Generative Verifiers: Reward Modeling as Next-Token Prediction

Abstract

验证器或奖励模型通常用于增强大型语言模型（ LLMs ）的推理性能

- 一种常见的方法是 Best-of-N 方法　其中由验证模型对LLM生成的 N 个候选解决方案进行排序，然后选择最好的一个
- 虽然基于LLM的验证器通常被训练为判别式验证器来对解决方案进行评分　它们没有利用预训练的LLMs的文本生成功能
- 为了克服这一限制　本文使用常用的下一个token预测来训练验证器
  - 联合验证和答案生成
  - 与传统验证器相比，此类生成验证器（ GenRM ）可以受益于LLMs的多个优势
    - 它们与指令微调无缝集成
    - 支持思维链推理
    - 可以通过多数投票利用额外的推理时间计算来实现更好的评分
- 效果
  - Best-of-N 求解算法和小学数学推理任务问题的GenRM作为验证器性能比判别式验证器性能提高16–64%
  - GenRM在数据集大小、模型容量和推理时间计算方面具有良好的扩展性

背景 / 问题 / 方法

1Introduction

- 虽然大型语言模型（ LLMs ）表现出卓越的能力，但它们经常自信地犯逻辑和事实错误
- 解决这个问题的常见策略是 Best-of-N
  - LLM为给定问题生成 N 个候选解决方案
  - 以及一个学习奖励模型，称为 “ verifier”
  - 对这些解决方案进行排名并选择最合适的一个
  - 策略的有效性取决于验证器的准确性
- 传统推理领域
  - 基于LLM的验证模型通常被训练为判别式奖励模型 (RM)
  - 为候选解决方案分配分数数值
  - 用于将它们分类为正确或错误
- 问题
  - 这种评分方法并没有利用LLMs根本设计目的文本生成功能
  - 没有利用生成式LLMs的固有优势
    - instruction tuning
    - 思维链推理
    - 利用额外的推理阶段的计算来提高性能
- LLM-as-a-Judge方法
  - 通过提示现有的生成式LLMs
  - 也提供了上述优势
  - 它通常不如经过基于LLMs训练的验证模型
- 本文工作
  - 建议使用下一个token预测来训练验证模型
    - 称之为GenRM
    - 利用LLMs的文本生成功能

> **Problem**: Tim decides to cancel his cable subscription and get streaming services. He gets Netflix for $10 a month. Hulu and Disney Plus normally cost $10 a month each but he saves 20% for bundling. How much money does he save by cancelling his $60 cable package?
>
> **Solution**: Tim pays $60 for cable. He gets Netflix for 10 and the bundle of Hulu and Disney Plus costs $10 * 80% = $8. So he pays $10 + $8 = $18 for Netflix and the bundle. Now he saves $60 - $18 = $42. The answer is 42.
>
> **Discriminative RM** correctness score: 0.999
>
> **GenRM-CoT**: Let's verify step by step.
>
> Step 3. **The bundle of Hulu and Disney Plus costs $10 * 80% = $8.** (Incorrect)
> - Expected Answer: The bundle of Hulu and Disney Plus costs $10 + $10 = $20. With the 20% discount, the total cost is $20 * 0.8 = $16. · · ·
> Verification: Is the answer correct (Yes/No)? No
>
> **GenRM-CoT** (Majority Voting) score: 0.0015

图 2：在 GSM8K 测试中使用生成验证器的示例

  - 在这里，解决方案是不正确的，因为它忽略了问题中的 “each” 一词
  - 示例
    - 判别式 RM 无法识别解决方案中的这种微妙错误
    - GenRM -CoT 验证器可以可靠地检测到该错误
    - 因为GenRM -CoT 接受了基于合成思维链原理的下一个标记预测的训练，使其能够明确地推理解决方案
  - 为了生成解决方案的分数值
    - 验证模型现在使用提示词　'Is the answer correct?
    - 并将分数值表示为单个文本token预测的概率（例如 “是” 或 “否” ）
  - GenRM天然支持 CoT 推理
    - 假设在训练期间的数据有思维过程可用
    - 它可以在使用 “是” 或 “否” 标记预测正确性之前生成思维过程来训练它的推理性能

图 3：生成验证器的图示

    - 给定一个问题和一个候选答案
    - GenRM 通过 SFT 直接微调 LLM来回答问题 “答案是否正确（是/否）？ ”
    - 示例
      - 推理过程中，通过提取 ‘Yes’ 标记的概率来获得验证模型分数
      - GenRM-CoT 微调 LLM，以在产生最终的 Yes/No token之前生成验证链 （CoT） 基本原理
      - 推理时，我们对多个 CoT 思维过程进行抽样，并使用多数投票来计算 “是” 的平均概率，使 GenRM-CoT 即使利用额外的推理计算来更好地验证

