Vibe Coding 介绍

2022年11月30日，ChatGPT的发布，标志着大语言模型（Large Language Model，LLM）第一次以现象级的规模进入大众视野

那时，大多数人并不理解什么是Transformer，不知道GPT如何训练，也不了解注意力机制的工作原理。

但他们发现，自己只需要在聊天框中输入一句话，计算机就能够写出文章、解释数学问题、生成程序，甚至与自己连续讨论复杂的问题。

对于研究者而言，这些能力并非完全凭空出现，而是多年人工智能研究不断积累的结果。

但对于普通人而言，这是他们第一次如此直接地感受到先进人工智能的能力。

ChatGPT最重要的贡献之一，不是让人工智能第一次拥有了生成语言的能力，而是让数以亿计的人第一次有机会亲自体验这种能力。

那么，使这一切成为可能的技术，究竟是什么？



Sam Altman于2022年11月30日发布的推文截图

Attention Is All You Need ——Transformer的诞生

2017年6月12日，Google的八位研究者向arXiv提交了一篇论文，标题是《Attention Is All You Need》。

他们当时想解决的，仅仅是如何让计算机更高效地理解和翻译语言。

论文提出了一种全新的神经网络架构：Transformer。



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

《Attention Is All You Need》论文首页

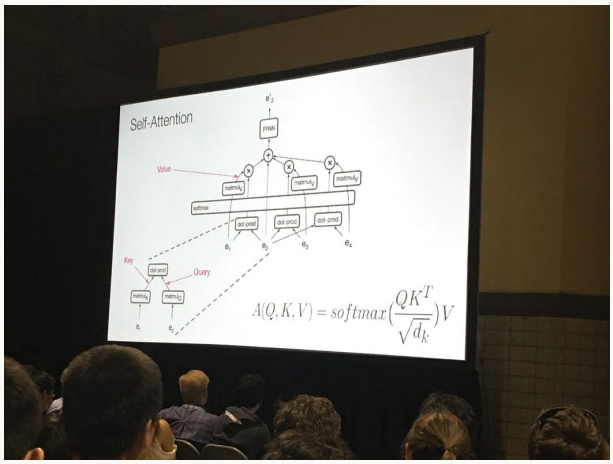
https://arxiv.org/abs/1706.03762?utm_source=chatgpt.com

原始 Transformer 由编码器和解码器组成。编码器处理输入句子，解码器逐步生成译文，并借助注意力机制读取输入中的信息。[1]

自注意力与上下文

它抛弃了传统序列模型对循环结构的依赖，让注意力机制成为信息交互的核心，并显著改善了并行训练效率。

当时的人们还不知道，这项最初面向机器翻译的研究，会成为后来GPT、BERT以及大语言模型浪潮的关键基础。



2017年NIPS会议上讲解Self-Attention的现场画面

自注意力让一个位置的表示结合其他位置的信息，词序则由位置编码提供。[1] 下图用 bank 展示上下文的作用：river 和 money 分别为“河岸”与“银行”的词义提供线索。

We sat on the bank of the river.	→ 河岸
She deposited money in the bank.	→ 银行

图 1 同一个词在两个句子中的含义。下划线标出目标词与上下文线索。

语言模型与代码生成

生成式语言模型把需求和代码切成 Token，再逐步生成后续 Token。Token 是模型处理文本的基本单位，包括字词片段和符号。

01 **输入：要求与书单**

“把已读的书单独列出来。”

小王子 · 已读　　海底两万里 · 未读　　城南旧事 · 已读

数据中的 read 字段记录是否已读。

02 **生成：按阅读状态筛选**

```
function readBooks(books) {
  return books.filter(book => book.read);
}
```

这段函数遍历书单，留下 read 为真的条目。

03 **运行：检查返回的书目**

小王子　　城南旧事

图 2 JavaScript 筛选函数示例。输入的每本书都有标题和阅读状态。

训练让模型学到代码的语法和常见写法；本轮输入说明具体要做什么，并提供现有数据。调整输入中的任务说明与示例，就能引导本次生成。[9]

生成后还要执行代码。2021 年的代码生成研究用函数测试判断实现是否正确；图中的函数也要通过运行，检查返回的书目是否符合要求。[11]

SWE-agent：
AI开始在数字世界中长出血肉



2024年4月2日，普林斯顿大学的研究团队公开发布了SWE-agent。SWE-agent提出了专门为大语言模型设计的智能体—计算机接口（ACI），AI开始从一个能够回答问题的语言模型，走向一个能够操作数字环境、完成实际任务的智能体

