Reflexion: 使用口头反思强化学习的语言智能体

秦海明 2453103002

2025 年 1 月

摘要

大语言模型在撰写和推理任务中展现出显著潜力，但其常常产生幻觉，自信地给出给出错误的答案。在基于语言模型的智能体中，一次错误的推理可能会导致灾难性的后果。Reflexion 是一种基于语言反思的智能体增强方法，其灵感来自于人类学习过程中通过反思总结经验。该框架使智能体能够从错误中学习并持续改进。本文复现了 Reflexion 框架，实现了多种反思策略。在 HotpotQA 数据集上的实验表明，反思机制能显著提升模型性能：与基础 CoT 和 ReAct 智能体框架相比，引入反思后准确率分别提升了 28.6% 和 44.8%，结合 Reflexion 和错误记忆的 CoT 实现提升了 21% 的准确率。此外，错误记忆与反思机制的结合能有效加速学习过程。本文的工作不仅验证了原论文的核心结论，还通过改进的工程实现和详细的实验分析。

关键词： 反思；大模型；智能体

1 引言

近年来，大语言模型在自然语言处理领域取得了突破性进展。特别是在问答和推理任务中，基于 Chain-of-Thought (CoT) [11] 和 ReAct [14] 等提示方法的语言智能体展现出强大的潜力。然而，这些方法仍存在一个关键局限：智能体缺乏从错误中学习和自我改进的能力。在实际应用中，当智能体遇到复杂问题或做出错误决策时，往往无法通过反思来调整其策略，这显著限制了其在实际场景中的应用效果。

Reflexion 框架 [10] 通过引入语言反思机制来解决这一问题。该框架允许智能体在执行任务后进行自我反思，总结经验教训，并在后续尝试中应用这些见解。这种方法的独特之处在于：它不需要更新模型参数，而是通过在提示中注入自然语言反思内容来增强模型的决策能力。这种设计既保持了方法的简洁性，又显著提升了模型的性能，如图 1所示。

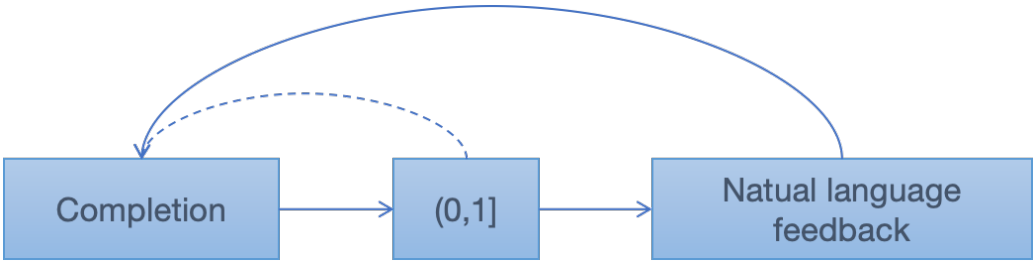

图 1. 自然语言反馈示意图

为了系统验证该框架的有效性，我们选择了 HotpotQA [13] 数据集作为测试基准。
本文的主要贡献包括：

1. 基于 ReAct 和 CoT 智能体框架，完整复现了 Reflexion 反思机制，还通过函数式编程范式重新设计了系统架构，实现了高效的异步并发处理；
2. 通过在 CoT 和 ReAct 两个主流框架上实现反思机制，系统地分析了反思策略在不同场景下的效果差异；
3. 在 HotpotQA 数据集上进行系统实验，使用了更新的模型组合，提供了与原论文的对比视角。

复现工作不仅验证了原论文的核心结论，还通过改进的工程实现和详细的实验分析，为 Reflexion 框架的实际应用提供了有价值的参考。

2 相关工作

2.1 大语言模型的推理能力增强

近年来，提升大语言模型推理能力的研究取得了显著进展。Chain-of-Thought (CoT) [11] 提示方法通过引导模型显式地展示推理过程，显著提升了模型在复杂任务中的表现。ReAct [14] 框架则进一步将推理和行动相结合，使模型能够在思考的同时执行信息搜索等操作，这种设计使智能体能够主动获取信息，而不是仅依赖于预先提供的上下文。

