TRPrompt: 从文本奖励中引导查询感知的提示优化

Andreea Nica
EPFL
andreea.nica@epfl.ch

Ivan Zakazov*
EPFL
ivan.zakazov@epfl.ch

Nicolas Mario Baldwin*
EPFL
nicolas.baldwin@epfl.ch

Saibo Geng
EFPL
saibo.geng@epfl.ch

Robert West
EPFL, Microsoft Research
robert.west@epfl.ch

Abstract

提示优化在不需要更新目标模型参数的情况下，提高了大语言模型（LLM）的推理能力。遵循基于启发式的“逐步思考”方法，该领域主要朝着两个方向发展：一类方法使用文本反馈从通用 LLM 中无需训练地引出改进的提示，而另一类研究则依赖于数值奖励来训练一个特殊的提示模型，以提供针对目标模型的最佳提示。在本文中，我们介绍了文本奖励提示框架（TRPrompt），通过将文本反馈直接整合到提示模型的训练中，统一了这些方法。我们的框架不需要先前的数据集收集，并通过对生成提示的反馈进行逐步改进。当结合 LLM 内化什么是“好”提示的能力时，文本奖励提供的高分辨率信号使我们能够训练一个提示模型，为 GSMHard 和 MATH 等具有挑战性的数学数据集的问题提供最先进的特定查询提示。

1 介绍

最近的最先进的大型语言模型（LLMs）的成功，不仅应归因于参数的大规模扩展，还应归因于将这些模型与人类反馈对齐的努力 [13] 。从人类反馈中进行强化学习（RLHF）被证明是一种极其有效的方法，可以将人类知识注入到 LLMs 中，使它们能够在各种基准测试中实现人类水平的表现 [12] 。RLHF 采用一种奖励模型，该模型生成用于指导语言模型优化的数值奖励。数值奖励充当人类偏好的代理，并在训练中起着至关重要的作用。然而，根据下游任务的不同，数值奖励可能会稀疏、信息不足且难以定义。而文本反馈则利用语言的丰富性，可以提供更具信息性和细致入微的学习信号。在某些任务中，数值奖励很难建模，比如依赖查询的提示优化问题，其目标是生成适应每个输入查询的提示，以帮助指导语言模型生成高质量的输出。

尽管大型语言模型拥有显著的能力，但在数学和逻辑推理任务方面似乎面临挑战 [8] 。提示工程（将输入查询与指令连接起来）作为一种轻量级的解决方案，出现以增强大型语言模型的推理能力 [17, 6] ，因为这无需调节参数。研究努力探索了寻找最优提示的不同方法。一个方向集中在将大型语言模型用作黑箱优化器，通过文本反馈迭代更新最优提示 [22, 11, 14, 19] 。这种无训练方法完全依赖于与现成的大型语言模型的交互。相比之下，其他方法则探讨通过强化学习训练一个策略模型，以便生成最优提示 [16, 3, 10, 7] 。尽管大多数这些方法专注于寻找能够在整个数据集内实现最佳平均表现的任务级最优提示 [3, 22] ，但查询级提示优化技术 [7, 10, 16] 可以产生更好的结果，因为提示是专门调整以适应特定查询。考虑到查询依赖问题的二元性（提示指导大型语言模型得出正确/错误答案），为某一具体查询设计数值奖励以成功指导提示空间探索是极具挑战性的，因为在稀疏信号中编码细微偏好的难度很大。图 1 展示了一个激励示例。

*IZ and NMB co-supervised the project and contributed equally.

Under review at NeurIPS 2025.

文本奖励是以自然语言表达的对提示的评估，作为其在实现特定目标方面有效性的一种定性衡量。一些成功应用文本奖励的方法专注于通过将合成评论与输入串联来提高奖励模型的质量 [18] 。这些文本评论提高了奖励模型的性能和数据效率。然而，涉及文本奖励用于提示优化问题的研究仍然极其稀缺。

解决方案：TRPrompt。为了解决现有文献中的上述限制，在本文中，我们提出了 TRPrompt，一种文本奖励提示优化方法，研究在提示模型训练中使用文本奖励而非数值奖励的潜在好处。我们相信，与数值相比，文本奖励的丰富性能够涵盖有关给定提示性能的更多信息。文本奖励应为生成最佳提示所用的提示模型的微调提供更强的信号和更稳健的特征。

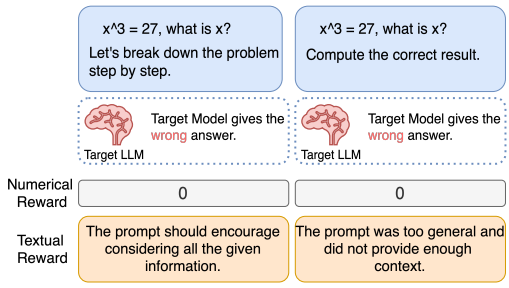

Figure 1: 激励示例：文本奖励比数值奖励更具表现力。当同时使用两个提示时，目标模型会得到错误答案。虽然数值奖励相同（0），但文本奖励清楚地显示每个提示未能有效指导目标语言模型正确回答。

我们开发了一个概念验证框架，该框架微调一个相对较小的模型（Llama-3-8B-Instruct [1] ）以生成查询级最优提示。由于此领域的工作较少，我们开发了整个在训练期间应用文本奖励的的方法，以用于提示优化问题，并对查询相关的目标进行调整，其中数值奖励更难以建模。我们通过监督微调（SFT）训练提示模型，使用一个合成生成的包含提示和其对应文本奖励的数据库。这个管道是全自动化的，文本奖励由另一个充当黑箱奖励模型的语言模型生成。在我们的实验中，我们使用同一系列的模型（Llama-3-8B-Instruct）来生成文本奖励，促进自我改进的过程。因此，我们建立了一个迭代的过程，其中微调的提示模型高效生成改进的、更好的提示，然后用这些提示进一步微调提示模型，形成一个循环反馈回路。这样，在每次迭代中，提示模型通过文本奖励获得信号，了解被视为最优提示的实际表现，从而促进改进。

