

本地提示优化

Yash Jain Vishal Chowdhary

Microsoft

Abstract

近年来，使用提示来引导大型语言模型的输出已显著增加。然而，即使是最优秀的专家也难以选择正确的词汇来构建所需任务的提示。为了解决这个问题，驱动 LLM 的提示优化成为一个重要的问题。现有的提示优化方法是在全局上优化提示，在解决复杂任务时，所有提示标记都必须在庞大的词汇中进行优化。庞大的优化空间（标记）导致对更好提示的指导不足。在这项工作中，我们引入了本地提示优化（LPO），它可以与任何通用自动提示设计方法集成。我们识别出提示中的优化标记，并促使 LLM 仅在其优化步骤中关注这些标记。我们观察到，在数学推理（GSM8k 和 MultiArith）以及 BIG-bench Hard 基准上，各种自动提示设计方法的表现显著提高。此外，我们展示了 LPO 比全局方法更快地收敛至最佳提示。

1 介绍

大型语言模型 (LLM) 无处不在。LLM 正在自动化几年前需要专业模型完成的所有任务 (??)。控制 LLM 输出的最简单、最便宜的方法是进行提示工程 (????)。不幸的是，编写提示是极其棘手的 (?)。虽然提示是用英文写的，但那些有效地具有相同意义的词语选择在提示在任务中的性能上会产生巨大差异 (???)。此外，LLM 本质上对其自身的词汇存在偏向，这使得任务更具挑战性。因此，LLM 被用于在一个称为提示优化的过程中修改提示 (?)。

提示优化技术遵循如图 1 所示的两步过程。首先，提示在训练集上进行验证，识别错误预测项。可选择地，增加一个反馈步骤，通过查询 LLM (??) 获得自然语言反馈，即“文本梯度”。最后，使用文本梯度（错误示例或自然语言反馈）优化提示，从而获得优化后的提示。该循环重复固定次数。传统的提示优化技术 (????) 在全球范围内优化提示，即对提示中的所有标记进行变异。然而，在复杂问题的词汇搜索中优化提示中的所有标记，使得提示优化非常具有挑战性。此外，对于许多生产应

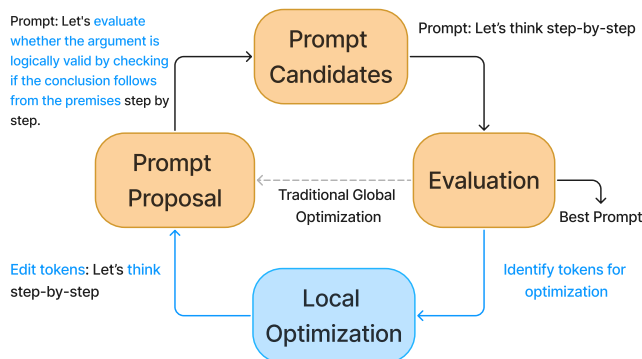


Figure 1: 本地提示优化集成在一个通用的自动提示工程框架中。

用，期望仅优化提示的某些子部分，而保持其他部分静止。这样做需要我们限制“提示提议”在提示的某些子部分的范围，因此需要局部提示优化 (LPO)。因此，我们减少了 LLM 的优化空间（标记）以简化问题并控制提示的编辑方向。在这项工作中，我们对比局部提示优化与全局提示优化，评估其效力和缺陷。我们将局部优化整合到三个自动提示优化算法中，并在 GSM8k (?)、MultiArith (?) 和 BIG-bench hard (?) 基准上进行评估。我们强调局部优化能更快地收敛到最佳提示，同时提升提示性能。最后，我们通过在一个生产提示上评估局部优化在实际应用中的效果。