SWE-agent 为搜索仓库、查看文件、编辑代码和执行程序设计了专门接口。模型通过这些工具接触实际项目，再根据返回结果继续工作。[2]



Preprint

SWE-AGENT: AGENT-COMPUTER INTERFACES ENABLE AUTOMATED SOFTWARE ENGINEERING

John Yang*[1] Carlos E. Jimenez*[1] Alexander Wettig[1]
Kilian Lieret[1] Shunyu Yao[1] Karthik Narasimhan[1] Ofir Press[1]
[1]Princeton Language and Intelligence (PLI), Princeton University

arXiv:2405.15793v1 [cs.SE] 6 May 2024

ABSTRACT

Software engineering is a challenging task requiring proficiency in both code generation and interacting with computers. In this paper, we introduce SWE-agent, an autonomous system that uses a language model to interact with a computer to solve software engineering tasks. We show that a custom-built agent-computer interface (ACI) greatly enhances the ability of an agent to create and edit code files, navigate entire repositories and execute programs. On SWE-bench, SWE-agent is able to solve 12.5% of issues, compared to the previous best of 3.8% achieved with retrieval-augmented generation (RAG). We explore how ACI design impacts an agent's behavior and performance, and provide insights on effective design.

1 INTRODUCTION

Language models (LMs) have become helpful assistants for software development (Chen et al., 2021; Austin et al., 2021; Li et al., 2022), with users playing the role of mediator between the LM and the computer to complete programming tasks. For instance, after executing LM-generated code, a user may request refinements from the LM based on computer feedback such as error messages. More recently, LMs have been used as autonomous *agents* that interact with computer environments without human intervention (Yang et al., 2023a; Wu et al., 2024; Xie et al., 2024). This approach has the potential to accelerate software development, but remains largely unexplored in realistic settings.

Figure 1: An agent interacts with a computer through an agent-computer interface (ACI), which includes the commands the agent uses and the format of the feedback from the computer.

This work introduces SWE-agent, an LM-based autonomous system that can interact with a computer to solve challenging real-world software engineering problems from SWE-bench (Jimenez et al., 2024). At every turn, SWE-agent outputs a thought and a command, and then receives feedback from the execution of the command in the environment (ReAct; Yao et al. (2023b)). As our main contribution, we establish the importance of designing an *agent-computer interface* (ACI) to enhance the LM agent's performance. We find that custom ACIs tailored to LMs can outperform existing user interfaces (UIs), such as the Linux shell, designed for human users.

Consider the baseline of using a Linux shell interface (Yang et al., 2023a), which naturally lends itself to turn-based interaction with an LM agent due to its text-only input commands and outputs.

*Equal contribution. Correspondence to {jyl482,carlosej}@princeton.edu.
Code and demo at swe-agent.com.

1

例如，用户报告“刷新后书单消失”。
模型查看保存逻辑，修改代码，再打开页面检查结果。

ACI 把查看、编辑和执行变成明确的操作；文件内容和命令输出，再返回给模型。

https://arxiv.org/pdf/2405.15793v1

SWE-agent 论文首页；点击链接可阅读全文。

工具调用与环境反馈

下面的例子分三步修复“刷新后书单消失”：读取代码、修改文件、操作页面。模型提出请求，工具执行操作，结果再返回模型，供下一步判断使用。



图 3 保存功能的修复示意。向右是操作请求，向左是执行结果。

在这个例子里，原来的书单只放在页面内存中，刷新就会丢失。修改后，添加书目时写入浏览器本地存储，打开页面时读回；刷新检查验证保存是否生效。

ReAct 将推理与行动交替组织，让环境反馈参与后续判断。[10] SWE-agent 则进一步研究了软件任务中的工具接口。[2]