Table 1: 按奖励类型和训练设置分类的提示优化方法概述。TRPrompt 是唯一将文本奖励与可训练方法相结合的方法。

	训练	Train-free
文本奖励	TRPrompt (ours)	Textgrad [19], APO[14]
Numerical Rewards	Prompt-OIRL [16], QPO[7]	APE[22]

我们的框架可以应用于广泛的任务，以评估大语言模型（LLM）的数学推理能力。TRPrompt 可以在任何现成的大语言模型上初始化，并且不需要先前的数据集收集，这使得它可以立即适应任何数据集。我们在多个数据集上进行了实验，并分析了我们的方法在为期望的目标生成最佳提示方面的有效性。我们的贡献可以总结如下：

- 我们提出文本奖励的概念，首次将其作为训练期间直接使用的监督信号——超越了之前不需要训练的方法，例如 TextGrad [19] 。
- 我们提出并开发了一种新颖的方法 TRPrompt，该方法能够利用文本奖励的优势，实现依赖查询的提示优化目标，以提高提示性能。表 1 将之前的方法与我们的方法进行比较，展示了 TRPrompt 如何首次将文本奖励直接嵌入到训练循环中。
- 我们在各种数据集上验证了我们方法的有效性，并提供了对其性能的全面分析。
- 我们在更具挑战性的数据集 GSMHard 和 MATH 上达到了在提示优化方面的最新性能，展示了我们方法的有效性。

2 查询依赖提示问题

查询依赖型提示问题的核心是引导目标模型正确解决给定的问题。在这项工作中，我们特别关注数学问题。为此，我们在问题前加上一个查询依赖型的提示——这是专门为给定问题定制的指令，以改善模型的推理能力。这个提示由一个单独的提示模型生成，该模型经过训练以产生有效的查询依赖型指令。提示模型根据文本奖励条件来生成提示——这些奖励

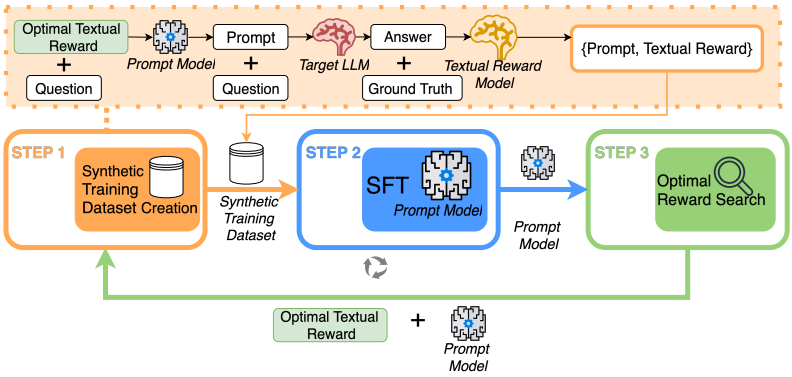

Figure 2: 我们基于查询的 prompt 优化迭代流程概述，包含三个步骤：（1）合成训练数据集的创建，（2）prompt 模型的微调，（3）最优奖励搜索。

是指示期望提示质量的自然语言反馈。为了生成可靠地引导目标模型得到正确答案的提示，我们让提示模型受最佳文本奖励的约束，即给定问题的提示所能获得的最佳反馈。这个过程在图 2 的步骤 1 中进行了说明。一个具体的例子在图 3 中展示。因此，训练提示模型涉及两个相互关联的目标：（1）微调提示模型和（2）识别最佳文本奖励。这个过程构成了我们所称的查询依赖型提示问题，现在我们将其形式化。

问题和答案。我们考虑解决数学问题的任务 $q \in \mathcal{Q} = \mathcal{V}^{\infty}$ ，其中每个问题使用词汇 $\mathcal{V}$ 以自然语言表达。每个问题 q 预计会有一个数学上正确的答案 $y^* \in \mathcal{Y}_{gt} \subset \mathcal{Y}$ ，这将被视为标准答案。

提示。通过使用适当的提示，目标大型语言模型在回答问题时的性能可以得到增强。提示 $p \in \mathcal{P} =\in \mathcal{V}^{\infty}$ 是词汇 $\mathcal{V}$ 的组合，是一种自然语言中的指令，用于解释如何解决任务。在这项工作中，提示附加在问题的末尾。查询相关提示是专门生成的提示，用于指导目标模型针对具体问题得到正确结果。

目标 LLM。数学问题使用目标 LLM $M_{\text{target}}: \mathcal{Q} \to \mathcal{Y}$ 来获得答案。生成的答案可以数学地描述为 $\hat{y}_i = M_{\text{target}}(q_i, p_i)$ ，其中 q_i 是第 i 个问题，p_i 是对应于问题 q_i 的查询相关提示。

这些生成的质量可以通过以下指标 $r(y_i^*, \hat{y}_i) = \mathbb{1}\{\hat{y}_i = y_i^*\}$ 进行评估，该指标检查生成的答案与真实值的相等性。

文本奖励。为了评估与查询相关的提示的质量，我们将使用文本奖励 $t \in \mathcal{T} = \mathcal{V}^{\infty}$ 。文本奖励代表以自然语言形式表达的反馈，该反馈将综合查询相关提示在引导目标大型语言模型走向正确答案时的表现。

最优文本奖励。我们将 $t^* \in \mathcal{T}$ 定义为最优文本奖励，对应于查询相关提示可以获得的最佳可实现反馈，符合目标范围。

文本奖励模型。一个文本奖励模型 R_{textual} ，其中 $R_{\text{textual}}: P \times \mathcal{Q} \times \mathcal{Y}_{\text{gen}} \times \mathcal{Y}_{\text{gt}} \to \mathcal{T}$ 是一个语言模型，它能够基于目标 LLM $\hat{y}_i \in \mathcal{Y}_{\text{gen}} \subset \mathcal{Y}$ 生成的答案和问题的真实答案 $y_i^* \in \mathcal{Y}_{\text{gt}} \subset \mathcal{Y}$ ，为查询相关的提示 $p_i \in \mathcal{P}$ 生成文本奖励 $t \in \mathcal{T}$ 。我们有 $t = R_{\text{textual}}(q_i, p_i, \hat{y}_i, y_i^*)$ 。