2 背景与方法

在本节中，我们将描述一个自动提示工程的一般框架(?)并重点突出该框架中的不足。在此基础上，我们将介绍局部提示优化。

2.1 自动提示构建

给定一个数据集 $D = (x, y)$ ，提示工程系统旨在找到一个提示 p^* ，以最大化评估函数 f 上的得分。具体来说，

$$p^* = \arg \max_p \sum_{(x,y) \in D} f(\mathcal{M}_{task}(x;p), y) \quad (1)$$

，其中 $\mathcal{M}_{task}(x; p)$ 是任务模型 $\mathcal{M}_{task}$ 在以提示 p 为条件时生成的输出。

一个通用的自动提示工程系统由三个部分组成：提示初始化、提示提议和搜索过程。

提示初始化： 提供了一个需要优化的初始提示给自动提示系统。提示初始化可以通过人工书写的指令来完成，也可以是来自数据集的几个示例 D (?)。

(2) 提示提案： 在这一步，新提示生成开始。在任何时间步 t ，都会从一组候选提示 p^t 生成一组新提示 $p^{(t+1)}$ 。使用一个提议 LLM $\mathcal{M}_{proposal}$ 来提议新提示，基于当前提示 p^t 获得的“文本梯度” g^t 。这些“文本梯度”由一个元提示以及附加信息组成，不同的自动提示工程技术之间有所不同。这些信息包括不正确的示例 (?)，或不正确示例的自然语言 LLM 反馈 (?)，或两者的组合以及之前的提示 $p^{(t-1)}$ 及其评分 (?)。

$$p^{(t+1)} = \mathcal{M}_{proposal}(p^t, g^t). \quad (2)$$

然而，提示 $p^{(t)}$ 中的编辑可以发生在提示的任何地方，包括在每个时间步完全重写提示，导致向最佳提示的更新速度缓慢。此外，它不提供任何在典型生产提示工程中所需的控制，其中专业人士希望提示编辑发生在提示的特定范围内。因此，全局优化导致提示收敛缓慢，并且无法控制提示优化的方向。

最后，在所有时间步 $p^0 \cup p^1 \cup \dots \cup p^t$ 中，部分表现最好的候选提示会被保留，然后重复这一过程。

2.2 局部提示优化

‘文本梯度’ g^t 的基本功能是指示根据模型的性能 (?) 应该如何调整优化过程 (梯度值)。然而，它并没有具体说明优化应在何处进行，或者在深度学习中梯度下降应该在哪些参数上进行。我们结合了这种参数选择的直觉，通过局部提示优化来减少优化空间。

遵循链式思维逻辑的直觉，我们首先在提示建议步骤之前在元提示中添加指令，如图 1 所示，以识别导致预测错误的潜在标记。我们使用 `<edit>` 标记来突出显示需要编辑的标记，元指令如图 2 所示。其目的是识别提示中提案 LLM $\mathcal{M}_{proposal}$ 应该优化的标记。

一旦识别出提示编辑标记，我们就进行提示提案步骤。提供指令 ‘Reply with the new instruction without the `<edit>`, `</edit>` tags.’ ‘给 $\mathcal{M}_{proposal}$ ，以输出更新后的提示 $p^{(t+1)}$ 。表 1 展示了具有局部和全局优化的完整提示演变。

```
First, identify the scope of tokens within the
prompt where edits should take place.
Prompt edits include adding, deleting or
modifying tokens.
Mark the scope of the prompt that needs editing
by putting <edit>, </edit> tags.
You can have multiple <edit> tags and each <edit>
tag should not entail more than 5 words.
Do not cover the whole sentence with multiple
<edit> tags.
Reply with the prompt with <edit>, </edit> tags.
Do not include any other text.
```

Figure 2: 识别潜在优化标记的提示示例。

3 实验

本节的目标是强调在不同的自动提示工程方法中，局部优化相对于现有全局优化的有效性。

3.1 数据集

紧跟 (?)，我们在三个任务集上进行评估，这些任务在目标和领域上各不相同。我们使用了 (?) 提供的相同训练、验证、测试划分。

BIG-bench 硬 或 BBH (?) 是一组 23 个任务 (27 个子任务)，可以分为算法任务、自然语言理解任务、世界知识任务和多语言推理任务。

(2) 数学推理 包含两个数据集 MultiArith (?) 和 GSM8K (?)。这两个数据集都包含需要 2 至 8 步代数推理才能达到最终答案的小学数学问题。