最近的一些工作进一步探索了提升模型推理能力的新方法。Self-Refine [8] 采用迭代框架进行自我改进，通过自我评估来优化生成结果。这种方法虽然有效，但仅限于单次生成任务。Xie 等人 [12] 提出使用随机束搜索来实现更高效的决策搜索策略，通过自评估组件实现前瞻性优势。Paul 等人 [9] 通过微调评论模型来提供轨迹内的中间反馈，以改进推理响应。Kim 等人 [5] 则探索了在固定步数内进行重试的模式，而 Goodman [4] 则提出了一种定性评估步骤，用于对先前生成的内容提出优化建议。

2.2 语言模型的自我改进机制

在语言模型的自我改进机制方面，现有研究主要探索了几个重要方向。一些工作专注于使用评论模型提供中间反馈以改进推理响应 [9]。特别值得注意的是，一些研究提出了定性评估步骤，用于提出对先前生成内容的优化建议 [2]。在编程领域，AlphaCode [7] 和 CodeT [1] 展示了测试执行的重要性，而 Self-debugging [3] 和 CodeRL [6] 则进一步探索了调试和强化学习在代码生成中的应用。

然而，这些方法往往缺乏持久的记忆机制来存储和利用经验。Reflexion 框架 [10] 的创新之处在于引入了自反思文本缓冲区，使智能体能够维护和利用过往经验。这种设计不仅允许智能体识别自身错误，还能从错误中总结经验教训，并在后续尝试中应用这些见解。Reflexion 基于语言的强化学习方法，无需更新模型参数就能实现策略优化。

