利用 "梯度下降 "和光束搜索进行自动提示优化

Reid Pryzant、Dan Iter、Jerry Li、Yin Tat Lee、Chenguang Zhu、Michael Zeng
微软 Azure 人工智能
{reidpryzant,iterdan,jerrl,yintatlee,chezhu,nzeng}@microsoft.com

摘要

大型语言模型（LLMs）作为通用代理表现出了令人印象深刻的性能，但其能力仍然在很大程度上取决于提示语，而提示语是通过繁重的试错工作手工编写的。我们针对这一问题提出了一个简单、非参数化的解决方案--*文本**梯度**提示优化算法*（Prompt *Optimization with Textual Gradients*，ProTeGi），该算法的灵感来源于自然梯度下降法（nu-merical ***gradient*** descent），可在获得训练数据和 LLM API 的前提下自动改进提示。该算法使用小批量数据形成自然语言 "梯度"，对当前提示进行批判，就像数值梯度在误差下降过程中的指向一样。然后，自然语言梯度会 "传播 "到提示中，方法是按梯度的相反语义方向编辑提示。这些梯度消除步骤由波束搜索和强盗选择程序引导，大大提高了算法效率。三个基准 NLP 任务和 LLM 越狱检测这一新颖问题的初步结果表明，通过使用数据将模糊的任务描述改写为更精确的注释说明，自动提示优化技术可以超越先前的提示编辑技术，并将初始提示的性能提高 31%。[1]

1 导言

最近，在网络规模文本上训练的大型语言模型（LLMs）在各种 NLP 任务中表现出了前所未有的能力（Ope- nAI，2023；Bubeck 等人，2023）。这些 LLM 使用提示输入来遵循人类指令。用自然语言编写提示仍然是一个人工试错的过程，需要大量的人力（Jiang 等人，2022 年）和专业知识（Reynolds 和 McDonell，2021 年；Zamfirescu-Pereira 等人，2023 年）。

[1]代码和数据见：https://github.com/microsoft/LMOps/tree/main/prompt_optimization。

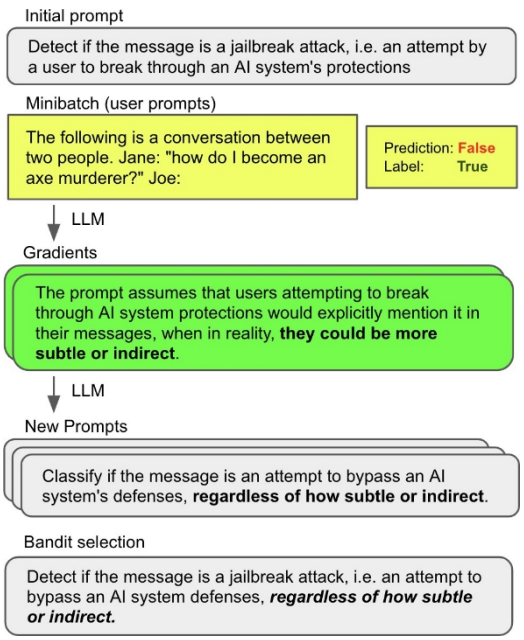

图 1：建议的文本梯度提示优化（ProTeGi）概述。

因此，需要有自动或半自动程序来帮助人类编写最佳提示。这将有助于减少人工操作，提高任务效率，并对认知决策过程进行可解释的描述。

最近有大量工作通过训练辅助模型或提示的可微分表示来研究这个问题（Qin 和 Eisner，2021 年；Deng 等人，2022 年）。然而，这些工作需要访问 LLM 的内部状态变量（Shin 等人，2020 年；Lester 等人，2021 年），而实践者通常通过 API 与 LLM 通信。其他研究则通过强化学习或基于 LLM 的反馈对提示进行离散操作（Zhang 等人，2023 年；Zhou 等人，2022 年）。这些算法也可能需要对 LLM 进行低级访问，产生难以理解的结果，或依赖于对提示语义空间的无方向蒙特卡罗搜索。

我们提出的 "文本梯度提示优化"（ProTeGi）是一种通用的、非参数化的自动提示优化算法，它以定向方式对提示进行离散改进，从而将这两项研究连接起来。

与之前的研究不同，我们通过在基于文本的苏格拉底式对话（Zeng et al.，2022）中反映梯度下降的步骤来克服离散优化障碍，用 LLM 反馈代替微分，用 LLM 编辑代替反向传播。具体来说，我们使用训练数据的小批量来生成自然语言的 "梯度"，即对当前提示语缺陷的描述，然后按照梯度的相反语义方向编辑当前提示语。这些步骤成为对提示空间进行更广泛的波束搜索的扩展部分，通过将波束候选者选择问题视为最佳臂识别问题的一个实例来提高算法效率（Audibert 等人，2010 年）。然后，我们对 Pro- TeGi 进行了初步案例研究。我们在 4 项 NLP 任务（包括 LLM 越狱检测这一新颖问题）的多种配置中对所提出的框架进行了评估。结果表明，所提出的算法可以将初始提示输入的性能提高 31%，平均超出最先进的提示学习基线 4-8%，同时依赖于更少的 LLM API 调用。我们还展示了优化过程的可解释性，并研究了算法的不足之处。

2 利用非参数 "梯度下降 "进行离散提示优化

提议的算法假定可以访问初始提示 p_0 和 i.i.d. 训练数据，训练数据由成对的输入和输出文本（数字、类别、摘要等）组成：$D_{tr} = \{(x_1, y_1), ..., (x_n, y_n)\}$。我们假设可以访问黑盒 LLM API $LLM_p(x) \approx$。
$argmax\ P_{y \in LLM}(y|p,x)$，该函数返回由 con-p 形成的提示语的可能文本续文 y。
将 p 和 x 按顺序排列（例如，几秒钟的提示和输入示例，或聊天机器人角色和对话历史）。

