arXiv:2602.00997v1 [cs.AI] 1 Feb 2026

错误分类法引导的提示优化

Mayank Singh [1] **Vikas Yadav** [2] **Eduardo Blanco** [1]

摘要

自动提示优化（APO）是一种强大的方法，无需修改大语言模型的权重即可提升其性能。许多现有方法依赖反复试错，测试不同的提示词或上下文示例，直到找到合适的配置，而这通常会消耗大量计算资源。近期，基于执行日志生成的自然语言反馈展现出潜力，可用于识别如何改进提示词。然而，大多数先前方法采用自底向上的方式，根据单个问题的反馈迭代调整提示词，这可能导致它们失去全局视角。在本研究中，我们提出错误分类法引导的提示优化（ETGPO），这是一种采用自顶向下方法的提示优化算法。ETGPO着眼于全局失败分布：收集模型错误，将其归类为分类法，并通过补充指导来针对最常见的失败模式。在涵盖数学、问答和逻辑推理的多个基准测试中，ETGPO取得了与最先进方法相当或更高的准确率，同时所需的优化阶段令牌用量和评估预算约为其三分之一。



图 *1.* ETGPO 概览。我们首先从失败轨迹中收集错误，然后将其整理成分类法。在筛选出最常见的错误类别后，我们为每一类生成可执行的指导，并将其整合为最终提示词。整个流程均实现了端到端自动化。

1. 引言

大语言模型（LLM）对提示措辞、结构和示例十分敏感，因此，提示设计可以在不修改模型权重的情况下有效提升任务性能。然而，手动设计提示需要专业知识、反复试错和投入精力。这推动了越来越多关于自动提示优化（APO）的研究，这类研究将提示设计视为针对离散自然语言字符串的优化问题。早期方法包括 APE（Zhou等人，2023年），它将提示设计表述为黑盒优化问题；OPRO（Yang 等人，2024年）表明，大语言模型可以通过从以往尝试中学习来优化提示词；ProTeGi（Pryzant 等人，2023年）则提出了利用自然语言批评进行反馈驱动式优化的方法。

此后，各种方法相继涌现：通过变异和组合来迭代提示词的进化方法（Guo 等人，2024年；Fernando 等人，2024年）、利用贝叶斯优化指导搜索的历史信息驱动方法（Opsahl-Ong 等人，2024年），以及根据执行错误来改进提示词的反馈驱动方法（Yuksekgonul 等人，2024年；Agarwal 等人，2025年；Agrawal 等人，2025年）。

我们关注反馈驱动方法，这类方法已展现出显著的实证收益，但采用自下而上的方式运作：它们在小批量数据上收集错误并立即修改提示词，存在过拟合特定示例的风险。为缓解这一问题，

[1]亚利桑那大学 [2]ServiceNow。通信请联系：Mayank Singh <mayanks43@arizona.edu>。

预印本。*2026*年*2*月*3*日。

它们投入大量预算，在验证集上反复评估每个候选提示词（Agrawal 等人，2025年）。

为了解决这些问题，我们提出了**错误分类法引导的提示优化（ETGPO）**，这是一种自上而下的自动提示优化方法，通过错误分类法明确建模数据集层面的错误分布。该分类法将基础提示执行轨迹中发现的错误归入不同错误类别，并统计其出现频率，从而以可解释的方式呈现大语言模型在相关领域中的错误。这种结构使我们能够仅针对出现频率最高的错误类别生成有针对性的指导。通过过滤长尾错误，我们确保提示编辑更有可能泛化到具体问题之外，从而减少迭代验证的需求并提高整体效率。

总而言之，我们的贡献如下 [1]：

1. 我们提出了 ETGPO，这是一种自上而下的自动提示优化方法，能够构建全局错误分类体系，并针对常见失败模式生成指导。ETGPO 不依赖于具体任务或大语言模型的选择，同时还能以可解释的方式呈现目标大语言模型的弱点。2. 我们表明，在涵盖数学、问答和逻辑推理的基准测试中，ETGPO 的表现与最先进的方法相当或更优。3. 我们以约三分之一的优化成本实现了这一成果，证明更简单的方法可以取代昂贵的迭代改进。

2. 相关工作

文献中存在许多自动提示优化方法，并且可以用多种不同方式对它们进行分类。就本文而言，我们根据新候选项的提出方式，将其分为三类：反馈驱动方法、进化方法和历史知情方法，并分别讨论每一类中的代表性方法。

反馈驱动方法会分析提示词在示例上的性能，并利用这些批评来指导修改，从而迭代优化提示词。ProTeGi（Pryzant 等人，2023年）会迭代收集训练示例小批次中的错误，使用大语言模型总结问题所在，并据此修改提示词。它通过束搜索维护多个候选项，并根据验证性能选出最佳项。TextGrad（Yuksekgonul 等人，2024年）将提示优化视为梯度下降：一个大语言模型生成对输出的批评，另一个大语言模型则修改提示词以解决这些问题。它沿着单一优化轨迹进行，而不是通过束搜索维护多个候选项。PromptWizard（Agarwal 等人，2025年）也会使用大语言模型分析推理失败、识别提示词的弱点，并改进提示词。不过，与 TextGrad 不同，它还会从成功的运行中收集上下文示例，并联合优化提示词和上下文示例。

进化方法会维护多个提示候选项，通过组合和变异使其不断进化，同时淘汰表现较差的候选项。EvoPrompt（Guo 等人，2024年）维护一个提示候选项池，并使用大语言模型对其进行组合或改写，每轮筛选出排名前 k 的候选项。PromptBreeder（Fernando 等人，2024年）在 EvoPrompt 的基础上进一步扩展，同时进化任务提示词和用于变异这些提示词的提示词，使系统能够随着时间推移学会更好的变异策略。GEPA（Agrawal 等人，2025年）是一种混合方法，它依据执行反馈对提示词进行变异和组合，同时维护一个多样化的人口，其中每个候选项都擅长处理不同的示例子集，并根据总体覆盖率选出最佳提示词。

历史知情方法会迭代改进提示词，利用以往尝试及其得分的轨迹为新的候选项提供依据。OPRO（Yang 等人，2024年）使用大语言模型提出新的提示词，同时向其展示以往提示词及其得分的历史，使其从先前的尝试中学习。APE（Zhou等人，2023年）使用大语言模型生成提示候选项，并通过改写表现最佳者来迭代优化它们，将成功的提示词视为隐含的历史指导。MIPRO（Opsahl-Ong 等人，2024年）同样使用大语言模型生成提示候选项，但还会优化来自成功运行的上下文示例，并使用贝叶斯优化来选择有前景的组合。

ETGPO是我们的方法，属于反馈驱动，但采取了与现有方法不同的途径。它并不像进化方法和历史知情方法那样维护多个提示候选项，也不像其他反馈驱动方法那样基于个别错误进行迭代编辑，而是构建全局错误分类体系，并为最普遍的失败模式生成有针对性的指导。这种自上而下的方法能够实现精准、高影响力的干预，减少反复评估候选项的需要，并降低总体优化成本。

3. 方法

在介绍我们的方法之前，我们先说明问题设定并给出一些必要的定义。

我们考虑一个问题数据集，其中每个问题都有一个真实答案。同时，我们还给定一个基础系统提示，它与问题文本一起用于向大语言模型发起查询，从而生成预测答案。自动提示优化的目标是通过附加额外文本来改进这个基础系统提示，使模型在整个相关数据集上的回答准确率（以真实答案为参照进行衡量）得到提升。

[1] 代码地址：https://github.com/mayanks43/etgpo

现在我们介绍相关定义。首先，我们区分方法中使用的两类大语言模型。*骨干大语言模型*用于解决相关数据集中的问题（即生成推理轨迹和答案）。*优化器大语言模型*用于改进传递给骨干大语言模型的系统提示。

