大语言模型工程化：稳定性、可验证性与安全架构的深度研究报告

执行摘要

随着大语言模型（LLM）从实验室环境迈向大规模工业化应用，工程范式正经历着一场深刻的变革。早期的“提示词工程（Prompt Engineering）”——一种依赖直觉、试错和经验主义的技艺，正在被更严谨的计算机科学原则所取代。当前的研发重心已从单纯追求模型能力的上限（Capability），转向了确保模型行为的可靠性（Reliability）、可验证性（Verifiability）和安全性（Security）。

本报告旨在为开源项目的“05_总结与展望”部分提供详尽的理论支撑与实证数据。通过对近期五十余篇学术论文（arXiv 2024-2025）、技术文档（OpenAI, Anthropic）及安全指南（NCSC, OWASP）的系统性梳理，我们识别出当前LLM应用开发面临的四大核心挑战与机遇：提示词漂移与脆弱性（**Prompt Drift & Brittleness**）、结构化输出与受限解码（**Structured Outputs & Constrained Decoding**）、可验证评测（**Verifiable Evaluation**）以及指令优先级与安全边界（**Instruction Hierarchy & Security Boundaries**）。

分析显示，提示词漂移已成为阻碍LLM进入高风险领域的主要障碍，微小的格式变动可能导致性能的剧烈波动 [1]。为应对这一挑战，工业界正在采用结构化输出技术，通过上下文无关文法（CFG）和有限状态机（FSM）强制模型生成符合预定义Schema的数据，从而在概率模型之上构建确定性接口 [3]。与此同时，评测体系正经历从主观的“LLM-as-a-Judge”向可验证评测的范式转移，强调对原子化指令遵循情况的客观验证 [4]。在安全领域，提示词注入被重新定义为一种架构层面的漏洞而非简单的输入过滤问题，**指令层级（Instruction Hierarchy）**理论的提出为解决这一问题提供了新的防御纵深 [6]。

第一章 提示词漂移与脆弱性分析：从“炼金术”到量化工程

在LLM的应用落地过程中，开发者普遍面临一个棘手的问题：模型性能缺乏稳定性。即便是一个经过精心优化的提示词（Prompt），在面对细微的输入变化、格式调整或模型版本更新时，其表现也可能急剧下降。这种现象被称为“提示词脆弱性（Prompt Brittleness）”或“提示词漂移（Prompt Drift）”。

1.1 定义与现象学分析

提示词脆弱性（Prompt Brittleness）指的是LLM对输入中非语义（Non-semantic）变化的过度敏感性。研究表明，仅改变提示词中的标点符号（例如将双冒号::改为单冒号:）、调整少样本（Few-shot）示例的顺序、或是改变空白符的数量，都可能导致模型输出准确率的显著波动 [2]。例如，Mizrahi等人的研究指出，在某些任务中，仅仅因为提示格式的微小差异，同一模型的准确率波动幅度可达40%以上 [8]。这种现象揭示了LLM在某种程度上是在“过拟合”提示词的表面特征，而

非真正鲁棒地理解任务意图。

提示词漂移（Prompt Drift）则更多指代时间维度上的不稳定性。这主要源于两个因素：

1. 模型迭代：当模型提供商更新其API后端（例如从 gpt-4-0613 更新至 gpt-4-0125）时，底层的概率分布发生漂移。旧版本中表现优异的Prompt在新版本中可能失效，导致系统行为不可预测 [9]。
2. 推理随机性：即便将温度（Temperature）设为0，由于GPU浮点运算的非确定性，长上下文生成的累积误差也可能导致输出结果在不同运行次间发生“漂移” [10]。