(3) 生产提示 是一个内部分类提示，开发用于协调进一步 LLM 调用的正确工具。该提示将接收用户查询并识别查询的“意图”。然后它会输出一个带有适当参数的函数调用。它由领域专家开发，包含 8k 个标记。提示包括技能定义部分、具体的分类说明、安全说明等，因此是理想的评估候选。

3.2 提示优化方法

为了进行公平比较，我们选择了三种具有代表性的提示优化技术，并用我们在 Sec. 2 和 Fig. 1 中解释的局部优化步骤替换它们的全局优化步骤。(1) APE (?) 利用 LLM 来生成输入提示的变体，给出几个示例，然后选择表现最好的提示。APE 的改进版本称为迭代 APE，重复这一过程几次以获得更优的优化提示。我们在论文中使用迭代 APE 进行比较。(2) APO (?) 基于迭代 APE，并在提示优化过程中添加了不正确预测反馈。这个反馈通常称为“文本梯度”，用于在候选提示上进行正确方向的编辑。他们最近的草案中将 APO 命名为 ProTeGi。

Initial Prompt	Let's think step by step.
Global Optimization	
Optimum	Ensure all given initial values and specific contexts (e.g., rounding rules, phrase interpretation) are considered, and explain the arithmetic operations logically and clearly, step-by-step.
Local Optimization	
Identifying edit	Let's <edit> think </edit> <edit> step by step </edit>.
Optimum	Let's carefully read and clearly understand the problem. Next, let's think through each step and verify each calculation carefully.

Table 1: 通过将传统的全局优化方法与我们提出的局部优化进行比较，找到的 MultiArith 提示。

Method	LPO	GSM8k (↑)	MultiArith (↑)	# steps (↓)
APE	-	77.7	93.2	2.5
	✓	78.0	96.2	4
APO	-	77.7	96.0	4
	✓	79.7	97.5	2
PE2	-	78.7	97.0	2.5
	✓	80.6	97.5	2

Table 2: 在数学推理基准上的局部提示优化 (LPO) 结果。

(3) PE2 (?) 在“文本梯度”方面进行了更进一步的创新，通过添加旧提示及其反馈历史来丰富这些梯度，以指导编辑过程。他们也限制了提示中的编辑次数。

3.3 实现细节

在所有实验中，我们一致使用 gpt-3.5-turbo 作为任务求解模型，使用 gpt-4o 作为提示优化器。其余设计细节遵循 PE2 (?) 的设计。我们将搜索预算限制为 3 个优化步骤，使用 4 的束大小，并在每个步骤生成 4 个提示。此外，我们用标准提示“让我们一步一步思考” (??) 初始化 BBH 和数学推理数据集的提示。我们在全局和局部优化中为所有提示优化方法保持相同的超参数。

4 结果与分析

我们在 GSM8K 和 MultiArith 数据集上评估了 APE、APO 和 PE2 算法，分别在使用和不使用局部优化的情况下，如表 2 所示。我们观察到局部提示优化能够平均提高 1.5% 的数学推理任务的提示效果，同时减少所需的优化步骤。此外，我们展示了局部优化在 BIG-bench Hard 基准 (27 个子任务) 上的广泛适用性。在图 3b 中，我们展示了局部优化支持在各种任务上使用各种自动提示技术。我们在各种方法上平均超过传统的全局优化方法 2.3%。我们假设，

