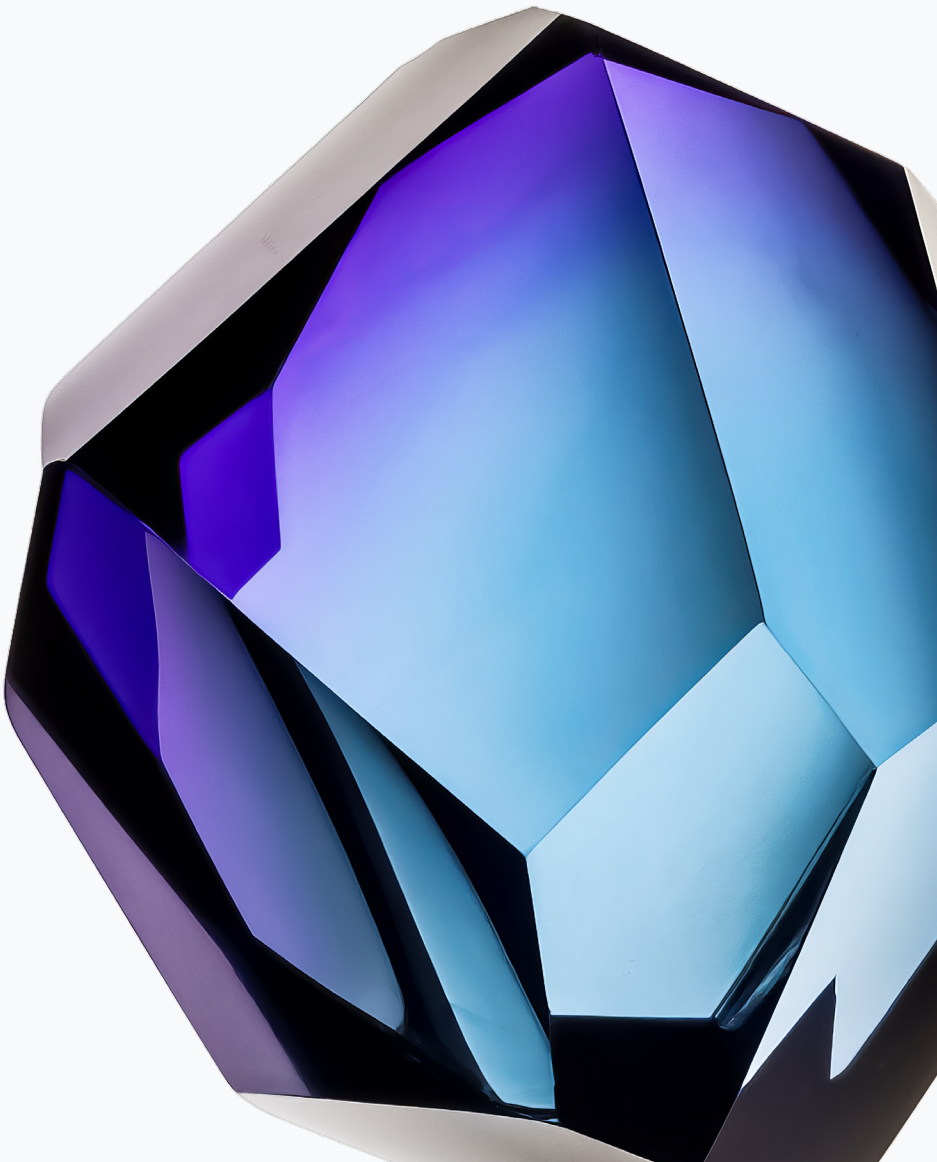
Prompt Engineering

Author: Lee Boonstra



Google

致谢

审稿人和撰稿人

迈克尔・谢尔曼

曹元

埃里克・阿姆布鲁斯特

阿南特・纳瓦尔加里亚

安东尼奥・古利

西蒙娜・卡梅尔

策展人和编辑

安东尼奥・古利

阿南特・纳瓦尔加里亚

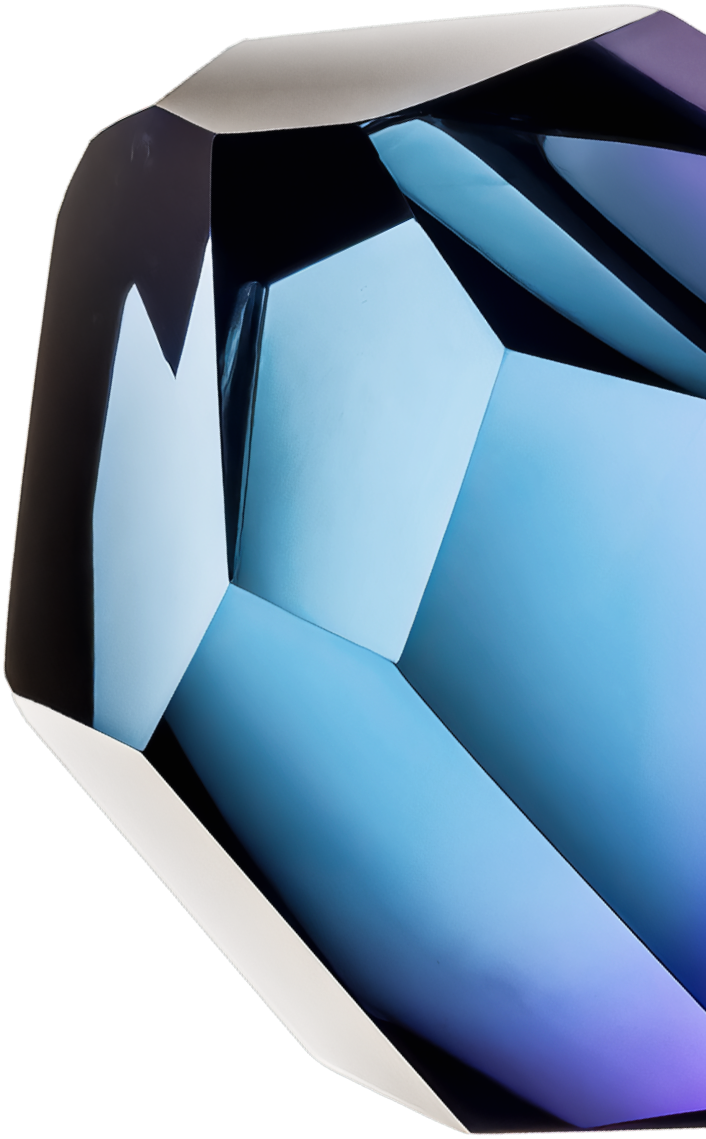
格蕾丝・莫里森

技术撰稿人

乔伊・海梅克

设计师

迈克尔・兰宁



目录

介绍 6

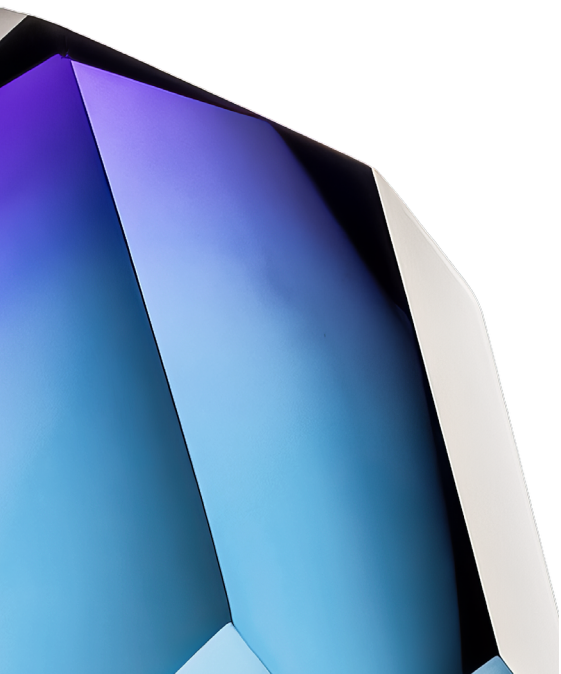
迅速工程 7

LLM 输出配置 8

- 输出长度 8
- 抽样控制 9
  - 温度 9
  - Top-K 和 top-P 10
  - 把所有东西整合起来 11

提示技巧 12

- 一般提示/零射击 13
- 单发和少发 14
- 系统、情境和角色提示 17
  - 系统提示 18
  - 角色提示 21
  - 情境提示 23



后退提示 25
思维链（CoT） 29
自我一致性 32
思想之树（ToT） 36
ReAct（理性与行动） 37
自动提示工程 40
代码提示 42
编写代码的提示 42
解释代码的提示 44
翻译代码的提示 46
用于调试和审查代码的提示 48
多模态提示如何? 54
最佳实践 54
请举例说明 54
设计简约 55
请具体说明输出结果。 56
遵循指令而非约束 56
控制最大令牌长度 58
在提示中使用变量 58
尝试不同的输入格式和写作风格 59
对于包含分类任务的少量提示，应混合使用不同的类别。 59
适应模型更新 60
尝试不同的输出格式 60

与其他快速工程师一起进行实验 61
CoT最佳实践 61
记录各种提示尝试 62
概括 63
尾注 65

> 你不需要是数据科学家或机器学习工程师——每个人都可以编写提示词。

介绍

在考虑大型语言模型的输入输出时，文本提示（有时会结合其他模态，例如图像提示）是模型用来预测特定输出的输入。你不需要是数据科学家或机器学习工程师——每个人都可以编写提示词。然而，编写最有效的提示词可能很复杂。提示词的许多方面都会影响其有效性：你使用的模型、模型的训练数据、模型配置、你的用词、风格和语气、结构以及上下文都至关重要。因此，提示词设计是一个迭代过程。不恰当的提示词会导致模棱两可、不准确的回答，并会阻碍模型提供有意义输出的能力。

