**提示词编写方法论**

**提示词设计要素：**

一份完整的应用级提示词包含5个主要素。分别是角色、任务、目标、背景、约束。

下面我将会对这几个要素进行详细的说明。

**角色：**

对角色的理解，我是这样的：大模型是一个拥有庞大知识库的多面助手。由于知识库过于庞大，没办法根据你的需求及时找到你想要的内容。但是我们如果将这个助手设置为特定领域的专家。他就会记起关于这个领域的相关知识。

设置角色可以有助于设定语气、明确目标、设定风格和提高专业性，从而提高大模型回答的准确率。

所以我们在编写时，可以将大模型设置为与任务目标相符的角色。

在设置角色的同时，设置需要大模型为我们完成的任务可以让角色的定位更加清晰。

**案例：**

我要编写一份文兜智写功能宣传稿提示词。

可以这样设置角色：

|  |
| --- |
| Shell 你是一位专业的视频脚本编辑，你能够根据我的需求编写剪辑视频所需要的口播脚本。 |

**任务：**

任务这一要素很好理解，就是大模型需要完成的具体工作或活动。也是整个提示词的工作方向。

需要注意的是，任务描述应清晰具体，包含需要执行的行动或解决的问题。

**案例：**

我要编写一份文兜智写功能宣传稿提示词。

可以这样设置任务：

|  |
| --- |
| Shell 现在需要你帮我编写一份用于宣传文兜智写新功能的视频口播稿。 |

**目标：**

目标这个要素是提示词的最终目的或期望达到的结果，是整个提示设计的核心驱动力。

明确目标有助于引导内容生成的方向，确保输出结果符合预期。

需要注意的是，设置的目标除了要清晰明确还要有可实现性。

**案例：**

我要编写一份文兜智写功能宣传稿提示词。

可以这样设置目标：

|  |
| --- |
| Shell - 介绍文兜智写的新功能，帮助用户理解如何使用新功能。  - 吸引更多的用户使用文兜智写。 - 针对关注AI编标领域的人群，提供有价值的内容，满足他们的知识需求。 |

**背景：**

背景这一要素是为了向大模型说明任务的环境，便于模型生成的内容更加贴合实际需求。

我们在编写背景时，要确保背景信息简明扼要，但包含所有关键细节，以避免误解。

**案例：**

我要编写一份文兜智写功能宣传稿提示词。

可以这样编写背景：

|  |
| --- |
| Shell 我是文兜智写（注释：国内领先的超长文生成式投标ai工具）的开发者，我们上线了一个新功能，现在需要编写一份用于宣传文兜智写新功能的视频口播稿。 |

**约束：**

约束这个要素的定义较为宽泛，在我认为凡是在执行任务过程中必须遵循的规则、限制或条件都可以被称之为提示词的约束。它会将输出结果的范围不断的缩小直至找到我们需要答案。

需要注意的是，约束基本上是提示词调优的重灾区，所以一份合理的约束需要我们不断的去调试。

约束可能包括：技能、标准、工作流程、要求等等。在实际使用的时候需要根据需求进行设置。

**案例：**

我要编写一份文兜智写功能宣传稿提示词。

可以这样编写约束：

|  |
| --- |
| Shell ## Constraints： - 文章目的：用于宣传文兜智写新功能。 - 目标读者：投标行业从业者，文兜智写的用户。 - 可读性强：简短句式，多用清晰、具体的表达。避免过度官方，增加读者亲近感，同时保持行业权威性。 - 视频模式：仅有功能教学界面和口播声音。  ## Function Description： - 我们上线了一款新功能：一键扩写。用户可以上传自己需要扩写的文档，点击确认扩写，文兜智写会立即将上传的文档的文本字数扩写到原来的2倍。 - 文档大小要低于10M。 - 上传文档只支持word格式。 - 从上传到扩写完成大约需要5分钟左右。  ## Skills： - 专业的软件教学视频文案编辑技能。 - 擅长使用通俗易懂的语言来让用户快速了解我们要表达的内容。  ## Require - 编写一份不少于3000字的视频脚本，其中口播内容不少于2000字。 - 仅输出视频脚本，避免输出其他内容。 |

**提示词框架模版：**

结构清晰的提示词可以有助于大模型对需求的理解。

下面我来介绍几我常用的提示词模版：

**框架一：**

如果你的需求涉及复杂的推理，建议使用以下完整的提示词框架：

|  |
| --- |
| Shell # Role：[角色]  ## Description: - 你是一位[角色]，你能够根据我的需求帮我[任务]  ## Background :  - [背景]  ## Goals: - [目标1] - [目标2] - [目标3] ……  ## Constraints： - [约束1] - [约束2] - [约束3] ……  ## Skills： - [技能1] - [技能2] - [技能3] ……  ## Require: - [要求1] - [要求2] - [要求3] ……  ## Workflow:(如果编写的提示词涉及对话，可以设置工作流来引导用户使用) 第一步，XXX 第二步，XXX 第三步，XXX …… |