提示模型。提示模型 $\Pi_{\text{query}}: \mathcal{Q} \times \mathcal{T} \to P$ 是一个用于生成查询相关提示的语言模型。对于某个查询 $q_i \in \mathcal{Q}$ 和文本反馈 $t \in \mathcal{T}$ ，我们有 $\Pi_{query}(q_i, t) = p_i \in \mathcal{P}$ 。提示模型的作用是生成查询相关的提示或指令，以便提高目标模型的性能，这种改进方式由反馈 t 针对特定问题 q_i 指定。

双重目标。为了最大化目标 LLM 在推理任务中的性能，查询相关提示问题可以被表述为一个双重目标优化问题。

1. 提示模型目标。第一个目标是优化提示模型 Π_{query} ，以生成与给定问题的最佳文本奖励 t^* 相对应的查询相关提

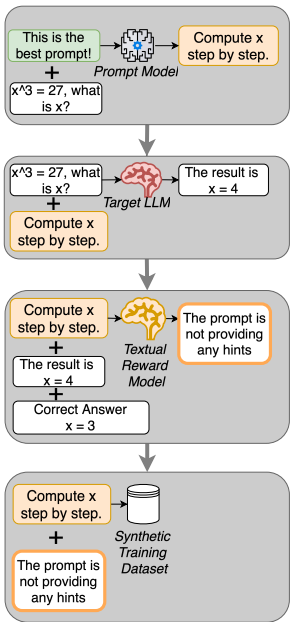

Figure 3: 步骤 1: 提示生成和文本奖励计算

示。这些提示应该引导目标 LLM 生成正确答案。目标是通过确保提示满足最高质量的反馈信号来最大化目标模型的性能。

$$\Pi^*_{\text{query}} = \arg\max_{\Pi_{\text{query}}} r(y_i^*,\ M_{\text{target}}(q_i, \Pi_{\text{query}}(q_i, t^*)))_{i\in[N], N=|\mathcal{Q}|} \tag{1}$$

2. 最佳文本奖励目标。第二个目标侧重于识别给定提示模型 Π_{query} 的最佳文本奖励 t^* 。该奖励代表最具信息量和最有效的自然语言反馈，当用于条件提示模型时，会生成查询相关的提示，从而最大化目标模型的准确性。

$$t^* = \arg\max_{t\in\mathcal{T}} r(y_i^*, M_{\text{target}}(q_i, \Pi_{\text{query}}(q_i, t)))_{i\in[N], N=|\mathcal{Q}|} \tag{2}$$

3 用文本奖励进行提示优化

现有依赖于查询的方法完全依靠数值奖励来指导训练 [16, 7] 。相反，我们认为文本奖励提供了更丰富和更具表现力的监督信号，能够更有效地引导学习过程。我们引入了 TRPrompt，这是一个利用文本奖励作为主要训练信号的提示优化框架。我们的迭代、多循环设计实现了持续改进，文本反馈在整个学习过程中促进了对提示模型的有效知识传递。

图 2 显示了我们针对查询依赖的优化问题的迭代方法。该流程包含三个重要步骤：步骤 1。合成训练数据集创建、步骤 2。 提示模型微调以及 步骤 3。 最优奖励更新。

步骤 1. 查询相关的提示生成 & 文本奖励计算。合成训练数据集包含查询相关的提示及其文本奖励。为了生成特定问题的提示，将最佳反馈信号 $t^* \in \mathcal{T}$ 与问题 $q \in \mathcal{Q}$ 一起提供给提示模型 Π_{query} ：

$$p_i = \Pi_{\text{query}}(q_i, t^*) \in \mathcal{P}, \quad \forall q_i \in \mathcal{Q}$$

查询特定生成的提示然后与问题连接，以从目标 LLM 生成答案。

$$y_i = M_{\text{target}}(q_i,\ p_i), \quad \forall q_i \in \mathcal{Q},\ p_i = \Pi_{\text{query}}(q_i, t^*)$$

分析生成的答案和真实值，文本奖励模型 R_{textual}: $P \times \mathcal{Q} \times \mathcal{Y}_{\text{gen}} \times \mathcal{Y}_{\text{gt}} \to \mathcal{T}$ 对特定任务的查询相关提示的性能产生详细的文本反馈。对应于具体任务提示的文本奖励被收集用于微调 $\mathcal{D}_{\text{train}} = \{(p_i, q_i, t_i)\}$ 。STEP 1 的整个流程的具体示例可以在图 3 中看到。

步骤 2. 提示模型微调。对于特定查询任务，我们基于文本奖励 $t \in \mathcal{T}$ 和问题 $q \in \mathcal{Q}$ 对提示模型 Π_{query} 在提示 $p \in \mathcal{P}$ 上进行微调。具体来说，我们最大化对数似然：

$$\mathcal{L}_{SFT} = -\mathbb{E}_{(p_i,q_i,t_i)\sim\mathcal{D}_{\text{train}}} \log \mathbb{P}(p_i|q_i, t_i) \tag{3}$$

步骤 3. 最优文本奖励更新。由于最优奖励是训练过程中尚未见过的静态值，因此找到能够利用整个微调过程中积累的所有知识的最佳最优奖励是一个独立的优化问题。具体来说，我们希望找到与方程 2 的目标一致的文本奖励。为了寻找最佳的最优文本奖励，我们采用了一种无训练的优化策略，Textgrad。[19] 。使用 Textgrad，我们可以探索奖励空间。这个步骤在每轮微调后完成，以获得每个模型对应的最优奖励。在微调提示模型 Π_{query} 结束时，我们得到了：

$$t^* = Textgrad(\Pi_{\text{query}})$$

关于查询相关目标的迭代过程，即在 K 次迭代中重复步骤 1、2 和 3，详见算法 1 。

4 相关工作

提示优化主要遵循两条路径：基于训练的方法通过数值奖励 [16, 7] ，以及无需训练的方法利用自然语言在不更新权重的情况下优化提示输出 [19, 14] 。TRPrompt 通过直接从文本奖励中实现基于梯度的训练，桥接了这两种模式。

