Published on
Tuesday, November 7, 2023

利用大语言模型开发先进的推理与规划算法 [译]

Authors

- Name

宝玉
Twitter
@dotey

原文：Developing Advanced Reasoning and Planning Algorithms with LLMs

本文介绍了 Branches，这是我们开发的一款工具，用于构建和展示先进的大语言模型（LLMs）推理和规划算法的原型。我们利用 Branches 来解决为 HumanEval 生成 Python 代码的挑战。

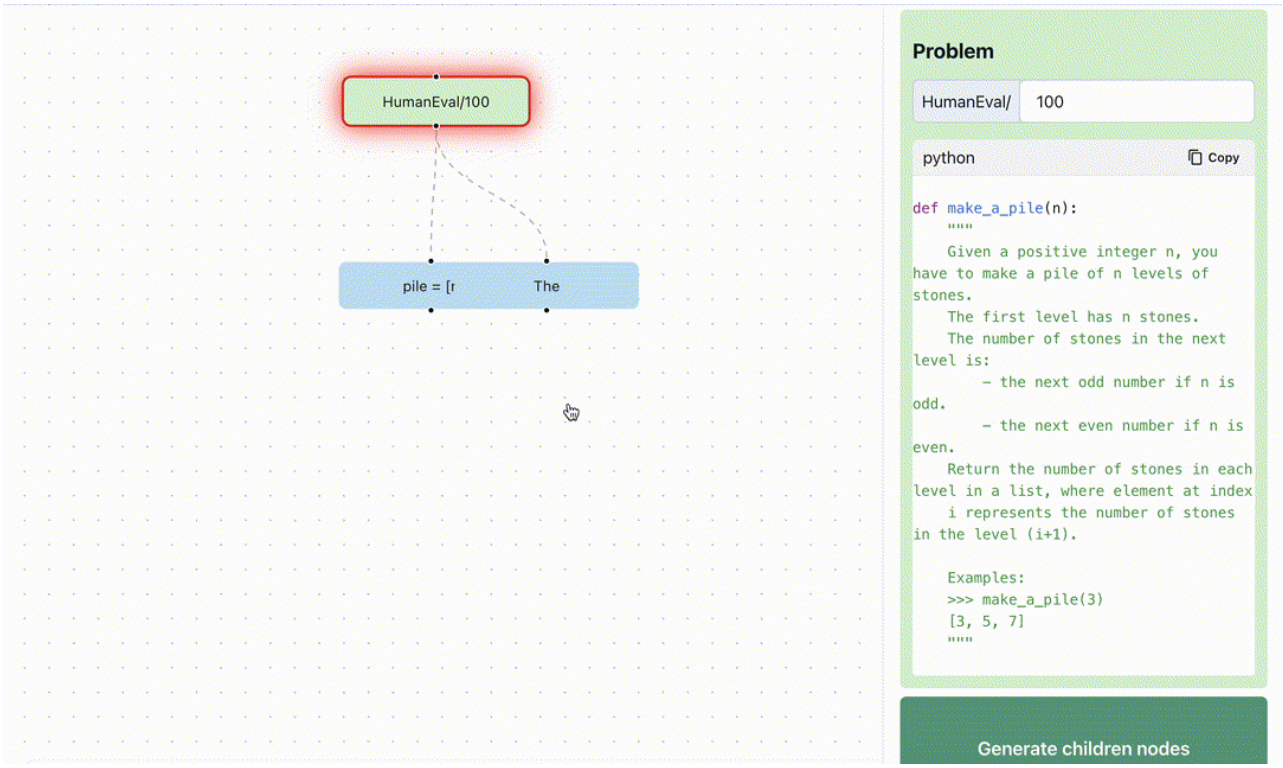
作者：Arun Patro、Rami Sbahi 和 Thomas Ahle

发表日期：2023 年 11 月 5 日

大语言模型（LLMs）在各种任务上展现出了卓越的性能，从写作小说到解释复杂的编程代码。不过，当任务涉及到需要长期规划或对世界最新知识的理解时，LLMs 的表现会大幅下降。