**框架二：**

如果你需求较为简单，就不需要编写复杂的提示词，只需将任务和需求说明清楚即可。

|  |
| --- |
| Plain Text 你是一位[角色]，你能够根据我的需求帮我[角色能力]。 下面，请你根据“要求”帮我[任务]。 要求： - [要求1] - [要求2] - [要求3] …… |

**提示词技巧：**

**1.使用有序列表与无序列表列出不同的项**

在使用大模型的过程中，我们有时需要将一个任务列出很多项，如做一件事情的步骤、提醒大模型在回答中需要注意哪些事项等，就需要大模型按条列出，这样才更清晰和醒目。这时，我们可以使用有序列表或无序列表来提示大模型。

**（1）有序列表**

有序列表的使用非常简单。对于有顺序的元素，如做一件事情的步骤，我们可以使用有序列表来列出它们的顺序，这样步骤会更清晰。对于有序列表，可以用数字序号的形式来表示。

例3-4 使用有序列表列出不同的项

Prompt：

|  |
| --- |
| Shell 请根据下面的“把大象塞进冰箱的步骤”，写一篇记叙文，描述今天早上我是如何把大象塞进冰箱的。 把大象塞进冰箱的步骤：  ###  1.打开冰箱门。 2.把大象塞进冰箱。 3.关上冰箱门。  ### |

在这个示例中，“把大象塞进冰箱的步骤”是有先后顺序的，我们可以用数字序号标出它们。

**（2）无序列表**

如果一个列表并没有先后顺序或重要程度可言，而只是列出一些项，如注意事项，就可以使用无序列表来列出不同的项。对于无序列表，可以使用“-”来列出它们。

例3-5 使用无序列表列出不同的项

Prompt：

|  |
| --- |
| Shell 请根据下面的“把大象塞进冰箱的步骤”，写一篇记叙文，描述今天早上我是如何把大象塞进冰箱的。 把大象塞进冰箱的步骤：  ###  1.打开冰箱门。 2.把大象塞进冰箱。 3.关上冰箱门。  ###  -使用夸张，生动的语言，突出故事的戏剧性。 -对大象的外貌与体态进行详细的描写。 |

在这里，我们对结果的要求做了额外补充，但是这两项要求并没有先后顺序或重要程度之分，因此可以使用无序列表。

有序列表的格式与无序列表的格式仍然来自Markdown，读者可以根据自己的兴趣阅读更多Markdown的语法规则。

**2.定义未知名词**

由于大模型的知识是有时间截止日期的，例如chatgpt的认知是截止到2023年10月，对于这以后发生的事是不知道的。所以我们要定义大模型不知道到的名词。

例如：在编写文兜智写功能宣传稿提示词的时候，我会提到文兜智写，但是大模型并不知道什么是文兜智写，如果这样直接提问，获得的结果会不理想，所以我们对文兜智写进行备注：国内领先的超长文生成式投标ai工具。至于备注的形式多种多样，只需让大模型能够理解即可。

**3.否定句式**

大模型对于例如：不要、不用等否定句式的执行能力较差。我可以用“避免”等非直接否定词来限制大模型的生成。

**4.token**

大模型处理文字的单位不是字数，所以实际生成的字数会和提示中规定的字数有所差异（类似于ChatGPT的大语言模型处理文字时使用的单位是“token”，例如，GPT-4-32k的上下文处理能力是32000token，约为25000个英文单词）。不过，总的来说，我们规定的字数还是会影响回答的长度，所以这仍然是控制回答长度的一个有效的手段。

在实际应用中，我们需要进行一定量的测试来确定我们最终的字数限制。

**5.输入输出的限制**

大模型的输入输出能力是有限制的，关于输入，目前很多模型都支持128k的输入量，单次可以接收几万字数的输入。关于输出，目前大部分的模型输出极限在1500字左右，也有少部分模型能够达到3000字左右的输出。例如：gpt-4o-2024-11-20

所以我们在输入文本和设置字数限制的时候要考虑模型的输入和输出能力。

**6.提示链(Prompt Chain)**

在我们处理复杂问题的时候，大部分模型没办法同时处理多个问题，所以我们可以将复杂问题拆解成多个单一的简单问题，然后分别用提示词进行实现和组合，最后完成复杂问题。这一方法在Epe中十分的重要。像文兜智写就是由几份核心提示词和四十多份辅助提示词组合而成。