提示优化。离散提示有一个重要特征：可解释性。关于零样本提示的研究表明，LLMs 仅通过使用提示 “让我们逐步思考” (CoT) [6] ，就能成为出色的零样本推理器。虽然大多数提示工程研究的努力集中在寻找任务级别（对整个数据集有效的提示）的最佳提示，但这种方法

Algorithm 1 查询依赖的提示优化

Require: Question dataset $\mathcal{Q}$, number of iterations K , a language model as textual reward model R_{textual} , a Target LLM M_{target} , SFT($\cdot$) denotes the supervised fine-tuning process, Textgrad($\cdot$) denotes applying Textgrad for optimal reward search, an initial optimal textual reward t_0^* , the number of questions to analyze the textual reward N , a Prompt Mode Π_{query} that generates query-dependent prompts.

for $k = 1$ to K **do**
  $\mathcal{D}_{\text{train}} \leftarrow \emptyset$
  for q_i in $\mathcal{Q}$ **do**
    Prompt Generation: $p_i \leftarrow \Pi_{\text{query}}^{k-1}(q_i, t_{k-1}^*)$
    Question Answering: $y_i = M_{\text{target}}(q_i, p_i)$
    Textual Reward $t_i \leftarrow R_{\text{textual}}(p_i, (q_i, y_i, y_i^*)$
    $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \{(p_i, q_i, t_i)\}$
  end for
  Fine-tuning: $\Pi_{\text{query}}^k \leftarrow SFT(\Pi_{\text{query}}^{k-1}, \mathcal{D}_{\text{train}})$
  Optimal Reward Update : $t_k^* = Textgrad(\Pi_{\text{query}}^k, t_{k-1}^*)$
end for
return Prompt Model Π_{query}^K and optimal reward t_K^* which can generate optimal prompts for a specific task.

可能在每个实例级别导致次优结果。一种通常有效的提示，虽然平均表现良好，可能会引导 LLM 得到错误的结果 [16] 。这就是为什么现在越来越关注查询依赖的方法 [16, 21, 10, 7] 。Prompt-OIRL [16] 使用逆向强化学习训练奖励模型，以在 n 个候选提示中为查询选择最佳一个。QPO [7] 使用以自定义数值奖励为指导的强化学习，对一个小型预训练语言模型进行微调，生成与输入查询定制的查询依赖最佳提示。上述所有方法都独特地依赖数值奖励来指导训练过程。

基于奖励的微调。在监督学习过程中，生成时根据奖励信号进行条件化的研究已在之前的工作中得到探索 [20, 15] ，并在代码优化等领域取得了成功 [15] 。然而，这些方法仅依赖于数值奖励标签，并未利用文本奖励所提供的更丰富的监督信息。

使用自然语言反馈进行优化。最近的研究方向调查了使用自然语言反馈来提高 LLM 整体性能的方法。一种研究方向通过润色模型输出，使其变得更好，利用文本反馈进行自我改进。这些应用将 LLM 作为一个黑箱优化器，以无训练方法部署，其中文本反馈用于获得模型输出中的迭代改进 [14, 19] 。Textgrad [19] 以梯度下降为灵感，提供了一种可定制框架，可以应用“文本梯度”（文本反馈）来改进 LLM 在各种任务上的生成，包括提示优化。TPO [9] 将奖励模型生成的数值奖励信号转化为文本批评，利用这些文字批评作为文本奖励来迭代地改进模型的最终输出。文本奖励用于帮助策略模型提出建议，指导下一次迭代的生成。另一种研究方向探讨使用批评来训练奖励模型 [18] ，从而将合成批评的丰富信号转换为数值表示。然而，据我们所知，目前没有方法在训练期间直接使用文本奖励作为信号。

5 实验

在本节中，我们提供实证证据，展示了我们的方法在使用文本奖励引导最佳查询相关提示模型训练中的效果。我们首先概述一般的实验设置，然后对结果进行深入分析。

任务。我们在三个具有不同复杂度的数学推理数据集上进行实验：GSM8K [2] ，GSMHard [4]（这是 GSM8K 的一个更难版本）和 MATH [5] 。这些算术推理任务在零样本提示域中被广泛研究。

模型。我们的框架由三种不同的模型组成：1. 目标 LLM，用于搜索最佳提示。2. 用于生成文本奖励的文本奖励模型。3. 会生成目标 LLM 的最佳提示的提示模型，该模型正在进行微调。我们将所有这些模型设定为 Meta-Llama-3-8B-Instruct [1] 。我们的方法是与模型无关的，因此可以为不同的组件更换不同的模型实例。然而，我们选择探索相同模型的方法，因为我们希望研究该模型的自我改进能力。对于 Textgrad [19] ，我们使用 GPT-4o-mini 进行最佳文本奖励更新。关于实验的技术细节可以在技术附录中找到。

表格 3 对 TRPrompt 生成的查询依赖优化提示与几种 SOTA 方法进行了全面比较：包括 CoT “让我们一步一步思考”提示，QPO 在两种不同初始化配置下生成的查询依赖提示（初始数

Table 2: TRPrompt 与现有研究的不同之处在于：(1) 不需要一组人工定义的提示，(2) 完全独立于起始提示，因为我们的方法通过合成方式构建整个训练数据集，避免了初始提示的需求，(3) 提供一种与查询相关的方法来解决提示优化问题，(4) 依靠文本奖励作为训练过程中的主要信号 (5)。

Method	(1) Avoids Static/Manual Prompts	(2) Independent of Initial Prompts	(3) Query Dependent Prompt	(4) Textual Rewards	(5) Textual Rewards For Training
CoT[6]	✗	✗	✗	✗	✗
Prompt-OIRL [16]	✗	✗	✓	✗	✗
QPO [7]	✓	✗	✓	✗	✗
Textgrad [19]	✓	✗	✗	✓	✗
TPO [9]	✗	✗	✗	✓	✗
TRPrompt (ours)	✓	✓	✓	✓	✓