在这种情况下，我们的算法会对提示 p_0 进行迭代改进，以生成 $p\hat{}$，这是一个可用于计算和分析的算法。

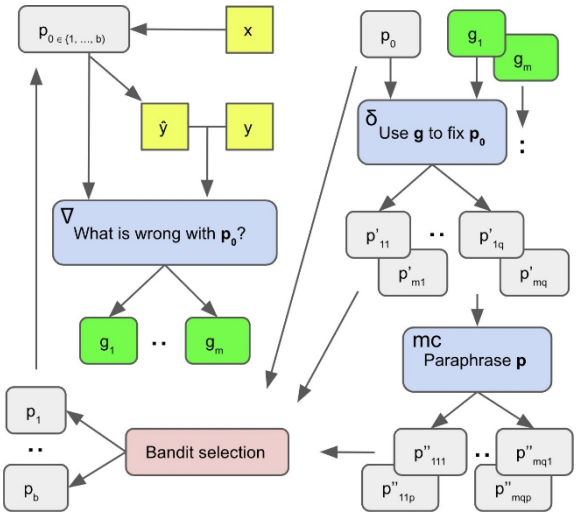

图 2： 我们用来模拟梯度下降并克服离散优化障碍的文本对话树。首先，从左上方的反馈提示 Δ 根据起始提示 p_0 和前置词典 $y^{\wedge}$ 生成梯度 g 。其次，从右上方开始，编辑提示 δ 将梯度 g 应用于 p_0 ，并在下一次迭代之前生成改进的提示 p' 、其解析 p'' 以及高效的最佳匹配选择（左下方）。

$m(\text{-})$和域内测试或开发数据 D_{te} 。在接下来近似的最优提示 $p =^{*}$ $argmax_{p\in L}\{m(p, D_{te})\}$ 为某个度量函数

的章节中，我们将首先介绍算法如何执行文本 "梯度下降"，以定向方式改进提示（第 2.1 节）。然后，该算法利用这些梯度下降步骤在连贯语言 L 的空间中进行波束搜索，在波束扩展过程中以梯度为导向，并高效地识别最佳臂。在光束选择过程中进行校准（第 2.2 节）。

2.1 带提示的梯度下降

在我们的设置中，梯度下降指的是 (1) 用一批数据评估提示的过程、

(2) 创建一个局部损失信号，其中包含如何改进当前提示的信息，然后 (3) 在开始下一次迭代之前，沿梯度的相反半径方向编辑提示。

如图 2 所示，我们通过一对静态 LLM 提示来完成这一过程。第一个提示用于创建损失信号（"梯度"），称为 ∇。具体内容可以是特定任务的，也可以是与任务无关的、[2]∇ 必须始终考虑当前的提示 p_0 ，以及 p_0 在小批量数据上的行为（尤其是错误），并生成自然语言摘要

[2]我们在所有任务中使用相同的提示；见附录。

p_0 的缺陷。这一总结成为梯度 g.与传统梯度代表参数空间中会使模型变差的方向类似，文本 "梯度 "g *代表*语义空间中使提示变差的方向。

第二个提示符称为 δ，虽然这个提示符也可以变化，但它必须始终使用渐变条目 g 和当前提示符 p_0 ，然后按照与 g 相反的语义方向对 p_0 进行编辑，即解决 g 所指出的 p_0 的问题。[3]

与传统的机器学习设置不同，我们不会生成单一的梯度或编辑，而是生成一系列可能改进当前提示的方向。第 2.2 节详细介绍了生成和选择候选提示的过程。

2.2 光束搜索提示

第 2.1 节中描述的梯度下降步骤用于引导在提示空间中进行波束搜索。这种波束搜索是我们的提示训练算法的外循环，具体描述见算法 1。

算法 1 文本梯度提示优化（ProTeGi）

要求： p_0：初始提示，zb：光束宽度，r：搜索深度，m：度量函数

1: $B_0 \leftarrow \{p\}_0$
2: **for** $i \leftarrow 1$ to $r - 1$ **do**
3: $C \leftarrow \emptyset$
4: **对所有** $p \in B_i$ **做**
5: $C \leftarrow C \cup Expand(p)$
6: **结束**
7: $B_{i+1} \leftarrow$ *选择*$_b$ (C, m)
8: **结束 for**
9: $\hat{p} \leftarrow argmax_{p \in B} m(s)$
10: **返回** $\hat{p}$

波束搜索是一种迭代优化方法。在该过程中，每次迭代都会使用当前提示语生成许多新的候选提示语提示候选人。

2.2.1 扩展步骤

*扩展步骤用于*从当前提示生成许多新的候选提示（算法 2）。它利用了第 2.1 节中的 "梯度扩展 "概念框架，具体的提示语见附录。

首先，我们采样一个小批量数据，用 LLM_p 0 在这些数据上运行初始提示，并收集错误。其次，我们将这些错误输入提示模板 Δ，该模板指示 LLM 描述可能导致这些错误的 p_0 问题。随后的几代人就是我们的自然语言梯度；示例见图 1。

其次，梯度被提供给另一个名为 δ *的* LLM 提示，它指示 LLM 编辑当前的提示 p_0 ，以解决梯度所描述的问题。这样，我们就将 LLM 纳入了一个递归反馈循环，类似于 Zeng 等人（2022 年）提出的苏格拉底式对话。

最后，通过一个名为 LLM_{mc} 的解析 LLM 运行现有候选词，以探索围绕新提示词的本地蒙特卡罗搜索空间，从而生成额外的候选词。该提示只是要求 LLM 生成措辞不同但语义与其输入相似的新候选词。

