以下为该文章《Prompt Engineering》的主要内容与结构梳理，并包含文章基本信息：

文章基本信息
- 标题：Prompt Engineering
- 作者：Lee Boonstra
- 机构：Google
- 出版日期：2025年2月
- 主要贡献者：Michael Sherman、Yuan Cao、Erick Armbrust、Anant Nawalgaría、Antonio Gulli、Simone Cammel
- 策划/编辑：Antonio Gulli、Anant Nawalgaría、Grace Mollison
- 技术撰稿人：Joey Haymaker
- 设计：Michael Lanning

---

1. Introduction（简介）

- 介绍什么是大型语言模型（LLM）及其输入（prompt），并说明prompt工程无需机器学习背景，但要高效编写prompt则较为复杂，需要考虑模型、训练数据、词语选择、风格语气、结构和上下文等多方面。
- 强调prompt工程是一个迭代过程，低质量prompt会影响模型输出的准确性和实用性。
- 本文聚焦于如何在Vertex AI或API中直接为Gemini等模型编写prompt，以便访问更多配置，并详细探讨prompt工程的技巧、最佳实践和挑战。
- 解释LLM如何通过顺序文本输入预测token，prompt工程的目标是优化模型输出。

2. Prompt Engineering（提示工程）

- 聚焦如何通过Vertex AI或API为Gemini模型等直接编写prompt，并访问如temperature等参数。
- 详细介绍prompt工程的技术、入门建议、最佳实践及常见挑战。
- 说明LLM的工作原理与prompt的作用，即通过高质量的prompt引导模型输出准确结果。
- prompt可用于多种任务，如摘要、信息抽取、问答、分类、翻译、代码生成等，并需针对不同模型优化prompt及配置参数。
- 输出配置是高效prompt工程的关键，包括输出长度等。

3. LLM Output Configuration（LLM输出配置）

- 介绍控制LLM输出的关键参数：输出长度（token数量）、temperature（温度）、top-K、top-P（nucleus sampling）。
- 详细解释每个参数的含义及相互影响，并提供推荐配置：
- 平衡创造性和连贯性：temperature=0.2, top-P=0.95, top-K=30
- 高创造性：temperature=0.9, top-P=0.99, top-K=40
- 低创造性：temperature=0.1, top-P=0.9, top-K=20
- 唯一正确答案：temperature=0
- 指出参数设置不当会导致输出冗余或重复，需平衡确定性与创造性。

4. Prompting Techniques（提示技术）

- 介绍多种prompting技巧：
- Zero-Shot Prompting：仅用任务描述和输入，无示例，适用于简单任务。
- One-Shot & Few-Shot Prompting：提供一个或多个高质量示例，适合复杂或结构化任务。
- 系统、上下文与角色提示（System, Contextual, Role Prompting）：分别设定模型目标、任务上下文及角色身份，三者可组合提升任务表现。
- 实践建议包括结构化记录各次prompt尝试，并精心设计示例。

5. System, Contextual, and Role Prompting（系统、上下文与角色提示）

- 详细说明三种关键提示技术：
- System Prompting：设定模型整体目标和输出规范（如格式、风格等）。
- Role Prompting：分配特定角色或身份，影响输出的风格和专业性。
- Contextual Prompting：补充即时任务相关的背景信息，提高响应准确性。
- 三种提示方式可灵活组合，提升LLM输出质量和针对性。

6. Step-back Prompting（反向推理/回溯提示）

- step-back prompting先让模型思考更一般性原则，再用该答案作为上下文完成具体任务，有助于激活知识、提升创意与输出多样性，适合复杂任务。

7. Chain of Thought (CoT) Prompting（思维链提示）

- CoT引导模型输出中间推理步骤，提升模型推理能力和结果可解释性，适合复杂、多步推理任务。
- 可结合few-shot和self-consistency技术进一步增强效果，但会增加生成成本。

8. Self-consistency（自洽性）

- Self-consistency通过高temperature多次采样，统计多数输出结果，以提升CoT推理的准确性和一致性。
- 优点在于提升鲁棒性，缺点为计算成本高。

9. Tree of Thoughts (ToT)（思维树）

- ToT是CoT的扩展，让模型并行探索多条推理路径，适合推理空间复杂、多解的问题，提升复杂任务的解决能力。

10. ReAct (Reason & Act)（推理与行动结合）

- ReAct让LLM结合自然语言推理与外部工具行动，模拟人类的“思考-行动-反馈”循环，适合需要外部信息和多步推理的复杂任务。

11. Automatic Prompt Engineering (APE)（自动提示工程）

- APE通过让模型自动生成、评估和优化prompt，减少人工干预，提高多样性和鲁棒性，适用于文本生成、问答等场景。

12. Code Prompting（代码提示工程）

- 介绍如何利用LLM生成、解释、翻译、调试和优化代码，包括代码生成、测试、理解、翻译、调试与优化等应用场景。
- 强调在prompt中加入示例可提升准确性和效果。

13. Best Practices in Prompt Engineering（最佳实践）

- 多模态提示：结合多种输入（文本、图片、音频等）。
- 提供示例：加入one-shot/few-shot示例。
- 简洁清晰设计：指令明确、避免冗余。
- 明确输出要求、优先正向指令、控制输出长度、在Prompt中用变量、尝试不同格式与风格、关注模型更新、实验输出格式（如JSON）、使用自动修复工具保证结构化输出完整。

14. JSON Repair

- JSON格式输出结构化清晰但易被截断，推荐用如`json-repair`自动修复不完整JSON，提升实际应用的可用性。

15. Working with Schemas

- 结构化JSON可用于输入和输出，通过schema定义输入结构，提升数据处理的准确性和效率。

16. Experiment together with other prompt engineers

- 鼓励多名工程师协作实验不同prompt，详细记录每次尝试，以便分析和持续优化。

17. Document the various prompt attempts

- 推荐用表格工具详细记录prompt尝试，包括内容、参数、模型、结果和反馈，便于复用和维护。

18. CoT Best practices

- CoT提示应在推理后给出答案，分离推理与答案，temperature设为0，详细记录所有尝试。

19. Summary（总结）

- 概括了本文介绍的各类提示技术与自动化方法，指出提示工程的挑战和最佳实践，旨在帮助成为优秀的Prompt Engineer。

20. Endnotes（尾注）

- 列出15条参考文献，涵盖Google产品文档、模型采样方法、核心学术论文与GitHub资源，便于进一步学习与查阅。