当你和Gemini聊天机器人聊天时，[1]您基本上是编写提示，但是本白皮书重点介绍如何在 Vertex AI 中或使用 API 为 Gemini 模型编写提示，因为通过直接提示模型，您可以访问温度等配置。

本白皮书将详细探讨提示设计。我们将研究各种提示技巧，帮助您入门，并分享成为提示专家的技巧和最佳实践。我们还将讨论您在设计提示时可能遇到的一些挑战。

迅速工程

记住LLM的工作原理：它是一个预测引擎。该模型以序列文本作为输入，然后根据训练数据预测下一个词元。LLM会不断重复这个过程，将之前预测的词元添加到序列文本的末尾，以预测下一个词元。下一个词元的预测基于先前词元的内容与LLM在训练过程中所观察到的内容之间的关系。

编写提示语时，您实际上是在尝试设置语言学习模型 (LLM)，使其能够预测正确的词元序列。提示语工程是指设计高质量提示语，引导 LLM 生成准确输出的过程。这个过程包括反复调整以找到最佳提示语、优化提示语长度，以及评估提示语的写作风格和结构是否符合任务要求。在自然语言处理和 LLM 的语境下，提示语是提供给模型以生成响应或预测的输入。

这些提示可用于实现各种理解和生成任务，例如文本摘要、信息提取、问答、文本分类、语言或代码翻译、代码生成以及代码文档或推理。

请随时参考谷歌的提示指南。[2,3]提供简单有效的提示示例。

在进行提示工程时，首先要选择一个模型。无论您使用的是 Vertex AI 中的 Gemini 语言模型、GPT、Claude，还是 Gemma 或 LLaMA 等开源模型，提示都可能需要针对您使用的特定模型进行优化。

除了提示之外，您还需要调整 LLM 的各种配置。

LLM 输出配置

选定模型后，您需要确定模型的配置。大多数 LLM 都提供各种配置选项，用于控制 LLM 的输出。有效的提示设计需要针对您的任务优化这些配置。

输出长度

一项重要的配置设置是响应中生成的令牌数量。生成更多令牌需要LLM进行更多计算，从而导致更高的能耗、可能更慢的响应时间和更高的成本。

缩短 LLM 的输出长度并不会使其输出在风格或文本上更加简洁，只会让 LLM 在达到限制后停止预测更多词元。如果您需要较短的输出长度，可能还需要对提示符进行相应的调整。

输出长度限制对于某些 LLM 提示技术（如 ReAct）尤其重要，因为 LLM 会在得到你想要的响应后继续发出无用的令牌。

抽样控制

LLM（逻辑逻辑模型）并不直接预测单个词元。相反，LLM 预测下一个词元的概率，LLM 词汇表中的每个词元都有一个概率值。然后对这些词元概率进行采样，以确定下一个生成的词元。温度、top-K 和 top-P 是最常用的配置项，它们决定了如何处理预测的词元概率，从而选择单个输出词元。

温度

温度控制着标记选择的随机程度。较低的温度适用于预期响应较为确定的提示，而较高的温度则可能导致更多样化或意想不到的结果。温度为 0（贪婪解码）时，结果是确定性的：总是选择概率最高的标记（但请注意，如果两个标记具有相同的最高预测概率，则根据平局打破机制的实现方式，温度为 0 时可能并不总是得到相同的输出）。

接近最高温度时，输出结果往往更具随机性。随着温度不断升高，所有代币成为下一个预测代币的概率都趋于一致。

Gemini 的温度控制可以类比于机器学习中使用的 softmax 函数。较低的温度设置相当于较低的 softmax 温度 (T)，强调一个高度确定的单一最佳温度。较高的 Gemini 温度设置则类似于较高的 softmax 温度，使得围绕所选设置更广泛的温度范围都可接受。这种更大的不确定性适用于一些不需要严格精确温度的场景，例如在尝试创作输出时。

Top-K 和 top-P

Top-K 和 top-P（也称为核取样）[4]LLM 中使用了两种采样设置，用于限制预测的下一个词元仅来自预测概率最高的词元。与温度类似，这些采样设置控制着生成文本的随机性和多样性。

- **Top-K**采样从模型预测的分布中选择最有可能的前 K 个词元。前 K 值越高，模型的输出就越有创意、越多样化；前 K 值越低，模型的输出就越静态、越客观。前 K 值为 1 时，相当于贪婪解码。

- **顶级球员**采样选择累积概率不超过某个值 (P) 的前几个标记。P 的值范围从 0（贪婪解码）到 1（LLM 词汇表中的所有标记）。

选择 top-K 还是 top-P 的最佳方法是尝试这两种方法（或将两者结合起来），看看哪种方法能产生你想要的结果。

另一个重要的配置项是响应中生成的令牌数量。请注意，生成更多令牌需要LLM进行更多计算，从而导致更高的能耗和可能更慢的响应时间，最终导致更高的成本。

把所有东西整合起来

选择 top-K、top-P、温度以及要生成的令牌数量取决于具体的应用场景和预期结果，而且这些设置之间相互影响。此外，务必确保您了解所选模型如何将不同的采样设置组合在一起。

如果温度、top-K 和 top-P 都可用（如在 Vertex Studio 中），则同时满足 top-K 和 top-P 条件的标记将作为下一个预测标记的候选标记，然后对满足 top-K 和 top-P 条件的标记进行采样，并应用温度。如果只有 top-K 或 top-P 可用，则行为相同，但只会使用其中一个 top-K 或 top-P 设置。

如果温度不可用，则从满足 top-K 和/或 top-P 标准的令牌中随机选择，以生成下一个预测令牌。

当某一采样配置值的设置达到极端值时，该采样设置要么会抵消其他配置设置，要么会变得无关紧要。

- 如果将温度设置为 0，则 top-K 和 top-P 将失效——最有可能出现的词元将成为下一个预测词元。如果将温度设置得非常高（高于 1，通常在 10 以上），则温度也将失效，所有符合 top-K 和/或 top-P 标准的词元都将随机抽取，用于选择下一个预测词元。

- 如果将 top-K 设置为 1，则温度和 top-P 将变得无关紧要。只有一个词元符合 top-K 标准，该词元即为下一个预测词元。如果将 top-K 设置得非常高，例如与 LLM 的词汇表大小相同，则任何具有非零概率成为下一个词元的词元都将符合 top-K 标准，并且不会有任何词元被选中。

- 如果将 top-P 设置为 0（或一个非常小的值），大多数 LLM 采样实现将只考虑最有可能满足 top-P 标准的标记，从而使温度和 top-K 值变得无关紧要。如果将 top-P 设置为 1，则任何具有非零概率成为下一个标记的标记都将满足 top-P 标准，并且不会有任何标记被排除。

一般来说，温度设为 0.2，峰值功率设为 0.95，峰值开尔文设为 30，可以得到相对连贯且富有创意的结果，但又不会过于离谱。如果想要更具创意的结果，可以尝试将温度设为 0.9，峰值功率设为 0.99，峰值开尔文设为 40。如果想要创意较少的结果，可以尝试将温度设为 0.1，峰值功率设为 0.9，峰值开尔文设为 20。最后，如果任务只有一个正确答案（例如，解答数学题），则将温度设为 0。

笔记：如果自由度更高（温度、top-K、top-P 和输出标记更高），LLM 可能会生成相关性较低的文本。

提示技巧

逻辑学习模型（LLM）经过精心设计，能够遵循指令，并基于大量数据进行训练，从而理解提示信息并生成答案。但 LLM 并非完美无缺；提示文本越清晰，LLM 预测下一个可能文本的准确率就越高。此外，利用 LLM 的训练方式和工作原理的特定技术，将有助于您从 LLM 中获得更准确的结果。

现在我们已经了解了什么是提示工程以及它需要什么，让我们深入探讨一些最重要的提示技巧的例子。

一般提示/零射击

一个*零发*[5]提示是最简单的提示类型。它只提供任务描述和一些文本，供LLM（学习者）开始写作。这些输入可以是任何内容：一个问题、故事的开头或一些说明。“零样本”指的是“没有示例”。

让我们在Vertex AI中使用Vertex AI Studio（用于语言学习），[6]它提供了一个测试提示的平台。表 1 展示了一个用于对电影评论进行分类的零样本提示示例。

下表所示的格式是记录提示信息的绝佳方式。您的提示信息在最终集成到代码库之前很可能会经历多次迭代，因此以规范、结构化的方式跟踪提示信息工程工作至关重要。关于此表格格式、跟踪提示信息工程工作的重要性以及提示信息开发流程的更多信息，请参阅本章后面的“最佳实践”部分（“记录各种提示信息尝试”）。

模型温度应设置为较低值，因为无需进行任何特殊设置，我们使用 gemini-pro 的默认 top-K 和 top-P 值，这实际上禁用了这两个设置（参见上文“LLM 输出配置”）。请注意生成的输出。 *令人不安*和*杰作*由于这两个词在同一句话中出现，因此预测会稍微复杂一些。

姓名	1_1_电影分类		
目标	将电影评论分为正面、中性或负面。		
模型	双子座-专业版		
温度	0.1	代币限额	5
Top-K	不适用	顶级球员	1
迅速的	将影评分为正面、中性或负面。影评：“她”是一部令人不安的影片，揭示了如果人工智能不受控制地持续发展，人类将会走向何方。我希望有更多像这部杰作一样的电影。情感：		
输出	积极的		