其次，我们将*失败轨迹* 定义为：对于数据集中的问题，骨干大语言模型未能给出正确答案时所产生的推理轨迹。第三，我们将*错误分类法* 定义为对失败轨迹中识别出的错误进行分类，并统计这些错误的出现频率。第四，我们将*指导* 定义为一组旨在防止错误分类法所描述的各类错误的指导文本。

接下来，我们介绍我们的方法 ETGPO。该方法包含四个步骤：(1) 错误收集、(2) 错误分类体系构建、(3) 错误类别选择，以及 (4) 指导生成。下面详细介绍每个步骤：

3.1. 错误收集

在第一步中，我们将当前数据集的验证集与基础系统提示结合起来，并针对这组问题向骨干大语言模型发起查询。随后，我们收集失败轨迹，以及相应的问题陈述、正确答案和预测答案。我们不会只对验证集进行一次遍历，而是运行 K 次，以捕捉由于大语言模型的随机性而导致同一问题可能出现的多种错误。

3.2. 错误分类体系构建

接下来，我们为失败轨迹中的错误创建一个分类法。对于每条失败轨迹，我们向优化器大语言模型提供问题陈述、正确答案、模型的推理轨迹和模型预测的答案，并提示其进行分析。我们要求优化器大语言模型找出推理首次出错的位置，判断错误的性质，并解释该错误为何导致了错误答案。根据这项分析，大语言模型会将该轨迹归入一个错误类别。

在实践中，失败轨迹的数量可能很大，存在超出优化器大语言模型上下文长度的风险。为解决这一问题，我们以批处理方式创建分类法。在第一次调用时，我们向优化器大语言模型提供一批 B 条失败轨迹，以及上述相关信息，并要求它们根据错误分析进行分类。作为回应，大语言模型会将每条轨迹分配到新创建的错误类别中，并为每个类别生成辅助信息，包括摘要、描述、示例、错误类型，以及错误为何会导致错误答案的解释。每个错误类别都必须是自包含的，无需参考原始问题即可理解，因此

算法1错误分类法引导的提示优化

输入： 验证集 V、基础提示词 P、骨干大语言模型、优化器大语言模型
参数： K（收集运行次数）、B（批次大小）、G（最大错误类别）
输出： 优化后的提示词 P^*

// 步骤 *1*：错误收集
$F \leftarrow \emptyset$
对于 $k = 1$ **至** K，**执行**
  对于 V 中的每个问题，**执行**
    使用 P 和问题查询骨干大语言模型
    如果预测不正确**那么**
      将（问题、轨迹、预测）添加到 F
    结束如果
  结束循环
结束循环

// 步骤 *2*：错误分类体系构建
$C \leftarrow \emptyset$
对于 F 中的每一批 B 推理轨迹**执行**
  使用优化器大语言模型识别错误类别
  用新的或已有的错误类别更新 C
结束循环

// 第 *3* 步：错误类别选择将 C 筛选为包含 ≥ 2 个问题的错误类别 按失败数量降序排列 $C\,S \leftarrow$ 个排名靠前的错误类别，取自 C

// 第*4*步：指导生成使用优化器大语言模型为 S、$P^* \leftarrow P$ + 前言和 + 指导文本生成指导**返回** P^*

以使所得指导能够泛化到优化期间遇到的特定示例之外。

在后续调用中，我们重复这一过程，并增加一项约束：只要可能，大语言模型就应重用现有错误类别。因此，在这些调用中，我们还会向大语言模型提供当前的错误类别集合，以及每个类别目前被分配的推理轨迹数量等元数据。此过程结束后，我们将得到一组构成错误分类体系的错误类别，以及这些类别的出现频率统计。此步骤的完整提示词见第 B 节。

3.3. 错误类别选择

构建错误分类法后，我们会选取部分错误类别来生成指导，帮助骨干大语言模型避免此类错误。由于多种原因，使用所有错误类别并不切实际：这会导致过于

原始系统提示

请逐步思考，然后完成任务。

ETGPO 生成的系统提示

请逐步思考，然后解决任务。为了最大限度地提高解决具有挑战性的数学问题时的准确率，请针对每种常见错误类型仔细遵循这些有针对性的指令。下面的每条指导都针对一种特定的失败模式，并提供可执行的步骤和示例，帮助你避免系统性错误。

1. 代数或几何计算错误*D*

escription: 在代数运算、几何计算或公式应用中的错误，即使已经列出了正确的方程或关系。这些错误可能包括符号错误、公式使用不当，或距离、交点计算错误。*A*

ctionable Advice:

- 仔细复核每一步代数运算和计算，尤其是多步推导。
- 明确写出每一步，并在继续之前核验中间结果。
- 对于几何问题，尽可能使用坐标或显式构造，并检查所有距离和关系是否准确。
- 得出结果后，将其代入原方程或几何设置中，以验证结果是否正确。

错误示例： 已知一个内接于圆的等边三角形和若干相切圆。你错误地计算了圆心或交点之间的距离，导致答案为 16，而正确答案是 378。正确示例： 你仔细地为所有相关点设定坐标，写出圆和直线的方程，并逐步检查所有计算。求出边长后，你将其代入几何关系中，确认其符合所有约束条件，最终得出正确答案 378。

2. 从模式或启发式方法进行概括错误*D*

escription: 基于有限的数据或直觉应用模式或启发式方法，却没有严格的依据。你可能根据少数几个计算结果进行外推，或依据与某个知名数列的表面相似性得出结论，从而导致错误的概括。*A*

ctionable Advice:

- 除非能够严格证明某种模式适用于所有相关情形，否则不要假定它成立。
- 不要仅凭少数几个计算结果就得出结论；应分析其内在结构或推导出通用公式。 • 如果怀疑存在某种模式，应尝试用归纳法证明，或通过分析问题的约束条件来验证。 • 务必检查是否存在可能打破该模式的例外或边界情形。*[. . . 为简洁起见，省略其他指导项。 ..]*

图*2.* 使用 GPT-4.1-mini 作为骨干模型、GPT-4.1 作为优化器模型时，原始系统提示与经 ETGPO 优化的提示词之间的比较。完整的优化后提示词包含 10 个区块，此处仅展示第一个区块和第二个区块的一部分。

提示词很长，而且许多类别很少见，或只出现在单个问题中。聚焦于最常见的错误可能会带来更大的影响。

基于这些原因，我们重点选择最重要的错误类别。对于每个错误类别，我们从上一步获取两类统计数据：问题数量，定义为出现该错误的不同问题数量；以及失败数量，定义为与该类别相关的失败轨迹数量。为避免针对个别问题过于具体的错误类别进行优化，我们会筛除仅与单个问题相关的类别。完成筛选后，我们按失败数量选出排名前 G 的错误类别，用于下一步的指导生成。

3.4. 指导生成

接下来，我们针对每个选定的错误类别生成可执行指导文本。我们向优化器大语言模型提供有关各类别的信息，包括其描述、一个示例，以及说明该类别为何会导致错误答案的解释，并要求它为每个类别生成指导文本。大语言模型需要生成可执行的建议，并附上具体的错误示例和正确示例。此外，大语言模型还会生成一段引出指导内容的前言。所有选定错误类别的指导文本及前言，均通过一次大语言模型调用生成。收到大语言模型的回复后，我们将前言和指导文本附加到原始系统提示中。此步骤的输出即为最终改进的系统提示词。本步骤中用于查询优化器大语言模型的完整提示词见第 C 节。图 2 展示了我们的方法生成的提示词示例。更多由 ETGPO 生成的提示词见第 D 节。