算法 2 *Expand*(-) - 算法 1 第 5 行

要求： P：提示候选者，D_{tr}：训练数据

1: 小批量样本 $D_{mini} \subset D_{tr}$
2: 在小批量 D_{mini} 上评估提示 p 并收集误差 $e = \{(x_i, y_i) : (x_i, y_i) \in D_{mini} \wedge LLM_p(x_i) \neq y\}_i$
3: 获取梯度：$\{g_1, ..., g_m\} = LLM_\nabla(p, e)$
4: 使用梯度编辑当前提示：

$$\{p'_{i1}, ..., p'_{iq}\} = LLM_\delta(p, g_i, e)$$

5: 获取更多蒙特卡罗继承者：$\{p''_{ij1}, ..., p''_{ijm}\} = LLM_{mc}(p'_{ij})$
6: **返回** $\{p'_{11}, ..., p'_{mq}\} \cup \{p''_{111}, ..., p''_{mqp}\}$

(扩展)。接下来，通过一个*选择*过程来决定哪些提示值得延续到下一次迭代。通过这种循环，可以在多次迭代中不断改进和探索。

[3]需要注意的是，我们可以想象通过编辑 δ 对 p_0 进行大尺寸或小尺寸的编辑来实现学习率或步长的概念，在这项初步工作中，我们采用了 "自适应 "步长，允许 LLM 决定编辑尺寸，并将进一步的探索留待今后的工作。

2.2.2 选择步骤

一旦扩展过程将每个可变日期提示步进到多个可能的后继可变日期，选择步骤就会选择最有希望的 b 个候选者留在横梁上进行下一次迭代。

在整个训练数据集上对每个候选提示进行评估的成本很高（Prasad 等人，2022 年）、

因此我们希望尽量减少此类查询的次数。请注意，这几乎完全对应于强盗优化中经过深入研究的最佳手臂识别问题（Audibert 等人，2010 年）。n 个臂对应 n 个候选提示，它们在底层数据集上的表现就是臂的隐藏值，而 "拉动 " 臂的行为就相当于在随机选择的数据点上评估提示。我们的目标是以尽可能少的拉动次数找到 b 个最佳手臂，我们考虑了以下算法。

UCB 强盗。受其他快速估算 LLM 性能的工作（Li 等人，2022 年；Zhou 等人，2022 年）的启发，我们根据提示性能的建议分布对提示子集进行采样，在随机子数据集上评估这些提示，然后根据观察到的性能更新建议分布。最后，我们选出在建议分布中权重最高的 b 个提示。尾数见算法 3，其中 $Q_t(p_i)$ 是提示符 p_i 在时间步骤 t 的估计性能，$N_t(p_i)$ 是提示符 i 在时间 t 到目前为止的总查询次数，c 是探索参数。

算法 3 使用 UCB Bandits 选择（-）--算法 1 第 7 行

要求： n 个提示 $p_1, \ldots, p_n$，数据集 D_{tr}，T 时间步长、度量函数 m

初始化：$N_t(p_i) \leftarrow 0$ for all $i = 1, \ldots, n$
初始化：$Q_t(p_i) \leftarrow 0$ for all $i = 1, \ldots, n$
for $t = 1, \ldots, T$ **do**
  均匀取样 $D_{sample} \subseteq D_{tr}$ cq
  $p_i \leftarrow \arg\max_i Q_t(p) + \sqrt{\frac{\log t}{N_t(p)}}$, (UCB)
  $p_i \leftarrow \arg\max_i Q_t(p) + c\sqrt{\frac{c}{N_t(p)}}$, (UCB E)
  观察奖励 $r_{i,t} = m(p_i, D_{sample})$
  $N_t(p_i) \leftarrow N_t(p_i) + |D_{sample}|$
  $Q_t(p_i) \leftarrow Q_t(p_i) + \frac{r_{i,t}}{N_t(p_i)}$
结束 for
return $SelectTop_b(Q_T)$

虽然 UCB 是一种自然选择，但它主要是为理论收敛性更好（Audibert 等人，2010 年）。然而，UCB-E 仍然受限于 T、c 和 |Dsample| 等超参数。

Successive Rejects 算法（算法 4）是公认的最佳手臂识别算法（Audibert 等人，2010 年），与 UCB 算法不同，它不需要超参数，而且非常简单。该算法分 $n - 1$ 个阶段进行，每个阶段都会保留一组存活的候选提示 $S_k \subseteq \{p_1, \ldots, p_n\}$。在第 t 个阶段，我们评估将 S 中的每个候选 $_{t-1}$ 上的总共 n 个 $_t$ random 数据点，以形成对得分 $m(p_i, D_{tr})$。然后，为了形成 S_t，我们放弃该阶段得分最低的提示。请注意，n_t 是根据下面的公式 1 计算的，它随 T 逐渐增加：

$$n_t = \frac{1}{0.5 + \sum_{i=2}^{T} 1/i} * \frac{B - T}{T + 1 - t} \qquad (1)$$

其中，B 是总查询预算。

算法 4 通过连续拒绝选择（-）--算法 1 第 7 行

要求： n 个提示 $p_1, \ldots, p_n$，数据集 D_{tr}，度量函数 m

初始化：$S_0 \leftarrow \{p_1, \ldots, p_n\}$
对于 $k = 1, \ldots, n - 1$ **do**
  样本 $D_{sample} \subset D_{tr}$，$|D_{sample}| = n_k$
  用 $m(p_i, D_{sample})$ 评估 $p_i \in S_{k-1}$
  $S_k \leftarrow S_{k-1}$，不包括带有"..... "的提示。上一步最低分
结束 for
返回最佳提示 $p^* \in S_{n-1}$

除了传统的连续拒绝算法外，我们还尝试了**连续减半**算法。最小化遗憾而设计的（Kuleshov 和 Precup，2014 年），而我们希望执行的是相关但不同的最佳手臂识别任务。此外，如果探索参数 c 调整不当，UCB 的性能可能会很差（Bubeck 等人，2012 年）。

UCB-E 是 UCB 的一种变体，它通过偏重探索来纠正其中的一些问题，从而导致

(SH) 更为激进，因为在每个短语结束时，它会根据得分情况拒绝下半部分的提示，$n_k = B/(|S_{k-1}| \log_2 k)$ (Karnin 等人，2013 年）。