表1. 零样本提示示例

当零次提示不起作用时，你可以在提示中提供演示或示例，从而过渡到“一次提示”和“少量提示”。通用提示/零次提示

单发和少发

在为人工智能模型创建提示时，提供示例很有帮助。这些示例可以帮助模型理解你的要求。当你想引导模型输出特定的结构或模式时，示例尤其有用。

一个**一次性提示**提供*单身的*例如，因此得名“单次建模”。其理念是，模型可以模仿一个示例，从而更好地完成任务。

一个**少枪提示**提供*多种的*向模型提供示例。这种方法向模型展示了它需要遵循的模式。其思路类似于单次训练，但多个所需模式的示例可以提高模型遵循该模式的概率。

少样本提示所需的示例数量取决于几个因素，包括任务的复杂性、示例的质量以及您使用的生成式人工智能（gen AI）模型的性能。一般来说，少样本提示至少需要三到五个示例。但是，对于更复杂的任务，您可能需要使用更多示例；或者，由于模型的输入长度限制，您可能需要使用更少的示例。

表 2 显示了一个少量提示示例，我们使用与之前相同的 gemini-pro 模型配置设置，只是增加了令牌限制以适应更长的响应时间。

目标	将披萨订单解析为 JSON 格式。		
模型	双子座-专业版		
温度	0.1	代币限额	250
Top-K	不适用	顶级球员	1
迅速的	将顾客的披萨订单解析为有效的JSON格式： 例子： 我想要一份小披萨，加芝士、番茄酱和意大利辣香肠。JSON 响应： ``` { "尺寸": "小号", "type": "normal", "配料": [["奶酪", "番茄酱", "意大利辣香肠"]] } ``` **（未完待续，请见下一页）**		

迅速的	例子： 我可以点一份大披萨吗？要番茄酱、罗勒和马苏里拉奶酪。 { "尺寸": "大号", "type": "normal", "配料": [["番茄酱", "bazel", "马苏里拉奶酪"]] } 现在，我想要一个大披萨，一半放奶酪和马苏里拉奶酪，另一半放番茄酱、火腿和菠萝。 JSON 响应：
输出	``` { "尺寸": "大号", "type": "half-half", 配料：[["奶酪", "马苏里拉奶酪"], ["番茄酱", "火腿", "菠萝"]] } ```

表 2. 少次提示示例

在选择示例时，务必使用与你要完成的任务相关的示例。示例应多样化、高质量且编写精良。哪怕一个小小的错误都可能使模型产生误解，并导致不理想的结果。

如果您希望生成能够应对各种输入的稳健输出，那么在示例中包含极端情况至关重要。极端情况是指那些不寻常或出乎意料的输入，但模型仍然应该能够处理这些输入。

系统、情境和角色提示

系统提示、情境提示和角色提示都是用于指导语言学习者生成文本的技术，但它们侧重点不同：

- **系统提示**它设定了语言模型的整体背景和目标。它定义了模型应该完成的“大局”，例如翻译语言、对评论进行分类等等。

- **情境提示**提供与当前对话或任务相关的具体细节或背景信息。这有助于模型理解问题中的细微差别，并据此调整响应。

- **角色提示**为语言模型指定一个特定的角色或身份。这有助于模型生成与所分配角色及其相关知识和行为相一致的响应。

系统提示、上下文提示和角色提示之间可能存在相当大的重叠。例如，为系统分配角色的提示也可能具有上下文信息。

然而，每种提示的主要目的略有不同：

- 系统提示：定义模型的基本功能和总体目标。
- 情境提示：提供即时、特定于任务的信息来指导响应。它与当前任务或输入高度相关，并且是动态的。
- 角色提示：界定模型的输出风格和语气，增添具体性和个性。

区分系统提示、上下文提示和角色提示，为设计具有明确意图的提示提供了一个框架，允许灵活组合，并更容易分析每种提示类型如何影响语言模型的输出。

让我们深入探讨这三种不同类型的提示。

系统提示

表 3 包含系统提示，我在其中指定了如何返回输出的附加信息。我提高了温度以获得更高的创造力水平，并指定了更高的词元限制。然而，由于我已明确指示如何返回输出，模型没有返回额外的文本。

目标	将电影评论分为正面、中性或负面。		
模型	双子座-专业版		
温度	1	代币限额	5
Top-K	40	顶级球员	0.8
迅速的	将电影评论分类为正面、中性或负面。标签必须全部大写。 影评：《她》是一部令人不安的影片，揭示了如果人工智能不受控制地持续发展，人类将会走向何方。它太令人不安了，我根本看不下去。感想：		
输出	消极的		

表3. 系统提示示例

系统提示可用于生成满足特定要求的输出。“系统提示”实际上是指“为系统提供额外任务”。例如，您可以使用系统提示生成与特定编程语言兼容的代码片段，或者使用系统提示返回特定结构。请参阅表 4，其中我以 JSON 格式返回了输出。

目标	将电影评论分类为正面、中性或负面，返回 JSON 数据。		
模型	双子座-专业版		
温度	1	代币限额	1024
Top-K	40	顶级球员	0.8
迅速的	将电影评论分类为正面、中性或负面。返回有效的 JSON 数据： 影评：《她》是一部令人不安的纪录片，揭示了如果人工智能不受控制地继续发展下去，人类将会走向何方。它太令人不安了，我根本看不下去。 模式： ``` 电影： { "sentiment": 字符串 "积极" | "消极" | "中性", "name": 字符串 } 电影评论： { "movie_reviews": [电影] } ``` JSON 响应：		
输出	``` { "movie_reviews": [{ “情绪”：“消极的”， "name": "她" }] } ```		

表 4. JSON 格式的系统提示示例

从提取数据的提示中返回 JSON 对象有一些好处。在实际应用中，我不需要手动创建这种 JSON 格式，我可以按排序顺序返回数据（这在处理日期时间对象时非常方便），但最重要的是，通过提示输入 JSON 格式，可以强制模型创建结构并减少错误。

系统提示对于安全和防止不良行为也非常有用。要控制输出，只需在提示中添加一行，例如：“你的回答应该尊重他人。”

角色提示

角色提示是提示工程中的一种技术，它为人工智能模型分配特定的角色。这有助于模型生成更相关、信息更丰富的输出，因为模型可以根据分配的特定角色来调整其响应。

例如，您可以让一个人工智能模型扮演图书编辑、幼儿园老师或励志演讲者等角色。模型被分配角色后，您可以向其发送与该角色相关的特定提示。例如，您可以提示老师创建一份教案，然后由老师进行审阅。请参阅表 5，其中模型扮演了旅行指南的角色。

目标	担任旅行向导，并提供 3 条旅行建议		
模型	双子座-专业版		
温度	1	代币限额	1024
Top-K	40	顶级球员	0.8
迅速的	我希望你能担任我的旅行向导。我会告诉你我的位置，请你推荐三个附近的景点。在某些情况下，我还会告诉你我打算去哪些类型的景点。 我的建议是：“我在阿姆斯特丹，我只想参观博物馆。” 旅行建议：		
输出	1. 荷兰国家博物馆：世界闻名的博物馆，以收藏伦勃朗的《夜巡》等荷兰艺术杰作而闻名。千万不要错过馆内令人惊叹的绘画和文物藏品，它们展现了荷兰的艺术和历史。 2. 梵高博物馆：沉浸在文森特·梵高的迷人作品中。该博物馆收藏了他数量最多的绘画和素描作品，包括《星夜》和《向日葵》。 3. 阿姆斯特丹市立博物馆：探索来自世界各地的现代和当代艺术。博物馆藏品包括毕加索、康定斯基和里特维尔德的作品，馆址位于一座引人注目的现代建筑内。		

表5.角色提示示例

为 AI 模型定义角色视角，可以为其提供你所寻求的基调、风格和专业领域的蓝图，从而提高输出的质量、相关性和有效性。

以下是一些我认为比较有效的风格，你可以选择一下：

对抗性的、描述性的、直接的、正式的、幽默的、有影响力的、非正式的、鼓舞人心的、有说服力的

让我们把表 6 中的提示语改成幽默励志的风格。

迅速的	我希望你担任我的旅行向导。我会告诉你我的位置，请你用幽默风趣的风格推荐三个附近值得一去的地方。 我的建议：“我在曼哈顿。” 旅行建议：
输出	1. 感受帝国之州的雄伟气魄：登上令人眩晕的帝国大厦之巅，尽情欣赏曼哈顿天际线的壮丽景色。准备好体验站在“大苹果”之巅的金刚感觉吧，只不过少了那根巨大的香蕉。 2. 在纽约现代艺术博物馆 (MoMA) 尽情释放你的艺术热情：在纽约现代艺术博物馆 (MoMA) 释放你内心的艺术爱好者。欣赏那些让你叹为观止的艺术杰作，并让你开始怀疑自己画的火柴人是否具有任何艺术价值。 3. 在第五大道尽情购物：在标志性的第五大道上尽情享受购物的乐趣吧。做好心理准备，在那些设计师精品店里，橱窗里的商品（或者真的去购物）会让你的钱包哭泣。但是，嘿，你身在曼哈顿，何不尽情享受这奢华的购物体验呢！

表6. 以幽默的语气和风格进行角色提示的示例

情境提示

通过提供上下文提示，您可以确保人工智能交互尽可能流畅高效。正如表 7 的示例所示，模型将能够更快地理解您的请求，并生成更准确、更相关的回复。

目标	请推荐一些关于复古游戏的博客文章		
模型	双子座-专业版		
温度	1	代币限额	1024
Top-K	40	顶级球员	0.8
迅速的	背景：你正在为一个关于 80 年代复古街机视频游戏的博客撰稿。 请提出三个您想撰写文章的主题，并简要描述文章应包含的内容。		
输出	1. 街机柜体设计的演变： 本文将探讨街机柜设计的演变，从 20 世纪 70 年代早期的木制和金属柜体到 20 世纪 80 年代及以后的时尚霓虹灯设计。 2. **重温经典：80年代标志性街机游戏：** 本文将带您了解 20 世纪 80 年代一些最具标志性和影响力的街机游戏的历史、影响和遗产，重点介绍它们的创新之处、流行的机制和持久的魅力。 3. 像素艺术的兴起与复古复兴： 本文将深入探讨像素艺术作为早期街机游戏标志性视觉风格的演变，追溯其起源，探索其技术，并讨论像素艺术在现代游戏和数字艺术中的复兴。		