表 *1.* 使用基本思维链和三种自动提示优化方法所获得的结果（所有运行的平均准确率及置信区间）；其中，**GPT-4.1-mini** 作为骨干模型，GPT-4.1 作为优化器。ETGPO（我们的方法）的平均表现优于以往方法，且在除一个基准测试外的所有基准测试上均表现更佳。注：**粗体** = 表示最佳；下划线 = 表示次佳。

方法	数学推理		通用	多跳	逻辑推理		Avg.
	AIME	HMMT	MMLU-Pro	Musique	AR-LSAT	FOLIO	
思维链	47.08 ± 1.43	47.14 ± 1.23	78.82 ± 0.30	74.25 ± 0.63	74.35 ± 0.91	75.09 ± 0.67	66.12
MIPROv2	47.66 ± 1.22	45.89 ± 1.31	78.37 ± 0.42	75.20 ± 0.60	76.36 ± 1.11	79.71 ± 0.92	67.20
GEPA	**49.06 ± 1.51**	47.86 ± 1.38	**79.55 ± 0.58**	76.35 ± 0.51	75.60 ± 0.98	77.83 ± 0.90	67.71
ETGPO（我们的方法）	**49.06 ± 1.36**	**50.47 ± 1.28**	79.40 ± 0.66	**77.30 ± 0.45**	**76.93 ± 0.76**	**81.34 ± 0.60**	**69.08**

表 *2*。 三种自动提示优化方法的优化阶段令牌用量（输入与输出令牌总数，截取至千位），其中以 **GPT-4.1-mini** 作为骨干模型，以 GPT-4.1 作为优化器。 Δ 列表示各方法的平均令牌用量与 ETGPO 的平均令牌用量之比。ETGPO（我们的方法）在所有基准测试中始终使用最少的令牌。注：**粗体** = 表示最低；下划线 = 表示次低。

方法	AIME	HMMT	MMLU-Pro	Musique	AR-LSAT	FOLIO	Avg.	Δ
MIPROv2	14,332	11,629	3,859	23,735	6,954	2,275	10,464	4.6×
GEPA	12,495	10,249	2,623	6,093	5,144	2,320	6,487	2.8×
ETGPO（我们的方法）	**4,377**	**4,025**	**595**	**2,849**	**1,372**	**453**	**2,279**	—

我们尝试了两种指导风格：简短风格，为每个错误类别提供一到两句话；详细风格，则包含完整描述、可执行的建议，以及错误示例和正确示例。之后，我们通过消融研究比较这两种风格。

4. 评估

在本节中，我们将讨论本方法在各种数据集上的评估设置和结果。

4.1. 数据集

我们在涵盖多个领域的六个数据集上评估本方法：两个数学数据集（AIME、HMMT）、通用问答（MMLU-Pro）、多跳推理（MuSiQue），以及逻辑与分析推理（AR-LSAT、FOLIO）。下面将对这些数据集进行介绍：

AIME（Art of Problem Solving）：美国邀请数学考试（AIME）是一项难度较高的高中数学竞赛，旨在测试高阶数学技能。我们使用 AIME 的题目来评估大语言模型的数学推理能力。在实验中，我们使用 2022–2024 年各届的 90 道问题作为验证集，并使用 2025 年届的 30 道问题作为测试集。

HMMT（哈佛-麻省理工数学竞赛）：哈佛-麻省理工数学竞赛（HMMT）是另一项颇具挑战性的高中数学竞赛。在实验中，我们使用2023年、2024年和2025年2月届次的90道问题作为验证集，并使用30道2025年11月届次的问题作为测试集。我们使用 Math-Arena 公开发布的数据集（Balunovi´c 等人，2026年）。

MMLU-Pro（Wang 等人，2024年）：MMLU-Pro 是 MMLU 基准测试的一个更具挑战性的变体（Hendrycks 等人，2021），这是一个通用领域的问答数据集。与原始基准测试每道题提供四个答案选项不同，MMLU-Pro 每道题提供十个选项。由于原始验证集仅包含 70 道问题，我们改为将完整测试集划分为两个子集，以创建更大的验证集和测试集。随后，我们分别从验证子集中抽取 100 道问题，从测试子集中抽取 500 道问题，构成验证集和测试集。

MuSiQue（Trivedi 等，2022）：MuSiQue 是一个多跳推理数据集，其中的问题需要回答多个隐含的子问题，才能得出正确答案。该数据集经过专门构建，避免出现只需回答单个子问题即可解决的问题，从而确保确实需要进行多跳推理。由于数据集规模较大，我们从训练划分中随机抽取 160 个问题组成验证集，并从验证划分中抽取 500 个问题组成测试集。

AR-LSAT（Zhong 等，2021）：AR-LSAT 是一个分析推理数据集，源自法学院入学考试（LSAT）中的问题。我们从验证划分中随机抽取 100 个问题作为验证集，并将测试划分中的全部 230 个问题纳入测试集。

FOLIO (Han 等人，2024年）：FOLIO 是一个旨在测试复杂逻辑推理能力的数据集。每个实例

表 3. 使用基础思维链和三种自动提示优化方法获得的结果（所有运行的平均准确率及置信区间）；其中，**DeepSeek-V3.1** 用作骨干模型，GPT-4.1 用作优化器。ETGPO（我们的方法）的平均表现优于以往方法，并在六个基准测试中的三个上表现更佳。注：**粗体** = 表示最佳；下划线 = 表示次佳。

方法	数学推理		通用	多跳	逻辑推理		Avg.
	AIME	HMMT	MMLU-Pro	Musique	AR-LSAT	FOLIO	
思维链	65.52 ± 1.25	53.65 ± 1.28	82.74 ± 0.53	76.92 ± 0.39	90.65 ± 0.61	75.58 ± 0.88	74.18
MIPROv2	68.23 ± 1.25	**60.68 ± 1.51**	83.73 ± 0.54	76.80 ± 0.61	**91.66 ± 0.68**	82.33 ± 0.57	77.24
GEPA	64.48 ± 1.33	57.55 ± 1.30	**83.76 ± 0.66**	76.38 ± 0.55	90.92 ± 0.57	**82.45 ± 0.40**	75.92
ETGPO（我们的方法）	**69.74 ± 1.33**	58.65 ± 1.62	83.65 ± 0.52	**79.03 ± 0.72**	91.44 ± 0.52	**82.45 ± 0.93**	**77.49**

表 4. 三种自动提示优化方法的优化阶段令牌用量（输入与输出 Token 总量，以千为单位截断），其中以 **DeepSeek-V3.1** 为骨干模型，GPT-4.1 为优化器。 Δ 列表示各方法的平均令牌用量与 ETGPO 的比值。ETGPO（我们的方法）在所有基准测试中始终使用最少的令牌。注：**粗体** = 表示最低；下划线 = 表示次低。

方法	AIME	HMMT	MMLU-Pro	Musique	AR-LSAT	FOLIO	Avg.	Δ
MIPROv2	17,400	17,027	4,785	26,096	10,735	3,276	13,220	6.5×
GEPA	9,899	9,923	3,300	6,449	5,523	2,550	6,274	3.1×
ETGPO（我们的方法）	**3,098**	**3,295**	**778**	**3,063**	**1,378**	**507**	**2,020**	—

由若干前提和一个结论组成，任务是判断该结论是否能从这些前提中逻辑推导出来。在我们的实验中，我们从训练划分中随机抽取100个问题作为验证集，并将验证划分中的全部203个问题作为测试集。