Harness工程：Agent迈向完全体

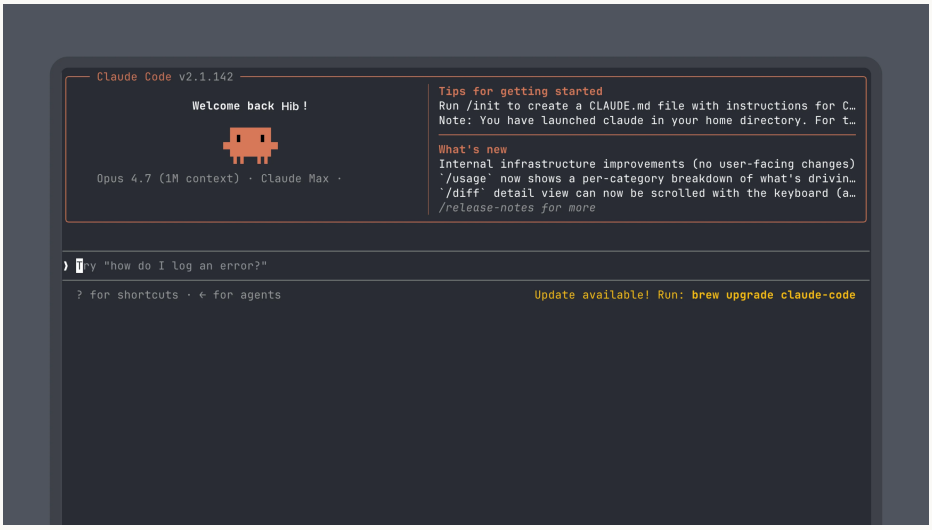
从拥有行动能力，到拥有完整的执行体系

2025年，Anthropic推出Claude Code，以简洁、克制的设计，让大语言模型开始承担复杂的软件工程任务。

2026年3月31日，Anthropic因软件发布失误，意外泄露了Claude Code约51万行TypeScript源码。

这次意外，让开发者得以深入研究一套成熟Agent系统的内部设计：它的核心行动循环精简克制，而支撑其运行的权限控制、工具调用、上下文管理与状态维护，却构成了一套庞大而精密的工程系统。

如果说大语言模型是Agent的大脑，SWE-agent让AI开始在数字世界中长出血肉，那么Harness工程，则为这副身体构建起骨架、神经与行动的秩序。



Claude code 页面所示

长期任务的工程设计参见 [3]；2026 年 3 月的发布事件参见 [8]。

任务怎样跨会话继续

上下文窗口限制一次模型调用能接收的信息量。长任务中的完整对话装不下时，代码、进度记录和 Git 历史提供了接续的依据。Anthropic 的长期任务实验，就用这些材料帮助新会话了解并继续工作。[3]