需要注意的是，提示词并不是万能的。在实际需要中，其实有相当一部分内容是提示词无法完成的，需要借助其他工具和提示词组合完成。

**7.提示词的迭代**

由于模型的更新升级，提示词也要进行迭代升级。我们需要定期的去维护提示词，确保其能够正常运行。这是Epe中非常见的工作。文兜智写的提示词在短短一年多的时间中迭代了数百个版本。

ps：随着模型的更新升级，对提示词的要求会越来越低，可能在几年后，只需要一些简单的语句就能够完成一项复杂的任务。

**8.文本分割符**

在编写提示词的过程中，我们使用的提示一部分是指令，另一部分是上下文。我们可以用“###”或“ “” ”，或者其他任何可以分割文本的分隔符将指令与上下文分割开。

举个例子，如果我们希望大模型根据一些材料来写一篇文章，可以使用下面的提示。

例3-1 使用文本分隔符分割指令和上下文

Prompt：

|  |
| --- |
| Shell 请根据下面的“把大象塞进冰箱的步骤”，写一篇记叙文，描述今天早上我是如何把大象塞进冰箱的。 把大象塞进冰箱的步骤：  ###  把大象塞进冰箱的步骤是，先打开冰箱门，然后把大象塞进冰箱，再关上冰箱门。  ### |

在这个示例中，“请根据下面的‘把大象塞进冰箱的步骤’，写一篇记叙文，描述今天早上我是如何把大象塞进冰箱的”是指令，而“把大象塞进冰箱的步骤”是上下文。上下文还有可能是需要大模型读的合同、程序源代码等信息，它们可能会非常长，使用分隔符可以让大模型抓住重点。

**9.使用标记语言标记输入格式**

在使用大模型的过程中，虽然我们的提示中往往会有一些重点，但是大模型偶尔会忽略它们，或者对我们想要强调的重点的注意力不够。这时我们可以使用“\*\*”来加粗文本，即在重点词或短语前后添加两个星号，让大模型注意到它们，如图所示。

这实际上是一个叫作Markdown的标记语言的加粗语法，在使用大模型时非常有用。



对需要强调的内容加粗后，大模型会对这些内容倾注更多的注意力，在回答中会非常明显地体现这一点。

下面是一个示例，如果我们希望提醒大模型要写记叙文，可以使用一对“\*\*”来强调关键词。

例3-3 使用“\*\*”在提示中强调内容

Prompt：

|  |
| --- |
| Shell 请根据下面的“把大象塞进冰箱的步骤”，写一篇\*\*记叙文\*\*，描述今天早上我是如何把大象塞进冰箱的。 把大象塞进冰箱的步骤：  ###   把大象塞进冰箱的步骤是，先打开冰箱门，然后把大象塞进冰箱，再关上冰箱门。  ### |

那么，为什么这种方法会有效呢？答案就在于我们使用的其实是Markdown语法——一种“标记语言”。标记语言是一种专门用于定义数据结构和展示方式的计算机语言。这种语言与我们通常所说的编程语言有所不同，它并不用于进行逻辑运算或控制程序流程，而是主要用于描述、组织和展示数据。

你可以将标记语言想象为一种为文本附加的“指示标签”，它告诉计算机应如何处理或显示这部分文本。由于大模型可以理解这些标记语言，它自然就会根据标记语言的指示来处理我们的输入。

**10.提供案例**

在编写提示词的时候，经常会出现明明已经描述的很清楚的，但是大模型就是没有按照自己的想法去输出内容。这时，你可以在提示词中加上一个结果案例，让大模型根据案例来生成我们想要的结果。

**11.逐步细化提示词：**

在编写提示词的时候，我们可以先从宽泛的提示词开始，逐步增加细节和限制条件。

例如：从“描述气候变化”开始，逐步细化为“描述气候变化对海洋生态系统的影响”。

**12.调整提示词的语气和风格：**

有时我们需要编写一些生成文案/文章的提示词，在编写这类提示词时。我们可以根据目标受众调整提示词的语气（如正式、非正式）和风格。

例如：对年轻用户使用更轻松的语气，而对专业人士使用正式语气。

**13.利用反向工程：**

反提示词工程是我们在分析提示词工作中非常重要的方法，通过分析成功的输出，反向推导出有效的提示词结构。

在分析其他ai应用软件时，反提示词工程可以帮助我们根据软件生成的效果反推出提示词的大概逻辑。

该方法需要大家在提示词编写工作中有一定的经验积累。

例如：如果某个提示词生成了高质量的文本，分析其结构并在其他场景中应用。

**14.上下文设定：**

我们可以在提示词中提供足够的上下文信息，以便模型更好地理解任务。

例如：在提示词中加入背景信息，如“假设你是一位历史学家，解释……”

