谷歌科学家万字长文：《改变你职业生涯的一篇文章，我如何运用人工智能完成工作》建议每个人都要读一遍

AI工作坊 AI深度研究员 2024年08月10日 10:00 上海

（关注公众号并设为✨标，获取最新人工智能资讯和产品）

全文约13,000 字，阅读约需 20分钟

在当今科技界,关于人工智能是否被过度炒作的争论从未停息。然而,很少有像谷歌 DeepMind 的安全研究专家和机器学习科学家 Nicholas Carlini 这样的专家，用亲身经历为我们提供了一个独特的视角。通过他的文章,我们看到了大型语言模型(LLM)在实际应用中的强大能力和多样性。这些并非空洞的营销宣传,而是切实可以改变工作方式、提高生产效率、激发创意的工具。

最近,Carlini 本人亲自撰写了一篇长达万字的文章,详细介绍了他如何在日常生活和工作中使用 AI。这篇洞察力十足的文章不仅展示了 AI 技术的广泛应用,更让我们得以一窥未来科技将如何重塑我们的生活方式。

Carlini 在文章中列举了 50 个他亲身使用 AI 的实例,涵盖了工作效率、创意写作、编程辅助等多个方面。然而,令人惊讶的是,这 50 个例子仅仅是他所有 AI 应用场景的冰山一角 - 据他估计,不到他实际使用情况的 2%。这一变革令人震撼,它不仅体现了 AI 技术的深度和广度,更凸显了我们可能还未充分认识到 AI 的潜力。

事实上,Carlini 的经历可能预示着一个更广泛的趋势:随着 AI 技术的不断进步和普及,我们可能正站在一个技术革命的风口浪尖。就像个人电脑和互联网彻底改变了我们的生活和工作方式一样,AI 可能会成为下一个推动社会变革的关键力量。

那么,面对这样的前景,我们应该如何看待 AI 技术的发展呢?是抱有审慎的态度,还是积极拥抱变革?

原文如下:

作者：DeepMind 安全研究专家和机器学习科学家Nicholas Carlini

我不认为“人工智能”模型（我指的是大型语言模型）被过度炒作了。

确实，任何新技术都会吸引骗子。许多公司喜欢说他们在“使用人工智能”，就像他们以前声称自己依靠“区块链”一样。（正如我们一次又一次看到的那样）同样，我们可能正处于一个泡沫中。互联网在2000年泡沫破裂，但我们现在拥有的互联网应用，曾是科幻小说中的东西。

但我认为最近我们取得的进展不仅仅是炒作的原因是，过去一年中，我每周至少花几个小时与各种大型语言模型互动，并且对它们解决我给出的越来越困难的任务的能力印象深刻。因此，

我可以说，这些模型使我在编写代码方面的速度提高了至少50%，无论是研究项目还是个人的编程副业。

我在网上找到的大多数谈论LLM实用性的人要么过于乐观，声称所有工作将在三年内自动化，要么极度悲观，认为它们没有任何贡献，永远也不会有。

所以在这篇文章中，我只是想让讨论更加实际。我不会对未来进行任何论断。我只想列出一个列表，这是我（一名研究机器学习的程序员和研究科学家）与不同的大型语言模型进行的50次对话，这些对话显著提高了我进行研究的能力，并帮助我从事随机的编程副项目。其中包括：

- 使用我之前从未使用过的技术构建整个网络应用程序。
- 教我如何使用各种框架，尽管我之前从未使用过它们。
- 将几十个程序转换为C或Rust以提高10-100倍的性能。
- 精简大型代码库，显著简化项目。
- 为我在过去一年中撰写的几乎每一篇研究论文编写初始实验代码。
- 自动化几乎所有单调的任务或一次性脚本。
- 几乎完全取代了帮助我设置和配置新软件包或项目的网络搜索。
- 在帮助我调试错误信息方面，网络搜索的替代率约为50%。

如果对这些应用实例进行分类，**它们大致可以分为“辅助学习”和“自动化日常琐事”两类。前者对我来说至关重要，因为它让我能够轻松应对之前觉得有难度的任务；后者同样重要，因为它使我能集中精力去做我最擅长的事，解决真正的难题。**

关键是，这些例子展示的是我如何实际利用大型语言模型（LLMs）的。它们并不旨在炫耀技术的惊人能力，而是基于我解决工作需求的实际情况。这意味着，这些例子或许看起来不够引人注目，但实际上，我日常工作的很大一部分也都是平凡无奇的，而现今可用的LLMs几乎帮我自动化了所有这些工作。