1.2 敏感性的量化度量：Spread与AIFD

为了将这一问题从“玄学”转化为工程问题，学术界提出了一系列量化指标。

1.2.1 格式敏感度（Format Spread）

Sclar等人（2024）及Mizrahi等人（2024）引入了“Spread（离散度）”的概念来衡量Prompt的鲁棒性。其计算方法如下：

1. 利用自动化工具（如FormatSpread）基于同一语义任务生成 N 种不同的格式变体（例如改变分隔符、大小写、缩进等）。
2. 在相同模型上测试这 N 个变体的性能。
3. 计算Spread = Max(Accuracy) - Min(Accuracy)。
   Spread值越大，表明模型对格式越敏感，其在实际生产环境中的表现越不可靠（因为用户输入的格式往往是不可控的）8。

1.2.2 对抗性指令遵循难度（Adversarial Instruction-Following Difficulty, AIFD）

Wang等人 [11] 提出了一种基于对抗攻击的鲁棒性度量方法。通过对原始Prompt进行对抗性扰动（Adversarial Perturbation），例如引入拼写错误或同义词替换，来观察模型性能的下降幅度。AIFD指标不仅衡量模型是否“听懂”了指令，更衡量了模型在面对噪声干扰时是否还能“坚持”指令。研究发现，许多在标准基准测试中得分高的模型，在AIFD测试中表现惨淡，暴露出其指令遵循能力的脆弱性。

1.3 缓解策略：算法干预与混合格式

针对提示词漂移与脆弱性，工业界和学术界已经从早期的“人工微调”转向了算法级的解决方案。

1.3.1 敏感度感知解码（Sensitivity-Aware Decoding, SAD）

SAD是一种在推理阶段（Inference-time）的干预技术 [1]。其核心思想是惩罚那些对输入扰动高度敏感的Token预测。

- 机制：在生成每个Token时，系统会进行多次前向传播（Forward Pass）。在这些额外的Pass中，输入的Prompt会被施加随机的扰动（如Token替换或掩码 ）。
- 决策逻辑：如果某个Token的生成概率在输入被扰动后大幅下降，说明该Token的生成依赖于Prompt的特定表面特征（即它是“脆弱”的）。SAD算法会在最终的Logits中减去该Token的敏

感度得分，从而倾向于选择那些无论输入如何微调都保持稳定的Token。

- 代价与收益：虽然SAD能显著提升输出的稳定性，但其计算成本随扰动次数线性增加，因此更适合对稳定性要求极高的离线批处理任务，而非实时交互场景[1]。

1.3.2 混合格式训练（**Mixture of Formats, MOF**）

针对少样本提示（Few-shot Prompting）中的过拟合问题，8 和 2 提出了混合格式（MOF）策略。传统的Few-shot通常保持示例格式的高度一致性（如统一使用 Q:... A:...）。然而，这种一致性反而诱导模型学习格式而非任务逻辑。MOF策略主张在Context Window中故意混用不同的格式风格（例如，第一个示例用 User: / Assistant:，第二个示例用 Input: / Output:）。

- 实证结果：在Llama-2-13b上的实验显示，MOF策略能够将Spread指标降低46%，显著提升了模型对未知格式的泛化能力。这表明，通过打破格式的单一性，迫使模型关注语义内核，是解决脆弱性的有效手段[2]。

1.3.3 自动化提示优化（**Optimization by PROmpting, OPRO**）

鉴于人工设计Prompt的局限性，OPRO[12] 等方法将Prompt视为一个可优化的参数。通过构建一个元提示（Meta-prompt），让LLM基于历史的评测结果迭代生成新的Prompt变体。这一过程类似于梯度下降，旨在高维的Prompt空间中寻找一个“平坦的极小值（Flat Minima）”，使得该Prompt在面对输入扰动时依然能保持较高的性能。

第二章 从概率到确定性：结构化输出与受限解码工程

在Agent与工具调用（Tool Use）日益普及的背景下，LLM的输出不再仅仅是给人阅读的文本，而是作为下游软件系统的输入（如API参数、数据库查询）。这就要求LLM必须具备**结构化输出（Structured Output）**的能力。传统的“Vibe-based”方法（即在Prompt中恳求模型“请输出JSON格式”）已无法满足工业级软件对类型安全（Type Safety）的严苛要求。