表7. 情境提示示例

后退提示

后退一步[8]提示法是一种提高学习能力的技巧，它通过引导学习能力者首先思考与当前具体任务相关的一般性问题，然后将该一般性问题的答案反馈到后续的具体任务提示中。这种“退一步”的方法使学习能力者能够在尝试解决具体问题之前激活相关的背景知识和推理过程。

通过考虑更广泛和更深层次的原则，学习管理者（LLM）可以得出更准确、更深刻的答案。退一步提示鼓励学习管理者进行批判性思考，并以新颖且富有创意的方式运用他们的知识。它改变了最终的提示方式，利用了学习管理者在任务参数中更多的知识，而这些知识在直接提示下是无法发挥作用的。

通过关注一般原则而不是具体细节，采用退后提示法，可以帮助减轻 LLM 回答中的偏见。

让我们来看几个例子，了解退一步提示法如何改善结果。首先，我们回顾一下传统的提示方法（表 8），然后再将其与退一步提示法（表 9）进行比较：

目标	为一款第一人称射击游戏编写一个关卡的故事情节。		
模型	双子座-专业版		
温度	1	代币限额	1024
Top-K	40	顶级球员	0.8
迅速的	请为一款第一人称射击游戏的新关卡撰写一段故事梗概，要求既具有挑战性又引人入胜。		
输出	关卡开始时，玩家的小队在人口密集的城市区域遭到敌方势力的伏击。玩家必须穿过迷宫般的后巷和破败的建筑，运用潜行和精准的射击技巧消灭敌人并避免被发现。玩家必须善用环境，设置陷阱、利用掩体并进行周密的行动才能生存下来。在此过程中，玩家会发现情报显示敌人正在策划一场毁灭性的袭击，玩家必须争分夺秒地收集证据并将其传回总部，阻止敌人实施计划。		

表 8. 传统提示与退一步提示的比较

当你把温度设为1时，你可能会得到各种各样的故事情节创意，但这些情节也相当随机和普通。所以，让我们退一步思考：

迅速的	根据流行的第一人称射击游戏，第一人称射击游戏中，哪些虚构的关键设定有助于构建具有挑战性和引人入胜的关卡故事情节？
输出	1. **废弃的军事基地**：一个庞大的后末日军事基地，到处都是变异士兵和失控的机器人，是进行高难度枪战的理想场所。 2. **赛博朋克城市**：霓虹闪烁的未来都市环境，高耸入云的摩天大楼和狭窄的巷道，以及经过赛博改造的敌人和黑客机制。 3. **外星飞船**：一艘巨大的外星飞船搁浅在地球上，里面有诡异的走廊、零重力区域，还有外星生物等着你去遭遇。 4. **丧尸横行的城镇**：一座荒凉的城镇，被成群的凶猛丧尸占领，玩家将进行激烈的近身战斗，并通过解谜找到安全通道。 5. **水下研究设施**：一个被水淹没的深海实验室，里面充满了变异的水生生物，需要隐秘行动和水下探索技能才能生存。

表 9. 提示自我一致性的示例

是的，这些主题似乎很适合第一人称视角游戏。让我们回到最初的问题，但这次我们将“后退一步”的答案作为背景，看看会得到什么结果。

迅速的	背景：第一人称射击游戏的 5 个引人入胜的主题： 1. **废弃的军事基地**：一个庞大的后末日军事基地，到处都是变异士兵和失控的机器人，是进行高难度枪战的理想场所。 2. **赛博朋克城市**：霓虹闪烁的未来都市环境，高耸入云的摩天大楼和狭窄的巷道，以及经过赛博改造的敌人和黑客机制。 3. **外星飞船**：一艘巨大的外星飞船搁浅在地球上，里面有诡异的走廊、零重力区域，还有外星生物等着你去遭遇。 4. **丧尸横行的城镇**：一座荒凉的城镇，被成群的凶猛丧尸占领，玩家将进行激烈的近身战斗，并通过解谜找到安全通道。 5. **水下研究设施**：一个被水淹没的深海实验室，里面充满了变异的水生生物，需要隐秘行动和水下探索技能才能生存。 从以下主题中选择一个，为第一人称射击游戏的新关卡编写一个段落式的故事情节，使其具有挑战性和吸引力。
输出	在幽暗深渊的中心，矗立着一座破败的水下研究设施，它见证了人类的野心及其带来的灾难性后果。黑暗笼罩着这里，故障机器的嗡嗡声不绝于耳，溺亡灵魂的回响萦绕不去，如今这里已成为深海噩梦的滋生地。玩家扮演一名装备精良潜水装备和实验性武器的精英海军陆战队员，肩负着深入这片恐怖水域的重任。凭借着坚韧不拔的意志和过人的智慧，玩家必须穿梭于危机四伏的走廊，解开神秘的谜题，并与潜伏在深海中的恐怖海怪展开殊死搏斗。从与深海的巨大压力抗争，到智胜狡猾的水生掠食者，在这片未知的海底世界，每一刻都在考验着人类的耐力和勇气。

表 10. 提示自我一致性的示例

那看起来像是一款很有趣的电子游戏！通过使用退后提示技巧，你可以提高提示的准确性。

思维链（CoT）

思维链（CoT）[9]提示是一种通过生成信息来提高语言学习模型推理能力的技术。**中间推理步骤**这有助于学习学习模型（LLM）生成更准确的答案。您可以将其与少样本提示相结合，以便在需要推理后才能作答的更复杂任务中获得更好的结果，因为零样本思维链对这类任务来说是一个挑战。

CoT（思维链）有很多优点。首先，它工作量小，效果却非常显著，而且能很好地与现成的学习学习模型（LLM）配合使用（无需进行微调）。CoT 提示还具有可解释性，因为您可以从 LLM 的响应中学习，并查看其推理步骤。如果出现故障，您也能轻松识别。思维链似乎还能提高在不同 LLM 版本之间切换时的鲁棒性。这意味着，与不使用推理链的提示相比，您的提示在不同 LLM 之间的性能波动应该更小。当然，它也有一些缺点，但这些缺点也比较容易理解。

LLM 响应包括思维链推理，这意味着更多的输出标记，这意味着预测需要更多的钱和更长的时间。

为了解释表 11 中的以下示例，我们首先尝试创建一个不使用 CoT 提示的提示，以展示大型语言模型的缺陷。

迅速的	我3岁的时候，我的伴侣比我大三倍。现在我20岁了。我的伴侣多大了？
输出	63岁

表 11. 一个试图解决数学问题的提示示例

哎呀，这显然是错误的答案。事实上，LLM（语言学习模型）在处理数学任务时经常遇到困难，甚至会给出错误的答案——即使是像两个数相乘这样简单的任务。这是因为它们接受的是大量文本训练，而数学运算可能需要不同的处理方法。那么，让我们看看中间的推理步骤能否改善输出结果。

迅速的	我3岁的时候，我的伴侣比我大三倍。现在我20岁了。我的伴侣多大了？我们一步一步来思考。
输出	1. 假设我现在的年龄是“x”岁。 2. 当我3岁的时候，我的年龄是3岁。 3. 当时我伴侣的年龄是我的 3 倍，也就是说，我伴侣的年龄是 3 * 3 = 9 岁。 4. 现在我20岁了，这意味着自从我3岁以来，我的年龄增加了20 - 3 = 17岁。 5. 因为我3岁时，我的伴侣9岁，而我的年龄已经增长了17岁，所以我的伴侣的年龄也一定增长了17岁。 6. 因此，我的伴侣目前的年龄是 9 + 17 = 26 岁。 我的伴侣今年26岁。

表 12. 思维链提示示例

好了，现在最终答案是正确的。这是因为我们明确指示LLM模型解释每个步骤，而不是直接返回答案。有趣的是，模型计算出了增加的17年。在我的脑海里，我会先计算我和伴侣年龄的差值，然后把差值加起来（20+(9-3)）。让我们帮助模型更像我一样思考吧。

表 12 展示了一个“零次”思维链的例子。当思维链提示与单次或少数几次提示结合使用时，效果会非常显著，如表 13 所示：

迅速的	问：我弟弟两岁的时候，我的年龄是他的两倍。现在我40岁了。我弟弟几岁了？我们一步一步来思考。 答：我弟弟2岁的时候，我2*2=4岁。我们年龄相差2岁，我比他大。现在我40岁了，所以我弟弟40-2=38岁。答案是38岁。 问：我3岁的时候，我的伴侣比我大三倍。现在我20岁了。我的伴侣多大了？让我们一步一步来思考。 步。 一个：
输出	我3岁的时候，我的伙伴3*3=9岁。我们相差6岁，我的伙伴比我大。现在我20岁了，所以我的伙伴20+6=26岁。答案是26岁。

表13. 单次提示引发思维链的示例

思维导图在各种应用场景中都非常有用。例如，在代码生成中，它可以将请求分解成几个步骤，并将这些步骤映射到特定的代码行。或者，当您拥有某种种子数据（例如“……”）时，它还可以用于创建合成数据。*该产品名为XYZ，请根据产品名称，写一段描述，指导模型如何进行假设。*一般来说，任何可以通过“讨论解决”的任务都适合用“思路链”的方法来解决。如果你能解释解决问题的步骤，那就试试“思路链”吧。

请参考笔记本。[10]托管在 GoogleCloudPlatform Github 存储库中，其中将更详细地介绍 CoT 提示：

在本章的最佳实践部分，我们将学习一些专门针对思维链提示的最佳实践。

自我一致性

虽然大型语言模型在各种自然语言处理任务中取得了令人瞩目的成功，但它们的推理能力通常被视为一种局限性，而这种局限性无法仅仅通过增加模型规模来克服。正如我们在之前的“思维链提示”部分中所了解到的，模型可以被提示生成类似人类解决问题的推理步骤。然而，“思维链提示”采用的是一种简单的“贪婪解码”策略，这限制了其有效性。自洽性[11]它结合了抽样和多数投票，生成多样化的推理路径，并选择最一致的答案。这提高了逻辑推理模型（LLM）生成答案的准确性和一致性。