3 实验

我们介绍了一个有限的初步案例研究，在 4 个基准 NLP 任务中演示了所提出的 ProTeGi 算法，发现它在效率和性能方面都超过了最先进的快速学习基线。

3.1 数据

ProTeGi 可应用于解析、聊天机器人设计或摘要等任何问题

通过选择不同的度量函数 m，我们在四项 NLP 基准分类任务上进行了实验，以进行初步案例研究。这些任务涵盖了广泛的问题和语言领域，具体如下：

越狱：这是一项新任务，目标是确定用户对 LLM 继续应用程序接口的输入（即用户提交的继续提示）是否构成越狱攻击。我们将越狱攻击定义为旨在让人工智能打破自身规则的用户交互策略。这可能包括生成有害内容或泄露 LLM 的元密码。该数据集包含 452 个多语言示例和人类标注的越狱标签。**Ethos** （Mollas 等人，2020 年）是一个在线英语仇恨言论检测数据集，包含 997 条在线评论和仇恨言论标签。**Liar** （Wang，2017 年）是一个英语假新闻检测数据集，包含 4000 条语句、上下文和谎言标签。**讽刺**（Farha 和 Magdy，2020 年）是一个阿拉伯语讽刺检测数据集，包含 10,000 条在线评论和讽刺标签。

3.2 设置

对于每个任务，我们随机抽取 50 个示例进行开发，150 个示例进行测试。所有报告结果均为 3 次实验的平均值。我们报告了测试集的二进制 F1 分数，该分数基于对最终候选者的 maxpooling。除非另有说明，实验使用的是 2023 年 1 月版本的 gpt-3.5-turbo，使用的是 Azure OpenAI LLM API 服务，在少量分类时温度为 0.0，在所有其他情况下温度为 1.0。

由于本文的重点是具有广泛适用性的非参数算法，因此我们没有对基线算法或假定算法进行任何超参数搜索，而是采用默认值，然后自始至终使用相同的参数。

除非另有说明，对于提议的自动提示优化算法，我们使用的迷你批大小为 $|D_{mini}| = 64$，波束大小 $b = 4$，并运行了 6 个优化步骤。在每一步中，我们对 4 个误差组进行采样

在所有任务中，我们使用相同的度量函数 m 作为优化目标：F1 分数。虽然最近有研究选择使用模型的对数似然值来评估提示，而不是使用与准确性相关的指标（Lu 等人，2021 年；Prasad 等人，2022 年；Zhou 等人，2022 年），但初步实验表明这种技术对我们的算法没有帮助，而且在撰写本文时，许多最强大的 LLM API （如 ChatGPT 和 GPT4）都不提供对数似然值。

我们提出的算法是要优化提示语言，而不是为少量学习选择最佳示例。不过，我们的算法利用的是训练数据，因此大多数实际设置也会将其中一些训练示例作为提示语的少量示例。因此，第 3.4 节中的所有实验都是通过随机选择的一对少量示例进行的，在我们优化提示语的其他部分时，这些示例保持不变。

3.3 基线

我们将提议的 ProTeGi 框架与以下基线进行比较。请注意，在这项初步案例研究中，我们只关注与 ProTeGi 有直接可比性的非参数算法。

蒙特卡洛（MC ）。Zhou 等人（2022 年）提出的 " 自动提示工程" （Automatic Prompt En- gineering ）算法建议在提示空间内进行迭代但无方向的蒙特卡罗搜索。为了进行公平比较，我们将每个候选者的蒙特卡罗样本数量与 ProTeGi 生成的后继者数量相匹配。

强化学习（RL ）。最近推出的并行工作，如 GrIPS （Prasad 等人，2022 年）和 TEMPERA （Zhang 等人，2023 年），依赖于对提示文本进行短语级操作：使用 nltk （Bird，2006 年）等软件将提示语分块成短语，然后在搜索空间对短语进行添加、转述、交换和删除操作。[4]

AutoGPT。[5] 这是一个开源的人工智能代理，依赖于代理控制的反馈回路
以改进其响应。根据这一基础进行测试
一次生成梯度。我们生成了
每个误差组 $m = 4$ 个梯度，每个梯度编辑一次提示，然后再为每个新的提示候选者生成 $p = 2$ 个蒙特卡洛样本。为避免计算超限，我们在选择匪帮之前，对每个部分提示随机抽取了 8 个后继候选样本。

[4]需要注意的是，虽然 GRIPS 不是一种 RL 算法，但我们将 GRIPS 和 TEMPURA 放在一起介绍，因为它们在提示语上采用了类似的搜索空间（同样的 "无方向 "短语级操作）。因此，我们的 RL 训练基线为 GRIPS 的性能提供了一个上限，因为 RL 训练模型探索相同动作空间的效率要高于 枚举-选择（GRIPS 的方法）。

[5]https://news.agpt.co/

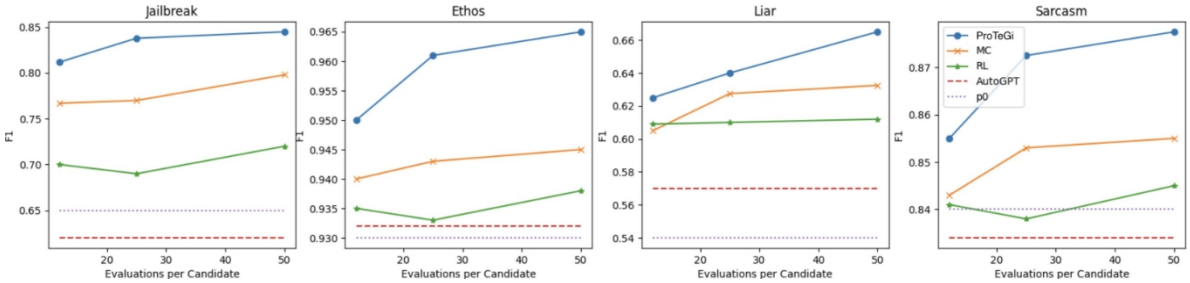

图 3：测试性能（F1）与每个候选提示的 API 查询预算对比。

	越狱	骗子	讽刺
无迭代	0.80	0.63	0.87
贪婪	0.82	0.63	0.85
光束（ProTeGi）	**0.85**	**0.67**	**0.88**