4.2. 基线

我们将思维链（Wei 等人，2023年）作为主要基线，因为本文中的其他自动提示优化基线都是在其提示词基础上改进的。我们还纳入了两种最先进的自动提示优化方法进行比较：GEPA（Agrawal 等人，2025年）和 MIPRO（Opsahl-Ong 等人，2024年）。我们使用 DSPy（Khattab 等人，2023年）对 GEPA 和 MIPRO 的实现；在 DSPy 中，MIPRO 以 MIPROv2 的形式实现。对于 GEPA 和 MIPRO，我们均采用重度优化模式，该模式使用最多的评估次数来搜索最佳提示词。我们的目标是与这些方法现有的最强版本进行比较。

4.3. 实验结果

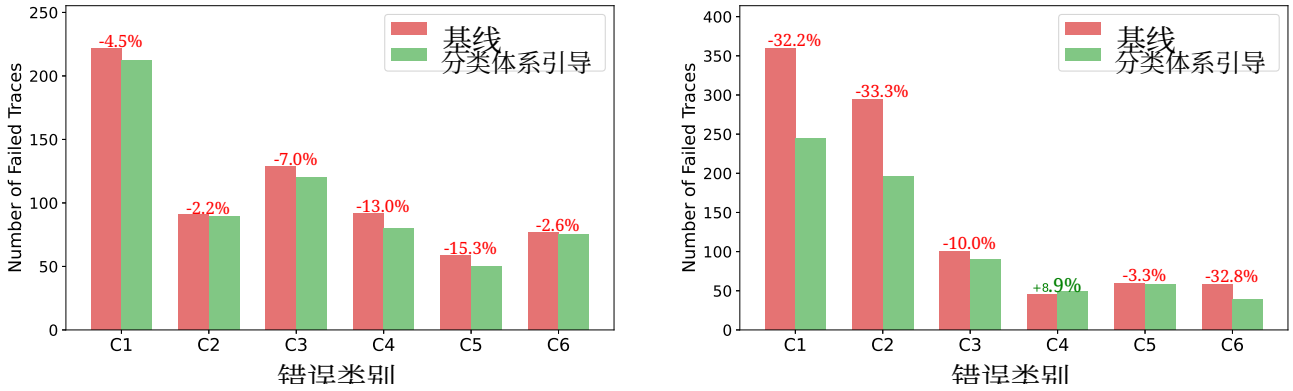
准确率结果 我们在所列数据集的测试集上评估我们的方法和基线。部分结果见表1 （骨干大语言模型：GPT-4.1-mini（OpenAI，2025b））和表3 （骨干大语言模型：DeepSeek-V3.1（DeepSeek，2025年））。所有报告的数值均对应于各方法生成的最终提示词在相应测试集上评估时取得的准确率。

对于这些表格中报告的所有自动提示优化方法，我们均使用 GPT-4.1（OpenAI，2025a）作为优化器大语言模型。为降低计算成本，我们对每种自动提示优化方法在每个数据集上只运行一次，但会对生成的提示词进行多次评估。对于较小的数学数据集，我们针对每个问题运行 64 次。对于较大的数据集，我们减少每个问题的运行次数：MuSiQue 和 MMLU-Pro 各运行 8 次，AR-LSAT 和 FOLIO 各运行 16 次。我们对每个提示词进行大量运行，以获得更窄的置信区间和更强的统计可靠性。遵循近期研究（Ho 等人，2025年），我们采用基于大语言模型的答案等价性检查：如果预测与真实答案完全匹配，则判定为正确；若未能完全匹配，则由强大的大语言模型（GPT-5（Singh 等人，2025年））判断两者是否等价。

对于我们的方法，我们在所有数据集上使用相同的超参数值，这些值是根据 AIME 验证集上进行的超参数搜索选定的。为降低计算成本，我们不针对每个数据集单独进行超参数搜索。具体而言，我们将 K（错误收集运行次数）设为 5，将 G（所选错误类别的数量）设为 10，并使用详细指导文本，而非简短版本。我们不对批次大小参数 B 进行搜索，而是采用默认值 6。针对每个数据集分别调优这些超参数，或许能进一步提升性能。

现在我们讨论表 1 和表3中的结果。我们的方法在这两个表中的平均性能均为最佳，始终达到或超过其他自动提示优化方法。在单个数据集上，使用 GPT-4.1-mini 作为骨干大语言模型时，我们的方法在 6 个数据集中有 5 个表现最佳；使用 DeepSeek-V3.1 时，则在 6 个数据集中有 3 个表现最佳。

就各个数据集上的绝对提升而言，



(a) *(b)*

图*3.* 在 (a) AIME 与 (b) FOLIO 的验证集上，错误分类法中前 6 个错误类别在使用 ETGPO 优化前后的失败轨迹数量。除一个类别外，其余类别的失败轨迹均有所减少，支持了我们的假设：针对性指导可降低错误发生率。

表*5.* 改变 K（即错误收集运行次数）对 AIME 与 FOLIO 验证集上最终提示词准确率的影响。准确率在 K的不同取值处趋于饱和，表明最优运行次数取决于数据集。

K	AIME	FOLIO
基线	52.35 ± 0.91	72.50 ± 0.08
1	52.61 ± 0.59	**79.70 ± 0.37**
5	**53.72 ± 0.28**	79.53 ± 0.53
10	53.43 ± 1.06	78.32 ± 0.38

两个骨干模型在 FOLIO 上的提升最大，准确率大约提高了 6–7 个百分点。AIME、HMMT 和 MuSiQue 也表现出强劲提升，在两个骨干模型上均稳定地大约提高了 2 到 5 个百分点。相比之下，MMLU-Pro 上的提升较为有限，我们认为这是因为这些问题往往需要更具体的领域知识，而仅靠通用指导不太可能带来改进；在这种情况下，基于检索的方法可能更为合适。当以 GPT-4.1-mini 作为骨干大语言模型时，AR-LSAT 显示出约 2.5 个百分点的显著提升。然而，DeepSeek-V3.1 在 AR-LSAT 上的表现似乎已趋于饱和，准确率接近 90%，进一步提升的空间有限。在这一区间内，执行日志几乎无法提供额外信号来提取有用的指导，因此仅带来边际提升。

Token 使用结果 在表 2和表4中，我们报告了所评估的每种 APO 方法（包括我们的方法）在优化阶段消耗的输入和输出 Token 总数。所有数值均以千为单位，并为简洁起见进行了截断。在所有情况下，报告的 Token 数量均为以下对象消耗的 Token 总和：骨干模型，以及在适用情况下优化器大语言模型。

在所有骨干大语言模型和数据集上，我们的方法在优化阶段始终使用最少的 Token。特别是，我们方法的 Token 总用量约为次高的 GEPA 的三分之一。这些结果表明，我们的方法在性能上可与最先进的自动提示优化方法相媲美，甚至更胜一筹，同时在优化过程中显著提高了 Token 效率。

4.4. 详细分析

接下来，我们介绍一组分析实验，旨在研究各种超参数的影响。所有实验均在 AIME 和 FOLIO 的验证集上进行，使用 GPT-4.1-mini 作为骨干模型，GPT-4.1 作为优化器模型。对于每种配置，我们让每种对比方法分别运行三次，每个生成的提示词进行 20 次评估运行。除非另有说明，其他所有超参数均固定为默认值。在这些分析中，我们还将基础思维链提示词作为基线。请注意，一项涉及 G的实验（该参数决定为指导生成选取的顶级错误类别数量）见第A节。

错误收集运行次数在第一个实验中，如表5所示，我们研究参数 K（控制错误收集运行次数）对最终提示词准确率的影响。参数 K 的影响因数据集而异。对于 AIME，随着 K 从 1 增至 5，准确率有所提升，但在 $K = 10$时趋于饱和。对于 FOLIO，在 $K = 1$时便已达到最佳性能，且准确率在更大的取值下趋于饱和。这表明，额外的执行轨迹带来的收益逐渐递减；这可能是因为对同一个问题进行多次运行，并不会产生差异显著的错误模式。