通过这篇文章，我的目的是用一个接一个的实例，向你展示我在过去一年如何有效利用LLMs提升工作效率，直到你感到有些疲惫为止。要知道，尽管你可能已经看过不少例子，但实际上我所展示的，还不到我利用LLMs的总数的2%。

细微差别

如果说互联网处理得不尽如人意的是什么，那一定是对细节的把握。我绝不认为现今的大型语言模型（LLMs）能够主宰世界，也不打算讨论未来的模型能做什么，不能做什么。我只想探讨，现阶段的模型对我是否真的有用。

你可能会问，为什么还需要一篇文章来证明语言模型的用途呢？这难道不是显而易见的事实吗？但事实上，无论是在学术界、软件工程领域还是媒体行业，都有不少人声称LLMs毫无用处，只是一时的炒作，几年后将默默无闻，对世界没有任何影响。我要反驳这些观点，因为目前的LLMs已经证明了它们的实用性。

但同时，我也需要说明，另有一批声音同样强烈的人士持相反观点，他们认为现有模型能够取代所有程序员，人们不应再学习编程，因为不久后大家都将面临失业。虽然反驳这些观点不是本文的主旨，但我必须明确，我并不支持这种极端的看法。

此外，我也不会主张“目的正当化手段”，即便这些模型的训练确实存在诸多负面影响，我不认为应当因此而推广使用。

我充分意识到这些模型可能带来的负面后果，这些后果可能非常严重，包括但不限于散布错误信息、滥用、监控以及取代工作岗位（甚至有人担忧到人类灭绝的地步）。我将在不久的将来撰写一篇文章，深入探讨LLMs可能引发的危害，届时会在此放上链接。然而，这与语言模型是否有用的讨论是两码事——这正是我此处想要探讨的问题。

我也清楚你可能因为语言模型偏向于生成不准确的信息、重复已知事实、在面对复杂情况时可能彻底失败的倾向而不愿使用它们——可能我对这些局限性的了解比你更深。但本文不会讨论这些。因为我认为，尽管有这些不足，模型仍然是有用的。

此外，我也深知培训这些模型所涉及的伦理问题极具争议。你可能不赞成在未经许可的情况下使用人们的数据来训练它们（我对此的理解可能比你更深刻）。或许你关注的是那些为直接训练这些模型而获得微薄报酬的人。我承认这些都是问题，但本文也不打算讨论这些。

正如我已经多次强调的：我在这里讨论的，仅仅是这些模型在当前状态下是否实际有用。

我的背景简介

通常，我不是那种轻易相信任何事物的人。比如，尽管我经历了十年前信息安全界的加密货币热潮，我却从未参与撰写任何关于区块链的研究论文。我也从未拥有过任何比特币，因为在我看来，它们除了用于赌博和欺诈之外，没有任何实际价值。我一直持怀疑态度，每当有人向我宣称“某项新技术将改变世界”，我的反应总是冷漠。

因此，当第一次有人告诉我人工智能将极大地提升我的工作效率并改变我的日常工作方式时，我同样持保留态度，我的回应是：“见到实际效果我才会信。”

此外，我是一名安全研究员。在过去近十年的工作中，我专注于展示人工智能模型在面对未曾训练过的各种环境时如何彻底失败。我证明了只需对机器学习模型的输入进行轻微扰动，就能使其输出完全错误的结果；或者这些模型往往只是记住了它们训练数据中的特定案例，并在实际应用时简单重复。我深知这些系统的种种局限。

然而，现在我却在这里说，我认为当前的大型语言模型是自互联网问世以来，给我的生产力带来的最大提升。坦白说，如果今天要我在使用互联网和一个最先进的语言模型之间选择一种工具来解决工作中随机一个编程任务，我可能会选择使用语言模型的次数超过一半。

如何利用语言模型

以下是我如何利用大型语言模型（LLMs）来提升工作效率。

你可能对我所述的使用案例不感兴趣，甚至认为它们有些荒谬。也可能这些案例与你的需求不相关，这一点我也能理解。但我只能从个人角度出发。这些使用实例都是我在过去一年里与LLMs的交流记录中直接摘录的。