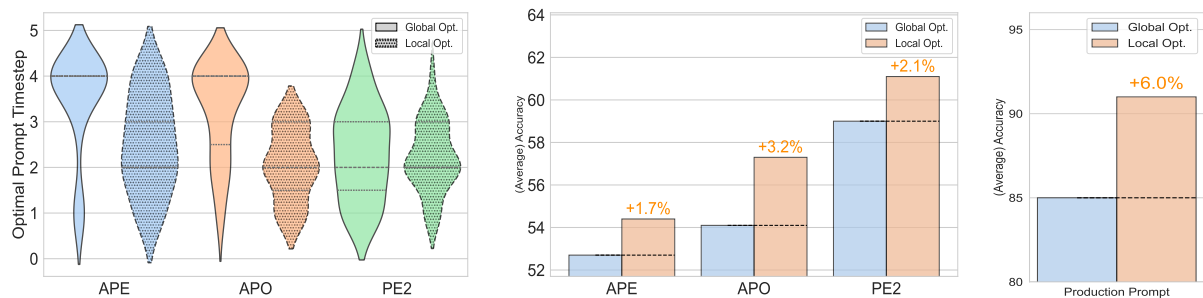
由于局部优化减少了建议的 LLM $\mathcal{M}_{proposal}$ 的优化代币，并在优化步骤中引入了链式思维方法， $\mathcal{M}_{proposal}$ 能够更有效地解决任务并提供更好的提示输出。

我们评估了在 BIG-bench Hard 基准的 27 个子任务中生成最佳提示的时间步数。迭代次数保持为 3，当初始化提示被发现是表现最佳的提示时，我们分配一个时间步数为 4。图 3a 描绘了最佳提示时间步的提琴曲线。值得注意的是，我们观察到大多数任务比全局优化方法更早收敛，节省了大量的 LLM 计算和时间。全局优化通常导致从头重写 LLM 的完整提示，使任务更具挑战性和复杂性。另一方面，我们认为，通过局部优化减少优化空间，使得梯度更新保持对准于最小值。

也许，LPO 最大的好处是可以控制编辑的范围。在领域专家编写的生产提示中，提示具有特定的部分，其中定义了不同的工具，并紧跟着是关于各个工具及其使用的指示。使用 LPO，我们可以指定需要更新哪个工具的指令，而不会影响其他工具。此外，它确保了由于优化过程不会导致提示在其他类别中的性能回退。在我们的评估中，正如图 3c 所示，我们在生产提示上获得了显著的 6% 提升。

在这项工作中，我们识别出现有自动提示工程技术的优化步骤中的缺口。传统上，提示会在每个步骤中整体变异。然而，这种全局优化增加了任务复杂性，因为优化器（大语言模型，LLM）必须在更多的参数（标记）上工作以寻找最佳更新。此外，许多生产提示仅需要优化提示的某个部分，而不是从头重写整个提示。作为解决方案，我们引入了局部提示优化 (LPO)，在其中我们识别出需要优化的标记，并引导优化器仅关注这些标记。我们观察到，在数学推理和 BIG-bench Hard 基准上性能有一致的提升。值得注意的是，我们观察到局部优化比传统方法显著更快地找到最佳提示。此外，LPO 可以很好地整合到较长的提示中，这在实际环境中更为普遍，进一步展示了我们方法的广泛性。展望未来，我们对从局部优化角度构建的提示优化技术感到乐观，它们可以从性能和效率的提高中受益。

我们相信我们的研究存在三个局限性，我们认为这些局限性可以在未来的工作中克服。(1) 多语言性：我们在这项工作中主要关注英语，从提示到数据集再到 LLMs。然而，我们相信论文中介绍的思想可以扩展到其他语言，并恳请学术界在我们的工作基础上进行拓展。(2) 局部优化有时会导致提示过拟合，开发分数接近 99%。我们认为更好的搜索策略可以解决这个问题，并希望能看到未来的研究对此进行解决。(3) 闭源模型：我们使用 GPT-4o 作为优



(a) 在 BBH 基准的 27 个子任务中，最佳提示时间步长。局部优化达到更快的收敛。

(b) 在 BBH 上的平均准确性。局部优化始终在各种方法中优于全局优化。

(c) 采用局部优化后的生产提示性能

Figure 3: 对 BBH 和 Production Prompt 的实验，展示了 LPO 在性能和效率上的优势。

化器来为大数据集建立基准。这给该工作的可重复性带来了挑战。然而，我们相信在专有模型上展示局部优化能力是对学术界和工业界的一个好信号，促使他们在提示工程方法中纳入这些想法。