表 *6.* ETGPO 各组成部分的消融研究。第一次消融将分类体系构建替换为原始失败采样。第二次消融将大语言模型生成的指导替换为直接插入错误类别描述。第三次消融比较简短指导与包含更完整解释和示例的详细指导。总体而言，每次消融都会降低平均性能，这表明所有组成部分都有助于提升 ETGPO 的有效性。

方法	AIME	FOLIO	Avg.	Δ
ETGPO	53.72 ± 0.28	79.53 ± 0.53	66.63	–
消融1：原始采样	53.39 ± 0.84	77.13 ± 0.45	65.26	-1.37
消融实验 2：直接使用类别	52.52 ± 1.27	78.33 ± 0.96	65.43	-1.20
消融实验 3：简短指导	52.20 ± 0.96	76.62 ± 0.90	64.41	-2.22

错误减少在第二个实验中，如图3所示，我们考察生成的指导是否能有效减少不同错误类别所涵盖的错误类型。我们使用优化器大语言模型进行分类，分别在 AIME 和 FOLIO 的验证集上比较优化前（基线）和优化后归入排名前六的错误类别的失败轨迹数量。如条形图所示，优化后所有错误类别中的失败轨迹数量均持续减少，证明了我们的方法生成的提示词能够有效缓解这些错误。

4.5 消融实验

我们还开展了一系列消融实验，以评估我们方法中不同组件的贡献。结果见表 6。与分析实验类似，所有消融实验均在 AIME 和 FOLIO 的验证集上进行，使用 GPT-4.1-mini 作为骨干模型，并使用 GPT-4.1 作为优化器模型。对于每种消融设置，我们运行该方法三次，每个生成的提示词进行 20 次评估运行。

错误分类体系构建 第一项消融实验研究错误分类体系构建步骤的重要性。具体而言，它考察在失败轨迹中显式地对错误进行分类，是否比直接使用原始失败轨迹能带来额外价值。在该消融实验中，我们跳过分类法构建，改为将10条失败轨迹样本（每个问题一条）直接传入指导生成步骤。结果表明，错误分类体系构建平均而言是有益的：移除该步骤后，FOLIO上的性能下降约2个百分点。相比之下，AIME上的下降并不那么显著，这表明少量失败轨迹样本可能足以覆盖该数据集中最显著的错误模式。

指导生成 第二项消融实验评估了指导生成步骤本身所发挥的作用。该消融实验不生成指导文本，而是直接构建最终提示词：将选定的错误类别描述以项目符号列出，并在其前面加上通用引言

鼓励模型留意常见的错误模式。结果显示，在两个数据集上，性能都明显下降，这表明，除了错误类别描述本身之外，显式的指导生成还能带来额外价值。具体而言，指导生成步骤提供了可执行的建议，以及具体的正确做法和错误示例，帮助模型更好地理解每个错误类别适用的时机和方式。

详细指导与简短指导 第三项消融实验考察指导粒度的影响。我们测试了最终提示词的一个变体，其中每个错误类别仅用一到两行进行概括，而非更详细的描述。如结果所示，该消融在两个数据集上造成了最大的性能下降，平均降幅约为两个百分点。这一发现凸显了在最终提示词中为每个错误类别提供描述性指导的重要性。

5. 结论

在这项工作中，我们提出了**错误分类法引导的提示优化（ETGPO）**，这是一种由反馈驱动的自动提示优化方法，它构建错误分类法，以可解释的方式呈现数据集级错误分布，并仅针对最常见的错误类别生成指导。通过聚焦于高频错误，我们确保提示编辑能够泛化到验证集之外。如第4节所示，这使我们得以避免成本高昂的迭代验证，从而显著提高了方法的效率。尽管如此，ETGPO 仍与最先进的方法相比极具竞争力，并在数学推理、问答和逻辑推理领域的多个数据集上取得了更好的表现。

参考文献

Agarwal, E., Magazine, R., Singh, J., Dani, V., Ganu, T., 和 Nambi, A. PromptWizard：通过任务感知、反馈驱动的自我进化来优化提示词。载于 Che, W.、Nabende, J.、Shutova, E. 和 Pilehvar, M. T.（编），计算语言学协会论文集：*ACL 2025*

tics: ACL2025，第 19974–20003 页，奥地利维也纳，2025年7月。计算语言学协会。ISBN 979-8-89176-256-5。doi: 10.18653/v1/2025.findings-acl. 1025。网址 https://aclanthology.org/2025.findings-acl.1025/。

Agrawal, L. A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M. J., Jiang, M., Potts, C., Sen, K., Dimakis, A. G., Stoica, I., Klein, D., Zaharia, M., and Khattab, O. Gepa: 反思式提示演化可胜过强化学习，2025年。URL https://arxiv.org/abs/2507.19457。

Art of Problem Solving。AIME竞赛题目与解答。URL https://artofproblemsolving.com/wiki/index.php/AIME_竞赛题目_与_解答。访问日期：2025年1月27日。Balunovi´c, M., Dekoninck, J., Petrov, I.,Jovanovi´c, N., and Vechev, M. MathArena：评估大语言模型在未污染数学竞赛中的表现，2026年。URLhttps://arxiv.org/abs/2505.23281。DeepSeek。DeepSeek-V3.1 发布，2025年。URL https://api-docs.deepseek.com/news/news250821。访问日期：2025年1月27日。Fernando, C., Banarse, D. S., Michalewski, H., Osin- dero, S., and Rockt¨aschel, T. PromptBreeder：通过提示演化实现自指式自我改进。载于 Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F.（编），第*41*届国际机器学习会议论文集，第235卷，机器学习研究论文集，第13481–13544页。PMLR，2024年7月21–27日。URL https://proceedings.mlr.press/v235/fernando24a.html。Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. 将大语言模型与进化算法相结合，可产生强大的提示优化器。载于第十二届国际学习表征会议，2024年。URLhttps://openreview.net/forum?id=ZG3RaNIsO8。Han, S., Schoelkopf, H., Zhao, Y., Qi, Z., Riddell, M., Zhou, W., Coady, J., Peng, D., Qiao, Y., Benson, L., Sun, L., Wardle-Solano, A., Szabo, H., Zubova, E., Burtell, M., Fan, J., Liu, Y., Wong, B., Sailor, M., Ni, A., Nan, L., Kasai, J., Yu, T., Zhang, R., Fabbri, A. R., Kryscinski, W., Yavuz, S., Liu, Y., Lin, X. V., Joty, S., Zhou, Y., Xiong, C., Ying, R., Cohan, A., and Radev, D. FOLIO：使用一阶逻辑进行自然语言推理，2024年。URLhttps://arxiv.org/abs/2209.00840。