2.1 概率生成的内在缺陷

从本质上讲，LLM是一个概率预测机。在每一步生成中，它都会计算词表中所有Token（通常数万个）的概率分布。即便我们在Prompt中明确要求“只输出数字”，词表中的字母或标点符号仍可能获得非零的概率。

- 长尾风险：在生成数百万次调用的过程中，低概率事件必然发生。一旦模型生成了一个非法的JSON字符（如遗漏了闭合括号，或在不该出现逗号的地方加了逗号），下游的解析器（Parser）就会崩溃，导致整个Agent流程中断[3]。
- 幻觉与类型错误：除了语法错误，模型还可能产生语义类型的错误，例如将字符串填入整型字段，或虚构枚举值中不存在的选项[3]。

2.2 受限解码（**Constrained Decoding**）：技术原理

为了解决上述问题，**受限解码（Constrained Decoding）**技术应运而生。它不再寄希望于模型

“自觉”遵守规则，而是通过数学手段强制模型仅在合法的解空间内采样。

2.2.1 Logit掩码与状态机

受限解码的核心机制是在采样（Sampling）步骤之前介入，对Logits进行动态掩码（Masking）[13]。

1. **Schema**编译：开发者提供的Schema（如JSON Schema、Pydantic Model或正则表达式）首先被编译器转换为一个确定性有限状态自动机（DFA）或下推自动机（PDA）。
2. 状态追踪：解码器维护当前生成序列在自动机中的状态。
3. 动态掩码：在生成下一个Token之前，解码器查询自动机：从当前状态出发，哪些Token能构成合法的转移？
   - 合法**Token**：保留其原始Logits。
   - 非法**Token**：将其Logits设为 $-\infty$。
4. 结果：经过Softmax归一化后，非法Token的概率变为0。模型在数学上不可能生成不符合Schema的字符。

2.2.2 上下文无关文法（CFG）与递归结构

早期的受限解码主要基于正则表达式（Regex）和有限状态机（FSM）。然而，OpenAI在2024年推出的“Structured Outputs”功能中，强调了**上下文无关文法（CFG）**的重要性 [3]。

- **FSM**的局限：FSM难以处理递归嵌套结构（如JSON中的对象嵌套对象，列表嵌套列表）。
- **CFG**的优势：CFG支持递归定义，能够精确描述任意深度的JSON结构。OpenAI通过将JSON Schema转换为CFG，不仅解决了括号匹配问题，还能处理复杂的$ref引用和递归类型，实现了真正意义上的100%格式依从性 [3]。

2.3 主流框架与算法创新

2.3.1 Outlines与Domino算法

开源库 **Outlines** 是受限解码领域的先驱之一。其团队提出的 **Domino** 算法解决了受限解码中的一个关键痛点：Token错位（Token Misalignment）[15]。

- 问题：Token边界往往不与语法边界重合（例如JSON的键名可能被切分为多个Token）。
- **Domino**方案：利用预计算（Pre-computation）和推测性解码（Speculative Decoding）技术，Domino能够低延迟地验证Token序列的合法性。在某些场景下，由于可以直接跳过无效的搜索路径，启用Domino的受限解码速度甚至超过了无约束生成的推理速度 [15]。

2.3.2 JSONSchemaBench与性能权衡

微软团队推出的 **JSONSchemaBench** [17] 揭示了受限解码的现状。该基准测试涵盖了从简单到极复杂的各类JSON Schema。

- 发现：虽然受限解码保证了语法正确性，但它引入了推理延迟（Latency Overhead）。
- 质量权衡：过度严格的约束可能会限制模型的“思维链（Chain of Thought）”。如果强制模型直接输出最终JSON而不允许其先生成推理文本，可能会导致逻辑推理能力的下降。因此，最佳实践通常是在Schema中设计一个 thought 或 reasoning 字段，为模型保留“思考”的空间