自洽性可以给出答案正确的伪概率，但显然代价很高。

它遵循以下步骤：

1. 生成多样化的推理路径：LLM模型会多次接收到相同的提示信息。较高的温度设置会鼓励模型生成不同的推理路径和问题视角。

2. 从每个生成的响应中提取答案。

3. 选择最常见的答案。

我们来看一个邮件分类系统的例子，该系统将邮件分为“重要”或“不重要”。我们会多次向LLM（邮件识别大师）发送一个零概率的思维导图提示，以观察每次提交后的回复是否有所不同。请注意邮件中友好的语气、用词以及讽刺的语气。所有这些都可能误导LLM。

迅速的	电子邮件: ``` 你好， 我看到你用WordPress建站。它是一个很棒的开源内容管理系统。我以前也用过。它自带很多优秀的插件，而且安装起来也很容易。 我发现联系表单存在一个错误，当您选择姓名栏时就会出现此错误。请查看附件中的屏幕截图，其中显示了我在姓名栏中输入文本的情况。请注意我弹出的 JavaScript 警告框。 但除此之外，这真是一个很棒的网站。我很喜欢阅读它。请保留网站上的漏洞，因为它能让我有更多有趣的内容可读。 干杯， 黑客哈利。 ``` 请将上述邮件归类为“重要”或“不重要”。让我们一步一步思考并解释原因。 **（未完待续，请见下一页）**

输出 尝试 1	**第一步：确定邮件用途。** 这封邮件的目的是告知收件人其网站联系表单存在一个漏洞。发件人还提到，他们很喜欢浏览该网站，并且暂时会保留这个漏洞。 **第二步：评估漏洞的潜在影响。** 联系表单中的漏洞可能允许攻击者在网站上执行任意 JavaScript 代码。这可能导致攻击者控制网站、窃取敏感信息或篡改网站内容。 **步骤3：考虑发件人的可信度。** 邮件发件人自称是名叫哈里的黑客。然而，我们无法验证发件人的身份。发件人有可能是一位真心想帮助收件人的黑客，但也有可能是一位恶意攻击者，试图诱骗收件人点击恶意链接或下载恶意文件。 **结论：重要** 鉴于该漏洞的潜在影响以及发件人的可信度，该邮件应被归类为**重要**邮件。收件人应立即采取措施修复该漏洞，并保护其网站免受攻击。
输出 第二次尝试	* * 解释：** 1. **缺乏紧迫感：** 这封邮件没有传达任何紧迫感或时间敏感性。它以随意的问候语开头，并以平淡的语气呈现信息。 2. **非关键性错误报告：**联系表单中报告的错误被描述为有趣但并非关键性错误。邮件发送者甚至建议网站所有者可以保留该错误。 3. **缺乏个人影响：** 这封邮件并未表明对网站所有者有任何直接或个人影响。邮件中只是提到发件人喜欢浏览该网站。 **（未完待续，请见下一页）**

输出 第二次尝试	4. **未提出行动请求：**该邮件并未明确要求网站所有者采取任何行动。它既没有要求修复漏洞，也没有提供任何具体的改进建议。 5. **发件人意图：** 发件人的意图似乎更侧重于分享观察结果和表达对网站的兴趣，而不是提出严肃的担忧或请求帮助。 **结论：不重要**
输出 尝试 3	**第一步：确定邮件用途。** 这封邮件的目的是告知收件人其网站联系表单存在一个漏洞。发件人还提到，他们很喜欢浏览该网站，并且暂时会保留这个漏洞。 **第二步：评估漏洞的潜在影响。** 联系表单中的漏洞可能允许攻击者在网站上执行任意 JavaScript 代码。这可能导致攻击者控制网站、窃取敏感信息或篡改网站内容。 **步骤3：考虑发件人的可信度。** 邮件发送者自称是名叫哈里的黑客。然而，我们无法验证发送者的身份。发送者有可能是一位真心想帮助收件人的黑客，但也有可能是一位恶意用户，试图利用漏洞谋取私利。 * * 结论：** 基于以上因素，该邮件应被归类为**重要**邮件。联系表单中的漏洞存在严重的安全风险，且发件人的可信度未知。收件人应立即采取措施修复该漏洞。

表14. 提示自我一致性的示例

您可以使用上述提示，并尝试查看其是否返回一致的分类结果。根据您使用的模型和温度配置，结果可能为“重要”或“不重要”。

通过生成许多思维链，并取最常见的答案（“重要”），我们可以从 LLM 获得更一致的正确答案。

本例表明，通过考虑多种视角并选择最一致的答案，可以利用自洽提示来提高学习管理者（LLM）回答的准确性。

思想之树（ToT）

既然我们已经熟悉了思维链和自我一致性提示，让我们回顾一下思维树（ToT）。[12]它推广了CoT提示的概念，因为它允许LLMs同时探索多种不同的推理路径，而不仅仅是遵循单一的线性思维链。如图1所示。

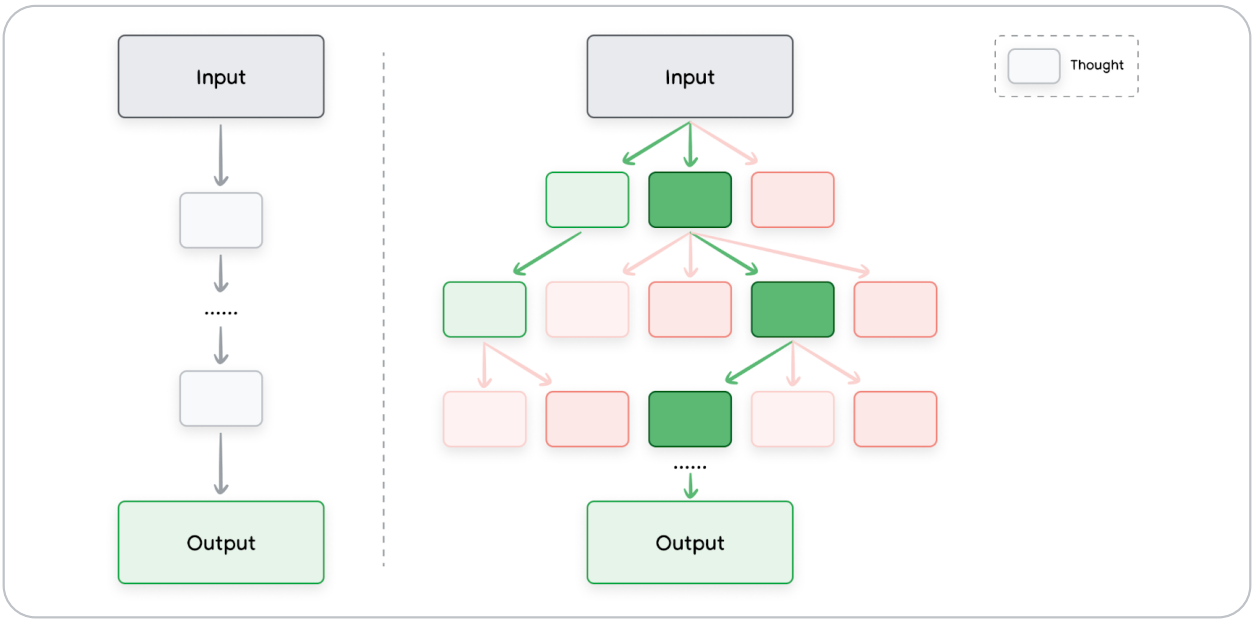


图 1. 左侧为“思维链”提示，右侧为“思维树”提示的可视化对比。

这种方法使得思维导图（ToT）特别适用于需要探索的复杂任务。它的工作原理是维护一个思维树，其中每个思维都代表一个连贯的语言序列，作为解决问题的中间步骤。然后，该模型可以通过从树中的不同节点分支出来，探索不同的推理路径。

有一本很棒的笔记本，更详细地展示了基于论文《大型语言模型引导的思维树》的思维树（ToT）。9

ReAct（理性与行动）

理性行动（ReAct）[10][13]提示是一种范式，它使 LLM 能够使用自然语言推理结合外部工具（搜 索、代码解释器等）来解决复杂任务，从而允许 LLM 执行某些操作，例如与外部 API 交互以检索信息，这是实现智能体建模的第一步。

ReAct 模拟人类在现实世界中的运作方式，例如我们进行语言推理并采取行动来获取信息。在多个领域，ReAct 的表现都优于其他快速响应式工程方法。

ReAct 提示的工作原理是将推理和行动结合起来，形成一个思维-行动循环。LLM 首先对问题进行推理，并生成行动计划。然后，它执行计划中的行动并观察结果。LLM 随后利用观察结果更新其推理，并生成新的行动计划。这个过程持续进行，直到 LLM 找到问题的解决方案。

要查看实际效果，您需要编写一些代码。在代码片段 1 中，我使用了 Python 的 langchain 框架以及 VertexAI（谷歌云人工智能平台）以及 谷歌搜索结果 pip 包。

要运行此示例，您必须创建一个（免费的）SerpAPI 密钥。https://serpapi.com/manageapi-key 并设置环境变量SERPAPI_API_KEY。

接下来，我们编写一些 Python 代码，让 LLM 学生来解决以下问题：*有多少孩子的爸爸是Metallica乐队的成员，而且是位名人？*

Python

```
从langchain.agents进口加载工具 从langchain.agents进口初始
化代理 从langchain.agents进口代理类型 从langchain.llms进口
VertexAI


提示 =Metallica乐队成员一共有几个孩子?

llm = VertexAI(温度=0.1) tools = load_tools(["serpapi"],
llm=llm)

agent = initialize_agent(tools, llm,
agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True)
agent.run(prompt)
```