Table 3: 对比当前最先进的方法和 TRPrompt 在 GSM8K、GSMHard 和 MATH 数据集上的表现。

Name	GSM8K (Acc.)	GSMHard (Acc.)	MATH (Acc.)
CoT [6]	85.59 %	27.98 %	39.35 %
Prompt-OIRL (6 prompts) [16]	84.53 %	28.61 %	21.31 %
QPO (1 starting prompt) [7]	84.76 %	27.98 %	28.37 %
QPO (500 starting prompts) [7]	86.05 %	30.80 %	37.31 %
TRPrompt Query-Dependent (Ours)	84.53 %	31.76 %	41.37 %

据集包含 1 个专家提示与 500 个专家提示），基于 6 个专家提示学习奖励的 Prompt-OIRL，以及通过应用 Textgrad 获得的任务级提示。

从结果中，我们可以得出结论：（1）在具有挑战性的数学基准测试中，例如 GSMHard 和 MATH，目标模型在生成正确答案方面表现出困难，TRPrompt 在 GSMHard 上比所有其他最先进的方法高出 1 %，在 MATH 上高出 2 %。在 GSM8K 的情况下，模型默认情况下回答大多数问题正确，相比之下，文本奖励在更具挑战性的数据集如 GSMHard 和 MATH 上的效果更强。(2) 尽管 QPO 和 Prompt-OIRL 表现出对初始专家提供提示的强烈依赖，TRPrompt 完全避免了这种限制，因为它不需要手工起始提示，从而避免了初始化偏差。Prompt-OIRL 选择预定义提示之一，而 QPO 生成的新提示则受到初始提示集的结构和内容的严重影响。准确性随着初始提示数量的增加而提高。TRPrompt 更加自由地探索提示空间，从而产生更高质量的查询相关提示。

5.1 分析

迭代优化的提示。在我们的流程中，迭代的特性对于确保提示模型从自身的局限中学习至关重要。对于每次迭代，都会使用最新版本的提示模型生成的提示来构建一个新的合成训练数据集，这些提示受当前最佳文本奖励的影响。这个设置保证了文本奖励针对模型的最新表现，使其能够接收到精准的批评。结果是，提示模型通过学习过去的错误逐步提高其生成能力，从而实现有针对性的改进。图 4 展示了随着迭代次数的增加，提示质量的不断提升，经过微调的提示模型在训练过程中表现得更好（+7.5 %）比基础模型更优，尤其是在较难的数据集 GSMHard 和 MATH 上。每次迭代通过文本奖励利用新学习的模式，逐步提高生成提示的质量。图 5 展示了对于 GSMHard 问题，提示在每次迭代中的进展。提示模型从包含显式计算开始，逐渐发展到生成通用且偶尔不相关的建议，最终收敛到有效的、与任务对齐的指令上。

对于难度较大的数据集（如 MATH 和 GSMHard），目标模型难以生成正确答案，这使得文本奖励能够提供高度针对性和建设性的反馈。因此，提示模型在每次迭代中都有显著的改进，能有效地从错误中学习。相比之下，对于像 GSM8K 这样较简单的数据集——在没有任

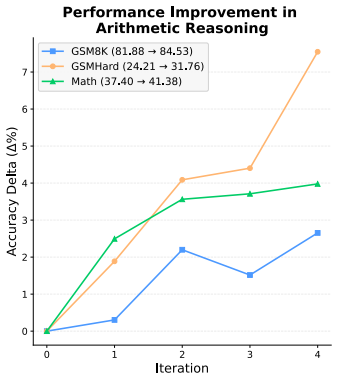

Figure 4: 与基准模型相比，在各个迭代中的准确性提升（以百分点差异表示）。每次迭代都提高了提示模型生成提示的能力，从而引导目标模型得到正确答案。

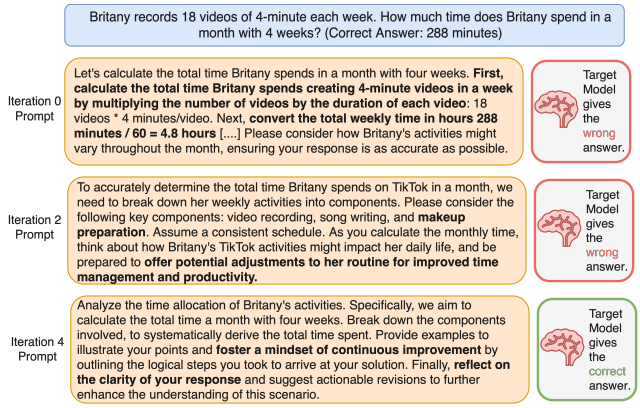

Figure 5: 针对 GSMHard 问题中查询相关提示的迭代改进。随着迭代次数的增加，提示模型生成的提示变得越来越结构化和通用，从而减少对部分解和不相关细节的依赖。在最后的迭代中，改进后的提示引导目标模型得出正确答案。

何提示的情况下目标模型已经表现良好——文本奖励的纠正能力减弱，导致获得的提升更为有限。我们假设在 GSM8K 上表现提升有限的部分原因是由于训练集不平衡，主要由正反馈主导。当大多数提示已经得出正确答案时，文本奖励会极大地增强成功，提供的对比有效和无效提示的内容很少。缺乏负面监督削弱了训练质量，因为提示模型得不到足够的信号来优化其行为。

消融见解：为何 SFT + 最优奖励搜索至关重要。我们研究了每个步骤对我们方法效率的影响。我们在流程中的每个中间步骤后计算准确率，并报告每个数据集的结果。图 6 展示了随着使用监督微调（SFT）对提示模型进行微调，并随后应用最优奖励搜索后，每个步骤对迭代中准确率持续增长的影响。性能的提升是这两个步骤结合效果的结果，共同使得对提示空间的有效探索成为可能。相比之下，图 7 展示了在我们框架中移除最优奖励搜索步骤并仅应用 SFT 的消融实验。在这种情况下，性能在所有数据集的每次迭代中逐渐下降，这表明如果不更新最优文本奖励，提示模型开始对次优反馈过拟合。这些结果突出了最优奖励搜索在引导提示模型对提示空间进行有意义的探索并在时间上保持性能改进方面的关键作用。