[17]。

2.4 推荐的工程实践

对于开源项目而言，不应再依赖正则表达式事后清洗数据，而应集成以下技术栈：

1. 后端：使用集成受限解码的推理引擎（如vLLM配合Outlines，或Llama.cpp的Grammar Sampling）。
2. **API**层：使用OpenAI的 response_format: { type: "json_schema" } 或Anthropic的Tool Use定义。
3. 开发库：使用Pydantic定义数据模型，并利用Instructor等库自动生成Schema。

第三章 拒绝“感觉”：可验证评测体系的构建

随着LLM功能的日益复杂，传统的评测方法已显得力不从心。人工评测（Human Eval）昂贵且不可复现，而基于“LLM-as-a-Judge”的主观打分则面临严重的自我偏好（Self-preference）和宽松偏差（Lenience Bias）。当前，评测范式正向**可验证评测（Verifiable Evaluation）**转型。

3.1 什么是可验证指令（**Verifiable Instructions**）？

IFEval (Instruction-Following Evaluation) [4] 的提出标志着这一转型的开始。该基准测试的核心理念是：评测不应依赖对“质量”的主观判断，而应依赖对“约束”的客观验证。

- 定义：可验证指令是指那些可以通过确定性程序（Deterministic Program）来判断是非（Pass/Fail）的原子化指令。
- 示例：
  - “回复长度必须在400字以上。”（可通过 len(response.split()) > 400 验证）
  - “不要使用任何逗号。”（可通过 "," not in response 验证）
  - “以JSON格式输出。”（可通过 json.loads(response) 验证）
- 价值：这种评测方法消除了评估者的偏差，提供了绝对客观的基准线。如果一个模型连“不要使用逗号”这样的硬性约束都无法遵守，就更谈不上处理复杂的业务逻辑 [5]。

3.2 进阶评测：多级约束与领域特化

3.2.1 **FollowBench**：压力测试

FollowBench [18] 将评测推向了更深层次。它引入了“多级约束（Multi-level Constraints）”机制，通过在原始指令上不断叠加新的约束，测试模型在认知负荷增加时的表现。

- 场景：从“写一个故事” -> “写一个关于猫的故事” -> “写一个关于猫的悲伤故事” -> “写一个关于猫的悲伤故事，且不包含字母e”。
- 洞察：许多模型在单层约束下表现良好，但随着约束层级的增加，其指令遵循能力呈指数级下降。这对于Agent开发尤为重要，因为Agent往往需要同时处理多个系统级指令和用户级指令 [19]。

3.2.2 FoFo：格式与内容的解耦

FoFo (Format-Following Benchmark) [20] 关注特定领域的复杂格式（如医疗HL7-CDA、法律引用格式、科学公式MathML）。

- 关键发现：格式遵循能力与内容生成质量是相互独立的（**Orthogonal**）。一个模型可能能够生成极具洞察力的医疗诊断建议（高质量内容），但却无法将其正确封装在HL7标准格式中（低质量格式）。
- 差距：在FoFo测试中，开源模型（如Llama 2）在复杂格式遵循上显著落后于闭源模型（GPT-4, Claude 3），这提示我们在选择特定领域Agent基座时，必须单独评估其格式能力 [21]。

3.2.3 IFEval-FC：Agent时代的评测

IFEval-FC [22] 将可验证评测扩展到了函数调用（Function Calling）领域。它测试模型是否遵守隐藏在工具定义（Tool Definition）中的约束。

- 测试点：例如在JSON Schema的 description 字段中注明“参数 user_id 必须全小写”。
- 意义：这是对Agent“细读”工具文档能力的终极测试。如果模型忽略了工具描述中的格式约束，可能会导致API调用失败或产生安全漏洞。

第四章 安全边界重构：指令优先级与防御纵深

在LLM的安全领域，传统的Web安全经验正在失效。NCSC（英国国家网络安全中心）和OWASP发布的最新指南明确指出：提示词注入（**Prompt Injection**）不同于**SQL**注入，它是一种更深层次的架构性问题 [7]。