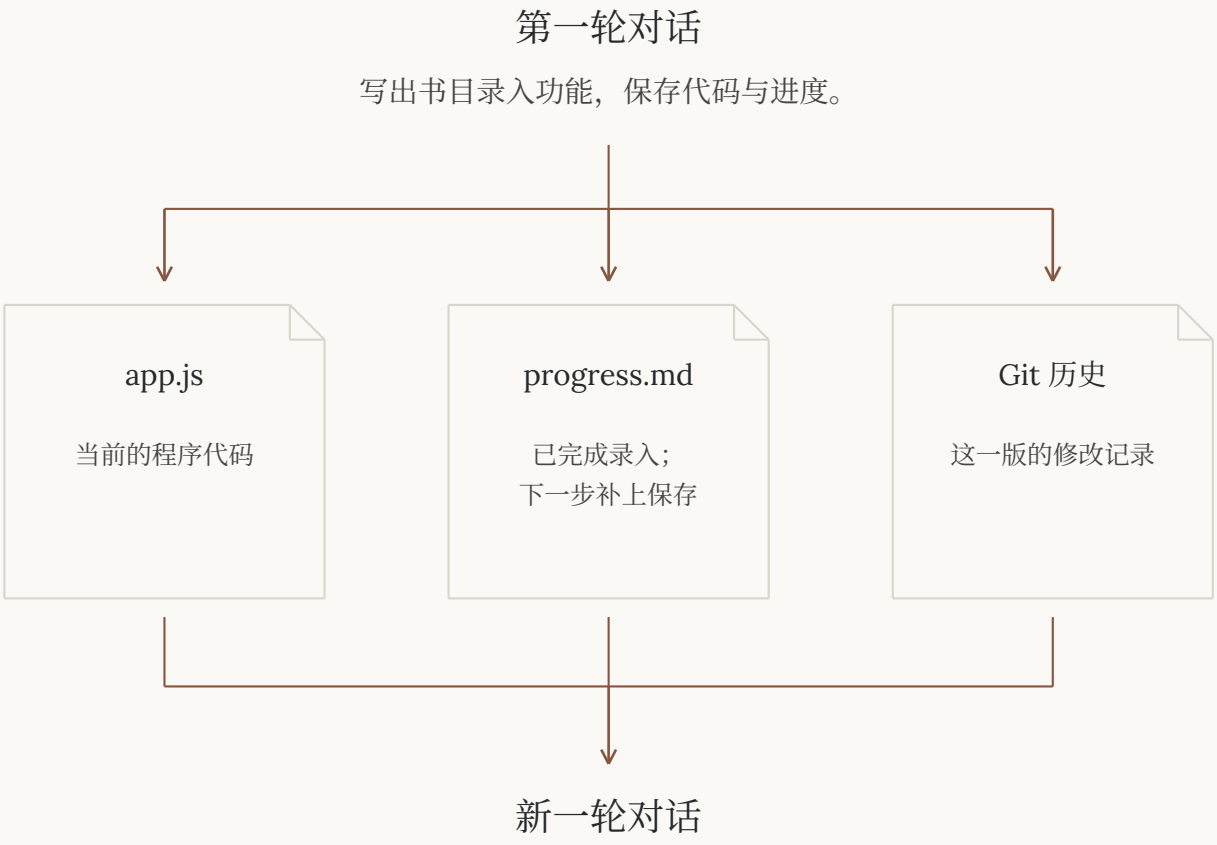


图 4 任务接续示意。文件保存在工作目录中，下一轮读取后进入上下文。

继续任务还需要恢复运行环境。例如，网页预览服务器停止后，需要重新启动；测试依赖的软件包尚未安装时，需要先安装。Harness 把模型调用、工具执行和这些准备工作组织在同一流程中。

目前比较知名的harness

1. Claude Code — 极简主义的终极兵器

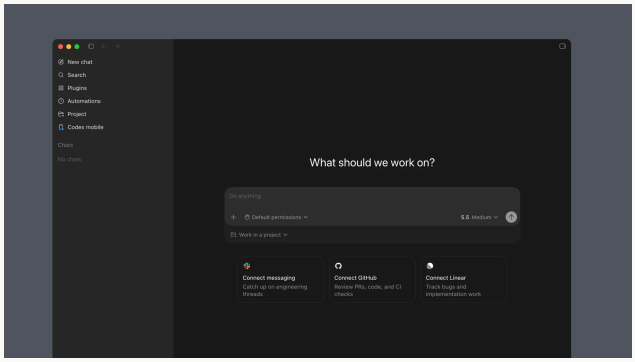
Anthropic为Claude模型量身打造的旗舰编程Agent，以克制的核心循环与模型主导的工具调用，充分释放大语言模型在复杂软件工程中的执行潜能

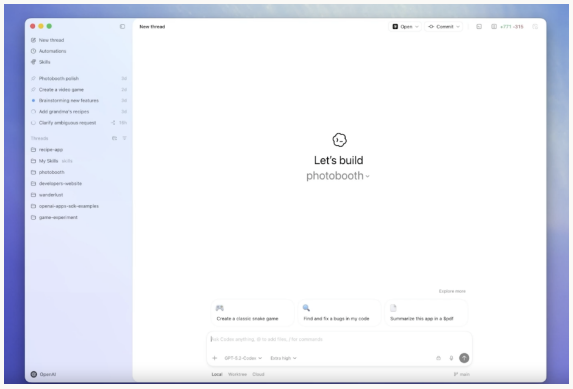
Claude Code 提供终端、IDE、桌面和浏览器等入口，支持读取代码、修改文件与执行命令。[4]

2. Codex — 工程化智能的集大成者

OpenAI打造的工程级Agent系统，以统一的执行内核、严格的沙箱隔离与多Agent并行协作，让AI从独立完成任务走向完整的软件工程交付。

Codex 的 Harness 管理对话状态、工具执行、沙箱与审批。App、CLI 等入口将这些能力用于具体开发任务。[5]