图 1：在 Best-of-N 性能方面，将GenRM与标准验证方法在多个推理任务上进行比较

- 效果
  - 通过将奖励模型重新设计为下一个token预测
  - GenRM在提高 Best-of-N 性能方面优于其他方法
  - 水平虚线对应于直接生成性能（Best-of-1）
  - GenRM利用LLMs的生成功能，使经过微调的验证器模型能够在推理时有效地利用思维链和多数投票来识别推理错误

2 Preliminaries

- Next-token预测　预训练和微调LMs的典型方法
  - 监督微调（ SFT ）目的是减少模型预测的下一个token与给定目标token之间的交叉熵损失
  - 给定一个数据集　$\mathcal{D} = \{(x, y)\}$
  - 输入上下文的x 和目标响应y

$$\mathcal{L}_{\mathrm{SFT}}(\theta, \mathcal{D}) = -\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\left[\sum_{i=1}^{|\mathbf{y}|}\log p_\theta(Y_i \mid \mathbf{x}, \mathbf{y}_{<i})\right].$$

- 判别式验证器
  - 正确答案和错误答案的数据集上微调LM作为分类器
  - 验证模型直接分配一个0-1分的数估计解决方案y对于某个问题x来说是正确的的概率
  - 判别式验证模型不利用LLMs的文本生成功

3 GenRM

- 概述
  - 基于LLM的判别性验证器不利用预训练的LLMs的文本生成功能
  - 建议使用标准的next-token预测来训练验证器同时这个验证器可以生成文本
  - GenRM使用LLM在各个token上的概率分布来表示答案的正确性，而不是单独预测一个数值
  - 这使GenRM的生成能力保持完整
    - 因为验证决策一个token
    - 同时还实现了LLMs “免费” 的多项优势
      - 答案生成和验证
      - 思维链推理
- 3.1直接验证器（ GenRM）　最简单的形式中
  - GenRM使用单个 “是” 或 “否” 标记来预测解决方案是否正确　图3 顶部
  - 数据集　$\mathcal{D}_{\text{Direct}} = \{(\mathbf{x}, \mathbf{y}^*, \mathbf{I}), \text{'Yes'}) \cup (\mathbf{x}, \mathbf{y}^-, \mathbf{I}), \text{'No'})\}$，　$\mathbf{I}$ = 'Is the answer correct (Yes/No)?'
  - 推理时，我们使用 “是” 标记的概率作为验证器对答案重新排名的分数　$r_{\text{Direct}}(\mathbf{x}, \mathbf{y}) = p_\theta(\text{Yes} \mid \mathbf{x}, \mathbf{y}, \mathbf{I})$.
- 3.2统一生成和验证　GenRM将验证答案正确或错误的建模与生成正确答案的有监督微调SFT 无缝集成
  - 可以通过简单地更改包括验证和生成任务的数据混合比例以更改 SFT 损失
  - 给定一个验证数据集　$\mathcal{D}_{\text{verify}}$　$\mathcal{D}_{\text{Direct}}$ / $\mathcal{D}_{\text{CoT}}$
  - 生成数据集　带有正确性标记的问题-答案对（可选地带有 CoT 原理）　$\mathcal{D}_{\text{correct}}$

$$\mathcal{L}_{\text{GenRM}}(\theta, \mathcal{D}_{\text{verify}}) = \mathcal{L}_{\text{SFT}}(\theta, \mathcal{D}_{\text{verify}}) + \lambda\mathcal{L}_{\text{SFT}}(\theta, \mathcal{D}_{\text{correct}}),$$

  - 其中 λ>0 是一个超参数　控制验证与生成任务的比例
  - 统一的训练可以提高验证器和生成器的性能　两任务具有一定的相关性
- 3.3思维链验证器（ GenRM -CoT）
  - 由于验证通常涉及复杂的推理　生成式验证器　可以从 CoT 推理中受益
  - 具体来说
    - 对解决方案的正确性做出决定之前生成中间推理步骤
    - 这可能会识别直接验证器遗漏的细微推理错误　图3 底部
  - 训练 CoT 验证器
    - 数据集 $\mathcal{D}_{\text{CoT}}$上最小化 SFT损失　$\mathcal{L}_{\text{GenRM}}$
    - 输入
      - problem-solution对
      - 提示词　$\mathbf{I}_{\text{CoT}}$ = 'Let's verify step by step.'
    - targets
      - 理由（思维过程）　$\mathbf{v}_{CoT}$　可以是人类生成的，也可以是LM生成的　探索了这两种情况
      - 最终的问题　$\mathbf{I}$ = 'Is the answer correct (Yes/No)?'
      - 最终答案　'Yes' or 'No' token

