agent 该看到什么、不该看到什么」。它把系统指令、工具定义、检索知识、会话历史、长期记忆当作一份有限的注意力预算来统一装配，目标是在预算内放入最小的高信号 token 集合。

二、核心原理

>> 提示词工程与上下文工程的分界

提示词工程优化单条指令的措辞，回答「怎么写」；上下文工程设计每次推理的完整输入装配，回答「放什么」。关键前提：模型的注意力是有限预算，token 越多，单个 token 分到的注意力权重越被稀释，哪怕远没到上下文窗口上限。

>> 上下文的四个来源，要分别装配

指令上下文：system prompt、角色设定、约束、输出格式。

知识上下文：RAG 检索片段、文档、数据库查询结果。

工具上下文：工具名、描述、参数 schema —— 最容易被忽视的膨胀源。

记忆与历史上下文：会话历史、长期记忆、任务中间状态。

>> 两种装配策略

预加载 (pre-load)：启动时把所有可能用到的信息一次性塞进去。实现简单，但 token 浪费大，极易触发上下文腐烂。

即时检索 (just-in-time)：只给 Agent 轻量标识（文件路径、查询语句、工具名），需要时再取。这是目前主流 Agent 框架推荐的路线，代价是增加一次工具调用往返。

>> 上下文腐烂 (Context Rot)

上下文越长，模型对中间位置信息的召回率越低 (Lost in the Middle 现象)。大量低相关 token 会拉低对关键信息的注意力。典型症状：任务准确率随无关上下文增长而下降，即使总长度远未触顶。

上下文工程的本质：注意力预算的分配问题，不是文字问题

先做消融删减，再考虑压缩 —— 压缩有损，删减无损

工具 schema 是隐形 token 黑洞，30 个工具约 6000 token/次

评估必须跑长会话衰减，不能只看单轮准确率

三、测试实战

测试一：上下文腐烂曲线 (Context Rot Curve)

◆ 目的：量化无关上下文对任务准确率的影响，找出可用长度拐点。

1. 准备一个固定任务，例如「从 500 笔交易流水中找出全部可疑笔」
2. 准备 1 份必要信息 + 5 档不同长度的无关背景 (0 / 2K / 8K / 32K / 100K token)
3. 同一模型、同一参数 (temperature=0)，每档跑 20 次
4. 记录准确率，画「准确率 vs 上下文长度」曲线

指标：准确率衰减斜率 (每增加 10K 无关 token 掉几个百分点)、拐点位置

工具：promptfoo (<https://promptfoo.dev>)、OpenAI Evals、自建 Python 脚本

测试二：工具描述膨胀实验

◆ 目的：验证工具数量与描述长度对工具选择准确率的影响。

1. 构造 5 / 15 / 30 / 50 个工具，其中只有一个能真正完成任务
2. 每档给 30 条自然语言需求，让 Agent 选择工具
3. 人工标注「正确工具」作为标准答案
4. 统计错误调用、参数填错、单次请求 token 数

指标：工具选择准确率 (Top-1)、错误工具调用率、参数填错率、单次调用 token 数

经验拐点通常在 20-30 个工具之间，超过后应改为「工具分组 + 两阶段检索」

测试三：大海捞针 (Needle in a Haystack)

◆ 目的：验证长上下文中间位置的信息召回能力。

1. 在长上下文里埋一条关键信息，例如「本次转账限额 5 万」
2. 埋放位置分别取 0% / 25% / 50% / 75% / 100%
3. 提问该信息，记录是否被正确召回

指标：按位置分组的召回率，重点观察 50% 中间位置是否为最低点

工具：Needle in a Haystack (https://github.com/gkamradt/LLMTest_NeedleInAHaystack)、RULER (<https://github.com/NVIDIA/RULER>)

测试四：上下文消融 (Ablation)

◆ 目的：量化每块上下文的价值，找出白占 token 的部分。

1. 从完整上下文中逐块删除：删工具描述、删历史、删检索片段、删示例
2. 每次只删一块，跑同一批用例
3. 对比基线准确率

指标：准确率下降幅度即该块上下文的价值；下降不明显的块直接精简

四、常见坑点

坑点1：把上下文工程等同于压缩历史

压缩是最后手段。真正的问题是「一开始就不该塞进去」。正确顺序是先做消融，删掉低价值块，再考虑压缩。压缩有损，细节一旦丢失无法恢复。

坑点2：工具 schema 是隐形 token 黑洞

30 个工具、每个 schema 200 token，就是 6000 token，每次请求都要付，而且工具越多模型选错概率越高。建议按场景分组加载，或先检索工具、再加载 schema。

坑点3：只用单轮准确率评估，忽略长会话衰减

Agent 是多轮的，第 1 轮与第 20 轮的准确率可能差很多（历史累积 + 上下文腐烂）。评估必须跑「同一会话连续 20 轮」的衰减测试，而不是 20 个互相独立的单轮请求。

五、延伸阅读

- Anthropic 《Effective context engineering for AI agents》: <https://www.anthropic.com/engineering/effective-context-engineering>
- Lost in the Middle 论文: <https://arxiv.org/abs/2307.03172>
- promptfoo 评测工具: <https://promptfoo.dev>
- RULER 长上下文基准: <https://github.com/NVIDIA/RULER>
- Needle in a Haystack 测试: https://github.com/gkamradt/LLMTest_NeedleInAHaystack