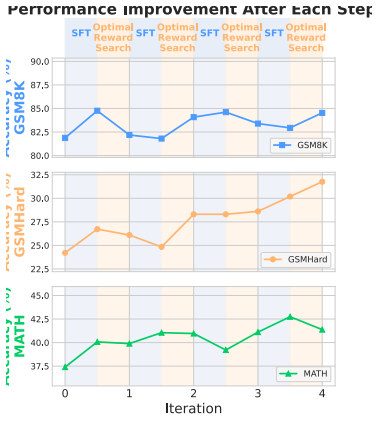

Figure 6: 在我们的流程中，每一步之后由提示模型获得的中间性能。SFT 和最优奖励搜索步骤都对增强提示模型生成高效提示的能力有贡献。

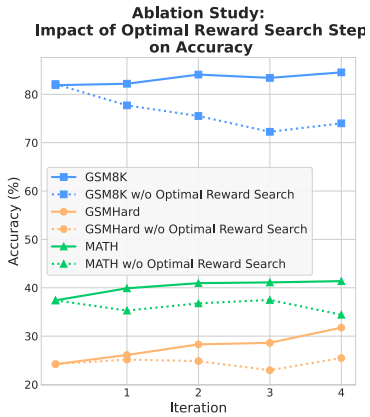

Figure 7: 移除我们流程中的最佳奖励搜索步骤的影响：准确率在每次迭代中持续下降。

评估跨数据集迁移：提示模型在未见任务上的准确性。我们通过测试一个在一个基准上训练的提示模型是否能为未见数据集生成有效提示来评估 TRPrompt 和查询相关的基线的泛化

能力。在此实验中，提示模型在单个数据集上训练，并在其余数据集上进行评估。如表 4 所示，所有方法在应用于分布外数据时性能都有所下降，这证实了提示优化对训练分布的敏感性。然而，当提示模型在 GSM8K 或 GSMHard 上训练时，TRPrompt 在 MATH 数据集上达到最高准确性，并且一致地优于 Prompt-OIRL 和 QPO。虽然 TRPrompt 在 GSM8K 上训练时没有超过在 GSMHard 上训练时的基线性能，但这一观察结果与之前的研究一致，表明在较简单的数据集上性能提升通常不太明显。这些结果总体上突显了 TRPrompt 的优越泛化能力，特别是在更具挑战的领域中。

Table 4: 跨数据集泛化：在一个数据集上训练的模型在另一个数据集上评估时的准确性。

Trained on	Method	Tested on	GSM8K Accuracy	GSMHard Accuracy	MATH Accuracy
GSM8K	Prompt-OIRL		-	29.55 %	22.14 %
	QPO		-	28.61 %	36.36 %
	TRPrompt (Ours)		-	27.67 %	37.20 %
GSMHard	Prompt-OIRL		84.15 %	-	26.53 %
	QPO		82.94 %	-	26.35 %
	TRPrompt (Ours)		83.09 %	-	39.41 %

6 局限性 & 未来工作

在较简单的数据集上的增益减少。TRPrompt 在 GSM8K 上显示出相对较低的性能增益，无论是在分布内还是分布外设置。正确答案和积极文本奖励的主导地位削弱了学习信号。解决此问题可能需要重新平衡训练数据或引入更加细致的文本反馈，以更好地反映提示质量的微妙变化。

计算成本高。涉及 Textgrad 的最优奖励搜索步骤计算代价昂贵且难以并行化，是流水线中的主要瓶颈。这限制了可扩展性并减缓训练速度，特别是在较大的数据集上。未来的工作可以集中于更高效的最优奖励搜索方法。

进一步利用文本奖励。我们的框架可以扩展到一些无法或不自然定义数值奖励的任务，例如创意写作或诗歌。在这些领域中，定量指标往往无法捕捉质量、连贯性或风格，而文本反馈则可以提供更丰富、更易于解释的信号。TRPrompt 为该领域的未来研究提供了一个有前景的基础。

我们引入了 TRPrompt，这是一种新颖的与查询相关的提示优化框架，利用文本上的奖励来直接引导提示模型的微调。与之前的工作不同，TRPrompt 消除了对初始专家提供的提示的强依赖，从而可以通过在文本反馈指导下的迭代过程从零开始学习提示。在算术推理任务的实验中，TRPrompt 在像 MATH 和 GSMHard 这样的具有挑战性的数据集上表现出色，其中丰富的文本反馈提供了有意义的指导。这些结果突显了 TRPrompt 在数值奖励有限或定义不明确的环境中对齐大型语言模型的潜力，并将文本监督定位为提示优化的强大工具。

Acknowledgments and Disclosure of Funding

References

[1] AI@Meta. Llama 3 model card. 2024.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

[4] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

[5] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[7] Yilun Kong, Hangyu Mao, Qi Zhao, Bin Zhang, Jingqing Ruan, Li Shen, Yongzhe Chang, Xueqian Wang, Rui Zhao, and Dacheng Tao. Qpo: Query-dependent prompt optimization via multi-loop offline reinforcement learning. *arXiv preprint arXiv:2408.10504*, 2024.

[8] Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*, 2024.

[9] Yafu Li, Xuyang Hu, Xiaoye Qu, Linjie Li, and Yu Cheng. Test-time preference optimization: On-the-fly alignment via iterative textual feedback. *arXiv preprint arXiv:2501.12895*, 2025.

[10] Zekun Li, Baolin Peng, Pengcheng He, Michel Galley, Jianfeng Gao, and Xifeng Yan. Guiding large language models via directional stimulus prompting. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2024.

[12] Josh OpenAI, Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[14] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

[15] Alexander Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob Gardner, Milad Hashemi, Graham Neubig, Parthasarathy Ranganathan, Osbert Bastani, and Amir Yazdanbakhsh. Learning performance-improving code edits. *arXiv preprint arXiv:2302.07867*, 2023.

[16] Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. In *The Twelfth International Conference on Learning Representations*, 2023.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[18] Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gallé. Improving reward models with synthetic critiques. *arXiv preprint arXiv:2405.20850*, 2024.

[19] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[20] Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. The wisdom of hindsight makes language models better instruction followers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[21] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*, 2022.