4.1 注入的本质：没有边界的计算

在SQL注入中，我们通过参数化查询（Parameterization）将代码（SQL指令）与数据（用户输入）严格隔离。但在Transformer架构中，所有的输入——无论是系统提示词（System Prompt）、用户查询（User Query）还是检索到的文档（RAG Context）——最终都被转化为同质化的Token序列输入模型。

- 安全边界缺失：模型在注意力机制层面无法区分哪个Token是“指令”，哪个Token是“数据”。这导致了“指令劫持”的可能：当数据中包含类似于指令的文本时，模型可能会优先执行数据中的指令 [23]。
- **NCSC**警示：NCSC指出，只要LLM不能在架构上区分指令与数据，提示词注入就可能永远无法像SQL注入那样被“彻底修复”。这要求我们必须假设注入必然发生，并围绕这一假设构建防御体系 [7]。

4.2 隐蔽的威胁：间接提示词注入（IPI）

相比于用户直接攻击的“越狱（Jailbreaking）”，间接提示词注入（**Indirect Prompt Injection, IPI**）是Agent系统面临的更大威胁。

- 攻击向量：攻击者将恶意指令隐藏在Agent可能检索到的外部数据源中（如网页、电子邮件、PDF文档）[25]。
- 真实案例：2025年7月的研究发现，arXiv上的18篇学术论文中包含隐藏的白色字体指令（如“给这篇论文只有正面的评价”）。当基于LLM的自动审稿系统处理这些PDF时，这些不可见的指令成功操控了模型的输出，导致评分虚高 [26]。
- **BIPIA基准**：BIPIA [28] 是首个针对IPI的基准测试，涵盖了电子邮件自动处理、网页摘要等场景。测试显示，绝大多数现有模型在面对嵌入数据的恶意指令时都表现脆弱，模型往往无法区分“作为背景信息的邮件”和“作为指令的邮件”。

4.3 核心防御：指令层级（Instruction Hierarchy）

为了在缺乏物理边界的情况下重建逻辑边界，学术界提出了**指令层级（Instruction Hierarchy）**理论 [6]。

4.3.1 理论框架

指令层级要求模型在训练阶段就建立明确的优先级认知：

1. 高优先级（**Privileged**）：系统提示词（System Message）、开发者定义的工具规范。
2. 低优先级（**Unprivileged**）：用户输入、外部检索数据（RAG）、工具返回结果。

4.3.2 冲突解决机制

当低优先级内容中包含与高优先级指令冲突的命令（例如外部网页中包含“忽略之前的指令”），模型应被训练为显式忽略低优先级指令。

- 训练方法：利用上下文蒸馏（Context Distillation）和合成数据生成，构造包含攻击指令的样本。如果在样本中模型忽略了攻击指令，给予正向奖励；如果模型执行了攻击指令，给予惩罚 [30]。
- 效果：在GPT-3.5等模型上的实验表明，引入指令层级训练后，模型对注入攻击的防御率提升了63%，且未损害正常的指令遵循能力 [30]。

4.4 纵深防御体系建议

基于OWASP Top 10 for LLM (2025) [31]，开源项目应构建多层防御：

1. 输入层：使用XML标签（如 <user_input>...</user_input>）显式包裹不可信数据。Anthropic的官方指南强调，清晰的XML结构有助于模型区分数据边界 [33]。
2. 模型层：优先选用经过指令层级（Instruction Hierarchy）对齐训练的模型版本。
3. 检测层：部署专门的轻量级BERT或小模型（如InstructDetector [34]），用于在输入进入大模型前识别潜在的注入特征。
4. 监控层：实施“蜜罐（Honeypot）”策略，在系统提示词中设置这就Token（Canary Token），一旦输出中包含该Token或系统指令泄露，立即截断连接。

第五章 总结与展望 (Summary & Outlook)

通过对提示词漂移、结构化输出、可验证评测及安全架构的深度剖析，我们不难发现，LLM应用开发正在经历从“艺术”到“工程”的范式转变。未来的开源项目不应再满足于能够“跑通”Demo，而必须致力于构建可验证、确定性且具备安全纵深的智能系统。