3 本文方法

3.1 整体架构

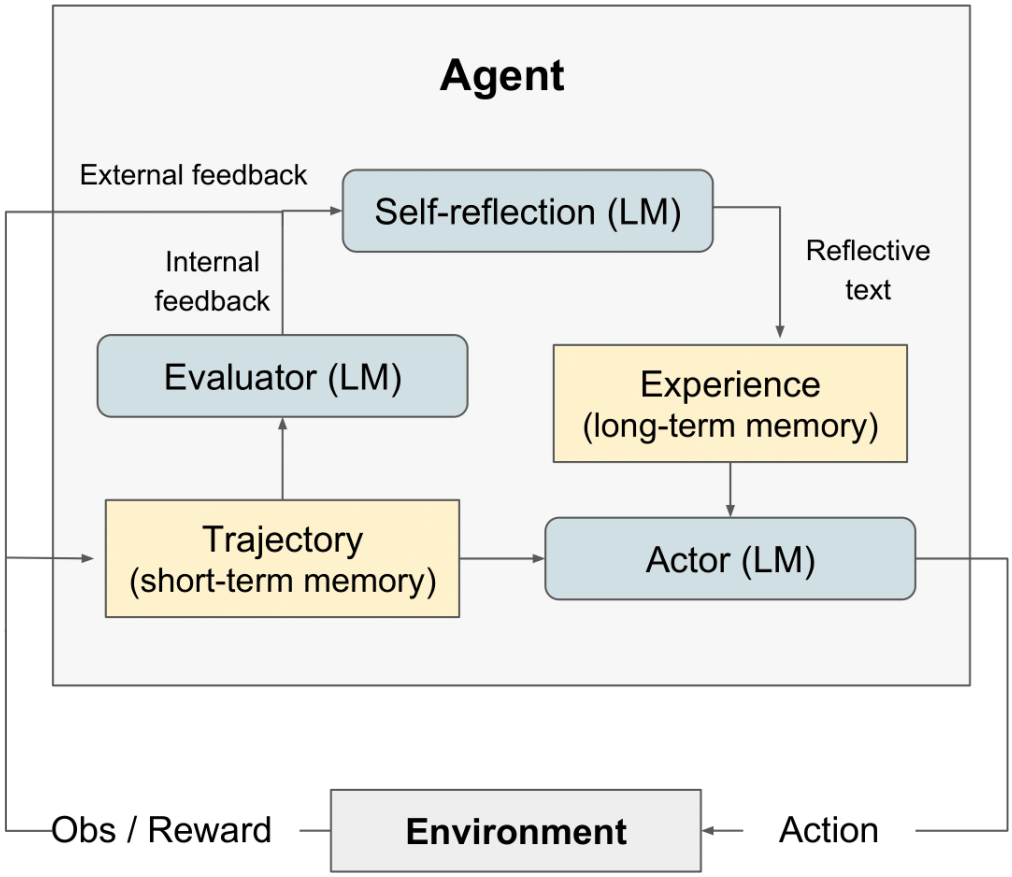

图 2. Reflexion 框架

如图 2所示，Reflexion 框架通过模拟人类的学习过程，将自然语言反思机制引入到语言智能体中。该框架中，大模型被赋予了三种紧密相关的功能，包含：执行（Actor）、评估（Evaluator）和自我反思（Self-Reflection）。这三种功能形成了一个完整的决策-执行-反思循环，再加上长短期记忆模块，使智能体能够不断从经验中学习和改进。

3.1.1 记忆系统

Reflexion 框架采用了双层记忆结构，这种设计借鉴了人类认知系统中工作记忆和长期记忆的概念。短期记忆（Trajectory）负责记录当前问题的解决过程，包括思考步骤、行动选择和观察结果。这些即时信息为当前决策提供直接支持，并在错误重试时为下一轮决策提供参考。而长期记忆（Experience）则存储了智能体在解决不同问题时积累的经验和教训，这些经验可以指导未来的决策过程。

在我们的实现中，短期记忆（Trajectory）记录了智能体在当前问题上的完整思考过程。例如，当回答”谁发明了电话？”这个问题时：

思考：我需要确认谁最早发明了实用的电话系统

行动：搜索电话发明历史
观察：亚历山大·格雷厄姆·贝尔在1876年获得了电话专利
思考：虽然有其他发明者也做出了贡献，但贝尔的专利最具影响力

而长期记忆（Experience）则存储了智能体解决不同问题时积累的经验和教训：

经验1：处理历史发明问题时，需要注意专利申请日期
经验2：多个发明者可能同时做出贡献，要仔细核实
经验3：重大发明往往伴随着法律争议，需要查证

3.1.2 反馈与学习机制

Reflexion 框架的另一个创新是将强化学习中的反馈机制与语言模型的推理能力相结合。这种机制包含两个关键环节：外部反馈和内部反思。外部反馈来自环境，提供了客观的评价信号；内部反思则是智能体对自身行为的分析和总结。这两种反馈相互补充，共同促进智能体的持续改进。

以解决一道复杂的数学题为例，框架的工作流程如下：

1．首次尝试：
问题：求解方程 $x^2 + 5x + 6 = 0$
思考：这是一个二次方程，可以用求根公式
计算：$x = (-5 \pm \sqrt{(25-24)})/2 = (-5 \pm 1)/2$
答案：$x = -3$ 或 -2

2．获得负面反馈（错误）后的反思：
反思：我的错误在于：
1．计算判别式时忽略了负号
2．没有验证最终答案
下次应该：
1．列出详细的计算步骤
2．代入结果验证

3.2 反思策略设计

在实现反思机制时，我们设计了一系列渐进式的策略，每种策略都针对特定的学习场景。纯反思策略（REFLEXION）模拟了独立思考的过程，不依赖外部资料，仅通过对自身解题过程的反思来改进。错误记忆策略（GT_EPM）则引入了系统化的错误记录和分析机制，使智能体能够从失败中学习。最后，组合策略（GT_EPM_REFLEXION）将这两种方法结合，既保留了反思的优势，又利用了错误记忆的经验积累。

3.2.1 纯反思策略（REFLEXION）

纯反思策略模拟了”闭卷思考”的场景：不依赖外部资料，仅通过对自身解题过程的反思来改进。这种策略特别适合培养智能体的自主思考能力。例如，在解决一个逻辑推理问题时：

第一次尝试:
问题: 如果所有的A都是B, 所有的B都是C, 那么所有的A都是C吗 ?
思考: 让我分析一下...
- A都是B, 说明A是B的子集
- B都是C, 说明B是C的子集
答案: 不确定

反思: 我的推理有误。如果A是B的子集, B是C的子集,
根据集合的传递性, A必然也是C的子集。
这说明我需要更好地理解集合关系的传递性。

第二次尝试:
思考: 根据上次的反思...
- 如果A B且B C
- 那么根据集合的传递性
- 必然有A C
答案: 是的, 所有的A都是C

3.2.2 错误记忆策略 (GT_EPM)