哈佛-麻省理工数学竞赛。HMMT 问题与结果档案。网址 https://www.hmmt.org/www/archive/problems。访问日期：2025年1月27日。Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. 和 Steinhardt, J. 大规模多任务语言理解测量，2021年。网址 https://arxiv.org/abs/2009.03300。Ho, X., Huang, J., Boudin, F. 和 Aizawa, A. 大语言模型即评审者：重新评估大语言模型在抽取式问答中的性能，2025年。网址 https://arxiv.org/abs/2504.11972。Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., San- thanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M. 和 Potts, C. DSPy：将声明式语言模型调用编译为自我改进流水线，2023年。网址 https://arxiv.org/abs/2310.03714。OpenAI。通过 API 推出 GPT-4.1，2025a年。网址 https://openai.com/index/gpt-4-1/。访-问日期：2025年1月27日。OpenAI。GPT-4.1 mini 模型 — OpenAI API，2025b年。网址 https://platform.openai.com/docs/models/gpt-4.1-mini。访问日期：2025年1月27日。Opsahl-Ong, K., Ryan, M. J., Purtell, J., Broman, D., Potts, C., Zaharia, M. 和 Khattab, O. 优化多阶段语言模型程序的指令和示例。载于 Al-Onaizan, Y., Bansal, M. 和 Chen, Y.-N.（编），*2024*年自然语言处理实证方法会议论文集，第 9340–9366 页，美国佛罗里达州迈阿密，2024年11月。计算语言学协会。doi: 10.18653/v1/2024. emnlp-main.525。网址 https://aclanthology.org/2024.emnlp-main.525/。Pryzant, R., Iter, D., Li, J., Lee, Y., Zhu, C. 和 Zeng, M. 使用“梯度下降”和束搜索进行自动提示优化。载于 Bouamor, H., Pino, J. 和 Bali, K.（编），*2023*年自然语言处理实证方法会议论文集，第 7957–7968 页，新加坡，2023年12月。计算语言学协会。doi: 10.18653/v1/2023. emnlp-main.494。网址 https://aclanthology.org/2023.emnlp-main.494/。Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A. 等。OpenAI GPT-5 系统卡，2025年。网址 https://arxiv.org/abs/2601.03267。

Trivedi, H., Balasubramanian, N., Khot, T. 和 Sabharwal, A. Musique：通过单跳问题组合构建多跳问题，2022年。网址https://arxiv.org/abs/2108.00573。

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., 和 Chen, W. Mmlu-pro：一个更稳健且更具挑战性的多任务语言理解基准测试，2024年。网址https://arxiv.org/abs/2406.01574。Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., 和 Zhou, D. 思维链提示词能够激发大语言模型的推理能力，2023年。网址https://arxiv.org/abs/2201.11903。Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., 和 Chen, X. 大语言模型作为优化器。载于 第十二届国际学习表征会议，2024年。网址https://openreview.net/forum?id=Bb4VGOWELI。Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Huang, Z., Guestrin, C., 和 Zou, J. Textgrad：通过文本实现自动“微分”，2024年。网址https://arxiv.org/abs/2406.07496。Zhong, W., Wang, S., Tang, D., Xu, Z., Guo, D., Wang, J., Yin, J., Zhou, M., 和 Duan, N. Ar-lsat：探究文本的分析推理能力，2021年。网址https://arxiv.org/abs/2104.06598。Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., 和 Ba, J. 大语言模型是达到人类水平的提示工程师。载于第十一届国际学习表征会议，2023年。网址https://openreview.net/forum?id=92gvk82DE-。

表 7。改变 G（为指导生成而选取的排名靠前错误类别数量）对 AIME 和 FOLIO 验证集最终提示准确率的影响。每个数据集的准确率在不同的 G处趋于饱和，这表明最优错误类别数量因任务而异。

G	AIME	FOLIO
基线	52.35 ± 0.91	72.50 ± 0.08
3	52.15 ± 0.50	78.35 ± 0.60
5	52.83 ± 0.12	**79.72 ± 0.21**
10	**53.72 ± 0.28**	79.53 ± 0.53

A. 错误类别数量实验

在本实验中，如表 7所示，我们分析参数 G对指导生成的影响。该参数决定选取多少个排名靠前的错误类别用于指导生成。对于 AIME，随着 G 增大，准确率稳步提升，反映出具有信息价值的错误类别超过十个。相比之下，FOLIO 的性能在 $G = 5$之后趋于饱和。这可能是因为 FOLIO 通常会产生约十个错误类别，而排名前五以外的类别对有效学习所提供的额外信号有限。

B. 错误分类体系创建提示

错误分类体系构建使用两个提示词，对失败轨迹中发现的错误进行分类。第一个提示词（见图 4和图 5）用于对第一批轨迹进行分类。第二个提示词（见图 6 和图 7）也包含迄今发现的错误类别，并提示大语言模型将错误归入现有类别；只有在遇到差异显著的错误时，才创建新类别。请注意，由于篇幅较长，两个提示词均拆分在多个图中展示。

C. 指导生成提示词

图 8 和图 9 展示了用于指导生成的提示词。还请注意，由于篇幅较长，该提示词被拆分在多个图中。

D. ETGPO 优化提示

我们在图 10、图 11、图 12、图 13和图14 中分别展示了 ETGPO 针对 HMMT、MMLU-Pro、Musique、AR-LSAT 和 FOLIO 数据集优化的提示词。

```
你是一位分析语言模型为何在{domain_description}上失败的专家。

## 失败 1
## 失败（问题 {problem_idx}，运行 {run_id}）

### 问题
{problem}

### 正确答案
{correct_answer}

### 模型推理
{reasoning}

### 模型答案
{predicted_answer}

---

## 失败 2
## 失败（问题 {problem_idx}，运行 {run_id}）
...

## 你的任务

分析每个失败，并找出每个错误的根本原因。请尽可能详细地描述
    可能。

对于每次失败，找出：
1. 推理过程中最早出现错误的地方
2. 具体出了什么问题（计算错误、错误方法、误解，
    等等）
3. 为什么这个错误导致了错误的最终答案

创建能够涵盖每种错误类型的问题类别。类别应通用
    得足以可能适用于其他推理轨迹，但又足够具体、有意义。

重要：每个类别必须是自包含的，并且能被这样的人理解：此人
    没有看过原始问题。

## 输出格式

返回一个JSON对象，其中包含：
“‘json
{
    "categories": [
        {
            "category_name": "对此类错误的简短描述性名称",
            "summary": "用一句话描述核心错误模式。",
            "description": "详细描述此类别中出现的问题。
    请尽可能具体。",
            "example": "一个具体且自成一体的示例。格式：’问题：[简单
    问题]。错误：[模型做错了什么]。正确做法：[应该如何处理]。’",
            "error_type": "错误类型（例如，计算错误，错误方法，
    概念性误解、步骤缺失、逻辑谬误、事实错误，
    推理不完整、误读问题）",
            "why_leads_to_wrong_answer": "解释此错误如何导致错误
    答案"
        }
    ],
```

图 *4.* 首批错误分类体系创建提示（第 1 部分，共 2 部分）

```
    "失败_分配": [
        {
            "失败       _id": 1    ,
            "问题_idx":  <问题_idx>,
            "run_id":  <运行 _id>,
            "类别_名称": "此失败所属类别的名称",
            "轨迹_详情": {
                "轨迹_具体_地点": "推理过程中错误发生的位置",

                "轨迹_具体_详情": "出错情况的具体详情"
            }
        }
    ]
}
```

图 *5.* 首批的错误分类法构建提示词（第 2 部分（共 2 部分））

```
你是一位专家，擅长分析语言模型在{domain_描述}方面失败的原因。

## 现有问题类别

### 类别：{category_name}
- 摘要：{summary}
- 描述：{description}
- 示例：{example}
- 错误类型：{error_type}
- 导致答案错误的原因：{why_leads_to_wrong_answer}
- 目前存在此问题的轨迹数：{trace_count}

### 类别：{category_name}
...

## 失败 1
## 失败（问题 {problem_idx},                    运行 {run_id})

### 问题
{problem}

### 正确答案
{correct_answer}

### 模型推理
{reasoning}

### 模型答案
{predicted_answer}

---

## 失败 2
## 失败（问题 {problem_idx},                    运行 {run_id})
...

## 你的任务

对于每次失败：
1. 判断错误是否符合某个现有类别
2. 或者，如果错误本质上有所不同，则创建一个新类别
```