**15.使用自动化工具：**

使用自动化工具来生成和评估提示词的效果。利用自动化脚本批量生成提示词并评估其性能。

**提示词分析方法：**

对提示词调优是提示词工程中最重要的一步，每份优秀的提示词都经历大量调优工作的检验，而调优提示词的核心能力就是分析提示词的逻辑与问题，下面我将详细说明如何分析提示词。

**1.明确目标：**

**（1）确定提示词的最终目标**

* **识别核心需求**：首先，明确提示词的核心需求是什么。你需要问自己或团队，提示词是为了达到什么目的？是为了生成特定的信息，还是为了引导用户完成某个任务？
* **定义成功标准**：明确什么样的结果被认为是成功的。例如，如果提示词是用于生成文本，成功的标准可能是生成的文本是否符合主题、是否具有逻辑性等。

**（2）目标的具体化**

* **SMART原则**：确保目标是具体的（Specific）、可衡量的（Measurable）、可实现的（Achievable）、相关的（Relevant）和有时限的（Time-bound）。
* **优先级排序**：如果提示词有多个目标，优先考虑哪些目标是最重要的，并专注于这些目标进行优化。

**（3）确保目标与上下文一致**

* **上下文分析**：提示词需要在特定的上下文中使用，因此分析提示词所处的环境和使用场景是关键。确保提示词的目标与其使用的情境相符。
* **一致性检查**：确保提示词的目标与整体产品或项目的目标一致，避免出现目标冲突。

**（4）目标沟通与验证**

* **团队沟通**：与团队成员沟通并确认目标，确保所有相关人员对提示词的目标有一致的理解。
* **用户验证**：通过原型测试或用户反馈，验证目标是否符合用户的期望和需求。

**2.逻辑结构分析：**

**（1）分解提示词**

* **元素识别**：将提示词分解成基本元素或组件，分析每个元素的功能和作用。例如，识别出目标、任务、约束、背景等。
* **模块化分析**：将提示词划分为不同的模块或段落，分析每个模块的独立性和与其他模块的关系。

**（2）因果关系检查**

* **前因后果**：检查提示词中是否存在因果关系，确保信息的前后顺序合理。例如，确保因果关系明确，避免“因为”与“所以”之间的逻辑断裂。
* **条件与结果**：如果提示词包含条件语句，确保条件与结果之间的因果关系清晰且合理。

**（3） 条件语句分析**

* **条件完整性**：检查条件语句是否完整，是否涵盖了所有可能的情况。
* **条件逻辑**：验证条件的逻辑性，确保条件的使用不会导致矛盾或偏差。

**（4）逻辑一致性**

* **一致性检查**：确保提示词在不同部分中使用的术语和概念一致，避免由于术语不一致导致的误解。
* **逻辑连贯性**：检查提示词的逻辑是否连贯，确保每个部分都自然地连接到下一个部分。

**（5） 信息优先级**

* **信息排序**：根据重要性对信息进行排序，确保最重要的信息在提示词中被优先传达。
* **层次结构**：建立信息的层次结构，帮助用户快速抓住重点信息。

**（6） 逻辑错误识别**

* **矛盾检查**：识别提示词中可能存在的矛盾或冲突，并进行调整。
* **歧义消除**：消除提示词中可能存在的歧义，确保每个信息点的含义清晰明确。

**3.语言清晰度：**

**（1）简洁性：**

* **避免冗余**：去除不必要的词语和重复的表达，确保每个词都有其存在的意义。
* **直截了当**：使用简单、直接的句子结构，避免复杂的句法。

**（2）具体性：**

* **明确指代**：使用明确的名词和代词，避免模糊不清的指代。
* **精确描述**：提供具体的细节以避免误解。

**（3）词汇选择：**

* **避免模糊词**：尽量避免使用“可能”、“某些”等模糊词语，除非必要。

**（4）清晰的指令：**

* **明确的操作步骤**：如果提示词涉及操作或步骤，确保每一步都清晰明确。
* **使用主动语态**：主动语态通常比被动语态更直接和清晰。

**4.用户反馈分析：**

* 收集用户对提示词的反馈，分析用户在使用提示词时遇到的问题。
* 根据反馈调整提示词的结构和内容。