示例 1：使用 LangChain 和 VertexAI 创建 ReAct Agent

代码片段 2 展示了结果。请注意，ReAct 执行了五次搜索。实际上，LLM 会抓取 Google 搜索结果来确定乐队名称。然后，它将结果列为观察结果，并将此思路串联起来进行下一次搜索。

未设置

进入新的 AgentExecutor 链……Metallica 共有 4 个
成员。
操作：搜索
输入操作：詹姆斯·海特菲尔德有几个孩子？ 观察结果：三个孩子

想法：Metallica乐队四分之一的成员有三个孩子。行动：搜索

输入操作：拉尔斯·乌尔里希有几个孩子？ 观察结果：3

想法：Metallica乐队四名成员中有两人有六个孩子。行动：搜索

操作输入：柯克·哈米特有几个孩子？
观察：哈米特自 1998 年以来一直与他的第二任妻子拉尼结婚。他们有两个儿子，安吉尔（生于 2006 年 9 月 29 日）和文森佐（生于 2008 年 6 月 28 日）。

想法：Metallica乐队四分之三的成员有8个孩子。行动：搜索

输入操作：罗伯特·特鲁希略有几个孩子？ 观察结果：2

思考题：Metallica乐队的4名成员共有10个孩子。最终答案：10

片段 2. 在终端中执行 ReAct Agent

代码片段 2 计算出 Metallica 乐队有四名成员。然后，它逐一查找每位成员，获取他们的子女总数，并将所有成员的子女总数相加。最后，它返回子女总数作为最终结果。

ReAct 提示的实际应用需要理解，您必须不断重新发送之前的提示/响应（并删减生成的额外内容），同时还需要使用适当的示例/说明来设置模型。请参阅笔记本。[14]托管在 Google Cloud Platform 的 GitHub 代码库中，其中更详细地展示了 LLM 的实际输入和输出，并提供了一个更详尽的示例。

自动提示工程

此时你可能意识到编写提示信息可能很复杂。如果能将这个过程自动化（编写提示信息来编写提示信息）岂不是更好？其实，有一种方法：自动提示信息工程（APE）。这种方法[15]不仅减轻了对人工输入的需求，而且还提高了模型在各种任务中的性能。

你会引导模型生成更多提示。评估这些提示，并可能修改其中较好的提示。然后重复此过程。

例如，你可以使用自动提示工程来帮助训练一个用于销售乐队周边T恤的网店聊天机器人。我们希望找出顾客在购买乐队周边T恤时可能使用的所有不同下单方式。

1. 编写用于生成输出变体的提示符。在本例中，我使用 gemini-pro 生成 10 条指令。参见表 15：

迅速的	我们有一个乐队周边T恤衫网店，为了训练聊天机器人，我们需要各种订购方式：“一件Metallica T恤衫，S码”。生成10个变体，语义相同但含义一致。
输出	1. 我想购买一件小号的 Metallica T 恤。 2. 我可以订购一件小号的 Metallica T 恤吗？ 3. 我想买一件小号的 Metallica T 恤。 4. 请给我一件 Metallica 乐队的 T 恤，小号的。 5. 我想要一件小号的 Metallica T 恤。 6. 我想订购一件小号的 Metallica T 恤。 7. 请给我一件小号的 Metallica T 恤。 8. 我想找一件小号的 Metallica T 恤。 9. 你能帮我弄一件小号的Metallica T恤吗？ 10.请给我一件Metallica小号T恤。

表15. 使用LLM生成提示

2. 根据选定的指标对所有教学候选人进行评分，例如，您可以使用 BLEU（双语评估替身）或 ROUGE（以回忆为导向的要点评估替身）。

3. 选择评估得分最高的指令候选方案。该方案将作为最终指令，供您在软件应用程序或聊天机器人中使用。您也可以对选定的指令进行微调并再次评估。

代码提示

Gemini 主要专注于基于文本的提示，其中也包括编写返回代码的提示。让我们前往 Vertex AI Studio 测试这些提示，并查看一些代码示例。

编写代码的提示

Gemini 还可以担任开发人员的角色，帮助您使用任何您选择的编程语言编写代码。作为开发人员，这可以帮助您加快代码编写速度。

想象一下，你电脑上有一个文件夹，里面有数百个文件需要重命名。逐个重命名会非常耗时。你懂一点 Bash，可以写个脚本来自动化这个过程，但这也可能需要一些时间。所以，我们来编写一个提示。你可以在面向用户的聊天机器人 Gemini 中编写提示，或者如果你更注重隐私，也可以在你的 Google Cloud 帐户中编写这些提示，然后打开 Vertex AI Studio。Vertex AI Studio 的优势在于你可以配置温度等参数。

姓名	1_bash_rename_files_in_folder		
目标	编写一个提示，指导用户使用 Bash 编写代码来重命名文件夹中的文件。		
模型	双子座-专业版		
温度	0.1	代币限额	1024
Top-K	不适用	顶级球员	1
迅速的	用 Bash 编写一段代码，提示用户输入文件夹名称。然后，它会获取该文件夹的内容，并将其中所有文件重命名，在文件名前添加“draft”字样。		
输出	`` bàsh # !/bin/bash # 询问文件夹名称 echo "请输入文件夹名称：" read folder_name # 检查文件夹是否存在 if [! -d "$folder_name"]; then echo "文件夹不存在。." exit 1 菲 # 获取文件夹中的文件列表 files=("$folder_name"/*) # 将每个文件重命名，在文件名前添加“draft” for file in "${files[@]}"; do new_file_name="draft_$(basename "$file")" mv "$file" "$new_file_name" 完毕 echo "文件重命名成功。"		

表 16. 使用 Gemini 进行编程

在我看来，这段代码看起来不错——甚至还有文档！但是，由于LLM模型无法进行推理，也无法重复使用训练数据，因此务必先阅读并测试你的代码。

我们翘首以盼的时刻，真的会到来吗？

我们先用一个只有几个文件的测试文件夹来尝试一下，这些文件需要从 filename.txt 重命名为 draft_filename.txt。

1. 复制表 16 中的输出（不包括“`**b灰**```将其粘贴到名为“text wrapper”的新文件中，然后将其粘贴到新文件中。**rename_files.sh**“
2. 打开终端窗口并输入：。**rename_files.sh**它会要求输入文件夹名称，例如 **测试**然后按回车键。
3. 脚本似乎运行正常。您将看到以下消息：**文件重命名成功。** 查看测试文件夹时，你会发现所有文件都已完美地重命名为…… **draft_filename.txt。**

成功了！

解释代码的提示

作为开发人员，在团队中工作时，您需要阅读其他人的代码。Gemini 也能在这方面为您提供帮助。让我们以表 16 中的代码输出为例，移除注释，然后让大型语言模型解释代码的运行机制，参见表 17：

姓名	1_bash_解释代码		
目标	请编写一个提示来解释 Bash 代码		
模型	双子座-专业版		
温度	0.1	代币限额	1024
Top-K	不适用	顶级球员	1
迅速的	请解释以下 Bash 代码： ``` # !/bin/bash echo "请输入文件夹名称：" read folder_name 如果 [! -d "$folder_name"]; 则 echo "文件 夹不存在。" 退出 1 菲 files=("$folder_name"/*) for file in "${files[@]}"; do new_file_name="draft_$ (basename "$file")" mv "$file" "$new_file_name" 完毕 echo "文件重命名成功。"		
输出	此 Bash 脚本执行以下操作： 1. **用户输入**： 它会提示用户使用 èchò 和 rèad 命令输入文件夹名称。输入的文件夹名称存储在 fòlder_namè 变量中。 2. **文件夹存在性检查**： - 它使用条件 [! -d “$folder_name”] 检查指定名称的文件夹是否存在。如果文件夹不存在，则显示错误消息（“文件夹不存在。”）并以退出代码 1 退出脚本。 （未完待续，请见下一页）		

输出	3. **文件清单**： 它使用 files=(“$folder_ name”/*) 命令列出指定文件夹中的所有文件。这将创建一个名为 files 的数组，其中包含文件夹中所有文件的路径。 4. **文件重命名**： - 它使用 for 循环遍历文件数组中的每个文件。 - 对于每个文件，它使用 ǹew_file_ name=”draft_$(basename “$file”)” ` 命令生成一个新的文件名。这个新文件名是通过在原始文件名前加上字符串“draft_”创建的。 然后，它使用 mv̀ “$file” “$new_file_name” 命令重命名文件，将文件移动到新的文件名。 5. **成功提示信息**： - 文件重命名完成后，会显示成功消息（“文件重命名成功。”），表明重命名操作已成功完成。

表 17. 代码解释提示

翻译代码的提示

表 16 中的 Bash 代码似乎运行正常。然而，如果该脚本能够提示输入文件名，并且理想情况下应该作为一个带有用户界面的独立应用程序运行，那么它的可重用性将会大大提高。作为起点，Python 比 Bash 更适合（Web）应用程序。语言学习硕士 (LLM) 可以帮助将代码从一种语言翻译成另一种语言。请参阅表 18 中的示例：

姓名	1_python_rename_files		
目标	编写一个提示，将 Bash 代码翻译成 Python 代码。		
模型	双子座-专业版		
温度	0.1	代币限额	1024
Top-K	不适用	顶级球员	1
迅速的	将以下 Bash 代码翻译成 Python 代码片段。 ``` bàsh # !/bin/bash echo "请输入文件夹名称：" read folder_name 如果 [! -d "$folder_name"]; 则 echo "文件 夹不存在。" 退出 1 菲 files=("$folder_name"/*) for file in "${files[@]}"; do new_file_name="draft_$ (basename "$file")" mv "$file" "$new_file_name" 完毕 echo "文件重命名成功。"		
输出	``` 蟒蛇 进口　　操作系统 进口　　闭 # 从用户处获取文件夹名称 folder_name = input("请输入文件夹 名称：") # 检查文件夹是否存在 if not os.path.isdir(folder_name): print("文件夹不 存在。") 　　退出(1) （未完待续，请见下一页）		