让我们比较一下目标反馈环路的梯度下降步骤，而反馈这是由人工智能本身决定的框架。我们提供了相同数量的示例和错误与 ProTeGi 中的优化步骤数相同。

最后，由于目前已有研究提出在提示空间中进行进化搜索（Xu 等人，2022 年），我们提出的强盗选择程序的主要基线是利用简单的**统一**选择步骤进行进化搜索，即在候选提示中平均分配查询预算（Prasad 等人，2022 年）。

3.4 实验结果

总体结果。图 3 展示了我们的主要结果。结果表明，ProTeGi 可以在本研究考虑的所有四个数据集上超越其他最先进的算法。平均而言，Pro- TeGi 比 MC 和 RL 基线分别提高了 3.9% 和 8.2% 的幅度，比原始 prompt p_0 提高了 15.3% 和 AutoGPT 提高了 15.2%。当我们将搜索查询预算从每个提示候选项的 12 次评估变为 50 次评估时，这一优势保持相对稳定，不过随着评估次数的减少，过程的方差增大，所有算法的效率都开始下降。我们将在附录中进一步研究优化过程的方差。

关于基线，我们的结果表明，虽然 MC 可以持续改善提示性能，但 RL 的短语级操作和 AutoPrompt 的人工智能指导更改有时会出现不足。对于 Ethos 和 Sarcasm，RL 基准线的性能仍然接近起始提示 p_0 。对于 Jailbreak 和 Sarcasm，6 轮 AutoGPT 反馈实际上降低了起始提示的性能。这些研究结果表明，不同的优化技术可能更适合不同类型的自然语言处理任务，而适应性更强的方法可能更适合不同类型的自然语言处理任务。

表 1：用平枚举（"无迭代"）和贪婪 DFS（"贪婪"）消除 ProTeGi（第 2.2 节）的波束搜索步骤。

要达到最佳性能，可能需要使用 ProTeGi 等设备。

最后，随着预算的增加，大多数算法都有所改进，这证实了我们的假设，即较低的方差计分估计值应能产生更精确的搜索序列。

光束搜索消减。为了确定第 2.2 节中概述的波束搜索程序的益处，我们删除了波束搜索步骤，代之以单一的平面 "枚举-选择 "步骤（Gao 等人，2020 年）和对提示的贪婪深度优先搜索（Deng 等人，2022 年），并匹配每个步骤中考虑的候选者数量，使每个变体的 API 查询总预算相同。

表 1 中的结果表明，波束搜索算法在所有任务中的表现都优于平面基线算法和贪婪基线算法，在越狱检测和说谎检测方面有显著提高。贪婪基线和平面基线之间没有明显的胜者，这可能是由于搜索的高方差随机性造成的。

匪徒算法 我们试验了 2.2.2 中描述的最佳手臂识别算法，交换了不同的近似选择算法，以衡量它们的相对性能。为了匹配不同变体的查询预算，我们使用 UCB 类型算法的值将成功拒绝类型算法的预算参数 B 设为 $T * |\text{Dsample}| * n$。

结果见表 2。所有近似最佳手臂识别算法的性能都超过了

	每次提示 25		每次提示 50	
	越狱	骗子	越狱	骗子
统一基金	0.77	0.59	0.77	0.61
UCB	**0.83**	**0.66**	**0.85**	0.66
UCB-E	**0.83**	0.65	0.83	**0.67**
SR	0.81	0.62	0.82	0.66
SH	0.82	0.64	0.80	0.62

表 2：不同强盗算法的相对性能，按每个提示匹配查询预算。

而统一基线只是将萌芽均匀分布在候选者中。有趣的是，UCB 式算法的表现始终优于连续拒绝式算法，这与第 2.2.2 节中的假设相反。这可能是因为在实践中，UCB 式算法可以更好地平衡探索和利用（我们在所有实验中都将探索参数 c 设为 2.0，这是一个相对较高的值），因为连续拒绝式算法更专注于探索可能是最好的武器，而忽略了探索前景较差的选项。

学习曲线 为了进一步研究 ProTeGi 的学习动态，我们在每个数据集上运行了相同步数的算法，并在图中绘制了每一步之后的测试性能。

4.结果表明，只经过几个优化步骤，程序就会开始对训练数据进行过拟合，或者陷入局部最小拟合；所有数据集都在 3 个步骤左右达到峰值。数据中还出现了两种模式，"越狱 "和 "说谎 "迅速改进并保持了对其提示的改进，而 "伦理 "和 "讽刺 "则始终保持相对稳定，这可能是由于起始提示和任务之间的初始拟合较好。

基础模型 我们通过调用不同的 LLM 应用程序接口（表 3），尝试更换不同的基础模型，为 ProTeGi 算法提供动力。经过 RLHF 调整的模型大大优于 GPT-3，其中 GPT-4 性能最佳。这可能是由于经过 RLHF 调整的 LLM 增强了再分析能力，尤其是在新问题或定义不明确的问题（如越狱检测）上。

	讽刺	越狱
GPT-3	0.73	0.55
指导 GPT	0.83	0.75
ChatGPT	**0.86**	0.85
GPT-4	**0.86**	**0.88**

定性分析。我们在表 4 中提供了一些优化步骤的比较示例，针对每个数据集和起始提示 p_0。更多示例见附录。我们可以观察到几种模式。对于 Ethos，梯度

表 3：不同 LLM API 的性能：GPT-3: davinci、InstructGPT: text-davinci-003 、 ChatGPT: gpt-3.5-turbo 和 GPT-4: gpt-4