代码*生成*正是这类任务之一，它涉及到多元的规范、约束以及测试。哪怕是这些看似简单的要求也会让自动代码生成变得复杂。而一旦规范之间出现冲突或模糊不清，难度就更大了。即使需求定义得非常清晰，语言模型本身的概率特性也可能让代码出错，因此需要仔细的人工监督。代码必须要能够*编译*1并且正确地解决问题。正确生成代码是一项*艰巨*的任务，即便是像 `GPT-4` 这样强大的系统也常常会在这方面遇到困难。

目标驱动的人工智能 是一种新范式，它承诺解决 LLMs 在规划和推理上的不足2。在本文中，我们将讲述树状推理和反思方法如何提高代码生成的质量，使其超越非目标驱动的方式。要看目标驱动推理如何在实际中运作，可以查看这个演示。我们展示了树状推理如何帮助模型解释并纠正错误代码，同时完成 HumanEval3的基准测试。

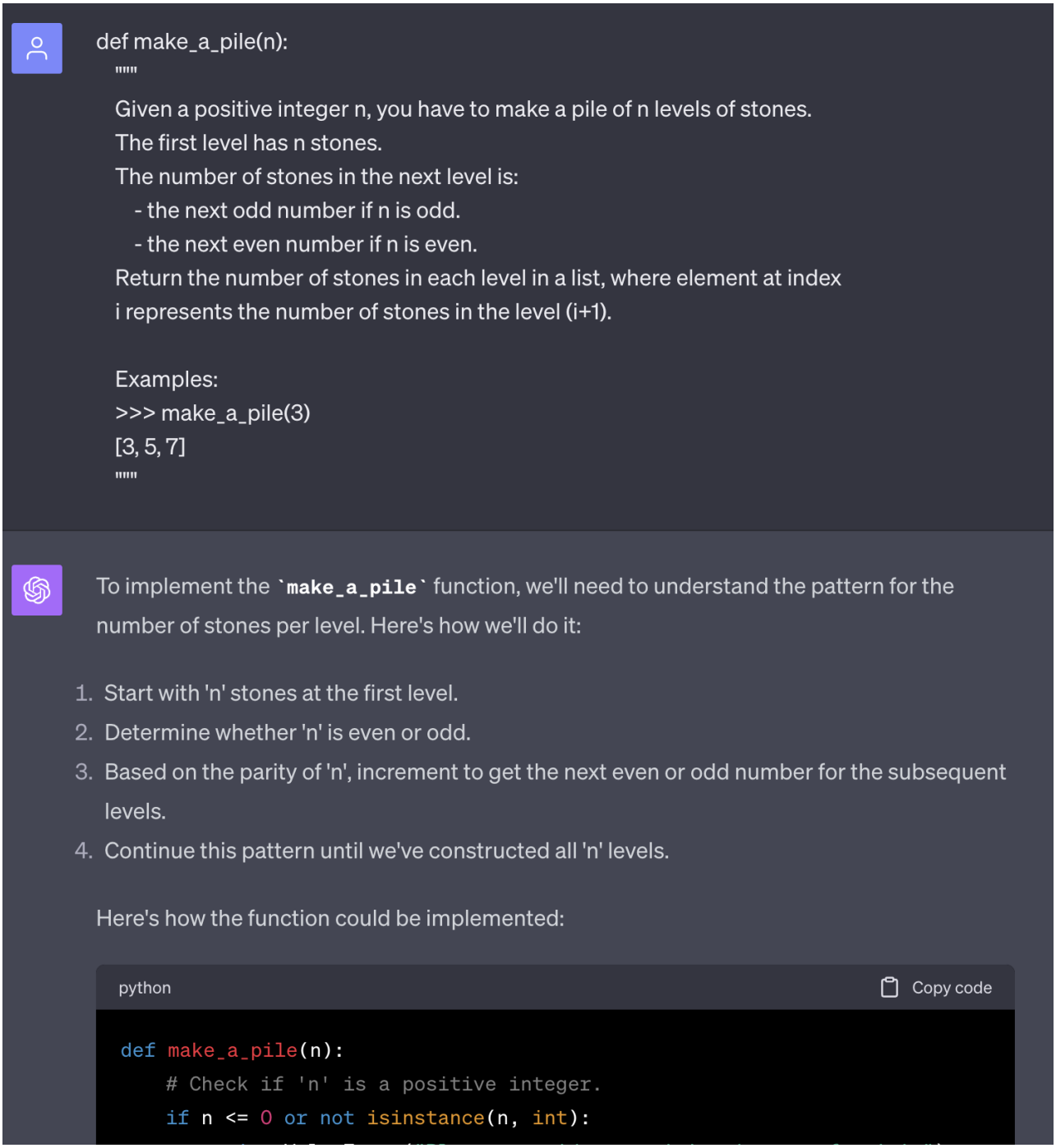


这是演示的动图！图中展示了模型如何自我修正错误。模型对错误进行推理，并生成修改后的代码以正确解决问题。橙色节点代表失败，紫色节点代表成功。

我们最近向公众开源了 `Branches` – 一个旨在帮助用户直观展示和探索大语言模型（LLMs）的规划与推理过程的工具，我们也用它来创建了演示。`Branches` 既可以作为一个原型开发工具，也可以作为一个网络应用程序，它设计上能与基于图的常用推理算法兼容。我们希望它能够成为社区中推动目标导向代理系统探索与开发的有力工具。

快速思考的隐患

在深入了解树状结构的处理方式之前，先来讨论一种更常见的做法，即直接让 `gpt-3.5-turbo` 或 `gpt-4` 这样的聊天机器人来编写代码。虽然这些模型有时能够正确地给出答案，但也仅仅是有时而已。其他时候，它们可能会出现逻辑错误或算法错误。在处理关键任务时，这些错误是不能接受的。



(a) 提示及其分解步骤

(b) 编写的代码，并未正确解决问题