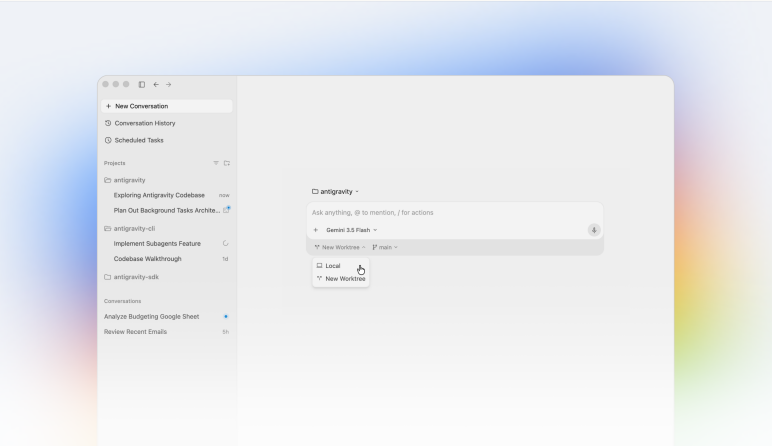


Codex 的深色与浅色界面。

开发平台与插件架构

3. Google Antigravity（反重力）— Agent时代的指挥中心

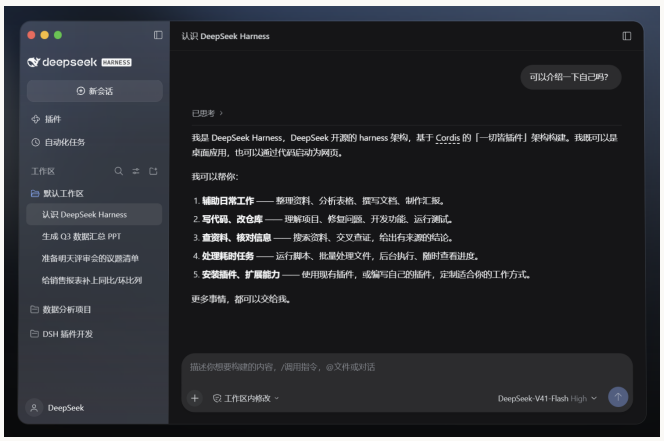
Google以Gemini模型为核心构建的智能体开发平台，凭借桌面化任务调度、动态子Agent与跨工具协作，让开发者从代码的编写者转变为智能体的指挥者。



Antigravity 将任务调度与多 Agent 协作放进开发界面，呈现各任务的运行状态与产物。[6]

4. DeepSeek Harness（DSH）— 万物皆可插件化

DeepSeek基于Cordis内核打造的开源Harness，以“一切皆插件”为设计哲学，将模型、工具、沙箱乃至执行循环解构为可自由组合的模块，探索Agent运行系统的无限可能。



DSH 基于 Cordis 将运行系统的能力组织成插件，便于组合与替换。官方仓库将其标注为开发者预览。[7]

从想法到可以试用的页面

把前面的技术放到一起，就能看见 Vibe Coding 的工作过程：用语言描述功能，AI 修改代码，使用者操作页面，再用具体表现提出下一轮要求。下面是同一份阅读清单的两个页面状态。



全部 · 3 本　　已读 · 2 本

图 5 阅读清单页面示意。点击“已读”后，未读条目被筛除。

这个筛选功能涉及三处实现：数据里记录每本书的阅读状态，函数按状态筛选书目，页面显示筛选后的结果。图 2 的函数负责筛选；这里的两个界面展示点击前后的变化。

接着提出“刷新后保留书单”，任务就进入图 3 的修复过程。这两项要求分别通过实际操作检查：点击“已读”，查看剩余书目；刷新页面，查看记录是否还在。

参考资料

正文中的编号对应以下资料，标题均可点击。产品资料核对于 2026 年 10 月 16 日。

[1] Attention Is All You Need

Vaswani 等；Transformer 原始论文。

[2] SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

Yang 等；面向软件任务的智能体—计算机接口。

[3] Effective harnesses for long-running agents

Anthropic；长期任务的环境、进度记录与增量工作。

[4] Claude Code Overview

Anthropic；产品入口和工具能力。

[5] Codex as a platform: build on the open agent harness

OpenAI；工具执行、状态与权限配置。

[6] Building the agentic future: Developer highlights from I/O 2026

Google；Antigravity 的任务调度与协作。

[7] DeepSeek Harness 官方仓库

DeepSeek；插件架构与开发者预览说明。

[8] Claude Code 发布事件报道

ITPro，2026-04-01；含 Anthropic 发言人的回应。

[9] Language Models are Few-Shot Learners

Brown 等；通过任务说明和示例进行上下文学习。

[10] ReAct: Synergizing Reasoning and Acting in Language Models

Yao 等；推理、行动与外部反馈。

[11] Evaluating Large Language Models Trained on Code

Chen 等；从函数描述生成代码及功能正确性评测。

字体：Lora / Noto Serif SC。纸色：#FAF9F6。阅读清单及其操作过程为教学示意。