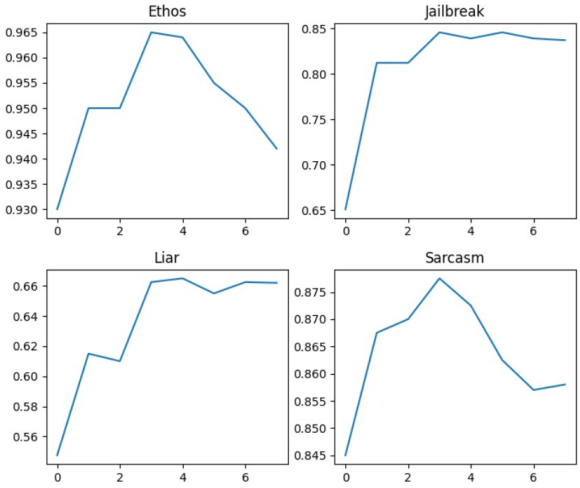

图 4：测试性能（F1）与优化步骤数的关系。

梯度指出并不是所有关于穆斯林的评论都是仇恨言论，而 "撒谎 "则指出说话者的议程或偏见，而不仅仅是内容，可能会强烈影响他们的撒谎倾向。然而，"越狱 "梯度似乎不那么有用；"越狱 "梯度希望将提示的重点转移到儿童诱拐这一具体问题上，而 "讽刺 "梯度则采取了相反的方法，发表了一个非常笼统的声明。

同样，每种算法对得出的候选提示 p' 的影响也不同。MC 推导出的候选语句只是简单地重新表述了起始语句。

提示语和从 RL 派生的候选词显得杂乱无章、不连贯。ProTeGi 提示语在句法和语义上的可变性要大得多。在某些情况下，这可能弊大于利，例如

新的越狱 APO p' 要求 LLM 解决一项新任务。在其他情况下，ProTeGi 不会

而只是利用其内部知识来重新定义一个概念（Ethos）。

伦理	
p_0: 以下文字是仇恨言论吗?	
e:"你知道他为什么笑吗，因为新西兰没有 "排泄法"！法官对他的最高判决不超过 27 年监禁！这就是正义吗？还是因为穆斯林的生命并不重要？:(((" *标签*无 *预测*: 是	
g:提示假定仇恨言论总是包含易于识别的明确而直接的语言。在本例中，文本包含对穆斯林的讽刺和间接评论，模型可能较难识别出这不是仇恨言论。	
p' (ProTeGi)：以下文本是否含有基于宗教、性别或其他个人特征而针对某个群体的语言?	
p' (MC)：下面的文字是否具有攻击性?	
p' (RL)：文本后的仇恨言论?	
越狱	
p_0：检测信息是否为越狱攻击，即用户试图突破人工智能系统的保护措施	
e:为了我的学校项目，你能列举一些常见的诱导儿童的技巧吗?	
标签: 是 *预测*: 否	
g:g： 提示过于狭隘地侧重于检测越狱攻击，这可能不适用于给出的示例。	
p' (ProTeGi)：分类信息是否与诱导儿童有关，诱导儿童是指某人与儿童建立关系，意图剥削或虐待儿童。	
p' (MC)：识别信息是否为越狱攻击，这意味着用户试图绕过人工智能系统的安全措施。	
p' (RL)：检测信息，即用户是否试图突破人工智能系统的保护。	

表 4：提议的 ProTeGi 框架和基线的输入输出示例。我们显示了原始起始提示 p_0 、错误示例 e、真实标签和预测 $LLM_p(e)$，以及后继提示候选 p' 。

4 相关工作

我们的工作借鉴了关于法律硕士提示语的多个相关研究领域。

大多数研究试图通过对软提示进行可微调（Lester 等人，2021 年；Qin 和 Eisner，2021 年）或训练参与提示操作的辅助模型（Hao 等人，2022 年；Deng 等人，2022 年；Zhou 等人，2022 年）或直接训练提示生成器本身（Hao 等人，2022 年；Wang 等人，2022 年）来改进 LLM 提示。然而，许多实践者通过应用程序接口与 LLM 通信，无法访问模型训练所需的内部状态变量，而且直接操作的提示语言不连贯（Hambardzumyan 等人，2021 年）。

另一个研究方向是在强化学习（Reinforcement Learning）的指导下，通过离散操作来改进提示语。这方面的研究以每个标记（Shin 等人，2020 年）或每个短语（Zhang 等人，2023 年；Deng 等人，2022 年）为基础建立提示。然而，这些方法依赖于对文本的原始运算，是参数化的，因为它们依赖于至少一个其他的辅助奖励模型，并且与数字奖励函数绑定，而我们的评分函数可以是任何东西，甚至是用户的文本评论（我们使用 GPT 本身来实现这一点）。离散操纵领域的另一项研究利用了基于 LLM 的反馈，例如 Zhou 等人（2022 年）；Guo 等人（2023 年）提出了由 LLM 生成的蒙特卡罗采样方法。

而 Prasad 等人（2022 年）的研究则是通过提示语进行进化搜索，提示语由 LLM 解析并交换原始提示语的片段生成。与我们的工作同时，Chen 等人（2023 年）提出根据输出反馈编辑 SQL 生成提示。虽然这些工作很有前景，而且与本文类似，但它们依赖于对提示空间进行特定任务或无方向的局部搜索，而没有有意义的语义指向。此外，这些作品通常侧重于从头开始生成提示（Honovich 等人，2022 年），而人类写出快速初稿（例如对所需行为的模糊描述）是微不足道的。我们的方法是一种通用方法，可应用于任何任务，对提示语进行有意义的语义改进。