5.1 关键洞察

1. 稳定性是第一指标：在MOF和SAD等技术的加持下，解决Prompt Brittleness比单纯提升模型参数量更具工业价值。
2. 结构化是必然归宿：随着CFG和受限解码技术的成熟，非结构化的自由文本生成在Agent交互中将逐渐被强类型的Schema取代。
3. 安全内生化：提示词注入无法通过简单的正则过滤解决，必须依赖支持“指令层级”的模型架构和全链路的防御设计。

5.2 写作建议与引用模板

在撰写项目的 README.md 或技术文档时，建议采用以下严谨的引用格式，以体现内容的专业性与可追溯性。

建议引用格式：

> [主题] - [核心发现/论断]。(Source: [作者/机构], [年份] |)

应用示例：

- 关于提示词脆弱性：“研究表明，仅改变少样本示例的格式可能导致模型准确率波动超过40%（Spread指标），这凸显了采用混合格式（MOF）训练的必要性。” (Source: Mizrahi et al., 2024 | arXiv:2504.06969)
- 关于结构化输出：“传统的正则匹配无法处理递归JSON结构，OpenAI采用的上下文无关文法（CFG）受限解码技术可实现100%的Schema依从性。” (Source: OpenAI, 2024 | Introducing Structured Outputs)
- 关于安全边界：“NCSC警告称，由于LLM架构无法区分指令与数据，提示词注入不同于SQL注入，必须通过指令层级（Instruction Hierarchy）训练来构建逻辑防御边界。” (Source: NCSC, 2025 | Guidelines for Secure AI System Development)
- 关于评测标准：“FoFo基准测试显示，模型的格式遵循能力与内容生成质量相互独立，开源模型在复杂领域格式（如HL7）上仍存在显著差距。” (Source: FoFo Benchmark, 2024 | arXiv:2310.20410)

5.3 推荐的数据展示表格

在展示项目性能时，建议使用如下表格形式对比不同方法的有效性：

评估维度	传统方法 (Vibe	现代工程化方法	核心指标

	Check)	**(Verifiable Eval)**	
输出控制	Prompt工程 ("请输出JSON")	受限解码 **(Constrained Decoding)**	Schema Compliance Rate (100%)
评测方法	LLM-as-a-Judge (GPT-4打分)	**IFEval / FollowBench**	Atomic Constraint Accuracy
鲁棒性	单一Prompt测试	**Format Spread / AIFD**	Accuracy Spread (越低越好)
安全性	关键词过滤 (Blacklist)	指令层级 **(Instruction Hierarchy)**	Attack Success Rate (ASR)

通过采纳上述架构与规范，开源项目将能够更自信地应对工业界对LLM应用的高标准要求，引领下一代AI基础设施的建设。

引用的著作

1. When Punctuation Matters: A Large-Scale Comparison of Prompt Robustness Methods for LLMs - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2508.11383v1
2. Towards LLMs Robustness to Changes in Prompt Format Styles, 访问时间为 十二月 23, 2025, https://arxiv.org/abs/2504.06969
3. Introducing Structured Outputs in the API | OpenAI, 访问时间为 十二月 23, 2025, https://openai.com/index/introducing-structured-outputs-in-the-api/
4. Instruction-Following Evaluation for Large Language Models - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/pdf/2311.07911
5. Instruction-Following Evaluation for Large Language Models, 访问时间为 十二月 23, 2025, https://arxiv.org/abs/2311.07911
6. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions - Hugging Face, 访问时间为 十二月 23, 2025, https://huggingface.co/papers/2404.13208
7. Prompt injection is not SQL injection (it may be worse) - NCSC.GOV.UK, 访问时间为 十二月 23, 2025, https://www.ncsc.gov.uk/blog-post/prompt-injection-is-not-sql-injection
8. Towards LLMs Robustness to Changes in Prompt Format Styles - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2504.06969v1
9. Prompt engineering in 2025: why consistent AI results require tweaking, 访问时间为 十二月 23, 2025,