[22] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

A 技术附录和补充材料

我们流程的核心组件是构建合成训练数据集。在每次迭代中，我们使用当前版本的提示模型和奖励模型生成一个新的数据集，然后用其来微调下一个版本的提示模型。这个过程使模型能够通过学习自己的输出和相应的反馈逐步改进。

合成数据集创建包括两个关键步骤：（1）提示生成，其中提示模型为给定问题生成特定查询的提示；（2）文本奖励生成，其中奖励模型为生成的提示产生文本奖励。

鉴于两个模型都依赖于 LLMs，明确和精确的指令对于确保每个模型在此过程中表现正确至关重要。为此，精心设计了元指令以使每个模型与其各自目标保持一致。

为了生成基于最佳文本奖励的提示，我们使用结构化输入格式来指导提示模型。元指令同时指定任务和相应的最佳奖励，作为指导反馈。如图 8 所示，提示模型被要求明确其输入和输出期望，并被明确指示生成与提供的文本奖励一致的查询相关提示。

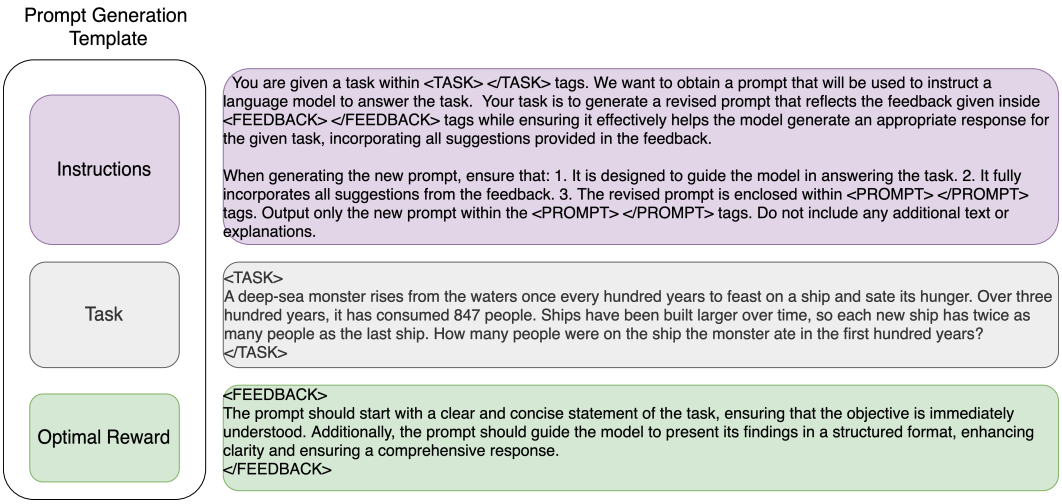

Figure 8: 元指令模板和示例用于生成基于最佳奖励的查询相关提示。

为了对特定的提示生成文本奖励，我们为奖励模型提供生成的提示、目标模型对相关问题的输出以及相应的真实答案。奖励模型由一个元指令引导，该指令旨在生成自然语言批评，根据生成答案的准确性和相关性来评估提示的质量。这个批评就是与查询相关的提示的文本奖励。这个元指令模板的一个例子以及一个具体实例如图 9 所示。

训练样本 每个合成的训练样本由一个与查询相关的提示和其对应的文本奖励组成。为了融入这种监督，我们进行监督微调（SFT），训练提示模型根据文本奖励生成原始提示。通过最小化生成提示与最初用于计算文本奖励的参考提示之间的损失来优化模型。一个由文本奖励及其关联提示组成的训练对例子如图 10 所示。

A.1 训练细节

在每个数据集中，我们在专门的测试集上计算测试分数，其中 GSM8K 有 1,319 个样本，GSMHard 有 319 个样本，而 MATH 有 3,369 个样本。训练集包括 GSM8K 的 7,473 个问题，GSMHard 的 1,000 个问题，以及 MATH 的 5,136 个问题。我们对每个训练集应用 90 % –10 % 的训练验证比例划分。