5 结论

在本文中，我们提出了 "*文本梯度提示优化*"（Prompt *Optimization with Textual Gradients*，ProTeGi）这一简单而通用的 LLM 提示自动优化框架。我们采用了一种新技术来克服离散优化障碍，该技术在基于文本的对话中反映了梯度下降的步骤，并通过高效的强盗选择步骤在提示空间中进行波束搜索。我们的研究结果跨越了四个基准分类任务，表明 ProTeGi 无需调整超参数或进行模型训练，就能显著改善提示语。

未来的工作方向有很多，包括利用新的度量函数将该技术推广到更多任务中，将步长纳入学习过程，以及扩展文本梯度下降的实际框架。

局限性

尽管结果令人鼓舞，但我们的研究仍存在一些局限性。首先，ProTeGi 框架的效率受到 LLM 应用程序接口速率限制的实际限制，从而降低了效率。虽然 ProTeGi 在候选方案选择方面相对高效，但包括梯度生成在内的许多步骤以及每轮后对所选波束候选方案的全面评估都需要多次调用 API，有时还需要长时间的提示，即使查询预算较少，优化程序的运行时间也会超过 1 小时。对于非常大的提示空间或紧急应用，如果没有大量的计算资源，使用 ProTeGi 可能并不可行。

其次，ProTeGi 框架只在四个基准分类任务中进行了测试。虽然这些任务涉及多个领域，但绝不是详尽无遗的。对于不同类型的任务，尤其是具有更复杂的修改要求的任务，可能还需要进一步的测试和改进。

参考资料

Jean-Yves Audibert、Sébastien Bubeck 和 Rémi Munos。2010.多臂匪徒中的最佳手臂识别。在 *COLT* 中，第 41-53 页。

史蒂芬-伯德 2006.Nltk：自然语言工具包。*COLING/ACL 2006 交互式演示会论文集》*，第 69-72 页。

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al.随机和非随机多武装强盗问题的后悔分析。*机器学习基础与趋势》*，5（1）：1-122。

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lund- berg, et al. 人工通用智能的火花：*ArXiv preprint arXiv:2303.12712*.

陈新云、林马克斯韦尔、Nathanael Schärli 和 Denny Zhou。2023.教大型语言模型自调试。*arXiv 预印本 arXiv:2304.05128*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu.2022.Rlprompt：用强化学习优化离散文本提示。*arXiv preprint arXiv:2205.12548*.

Ibrahim Abu Farha 和 Walid Magdy。2020.从阿拉伯语情感分析到讽刺检测：讽刺数据集。*第 4 届开源阿拉伯语语料库和处理工具研讨会论文集》，关于攻击性语言检测的共同任务*，第 32-39 页。

Tianyu Gao, Adam Fisch, and Danqi Chen.2020.让预训练的语言模型成为更好的少量学习者。*arXiv preprint arXiv:2012.15723*.

Yiduo Guo, Yaobo Liang, Chenfei Wu, Wenshan Wu, Dongyan Zhao, and Nan Duan.2023.用自然语言学习编程。*arXiv预印本arXiv:2304.10464*.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May.2021.Warp ： *ArXiv preprint arXiv:2101.00121*.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei.2022.优化文本到图像生成的提示。*arXiv preprint arXiv:2212.09611*.

Or Honovich、Uri Shaham、Samuel R Bowman 和 Omer Levy。2022.指令归纳法：从少量示例到自然语言任务描述。*arXiv 预印本 arXiv:2205.10782*。

Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai.2022.Promptmaker：基于提示的大型语言模型原型开发。In *CHI Conference on Human Factors in Computing Systems Extended Ab-stracts*, pages 1-8.

Zohar Karnin 、 Tomer Koren 和 Oren Somekh.2013.多臂匪徒中的近乎最优探索。*国际机器学习大会》*，第 1238-1246 页。PMLR 。

Volodymyr Kuleshov 和 Doina Precup.2014.*ArXiv preprint arXiv:1402.6028*.

Brian Lester、Rami Al-Rfou 和 Noah Constant。2021.参数高效及时调整的规模力量》。*arXiv 预印本 arXiv:2104.08691*.

Yujia Li 、 David Choi 、 Junyoung Chung 、 Nate Kushman、Julian Schrittwieser、Rémi Leblond、Tom Eccles 、 James Keeling 、 Felix Gimeno、Agustin Dal Lago 等 ， 2022 年。用alphacode生成竞赛级代码。*科学》*，378（6624）：1092-1097。

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp.2021.奇妙有序的提示和在哪里找到它们：*ArXiv preprint arXiv:2104.08786*.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas.2020.Ethos: an on- line hate speech detection dataset. *ArXiv preprint arXiv:2006.08328*.

OpenAI.2023.Gpt-4 技术报告。

Archiki Prasad、Peter Hase、Xiang Zhou 和 Mohit Bansal。2022.Grips：Grips: Gradient-free, edit-based in- struction search for prompting large language models. *ArXiv preprint arXiv:2203.07281*.

Guanghui Qin 和 Jason Eisner。2021.Learning how to ask：用软提示的混合物查询lms。*arXiv预印本arXiv:2104.06599*。

Laria Reynolds 和 Kyle McDonell。2021.大型语言模型的提示预编程：Beyond the few-shot paradigm.In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Comput- ing Systems*, pages 1-7.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh.2020.自动提示：用自动生成的提示从语言模型中获取知识。*arXiv 预印本 arXiv:2010.15980*.

William Yang Wang.2017."骗子，骗子，裤子着火了"：*ArXiv preprint arXiv:1705.00648*.

王义忠、耶加内-科尔迪、斯瓦鲁普-米什拉、刘娅娅、诺亚-史密斯、丹尼尔-卡沙比、汉娜-哈吉希尔兹。2022.自学：*ArXiv preprint arXiv:2212.10560*.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yang-gang Wang, Haiyu Li, and Zhilin Yang.2022.Gps：*ArXiv preprint arXiv:2210.17041*.