错误记忆策略 (Error Pattern Memory) 借鉴了" 错题本" 的概念。就像学生会专门记录和分析做错的题目一样, 这个策略让智能体能够系统地记录和学习从错误中得到的经验。

考虑以下例子:

问题: 计算 $(a+b)^2$ 的展开式
第一次尝试:
思考: 展开平方...
答案: $a^2 + b^2$

错误总结:
1. 遗漏了交叉项 2ab
2. 这是一个常见的代数展开错误
3. 需要记住完整的平方公式: $(a+b)^2 = a^2 + 2ab + b^2$

第二次尝试:
思考: 根据平方公式...
答案: $a^2 + 2ab + b^2$

3.2.3 组合策略 (GT_EPM_REFLEXION)

组合策略将错误记忆和深度反思结合起来, 这就像一个优秀学生不仅记录错题, 还会深入分析错误原因并总结改进方法。这种策略特别适合处理复杂的推理任务。

例如，在解决一个科学推理问题时：

问题：为什么北半球的河流右岸更容易发生侵蚀？
第一次尝试：
思考：可能与水流方向有关...
答案：因为水流总是向右

错误记忆：
- 答案过于简单化
- 忽略了地球自转的影响
- 没有考虑科学原理

反思：
1. 需要运用科学原理：科里奥利力
2. 考虑地球自转对流体运动的影响
3. 解释物理现象时要有完整的因果链

第二次尝试：
思考：由于地球自转，北半球的流体运动会受到科里奥利力的影响，
使其偏向右侧。这导致河流右岸受到更强的冲刷作用...
答案：因为科里奥利力使北半球流体偏向右侧，加强了对右岸的侵蚀

3.3 工程实现

本研究在工程实现层面采用了函数式编程范式。函数式编程的纯函数特性使得系统各个组件之间的依赖关系更加清晰，状态管理更加可控。特别是在处理复杂的反思链和推理过程时，函数式的不可变性和组合特性帮助我们构建了更可靠的推理流水线。

在并发处理方面，我们基于 asyncio 构建了高效的工作者池系统。通过实现异步 API 调用和细粒度的任务调度，系统能够同时处理多个问题，显著提升了整体吞吐量。处理 100 个测试样本的时间从单线程的 2 小时（估计）减少到了并行处理的 5 分钟（平均）。

状态管理是函数式架构中的一个关键考虑点。在处理长期运行的反思链时，我们通过设计不可变的状态传递机制和巧妙的函数组合来管理状态信息。这种方案既保持了函数式编程的纯粹性，又实现了高效的状态管理。特别是在处理反思过程中的中间状态时，这种设计显示出了显著优势。

4 实验

4.1 实验设置

4.1.1 模型选择

由于我们使用的模型与原论文不同，加上框架实现细节的差异，本研究的性能数据与原论文有一定出入。

原始论文采用了 ChatGPT 3.5 Turbo 作为基础模型，而复现过程中选择了不同的模型组合：使用 GPT4o-mini 作为主要的推理和反思模型，同时使用 Qwen 2.5 32B 作为判断模型。使用 GPT4o-mini 的主要考虑是性价比。该模型虽然参数量相对较小，但采用了更新的架构和训练方法，在推理任务上展现出了良好的性能。其更经济的定价策略使得大规模实验成为可能。此外，由于预训练数据更新，该模型在处理广泛的问题类型时表现出了相比 ChatGPT 3.5 Turbo 更好的知识覆盖度。

在判断模型的选择上，我们使用了部署在本地 GPU 服务器的 Qwen 2.5 32B 开源模型。使用它作为判断模型有两个主要优势：

1. 本地部署意味着零调用成本，这在需要频繁评估答案正确性的实验中特别重要；
2. 模型的参数量（32B）保证了判断的可靠性。

我们也尝试过使用 Qwen 2.5 32B 作为推理和反思模型，但该模型难以遵循 ReAct 格式进行主动式推理，需要做更多调节才能勉强完成反思，失败率非常高。

4.1.2 基础框架与实验实现

本研究的一个重要特点是在两个不同的智能体框架上实现并评估了反思机制：Chain-of-Thought (CoT) 和 Reasoning-Acting (ReAct)。这两个框架在信息获取方式上有本质区别：CoT 框架通过直接提供上下文信息（context）来支持推理，而 ReAct 框架则通过维基百科 API 主动搜索相关信息。这种设计允许我们从不同角度评估反思机制的效果。