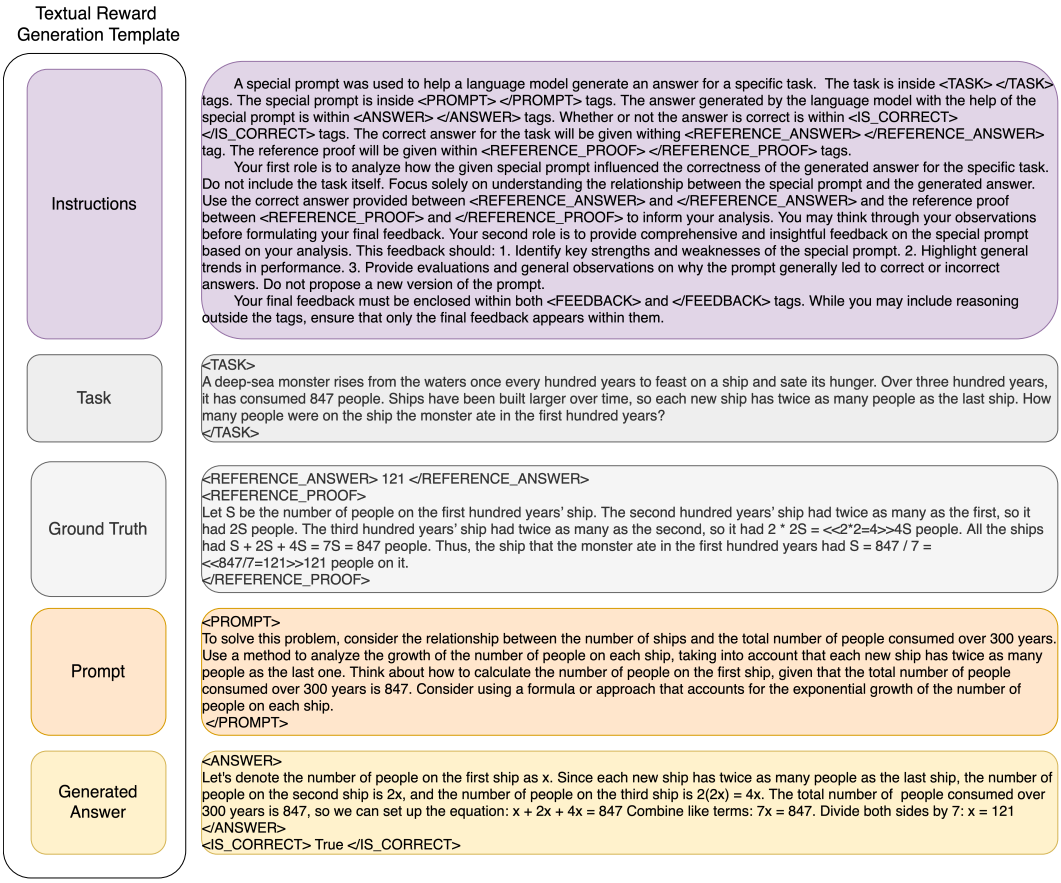

Figure 9: 指导奖励模型生成文本奖励的指令模板和示例。输入包括任务、查询相关提示、真实答案以及目标模型在使用查询相关提示时对问题的回答。

合成数据集。 在每次迭代中，创建一个包含 800 长度的合成数据集，其中包含提示及其对应的文本奖励。为了创建该数据集，我们在每次迭代中随机从训练集中抽取 800 个问题。我们在论文中报告的所有最终结果都是基于每个数据集的专用测试集。我们使用验证集来选择最佳检查点。

世代。 在评估提示的性能时，我们配置目标 LLM 的推理温度为 0.001，以便在生成针对问题的响应时，使用我们希望评估的查询相关提示。我们还将文本奖励生成的温度设置为 0.001。在通过提示模型生成新的查询相关最优提示以鼓励多样化和创造性生成新提示用于训练时，我们将默认温度设置为 0.9。我们使用 64 的批量大小进行生成。

训练细节。 我们训练提示模型进行 4 次迭代。

时域傅里叶变换。 我们通过监督微调 (SFT) 在包含提示及其奖励对的合成数据集上训练提示模型。由于内存和成本限制，我们选择使用低秩适应 (LoRA)，这是一种旨在加速微调同时保持低内存消耗的方法。LoRA 将权重更新矩阵分解为两个较低维度的矩阵，这些矩阵更容易计算并存储在内存中。由于其优点，LoRA 是优化微调 LLM 的合适选择，同时保持计算效率和模型性能。对于 LoRA 参数，我们利用 $r = 256$ 和 $\alpha = 256$ 。提示模型在每次迭代中以 2 个 epoch 训练，学习率为 2×10^{-5} 。我们使用 Adam 优化器，并采用线性学习率衰减。

Textgrad 对于 Textgrad，我们使用 GPT-4o-mini 来修改提示。Textgrad 运行 10 次迭代，并使用验证集来比较最佳奖励的质量。

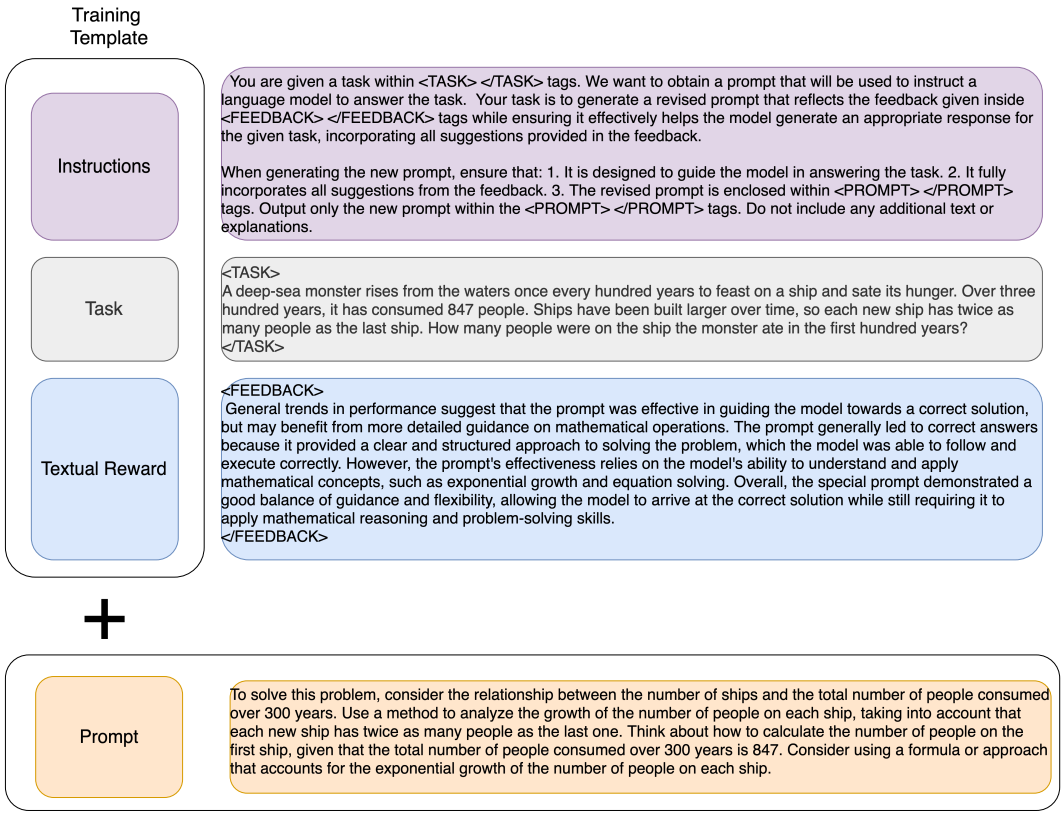

Figure 10: 受监督微调中使用的训练对示例。提示模型被训练以根据提供的文本奖励再现与查询相关的提示。

硬件要求。 训练是在具有 80 GB RAM 的 NVIDIA A100 GPU 上进行的。训练使用了 GPU 内存。根据数据集不同，训练时间为 48 小时至 72 小时。训练的瓶颈是 Textgrad 步骤，该步骤无法并行化。确定每次迭代的最佳奖励大约占总训练时间的 70 %，主要是由于 Textgrad 中不可并行化的评估步骤显著减慢了训练过程。