$$\mathcal{D}_{\text{CoT}} = \{(\mathbf{x}, \mathbf{y}^*, \mathbf{I}_{\text{CoT}}), (\mathbf{v}_{\text{CoT}}, \mathbf{I}, \text{'Yes'})\} \cup \{(\mathbf{x}, \mathbf{y}^-, \mathbf{I}_{\text{CoT}}), (\mathbf{v}_{\text{CoT}}, \mathbf{I}, \text{'No'})\}, \quad \mathbf{I}_{\text{CoT}} = \text{'Let's verify step by step.'}$$

  - 推理过程中
    - 从GenRM -CoT生成一个 CoT 基本原理　然后使用 “yes” 的概率来分配正确性分数
    - 使用 “是” 的概率来分配正确性分数：

$$r_{\text{CoT}}(\mathbf{x}, \mathbf{y}) = p_\theta(\text{Yes} \mid \mathbf{x}, \mathbf{y}, \mathbf{I}_{\text{CoT}}, \mathbf{v}_{\text{CoT}}, \mathbf{I}), \quad \text{where} \quad \mathbf{v}_{\text{CoT}} \sim p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}, \mathbf{I}_{\text{CoT}}),$$

    - 与直接使用相比　上述 CoT 奖励还取决于
      - $\mathbf{I}_{\text{CoT}}$ = 'Let's verify step by step.'
      - 自生成的 $\mathbf{v}_{CoT}$
  - CoT 验证器的推理时计算　当对验证器 CoT 进行采样时
    - 生成式验证器可能会使用不同的推理路径
    - 并对同一problem-solution对产生不同的正确性概率
    - 我们希望边缘化中间推理路径，以选择最一致的正确性答案
    - 使用称为个理由进行多数投票　并将这些理由的 CoT 验证器分数平均化

$$r_{\text{MajVote@k}}(\mathbf{x}, \mathbf{y}) = \frac{1}{K}\sum_{i=1}^{K} p_\theta(\text{Yes} \mid \mathbf{x}, \mathbf{y}, \mathbf{I}_{\text{CoT}}, \mathbf{v}_{\text{CoT}}^{(i)}, \mathbf{I}), \quad \text{where} \quad \mathbf{v}_{\text{CoT}}^{(i)} \sim p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}, \mathbf{I}_{\text{CoT}})$$

    - CoT 验证器的单个验证理由可能存在推理错误　进行平均来减轻此类错误的影响
    - GenRM -CoT 可以利用额外的推理时计算来提高其准确性　这是判别性验证器无法做到的
    - 默认根据 32 票的多数投票报告 GenRM-CoT 性能
  - CoT Verifier 的综合验证理由生成
    - 探索了在 GSM8K 上使用模型生成的原理
    - 如果简单地使用给定问题-解决方案对和 ‘Let's verify step by step’ 提示
    - 并根据它们是否正确验证答案的正确性来过滤生成的理由　准确率为 50%
    - 生成的理由质量很差
    - 除了要验证的问题和答案之外，我们还提供了一个参考提示词　使 LLM指出所提供的答案中的任何推理错误

> **Problem**: Carol spends 4 hours writing a song, half that much time recording it, and 90 minutes editing it. What percentage of her total work time did she spend editing?
>
> **Solution**: Half of 4 hours is 2 hours. So the time spent recording is 2 hours. Add the editing time of 90 minutes and the total time for the entire project is 4 + 2 + 1.5 = 7.5 hours. That means she spent (90 / 7.5) * 100 = 12% of the total time editing. The answer is 12.
>
> **Discriminative RM** correctness score: 0.8414
>
> **GenRM-CoT**: Let's verify step by step.
> · · ·
> Step 3: That means she spent (90 / 7.5) * 100 = 12% of the total time editing.
> Expected Answer: Incorrect. The editing time should be in hours, so it should be (1.5 / 7.5) * 100 = 20%.
>
> Verification: Is the answer correct (Yes/No)? No
>
> **GenRM-CoT (Majority Voting)** score: 0.2498

图 4：GenRM-CoT 捕捉到判别式验证器无法捕捉到的细微错误的示例

4. Experiments

- 候选解决方案没有将 90 分钟转换为 1.5 小时，然后除以 7.5
- 判别性验证器无法检测到这个错误
- 我们提出的 GenRM-CoT 模型能够使用分步生成验证来识别这个错误

参考资料