在 CoT 框架中，我们实现了以下策略变体：

1. CoT_ONLY：基础的思维链推理，不提供外部信息
2. CoT_GT：在推理过程中提供问题相关的支持性文档
3. CoT_REFLEXION：在基础推理基础上添加反思机制
4. CoT_GT_REFLEXION：结合支持性文档和反思机制
5. CoT_GT_EPM：结合支持性文档和错误记忆机制
6. CoT_GT_EPM_REFLEXION：完整框架，集成所有增强机制

在 ReAct 框架中，我们实现了类似的策略组合：

1. ReAct_ONLY：基础的推理-行动框架，通过 API 主动搜索信息
2. ReAct_REFLEXION：添加反思机制，允许智能体反思之前的搜索和推理过程
3. ReAct_EPM：引入错误记忆机制，记录和分析搜索策略的失误
4. ReAct_EPM_REFLEXION：完整的反思增强版本

不同策略的实验具有几个重要优势：

1. 允许我们比较被动接收信息（CoT）和主动搜索信息（ReAct）两种范式下反思机制的效果

2. 通过在不同框架中实现相似的策略组合，我们可以评估反思机制的普适性

3. 这种设计也能帮助理解反思框架的各个组件的相对重要性

在实验实现上，我们采用了一系列工程优化措施。对于 ReAct 框架，我们实现了高效的 API 调用机制和缓存系统，以减少外部查询的延迟。对于 CoT 框架，我们优化了上下文信息的处理流程，确保模型能够有效利用提供的支持性文档。两个框架都实现了基于 asyncio 的异步并行处理，支持多个 queue 同时处理不同的问题。此外，我们还实现了统一的错误处理机制，确保在网络波动或模型响应异常时能够自动恢复。

为了保证评估的客观性，我们在两个框架中使用了相同的判断模型（Qwen 2.5 32B）来评估答案的正确性。每个问题的推理过程被限制在最多 5 步内完成，以平衡推理深度和计算效率。这种统一的评估标准确保了不同框架和策略之间的性能数据具有可比性。

4.2 实验数据集和评估

在数据集选择方面，我们使用了 HotpotQA 评测集中的 100 个问题样本，这些样本与原始 Reflexion 论文中使用的样本保持一致。这种选择使得我们能够直接将实验结果与原始研究进行对比。

在评估指标方面，我们主要关注两个核心维度：答案准确率和解题效率。答案准确率直接反映了模型的推理能力，通过判断最终给出的答案是否正确来衡量。解题效率则通过记录达到正确答案所需的反思次数来评估，我们将反思次数超过 4 次的情况判定为求解失败。

4.3 实验结果与分析

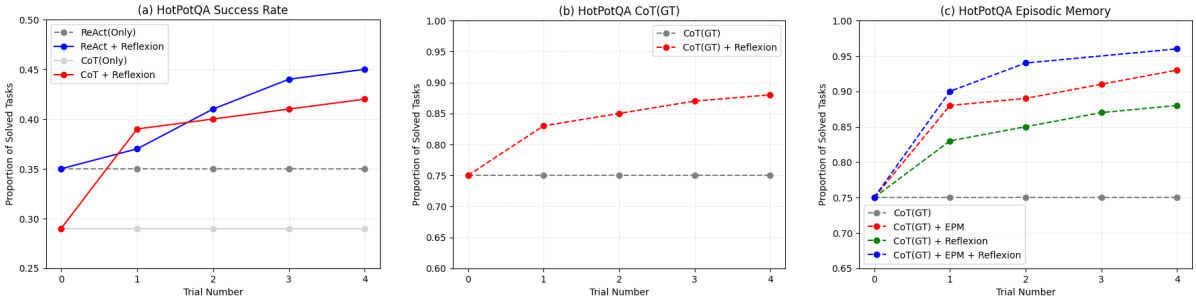

图 3. 实验结果

在基准性能评估中，CoT 和 ReAct 框架展现出了不同的起点和增长模式。如图 3(a) 所示，未经增强的 CoT 框架达到了 29% 的基准准确率，而 ReAct 框架则实现了 35% 的基准准确率。这种差异反映了两个框架在处理多跳问答任务时的固有特点：ReAct 框架通过主动信息搜索获得了初始优势，而 CoT 框架则受限于其对预设上下文的依赖。