图 *6.* 后续批次的错误分类体系创建提示（第 1 部分，共 2 部分）

```
## 输出格式

返回一个 JSON对象，其中包含：
```json
{
    "新_类别": [
        {
            "category_name": "新错误类型的简短描述性名称",
            "摘要": "用一句话描述核心错误模式。",
            "描述": "详细描述出错之处。",
            "示例": "一个具体示例。",
            "错误_type": "错误类型（例如，计算错误，错误方法，
    概念性误解、步骤缺失、逻辑谬误、事实错误，
    推理不完整、误读问题）",
            "为什么_会导致__错误_答案": "说明此错误为何会导致错误
    答案"
        }
    ],
    "失败_分配": [
        {
            "失败_id": 1,
            "问题_idx": <问题_idx>,
            "run_id": <运行 _id>,
            "is_new_类别": 错误，
            "类别_名称": "现有或新类别的名称",
            "轨迹_details": {
                "推理_中特定_地点": "推理过程中错误发生的位置",

                "推理_中特定_细节": "出错的具体细节"
            }
        }
    ]
}
```

注意："新_类别"应仅包含尚不存在的类别。
```

图 7. 用于后续批次的错误分类体系创建提示（第 2 部分（共 2 部分））

```
你是提升语言模型在{domain_描述}方面性能的专家。

我已根据模型失败情况确定了以下错误类别。请生成
    帮助避免这些错误的指导。

## 类别1：{category_name}

**统计数据：** {failure_count} 次失败（{coverage_pct}%），{problem_count} 个问题

**摘要：** {summary}

**描述：** {description}

**示例：** {example}

**错误类型：** {error_type}

**导致答案错误的原因：** {why_leads_to_wrong_answer}

---
```

图 8. 指导生成提示（第 1 部分，共 2 部分）

类别 2：{category_名称}
...

你的任务

生成指导文本，要求：
1. 针对每种失败类别，提供具体、可执行的建议
2. 以向模型发出指令的方式撰写
3. 在有帮助时使用具体示例
4. 按出现频率排序

提供包含示例的详细指导。每项应包括：
- 错误模式的描述
- 关于如何避免该错误的可执行的建议
- 展示该错误的错误示例
- 展示正确方法的正确示例

关键约束

- 目标是提高准确率，而非谨慎。绝不要生成鼓励模型
  在能够合理给出答案时拒绝回答、弃答或声称“未作说明”的指导

- 正确示例必须始终展示模型给出实质性回答。绝不要
  将弃答/拒绝作为正确行为。

输出格式

返回一个包含以下内容的 JSON对象：

```json
{
  "指导_items": [
    {
      "类别_名称": "类别的名称",
      "guidance_text": "此类别的完整指导文本"
    }
  ],
  "preamble": "1-2 句介绍",
  "full_prompt": "以基础指令开头的完整增强提示词"
}
```

"full_prompt" 应以以下内容开头：
"请逐步思考，然后解决任务。"

然后添加前言和指导项。

图 *9.* 指导生成提示（第 2 部分（共 2 部分））

请逐步思考，然后完成任务。

为了最大限度地提高解决数学问题时的准确率和可靠性，请认真遵循以下详细指令，这些指令针对常见错误模式。每条准则都针对一种特定的失效模式，并提供可执行的步骤和示例，帮助你避免犯错。

1. 多变量系统中的代数运算错误：
描述：在处理或求解含有多个变量的方程组时会出现错误，通常是由于误用代数恒等式、错误的代入或算术失误所致。这会导致关系错误，并产生无效的解。

可执行的建议：
- 仔细展开、化简和处理方程，尤其是在处理变量的和、积或幂。
- 明确写出代数恒等式（例如，(x + y)^2 = x^2 + 2xy + y^2）和检查每一步是否正确。
- 代入时，确保所有变量和项都已正确替换并化简。
- 解出后，将解代回原方程以验证正确性。

错误示例：
已知 x + y = 5 和 x^2 + y^2 = 13，写成 (x + y)^2 = x^2 + y^2 = 25，遗漏了交叉项。

正确示例：
(x + y)^2 = x^2 + 2xy + y^2 = 25，因此 13 + 2xy = 25 => 2xy = 12 => xy = 6。现在求解 x 和 y，其中 x + y = 5 且 xy = 6。

2. 未能识别结构约束：
描述：当忽略或错误应用问题的关键约束条件（例如互异性、可除性、极大性/极小性）时，便会出现此类错误，进而导致无效或不完整的解。

可执行的建议：
- 仔细识别并列出问题陈述中所有明确和隐含的约束条件

- 对每个候选解，在接受之前验证其是否满足所有约束条件接受它。
- 搜索极大值/极小值时，检查所有可能性，直到指定的界限。
- 对于涉及素性等性质的问题，互异性或可除性，请检查解中每个元素的每一项性质。

错误示例：
找出最大的整数 n <= 100 ，使得 n 与 n+1 均为素数。列出 97 和将 98 作为一对，却未检查 98 并非素数。

正确示例：
逐一检查每个 n <= 100，验证 n 与 n+1均为素数。满足条件的最大 n 为 2（因为 2 与 3 均为素数）。

3. 未能完成枚举或计算最终数值答案：
描述：模型建立了正确的公式或求和，但未能计算明确给出最终数值答案，或代入了错误的数值。

...

图 *10.* 使用 GPT-4.1-mini（骨干模型）和 GPT-4.1（优化器）针对 HMMT数据集优化的提示词。内容已截断。

```
请逐步思考，然后解决任务。

为使多项选择推理题的准确率达到最高，请仔细遵循以下
    针对每种常见错误模式的详细指导。这些指令将
    帮助你避免常见陷阱，并根据上下文选出最佳答案。

1. 对最合理或预期答案的误判
描述：当你选择的答案在事实上得到支持时，就会出现此错误
    例如，它得到文章或数据的支持，但你没有意识到问题要求的是
    在特定语境中最合理、最符合预期或最具代表性的答案（例如
    群体层面、文化层面或比较合理性方面）。你可能会默认选择
    得到最直接支持或最熟悉的事实，而不是考虑哪个答案
    最符合问题的比较或语境框架。

可执行的建议：
- 始终确认问题是否要求给出“最合理的”，“最可能的”，“预期的”或“最佳”答案，并注意具体语境（例如人口、文化或比较）。
- 结合上下文比较所有答案选项，并选出最符合语境的选项
    重点考虑比较性或预期性的表达，而不只是选择有事实依据的选项。
- 问问自己：“在这种语境下，哪个答案最常见或最符合预期，即使
    其他答案也有依据？”

错误示例：
文章：讨论在芬兰使用桑拿的健康益处。
问题：在芬兰，哪种结果最有可能？
模型选择：“经常使用桑拿者的心血管相关死亡率较低者”（事实有依据，但并非最符合语境的答案）。

正确示例：
模型选择：“一位深受社区敬仰的86岁男性市长”（更
    考虑到上下文，这种情况在芬兰作为群体层面或文化层面的结果是合理的）。


2. 多步定量推理中的程序性错误
描述：当你在多步定量问题的一个或多个步骤中出错时，就会发生这种错误，
    例如错误地套用公式、未考虑
    约束条件（如材料宽度或损耗），或误解应该使用哪些
    尺寸或单位。任何一步出错都可能产生连锁影响，导致
    最终答案不正确。

可执行的建议：
- 将问题分解为清晰、有序的步骤，并明确写出每一项计算。
- 仔细检查要求和约束条件（例如，卷材宽度、向上取整至
    整数单位、损耗）。
- 每完成一步后，在继续之前验证结果是否合理。
- 仔细核对单位换算，并确保使用正确的尺寸/
    每一步的公式。

错误示例：
问题：计算使用固定宽度卷材为房间铺地毯的成本。
模型计算：房间的总面积，再乘以单位面积成本，忽略
    卷材宽度和所需长条。

正确示例：
模型计算：根据卷材宽度和房间尺寸确定所需长条数，
    然后计算所需地毯的总面积，再乘以单位面积成本，同时考虑
    所有约束条件。

...
```