图 1: GPT-4 在解答一个简单问题时因为顺序性思维出错。当面对第 100 个 HumanEval 问题时，尽管文档字符串中已经明确提出了测试案例 `make_a_pile(3) == [3, 5, 7]`，代码错误地返回了 `[3, 5, 6]`。

这种情况和人类在首次尝试编程时不总是能得到正确结果的情况相似。问题的一部分在于大语言模型（LLMs）是逐行生成代码的，一旦某行代码出现错误，随后的代码生成也很可能被影响。这些错误的标记或序列就好比是开启了幻觉列车。我们需要找到方法，在大语言模型开始出错的那一刻即时发现并拦截它，防止错误像列车脱轨一样无法挽回。

利用树形结构和反思机制提升代码生成效率

在语言模型（LLMs）中，一个降低错误率的简便方法就是按步就班地思考 step by step。在这种方式下，我们把复杂的任务切分成一系列小步骤，然后一步步来解决。这大概模拟了人类分解问题并解决问题的方式。指导 LLM 按步骤思考，仅仅是一系列使 LLM 的回答步骤化的方法中的一个实例。
但是，传统的少次尝试（few-shot）和思维链（chain-of-thought）的提示方法仍然不够。它们还是会遭受到错误传播的问题，要想提高它们的成功率，往往需要多次重新生成答案。如果能从出现错误的那一点继续，而不是重新生成整个答案链，那么速度会快得多。
正是因为这样，树状方法应运而生。从错误发生的地方展开新的解决方案探索，以及追寻多条解决路径，正是 Tree of Thoughts (ToT) 4 的核心思想。在 Language Model Cascades 中也提出了将一个任务拆分为多个推理步骤，甚至可能包括验证步骤的概念。这些方法允许通过外部评估来揭示中间步骤，以此来确保目标能够正确引导整个推理过程。
在这篇文章中，我们专注讨论一个特殊的目标驱动型 AI 示例，它采用了树状方法，并结合了 Python 解释器的反馈进行了增强。通过自我评估和反思，这些方法为我们提供了能够自行检测并纠正错误的代码自动生成代理 5。