当引入真实背景信息（Ground Truth，GT）支持时，如图 3(b) 所示，CoT 框架的性能出现显著提升，准确率达到 75%。这一发现突出了高质量上下文信息对推理性能的关键影响。

特别值得注意的是，在此基础上引入反思机制后，准确率进一步提升至 88%，相较于基线提升了 17.3 个百分点。通过分析反思过程中的推理链变化，我们发现这种提升主要源于模型能够识别和修复推理链中的薄弱环节，优化信息整合方式。

在消融实验中，如图 3(c) 所示，错误记忆机制（EPM）展现出了显著的性能提升效果。EPM 使 CoT_GT 的准确率从 75% 提升至 93%，提升幅度达 24%。当将 EPM 与反思机制结合时，性能进一步提升至 96%，实现了 28% 的总体提升。这种协同效应表明，系统化的错误记录能够为反思过程提供更有效的指导。通过分析反思内容，我们发现包含明确错误诊断和具体改进建议的反思往往能带来更显著的性能提升。

5 主要发现

本研究通过系统性实验验证了语言反思机制在增强大语言模型推理能力方面的有效性。一个重要发现是，通过语言模型进行口头强化学习（Verbal Reinforcement Learning）能够在不更新模型参数的情况下实现策略优化，这种方法既高效又简单。我们的实验结果表明，反思机制能够与 ReAct 和 CoT 等主流 Agent 框架良好结合，在不同场景下带来 15%-45% 的性能提升。在 CoT 框架中，反思主要通过优化推理链和加强信息整合来提升性能；在 ReAct 框架中，反思则同时改进了信息搜索策略和推理过程。这种差异性表明，反思机制能够根据框架特点自适应地发挥作用。

特别值得注意的是，在消融实验中，我们发现短期记忆（上一轮的失败记录）和长期经验（累积的错误模式）都对性能提升有显著贡献，但短期记忆的效果更为明显。这一发现与原始论文的结果有所不同，可能是由于我们使用了更新的模型（GPT4-Turbo-mini）而非原论文中的 ChatGPT 3.5 Turbo，这表明模型能力的提升可能改变了不同记忆机制的相对重要性。

6 结论与展望

本研究通过系统性实验验证了反思机制在增强大语言模型推理能力方面的有效性，特别是在结合错误记忆机制后能够实现显著的性能提升。研究发现，反思机制能够根据不同框架的特点自适应地发挥作用，这为构建更强大的推理系统提供了新的思路。

尽管取得了积极的实验结果，本研究仍存在几个重要的局限性：

1. **反思能力的迁移问题：**当前框架没有提供有效的方案来解决如何将反思能力迁移到不能进行重试的场景，以及如何将在一个任务上获得的错误经验迁移到其他同类任务中。

2. **模型依赖性：**反思能力和结构化输出能力严重依赖底层语言模型的性能。在我们使用 Qwen 2.5 32B 模型进行测试时，发现成功率显著降低。这种依赖性不仅体现在模型的基础能力上，还表现在多个技术层面：提示词的质量、模型的上下文长度、输出格式要求、推理速度等

未来的研究可以从以下方向展开：

1. **探索更复杂的反思策略：**如引入元认知机制，使模型能够更好地理解和改进自身的推理过程。

2. **反思能力迁移：**探索如何将反思机制扩展到单次推理场景，以及如何实现跨任务的经验迁移。这可能需要设计更复杂的记忆机制和迁移学习策略。

3. **模型依赖优化：**研究如何降低框架对特定模型能力的依赖，可能的方向包括设计更鲁棒的提示模板、优化输出格式要求、改进错误处理机制等。

参考文献

[1] Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. *arXiv preprint arXiv:2207.10397*, 2022.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[4] Noah Goodman. Meta-prompt: A simple self-improving language agent. *noahgoodman.substack.com*, 2023.

[5] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*, 2023.

[6] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

[7] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[8] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

[9] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

[10] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[11] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[12] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023.

[13] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[14] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.