输出	# 获取文件夹中所有文件的列表 files = os.listdir(folder_name) # 遍历文件 for file in files: # 创建一个以“draft_”为前缀的新文件名 new_file_name = f"draft_{file}" # 将文件移动到新名称 shutil.move(os.path.join(folder_name, file), os.path.join(folder_name, new_file_name)) # 打印成功消息 print("文件重命名成功。") ```

表 18. 将 Bash 代码翻译成 Python 的提示

阅读并检查代码。复制提示符的输出并将其粘贴到一个新文件中： **file_renamer.py**打开终端窗口，执行以下命令来测试代码。**python file_renamer.py**。

笔记：在 Vertex AI 的语言工作室中输入 Python 代码时，您必须点击“Markdown”按钮。否则，您将收到纯文本，其中缺少正确的行缩进，而这对于运行 Python 代码至关重要。

用于调试和审查代码的提示

让我们手动修改一下表 18 的代码。它应该提示用户输入文件名前缀，并将前缀全部大写。参见示例代码片段 *3，但可惜的是，它现在返回 Python 错误！

Python

```
进口      操作系统
进口      闭

文件夹名称 =输入（请输入文件夹名称：）
前缀 =输入（“请输入要添加到文件名前面的字符串：”）text = toUpperCase(prefix)


如果not os.path.isdir(folder_name):
      打印（“文件夹不存在。”） 出口（1）


files = os.listdir(folder_name)

为了文件在文件：
      新文件名 =          f"{text}_{file}"

shutil.move(os.path.join(folder_name, file),
os.path.join(folder_name, new_file_name))

打印（文件重命名成功。）
```

片段 3：一个损坏的 Python 脚本

哎呀！这看起来像个bug：

未设置

```
以下Python代码会报错：

回溯（最近一次调用）：
  文件“/Users/leeboonstra/Documents/test_folder/rename_files.py”，第 7 行，在 <module> 中

      text = toUpperCase(prefix) NameError: 名
称“toUpperCase”未定义
```

片段 4. 我破坏了 Python 代码

我们来看看能否让大型语言模型来调试和审查代码。请看表 19：

姓名	1_python_debug_code		
目标	编写一个调试和审查 Python 代码的提示。		
模型	双子座-专业版		
温度	0.1	代币限额	1024
Top-K	不适用	顶级球员	1
迅速的	以下Python代码会报错： 回溯（最近一次调用）： 文件 ”/ Users/leeboonstra/Documents/test_folder/rename_files.py", 7, 在 线 <module> 中 text = toUpperCase(prefix) NameError: 名称“toUpperCase”未定义 找出代码中的错误并解释如何改进代码。 ``蟒蛇 进口 操作系统 进口 闭 # 从用户处获取文件夹名称 folder_name = input("请输入文件夹名称：") prefix = input("请输入要添加到文件名前面的字符串：") text = toUpperCase(prefix) # 检查文件夹是否存在 if not os.path.isdir(folder_name): print("文件夹不存在。") exit(1) # 获取文件夹中所有文件的列表 files = os.listdir(folder_name) # 遍历文件 for file in files: （未完待续，请见下一页）		

迅速的	# 创建一个以“draft_”为前缀的新文件名 new_filename = f"{text} _{file}" # 将文件移动到新名称 shutil.move(os.path.join(folder_name, file), os.path.join(folder_name, new_file_name)) # 打印成功消息 print("文件重命名成功。") ```
输出	代码调用了 `tǒUpperCasè` 函数将前缀转换为大写，但该函数未定义。要解决此问题，可以使用字符串类的 `ùpper()` 方法，该方法可以将给定的字符串转换为大写。修改后的代码如下所示： `` 蟒蛇 进口　　操作系统 进口　　闭 # 从用户处获取文件夹名称 folder_name = input("请输入文件夹 名称：") prefix = input("请输入要添加到文件名前面的字符串：") text = prefix.upper() # 检查文件夹是否存在 if not os.path.isdir(folder_name): 　　print("文件夹不存在。") exit(1) # 获取文件夹中所有文件的列表 files = os.listdir(folder_name) # 遍历文件 for file in files: 　　# 创建一个以“draft_”为前缀的新文件名 new_filename = f"{text} 　　_{file}" # 将文件移动到新名称 shutil.move(os.path.join(folder_name, file), （未完待续，请见下一页）

输出	os.path.join(folder_name, new_filename)) # 打印成功消息 print("文件重命名成功。") ``` 此外，代码还有一些其他的改进空间： 1. 新文件名的文件扩展名不会被保留。最好从原始文件名中提取扩展名，并将其附加到新文件名中。 2. 该代码假定文件夹名称中不包含空格。然而，最佳实践是妥善处理空格，将其替换为下划线或连字符。 3. 使用 f` 字符串语法进行字符串插值比传统的 +`运算符更易读、更简洁。 4. 这段代码没有处理重命名过程中可能出现的错误。最好将 shutil.move`调用放在 try...except`块中，以便捕获任何潜在的错误。 以下是根据这些建议改进后的代码： ``蟒蛇 进口 操作系统 进口 闭 # 从用户处获取文件夹名称 folder_name = input("请输入文件夹名称：") prefix = input("请输入要添加到文件名前面的字符串：") # 检查文件夹是否存在 if not os.path.isdir(folder_name): print("文件夹不存在。") exit(1) # 获取文件夹中所有文件的列表 files = os.listdir(folder_name) # 遍历文件 for file in files: （未完待续，请见下一页）

输出	# 获取文件扩展名 ext = os.path.splitext(file)[1] # 创建一个带有前缀和文件扩展名的新文件名 new_filename = f"{prefix}_{os.path.splitext(file)[0]}{ext}" # 构建原始文件和新文件的完整路径 old_path = os.path.join(folder_name, file) new_path = os.path.join(folder_name, new_filename) # 尝试移动文件 try: shutil.move(旧路径, 新路径) 除异常 e 外： print(f"移动文件 {file} 时发生错误：{e}") # 打印成功消息 print("文件重命名成功。") ``` 这些改进使代码更加健壮灵活，同时也更易于阅读和理解……（由于达到令牌限制，响应已被截断。如果需要更长的响应，请尝试增加令牌限制。）

表 19. Python 代码调试和审查提示

太好了。它不仅告诉我如何解决问题，还发现了我的代码中还有其他错误以及如何修复它们。提示的最后一部分还给出了改进代码的整体建议。

多模态提示如何?

代码提示仍然使用相同的常规大型语言模型。多模态提示是另一个不同的概念，它指的是使用多种输入格式来引导大型语言模型，而不仅仅是依赖文本。这可以包括文本、图像、音频、代码甚至其他格式的组合，具体取决于模型的功能和当前任务。

最佳实践

找到合适的提示需要反复尝试。Vertex AI 中的 Language Studio 是一个理想的工具，可以用来调整提示，并能针对各种模型进行测试。

遵循以下最佳实践，成为快速工程领域的专家。

请举例说明

最重要的最佳实践是在提示中提供（一次或几次）示例。这非常有效，因为它是一种强大的教学工具。这些示例展示了期望的输出或类似的反应，使模型能够从中学习并相应地调整自身的生成方式。这就像给模型一个参考点或目标，使其能够提高反应的准确性、风格和语气，从而更好地符合你的期望。

设计简约

提示语应该简洁明了，便于你和模特理解。一般来说，如果你自己都觉得困惑，模特很可能也会感到困惑。尽量避免使用复杂的语言，也不要提供不必要的信息。

例如：

前：

我现在正在纽约旅游，想了解一下有哪些好去处。我带着两个三岁的孩子。我们假期应该去哪里玩呢？

重写后：

担任旅游向导，向3岁儿童介绍纽约曼哈顿值得一游的好去处。

尽量使用描述动作的动词。以下是一些示例：

行动、分析、分类、归类、对比、比较、创造、描述、定义、评估、提取、查找、生成、识别、列出、测量、组织、解析、选择、预测、提供、排名、推荐、返回、检索、重写、选择、展示、排序、总结、翻译、写作。

请具体说明输出结果。

明确描述所需输出。过于简洁的指令可能无法充分指导语言学习模型，或者过于笼统。在提示中提供具体细节（通过系统提示或上下文提示）可以帮助模型专注于相关内容，从而提高整体准确率。

例如：

做：

请撰写一篇三段式的博客文章，介绍排名前五的电子游戏主机。文章应内容翔实、引人入胜，并以对话式的风格撰写。

不要：

撰写一篇关于游戏机的博客文章。

遵循指令而非约束

提示中使用指令和约束来指导 LLM 的输出。

- 一个**操作说明**它明确规定了响应所需的格式、风格或内容。它指导模型应该做什么或生成什么。

- A**约束**是一组对响应的限制或边界。它限制了模型不应该做什么或应该避免什么。

越来越多的研究表明，在引导过程中，侧重于积极的指导比过度依赖限制更为有效。这种方法符合人类更倾向于接受积极指导而非罗列禁忌清单的心理。

指令直接传达预期结果，而约束则可能让模型猜测哪些行为是允许的。指令赋予模型灵活性，鼓励在既定范围内进行创新，而约束则会限制模型的潜力。此外，一系列约束之间也可能相互冲突。