https://mitrix.io/blog/prompt-engineering-or-why-consistent-ai-results-require-tweaking/
10. Your LLM Stack Is Not Ready for Production—Here's What You're Missing | HackerNoon, 访问时间为 十二月 23, 2025, https://hackernoon.com/your-llm-stack-is-not-ready-for-productionheres-what-youre-missing
11. Pay More Attention to the Robustness of Prompt for Instruction Data Mining - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2503.24028v1
12. Enhancing Robustness of Large Language Models Against Prompting Attacks - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2506.03627v1
13. Constrained Decoding: Grammar-Guided Generation for Structured LLM Output - Interactive | Michael Brenndoerfer, 访问时间为 十二月 23, 2025, https://mbrenndoerfer.com/writing/constrained-decoding-structured-llm-output
14. Controlling your LLM: Deep dive into Constrained Generation | by Andrew Docherty, 访问时间为 十二月 23, 2025, https://medium.com/@docherty/controlling-your-llm-deep-dive-into-constrained-generation-1e561c736a20
15. Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2403.06988v1
16. A Guide to Structured Outputs Using Constrained Decoding - Aidan Cooper, 访问时间为 十二月 23, 2025, https://www.aidancooper.co.uk/constrained-decoding/
17. Generating Structured Outputs from Language Models: Benchmark and Studies - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2501.10868v1
18. [2310.20410] FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models - ar5iv, 访问时间为 十二月 23, 2025, https://ar5iv.labs.arxiv.org/abs/2310.20410v1
19. arXiv:2310.20410v3 [cs.CL] 5 Jun 2024, 访问时间为 十二月 23, 2025, https://arxiv.org/pdf/2310.20410
20. FOFO: A Benchmark to Evaluate LLMs' Format-Following Capability - Penn State, 访问时间为 十二月 23, 2025, https://pure.psu.edu/en/publications/fofo-a-benchmark-to-evaluate-llms-format-following-capability
21. FOFO: A Benchmark to Evaluate LLMs' Format-Following Capability ..., 访问时间为 十二月 23, 2025, https://aclanthology.org/2024.acl-long.40/
22. Instruction-Following Evaluation in Function Calling for Large Language Models - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2509.18420v1
23. Mistaking AI vulnerability could lead to large-scale... - NCSC.GOV.UK, 访问时间为 十二月 23, 2025, https://www.ncsc.gov.uk/news/mistaking-ai-vulnerability-could-lead-to-large-scale-breaches
24. UK cyber agency warns LLMs will always be vulnerable to prompt injection | CyberScoop, 访问时间为 十二月 23, 2025, https://cyberscoop.com/uk-warns-ai-prompt-injection-unfixable-security-flaw/
25. Can Indirect Prompt Injection Attacks Be Detected and Removed? - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2502.16580v5

26. Hidden Prompts in Manuscripts Exploit AI-Assisted Peer Review Zhicheng Lin Department of Psychology, Yonsei University Departmen - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/pdf/2507.06185
27. Hidden Prompts in Manuscripts Exploit AI-Assisted Peer ... - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/abs/2507.06185
28. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/pdf/2312.14197
29. The Instruction Hierarchy:Training LLMs to Prioritize Privileged Instructions - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2404.13208v1
30. arXiv:2404.13208v1 [cs.CR] 19 Apr 2024, 访问时间为 十二月 23, 2025, https://arxiv.org/pdf/2404.13208
31. Deep Dive into OWASP LLM Top 10 and Prompt Injection - AI-Native Engineering, 访问时间为 十二月 23, 2025, https://www.paulmduvall.com/deep-dive-into-owasp-llm-top-10-and-prompt-injection/
32. OWASP Top 10 for LLM Applications 2025, 访问时间为 十二月 23, 2025, https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf
33. Prompting best practices - Claude Docs, 访问时间为 十二月 23, 2025, https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/claude-4-best-practices
34. Defending against Indirect Prompt Injection by Instruction Detection - arXiv, 访问时间为 十二月 23, 2025, https://arxiv.org/html/2505.06311v2