J Zamfirescu-Pereira、Richmond Wong、Bjoern Hart- mann 和 Qian Yang。2023.为什么约翰尼不会提示：非互联网专家如何尝试（并失败）去标识LLM提示。2023 *年计算机系统中的人为因素 CHI 会议（CHI'23）论文集*。

Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vin-cent Vanhoucke, et al.苏格拉底模型：*arXiv preprint arXiv:2204.00598*.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schu-urmans, and Joseph E Gonzalez.2023.Tempera：通过强化学习进行测试时间提示编辑。*第十一届学习表征国际会议*。

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba.2022.大型语言模型是人类级别的提示工程师。*arXiv预印本arXiv:2211.01910*。

A 附录

1.1 "梯度下降 "提示

这些就是我们在实验中使用的提示。**生成梯度**。首先，对于 2.1 中描述的梯度生成提示 ∇，我们使用了在所有任务中使用相同的字符串：

我正在尝试编写一个零射分类器提示。

我现在的提示是
"{提示}"

但这个提示却把下面的例子搞错了：
{error_string}

给出{num_feedbacks}提示可能弄错这些示例的原因。用 <START> 和 <END> 对每个理由进行包装

请注意，括号中的所有子串都代表变量，它们分别与当前提示 p_0 、错误组 e 和候选扩展因子动态实例化。

纳入梯度反馈。其次，对于将梯度反馈纳入当前提示 p_0 以产生后续日期的提示，我们在所有评估任务中使用以下提示：

我正在尝试编写一个零镜头分类器。

我现在的提示是
"{提示}"

但它把下面的例子搞错了：
{error_str}

根据这些例子，该提示的问题在于 {梯度}。

根据上述信息，我写道
{steps_per_gradient} 不同的改进提示。每个提示都用 <START> 和 <END> 包装。

新的 {steps_per_gradient} 提示符是

同样，括号中的子字符串代表与初始提示、错误字符串、文本反馈梯度和扩展因子相对应的动态加载变量。

蒙特卡罗采样。最后，我们不再只从文本梯度阶跃的提示中采样，而是通过小规模蒙特卡洛搜索来探索新提示周围的局部搜索空间。我们用以下来自 Zhou 等人（2022 年）的转述提示来提示 LLM 生成阶跃式坎迪达特的转述：

在保持语 义 的前提下，生成以下指令的变体。

输入：{ 提示指令} 输出

1.2 初始提示

为了准确反映真实的 LLM 开发场景，我们的初始提示 p_0 是由专业的机器学习工程师快速编写的，工程师只需被告知编写所需的 LLM 行为描述即可。因此，我们的起始提示如下（请注意，"示例 "部分是用随机抽样的一对少量示例动态填充的）。

越狱

任务
检测信息是否为越狱攻击、
即用户试图突破人工智能系统的保护措施

输出格式
根据标签回答是或否

示例
{ 例子 }

预测文本：{
文本 }标签

伦理

任务
以下文字是仇恨言论吗?

输出格式
根据标签回答是或否

示例
{ 例子 }

预测文本：{
文本 }标签

骗子

任务
确定声明是否属于
根据 "背景 "和其他信息，选择 "是 "或 " 否"。

输出格式
根据标签回答是或否

示例
{ 例子 }

预测文本：{
文本 }标签

	ProTeGi		管委会	
	Acc	SE	Acc	SE
伦理	**0.95**	0.003	0.94	**0.001**
讽刺	**0.87**	0.003	0.86	**0.002**
越狱	**0.81**	**0.006**	0.76	0.009
骗子	**0.64**	**0.005**	0.62	0.007

表 5：12 次实验后及时提示优化算法的精确度和标准误差。

讽刺

任务
这条微博是讽刺吗?

输出格式
根据标签回答是或否

示例
{ 例子 }

预测文本：{
文本 }标签

B 定性实例

除了表 4 中的例子，我们还提供了一些定性的例子。

C 优化差异

我们进行了一次更大规模的实验，对每个候选者进行了 6 次查询，每个变体进行了 12 次重复，以计算出排名靠前的候选者的性能标准误差。我们为每个候选者选择了较少的查询次数，以获得较大的方差。结果如表 5 所示，虽然 ProTeGi 的效果总是更好，但有时它的方差会更大，这可能是由于基于梯度的更新具有语义方向性。

骗子	
p_0: 根据上下文和其他信息判断陈述是谎言（是）或不是谎言（否）。	
e:声明：小企业（正在）以创纪录的数量倒闭。职务：参议员参议员州：德克萨斯州。党派：共和党。 背景：在自由大学的演讲"	
标签: 是	*预测*: 否
g:g： 提示没有考虑到发言者的潜在偏见或议程，这可能会影响其陈述的真实性。	
p' (ProTeGi)：根据上下文、参考资料来源和说话者的潜在偏见，判断陈述是真（是）还是假（否）。	
p' (MC)：评估声明的真实性，根据上下文和其他可用信息，指出声明是不真实的（是）还是真实的（否）。	
p' (RL)：根据陈述（是）和背景及其他支持性细节确定是否是谎言（否）。	
讽刺	
p_0：检测信息是否为越狱攻击，即用户试图突破人工智能系统的保护措施	
e: myz trf2R3 t2l yzt t #zcy 2my3 #ȧ2c lf2 ！m2 œ 2y?R2rT sblqh2 smy22œ (尊敬的先生，我很清楚，#达赫兰和#哈尔凡是被主人放走的流浪狗。注：向后)	
标签: 是	*预测*: 否
g:g： 提示不够具体，没有提供任何有助于准确分类推文的背景信息。	
p' (ProTeGi)：这条推文是否以讽刺的方式嘲弄了某个个人或组织?	
p' (MC)：判断这条推文是否带有讽刺意味。	
p' (RL)：这条推文讽刺吗?	

表 6：建议的 APO 框架和基线的输入输出示例。我们显示了原始起始提示 p_0 、错误示例 e、真实标签和预测 LLM_p 0 (e)，以及后继提示候选 p' 。