约束条件仍然很有价值，但在某些情况下却并非如此。例如，为了防止模型生成有害或带有偏见的内容，或者当需要严格的输出格式或样式时。

如果可能，请使用肯定式指令：不要告诉模型不要做什么，而是告诉它应该做什么。这可以避免混淆，并提高输出的准确性。

做：

请撰写一篇关于排名前五的游戏主机的单段博文。文章内容仅需包含主机名称、制造商、年份和总销量。

不要：

请撰写一篇包含1段文字的博客文章，介绍排名前5位的游戏主机。请勿列出游戏名称。

最佳实践是，首先要确定指令的优先级，清晰地说明你希望模型实现的功能，并且仅在出于安全、清晰度或特定要求时才使用约束条件。通过实验和迭代，测试不同的指令和约束条件组合，找到最适合你特定任务的方案，并将这些方案记录下来。

控制最大令牌长度

要控制生成的 LLM 响应的长度，您可以在配置中设置最大令牌限制，或者在提示符中显式请求特定长度。例如：

“用一条推文的长度解释量子物理学。”

在提示中使用变量

为了重用提示并使其更具动态性，可以在提示中使用变量，这些变量可以根据不同的输入进行更改。例如，如表 20 所示，一个提供城市信息的提示。与其在提示中硬编码城市名称，不如使用变量。变量可以避免重复工作，从而节省时间和精力。如果需要在多个提示中使用相同的信息，可以将其存储在一个变量中，然后在每个提示中引用该变量。将提示集成到您自己的应用程序中时，这种方法非常实用。

迅速的	变量 {city} = "阿姆斯特丹" 迅速的 您是一名导游。请告诉我关于{city}这座城市的一个事实。
输出	阿姆斯特丹是一座美丽的城市，遍布运河、桥梁和狭窄的街道。它拥有丰富的历史、文化和夜生活，是值得一游的好地方。

表20. 在提示中使用变量

尝试不同的输入格式和写作风格

不同的模型、模型配置、提示格式、用词和提交方式都可能产生不同的结果。因此，尝试不同的提示属性，例如样式、用词和提示类型（零次机会提示、少量机会提示、系统提示），非常重要。

例如，一个旨在生成关于革命性视频游戏机世嘉Dreamcast的文本的提示，可以表述为：**问题**，一个**陈述**或**操作说明**从而导致不同的输出结果：

- **问题：**世嘉Dreamcast是什么？它为什么是一款具有革命性意义的游戏机？
- **陈述：**世嘉Dreamcast是世嘉公司于1999年推出的第六代家用游戏机。它……

- **操作说明：**请用一段话描述世嘉Dreamcast游戏机，并解释它为何具有革命性意义。

对于包含分类任务的少量提示，应混合使用不同的类别。

一般来说，样本顺序对小样本测试的影响不大。但是，在进行分类任务时，务必确保样本中所有可能的响应类别都交错排列。否则，模型可能会过度拟合样本的特定顺序。通过交错排列可能的响应类别，可以确保模型学习识别每个类别的关键特征，而不是简单地记忆样本顺序。这将使模型在未见过的数据上获得更稳健、更具泛化能力的性能。

一个好的经验法则是先拍摄 6 个小样本，然后从那里开始测试准确性。

适应模型更新

密切关注模型架构变更、新增数据和功能至关重要。尝试使用新版本模型，并调整提示信息以更好地利用新模型特性。Vertex AI Studio 等工具非常适合存储、测试和记录各种版本的提示信息。

尝试不同的输出格式

除了提示的输入格式之外，还可以尝试不同的输出格式。对于提取、选择、解析、排序、排名或分类数据等非创造性任务，可以尝试以结构化格式（例如 JSON 或 XML）返回输出结果。

从提取数据的提示中返回 JSON 对象有一些好处。在实际应用中，我不需要手动创建这种 JSON 格式，我可以按排序顺序返回数据（这在处理日期时间对象时非常方便），但最重要的是，通过提示输入 JSON 格式，可以强制模型创建结构并减少错误。

少样本提示部分的表 4 显示了如何返回结构化输出的示例。

与其他快速工程师一起进行实验

如果你需要想出一个好的题目，不妨让几个人都尝试一下。当每个人都遵循最佳实践（如本章所述）时，你会发现不同题目的尝试结果会有所不同。

CoT最佳实践

对于 CoT 提示，需要在推理之后给出答案，因为推理的生成会改变模型在预测最终答案时获得的标记。

使用 CoT 和自洽性，你需要能够从提示中提取最终答案，并将其与推理过程分开。

对于 CoT 提示，将温度设置为 0。

思维导图提示基于贪婪解码，根据语言模型分配的最高概率预测序列中的下一个词。一般来说，当使用推理得出最终答案时，可能只有一个正确答案。因此，温度应该始终设置为 0。

记录各种提示尝试

最后一个建议在本章前面已经提到过，但我们再怎么强调它的重要性都不为过：详细记录你的提示尝试，这样你就可以随着时间的推移了解哪些方面做得好，哪些方面做得不好。

提示输出在不同的模型、不同的采样设置，甚至同一模型的不同版本之间都可能存在差异。此外，即使是对同一模型使用完全相同的提示，输出句子的格式和用词也可能存在细微差别。（例如，如前所述，如果两个词元的预测概率相同，则可能会随机打破这种平局。这可能会影响后续的预测结果。）

我们建议以表 21 为模板创建一个 Google 表格。这种方法的优点在于，当您不可避免地需要重新审视提示工作时，就能拥有完整的记录——无论是为了将来继续进行（您会惊讶于短暂休息后会忘记多少东西），测试不同版本模型的提示性能，还是帮助调试未来的错误。

除了此表中的字段外，跟踪提示的版本（迭代次数）、记录结果（OK/NOT OK/SOMETHA OK）以及记录反馈的字段也很有帮助。如果您有幸使用 Vertex AI Studio，请保存您的提示（使用与文档中列出的相同的名称和版本），并在表中跟踪指向已保存提示的超链接。这样，您只需单击一下即可重新运行提示。

在从事某项工作时*检索增强生成*此外，您还应该捕获 RAG 系统的具体方面，这些方面会影响插入到提示中的内容，包括查询、块设置、块输出和其他信息。

当你觉得提示语接近完美时，就把它添加到你的项目代码库中。在代码库中，将提示语保存在与代码分开的文件中，这样更易 于维护。最后，理想情况下，你的提示语应该成为已运行系统的一部分，作为提示语工程师，你应该依靠自动化测试和评估流程来了解你的提示语在实际任务中的通用性。

提示符设计是一个迭代过程。设计并测试不同的提示符，分析并记录结果。根据模型的性能改进提示符。不断试验，直到获得理想的输出。当您更改模型或模型配置时，请返回并继续试验之前使用的提示符。

姓名	[提示符的名称和版本]		
目标	[用一句话解释本次尝试的目标]		
模型	[所用型号的名称和版本]		
温度	[取值范围为 0 到 1]	代币限额	[数字]
Top-K	[数字]	顶级球员	[数字]
迅速的	[请完整填写提示信息]		
输出	[写出输出结果或多个输出结果]		

表21. 提示信息记录模板

概括

本白皮书探讨了提示工程。我们学习了各种提示技术，例如：

- 零提示
- 少枪提示

- 系统提示
- 角色提示
- 情境提示
- 后退一步提示
- 思路链
- 自我一致性
- 思想之树
- React

我们甚至研究了如何实现提示信息的自动发送。

白皮书随后探讨了人工智能生成面临的挑战，例如提示信息不足时可能出现的问题。最后，我们总结了如何成为一名更优秀的提示信息工程师的最佳实践。

尾注

1. Google，2023，《Gemini by Google》。可访问：https://gemini.google.com。

2. Google，2024，《Gemini for Google Workspace 提示指南》。可访问以下网址获取：https://inthecloud.withgoogle.com/gemini-for-google-workspace-prompt-guide/dl-cd.html。

3. Google Cloud，2023，《提示简介》。可访问以下网址获取：https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design。

4. Google Cloud，2023，《文本模型请求正文：Top-P 和 Top-K 采样方法》。可访问：https://cloud.google.com/vertex-ai/docs/generative-ai/model-reference/text#request_body。

5. Wei, J. 等人，2023，《零样本——微调语言模型是零样本学习器》。可访问：https://arxiv.org/pdf/2109.01652.pdf。

6. Google Cloud，2023，《Google Cloud 模型花园》。网址：https://cloud.google.com/model-garden。

7. Brown, T. 等人，2023，《少样本学习——语言模型是少样本学习者》。可访问：https://arxiv.org/pdf/2005.14165.pdf。

8. Zheng, L. 等人，2023，《退一步思考：在大型语言模型中通过抽象激发推理》。可访问：https://openreview.net/pdf?id=3bq3jsvcQ1

9. Wei, J. 等，2023，《思维链提示》。可访问以下网址获取：https://arxiv.org/pdf/2201.11903.pdf。

10. Google Cloud Platform，2023，《Chain of Thought》和 React。可访问：https://github.com/GoogleCloudPlatform/generative-ai/blob/main/language/prompts/examples/chain_of_thought_react.ipynb。

11. Wang, X. 等，2023，《自洽性提升语言模型中的思维链推理》。可访问：https://arxiv.org/pdf/2203.11171.pdf。

12. Yao, S. 等，2023，《思维之树：基于大型语言模型的刻意问题解决》。可访问：https://arxiv.org/pdf/2305.10601.pdf。

13. Yao, S. 等，2023，《ReAct：语言模型中推理与行动的协同作用》。可访问：https://arxiv.org/pdf/2210.03629.pdf。

14. Google Cloud Platform，2023，《高级提示：思维链和 React》。可访问：https://github.com/GoogleCloudPlatform/applied-ai-engineering-samples/blob/main/genaion-vertex-ai/advanced_prompting_training/cot_react.ipynb。

15. Zhou, C. 等人，2023，《自动提示工程——大型语言模型是人类级别的提示工程师》。可访问：https://arxiv.org/pdf/2211.01910.pdf。