利用树形逻辑推理提升的 Python 代码生成效率。

分支：原型化基于图的目标驱动人工智能

在 Normal Computing，我们坚信，让人类参与进来是确保人工智能长远发展和实用性的关键。为了保持这种有效的反馈循环，我们提供了更好的工具和接口，帮助开发者构建和操作 AI 系统。秉承这个理念，我们推出了开源项目 `Branches`。在这个项目中，我们提供了一些工具，它们能够将基于图的推理算法与各种反馈驱动的基准进行可视化对比。目前，我们已经展示了两个示例，它们展示了如何将基于树的推理技术应用于 HumanEval 和 24 点游戏。AI 模型对每一个中间步骤进行打分，指出每步的错误，并据此决定如何优化搜索边界。

利用 `branches` 在 24 点游戏中寻找解决方案的过程进行了可视化展示。图中显示了搜索过程被分解为若干步骤，每一步骤都由语言模型自我评估。
我们采用 ToT（思维树）模式来解决 HumanEval 数据集 6 上的编程难题，用于生成 Python 代码。如下图所示，模型尝试多个解决方案并进行测试。如果它没有通过某个测试案例或产生了无效代码，外部的 Python 代码解释器会提供详尽的反馈，包括完整的错误追踪信息 7。模型会对这些错误追踪信息进行反思，总结错误，并找到改进代码的方向。这种通过融入现实世界反馈的方式，大幅提升了成功率。
在我们的交互式演示中，你可以探索应用于 HumanEval 的“思维树”推理过程。这与 OpenAI 的代码解释器不同，后者目前不能处理详细的反馈，因此只能知道代码没能成功运行。

这是一次树形探索，语言模型通过错误追踪反思自己代码中的错误。图中详细展示了输入信息、错误代码、错误追踪、对错误的反思以及最终修正的代码。橙色表示出错，紫色表示修正成功。

明天的 AI，走向何方？

这次的演示只是我们在拓展以目标为驱动的人工智能前沿的小小一步。我们不仅介绍了开发和评价 LLM 规划与推理算法的新路径，还特别深入探讨了在编程任务中，如何在反馈的帮助下运用树状结构方法。以下是我们对如何让这些树状方法运作得更迅速、精确及协同的展望：

1. **快速推理：** 鉴于提示的头部在各节点间是固定不变的，我们推测通过在各提示完成之间利用键值（KV）缓存技术[8]，可以实现推理速度的提升。虽然现今主流的 LLM 服务提供商还未广泛支持跨提示的 KV 缓存功能，但通过精细的工程调整，这一挑战是可以被克服的。
2. **扩展记忆：** 当任务变得更加长和复杂时，对反馈信息的需求很快就会超出许多模型的处理范围。记忆型变换器（Memorizing transformers）正是打破常规上下文限制的一种方法（我们最近通过扩展思维变换器（Extended Mind Transformers）做了进一步的推广）。这些扩展的记忆功能有可能被用来提升搜索过程的效率。
3. **人类反馈：** 引入人类反馈，能够让定制化的质性目标指引探索过程。未来的研究可以在图形搜索算法和演示可视化方面，更好地融合这类反馈。

致谢

我们特别感谢 Paradigm 的 Flux，它是我们这次可视化演示的基石。

脚注

1. 借助 `outlines` 工具包，我们能确保编写的代码可以成功编译，但这并不意味着代码就一定无误↩
2. 以 Yann LeCun 关于目标导向 AI 的 发言 为例↩
3. https://github.com/openai/human-eval ↩
4. ToT 扩展了诸如思维链条（thoughts）、验证过程（verification）、数据密集度（density）等多种方法↩
5. 想了解更多关于 AI 的可解释性和幻觉检测，不妨阅读我们的 近期博客文章↩
6. 这是一个由 164 个 Python 编程问题组成的测试集↩
7. 可以在这个链接找到对此的简要概述：https://realpython.com/python-traceback/ ↩
8. 想要深入了解 KV-Caching 的精妙设计吗？点击这里，一篇详细的解析等你发现。↩

重用版权信息

Discuss on Twitter • View on GitHub

Tags

llmai

Previous Article

Assistant API 文档 [译]

Next Article

大语言模型遭受的对抗性攻击 [译]

← Back to the blog

宝玉

•

•

宝玉的工程技术分享

Tailwind Nextjs Theme