图 *11.* 使用 GPT-4.1-mini（骨干模型）和 GPT-4.1（优化器）为 MMLU-Pro 数据集优化的提示词。已截断。

请逐步思考，然后解决该任务。

为了最大化多跳推理问题的准确率，请仔细遵循这些详细的
针对每种常见错误类型的指令。应用这些策略将有助于
确保你的答案精确，得到充分支持，并且在语境上恰当。

1. 未能识别最具体或最相关的答案
描述：当你给出通用或部分正确的答案，而
不是问题所要求的最具体或直接相关的答案时，就会出现这种错误。
你可能会止步于更高层级类别或更通用的答案，从而遗漏
更精确或更符合语境的回答。

可执行的建议：
- 始终检查问题是否要求以特定形式作答（例如符号、
date, name, 地点）。
- 如果答案有多个具体程度可选，请选择最精确且
能直接回答问题的那个。
- 重新阅读问题，找出“符号”、“精确”等关键’全名’，或’
特定的’。

错误示例：
问题：’原子序数为 8 的元素的化学符号是什么？’
答案：’氧。’

正确示例：
答案：’O.’

错误示例：
问题：’剑桥市位于哪个区？’
答案：’剑桥郡。’（这是郡，而不是区。）

正确示例：
答案：“剑桥区。”

2. 未能正确串联多跳推理步骤
描述：当你跳过必要的推理步骤、建立错误的联系，或未能在多跳
问题中保持正确的逻辑链时，就会发生此类错误。
这通常会导致答案基于部分
信息看似合理，但并未得到所需推理链的充分支持。

可执行的建议：
- 将问题拆解为各个推理步骤，并按顺序逐步解决。

- 在继续下一
步骤。
- 不要在得到第一个看似合理的答案后就停止；确保完成所有步骤。

错误示例：
问题：‘2010年，首都是巴黎的国家的总统是谁？’
答案：‘埃马纽埃尔·马克龙。’（现任总统，不是2010年的总统。）

正确示例：
步骤 1：巴黎是法国的首都。
步骤 2：2010年法国的总统是谁？尼古拉·萨科齐。
答案：‘尼古拉·萨科齐。’

...

图 *12.* 使用 GPT-4.1-mini（骨干模型）和 GPT-4.1（优化器）为 Musique数据集优化的 ETGPO 提示词。内容已截断。

请逐步思考，然后完成任务。

为最大限度地提高逻辑推理准确性，请仔细遵循以下详细指令
以避免常见错误模式。每条准则都针对一种特定的推理
陷阱，并提供具体步骤和示例，以确保正确答案
的选择。

1. 未能系统地枚举各种可能性：
- 在推导之前，务必系统地列出或考虑所有
符合给定约束条件的可能排列或情形。
- 使用表格、列表或图表来追踪各种可能性，并且只有在
彻底检查之后才排除选项。
- 错误做法：只检查少数几种排列，遗漏了有效选项。
- 正确做法：列出所有可能的分配，并逐一检查是否符合约束条件。

2. 约束条件的忽视或误用：
- 开始推理之前，仔细识别并重述每一项约束条件。
- 每一步都要检查，确保你考虑的每种配置都遵守所有约束条件
。
- 错误做法：允许违反已声明约束条件的排列。
- 正确做法：明确检查每种排列是否满足所有约束条件。

3. 必要结论的过度推导：
- 对于“必然为真”类问题，只选择在每一种可能的
有效配置中都为真的选项，而非仅在某些配置中为真。
- 逐一检验每个答案选项，考虑它在该条件下是否可能为假
给定约束条件。
- 错误做法：要求只选一个答案时，却选择了多个“必然为真”的答案。
- 正确做法：只选择在所有情况下都必然为真的答案。

4. “必然为真”的误用 vs. “可能为真”：
- 对于“必然为真”对于问题，验证该陈述在每一种可能的
约束条件所允许的配置中都成立。
- 对于「可能为真」问题，检查是否至少存在一种有效配置
使该陈述成立。
- 错误：仅仅因为某些情况下成立，就断言某事「必然为真」。
- 正确做法：检查所有情形，区分必然性和可能性。

5. 对“不可能”的误解或“例外”题：
- 对于“不可能为真”题，寻找在
约束条件。
- 对于“例外”题找出一个不符合给定模式的选项。

- 用自己的话重述问题，明确题目要求。
- 错误做法：题目要求选择不可能的选项时，却选了一个可能的选项。
- 正确做法：逐一检查每个答案是否不可能，并据此选择。

6. 未能识别互斥性或唯一性：
- 对于分配，确保每个实体只占据一个位置，并且每个
位置只由一个实体占据，除非另有说明。
- 在推理过程中明确检查是否存在重叠或重复分配。
- 错误：将两个实体分配到同一个唯一位置。
- 正确做法：确保每项分配都是唯一且互不重叠的。

图 *13.* 使用 GPT-4.1-mini（骨干模型）和 GPT-4.1（优化器）为 AR-LSAT 数据集优化的提示词。

请逐步思考，然后解决任务。

为提高逻辑推理准确性，请仔细区分哪些内容是前提所蕴含的，
哪些内容与前提矛盾，以及哪些内容尚不确定。请参考以下详细
指导，以避免常见的推理错误。

1. 逻辑推理中混淆“错误”与“不确定”
描述：当您将一个并非由
前提所蕴含的结论视为绝对错误，而不是认识到它仅仅是
在所提供信息下不确定的。如果前提并不能保证
结论的真或假，正确答案就是“不确定”。

可执行的建议：
- 不要仅仅因为一个结论不可能被证明为真，就假定它是错误的，依据来自
前提。
- 只有当结论与前提直接矛盾时，才回答“错误”。
- 如果前提与结论及其否定都相容，则回答“
不确定。”

错误示例：
前提：有些猫是黑色的。
结论：所有猫都是黑色的。
模型的（错误）答案：错误（因为这些前提无法证明该结论）

正确示例：
前提：有些猫是黑色的。
结论：所有猫都是黑色的。
正确推理：这些前提既不能保证结论成立，也不能排除结论成立。答案
是“不确定”。

2. 将直接矛盾误判为不确定
描述：当你未能识别出结论直接
与前提矛盾，却将其标记为“不确定”时，就会出现这种错误。如果结论
被前提明确排除，就必须将其标记为“错误”。

可执行的建议：
- 始终检查结论是否是任一前提的直接否定。
- 如果结论与前提不相容，请回答“错误”。
- 存在矛盾时，切勿回答“不确定”。

错误示例：
前提：所有狗都会叫。
结论：有些狗不叫。
模型的（错误）答案：不确定

正确示例：
前提：所有狗都会叫。
结论：有些狗不叫。
正确推理：结论与前提直接矛盾。答案是 ’
错误。”

3. 不当排除逻辑可能性
描述：当你错误地排除可能的情况，将
仅仅因为结论无法从前提中推出，就将其视为错误时，便会出现这种错误。如果
前提既没有排除某个结论，也没有推出该结论，那么答案
应为“不确定。”

可执行的建议：
- 不要将缺乏证据视为证伪的证据。

. . .

图 *14.* 使用 GPT-4.1-mini（骨干模型）和 GPT-4.1（优化器）为 FOLIO 数据集优化的 ETGPO 提示词。已截断。