1、为我开发完整应用

去年，我创建了一个测验，让人们评估GPT-4在处理几个特定任务上的表现。这个测验非常受欢迎，获得了超过一千万的页面浏览量。你可能想不到，我几乎是让GPT-4为我编写了这个应用的全部初始版本。这一过程是通过一连串的问题进行的，我从询问应用的基本架构开始，然后逐步扩展其功能。整个对话长达30,000字，极大地测试了当时的GPT-4模型的能力极限。如果你翻看这些对话，会发现从我描述需求并请求模型完成全部实现的信息，到我请求做出具体修改的信息（如“不要比较平均分数，而是用核密度估计表示它是哪个百分位”），以及我提出一些完全不明确的问题时复制粘贴的错误信息（例如，“绘图错误：numpy.linalg.LinAlgError: singular matrix”），还有我仅仅询问简单单次问题的情况（比如“如何用JavaScript在页面上添加一个iframe，内容是从字符串加载的？”）。

这种方法之所以有效，主要是因为语言模型擅长处理人们已经解决的问题，而这个测验的99%内容只不过是一些基础的HTML和Python后端服务器，这是任何人都可以编写的。这个测验之所以引人关注并受到喜爱，并非因为其背后的技术，而是因为测验内容本身。通过自动化所有重复性的部分，我能轻松地完成这个项目。

实际上，如果没有语言模型的帮助，我可能根本就不会去创建这个测验，因为我不愿意花时间从头编写整个网页应用。尽管我擅长编程！我相信，即使是现有的模型，也足以让大多数人仅凭提问就能解决他们以前无法解决的重要任务。

我还有一些类似的例子以后介绍，其中我让模型为我编写了整个应用，当这些应用发布时，我会明确指出它们是在语言模型的协助下完成的。

2、作为新技术的向导

我曾经总能赶上各种新兴框架的步伐。但是一个人的时间毕竟有限，因为我的职业特性，我大多数时间都在跟进最新的研究进展，而不是最新的JavaScript框架。

这意味着当我开始一个不属于我研究领域的新项目时，我通常有两个选择：一是利用我已知的技术，虽然这些技术可能已经过时十年或二十年，但对于小项目来说通常足够了；二是尝试学习新的（通常也是更好的）方法。

这就是语言模型派上用场的时候。对我而言较新的框架或工具，比如Docker、Flexbox或React，对其他人来说可能已经相当熟悉了。世界上可能有成千上万的人已经非常了解这些技术。当前的语言模型也能做到这一点。

这意味着，我可以通过与语言模型的交互式学习，来掌握解决任务所需的任何知识，而不必依赖那些假设读者具备特定知识、目标明确的静态教程。

比如，今年早些时候，我在构建一个LLM评估框架时，希望能在一个受限环境中运行由LLM生成的代码，以避免它删除我的计算机上的随机文件等问题。Docker是完成这一任务的理想工具，但我之前从未使用过。

重要的是，这个项目的目标并非是要学会使用Docker，Docker只是我实现目标所需的工具。我只需要掌握Docker的10%，确保我能以最基本的方式安全使用它。

如果是在90年代，我可能需要购买一本关于Docker的书籍，从头开始学习，阅读前几章，然后试图跳读以找出如何实现我想要的功能。过去十年情况有所改善，我可能会在线搜索如何使用Docker的教程，跟着操作，然后搜索遇到的错误信息，看是否有人遇到相同的问题。

但在今天，我只需要请一个语言模型教我使用Docker。

3、开始新项目

回想起来，我最初接触的编程语言是Java。我非常喜欢编程，但有一件事我绝对讨厌：面对新项目的空白屏幕。特别是在使用Java时！甚至仅仅是让程序编译一个hello world程序——这个“public static void main string args”到底是做什么的？括号应该怎么放？哪些字母应该大写？为什么这里用花括号，那里用方括号？

所以我做了任何孩子都会做的事——我让我父亲帮我做了。

二十年过去了，我仍然不喜欢使用我不熟悉的框架开始新项目。仅仅是为了搞定样板代码就需要花费大量时间，而且我对我在做的事情毫无头绪。

例如，我最近想尝试编写一些CUDA代码，来评估一种简单的贪婪搜索在GPU上的性能与某人的高效优化的CPU实现相比。

但我不懂CUDA编程。我会写C语言，了解GPU的工作原理、内核的功能以及内存布局等，但实际编写向GPU发送任务的代码？我不知道该从哪里开始。因此，我直接请求模型为我编写CUDA程序的初稿。完美吗？当然不是！但这是一个起点。这正是我需要的。你会注意到这里的代码有很多错误！实际上，我完全可以接受。我不是在寻找完美的解决方案，而是一个起点，我可以从那里继续。如果未来的模型能做得更好，那将是惊人的。但我现在拥有的已经大有帮助了。

另一方面，对于我在家进行的一些其他个人项目，我正在使用一个树莓派Pico W。这是我第一次使用它。我希望它能为我做一些事情，特别是一些网络相关的事情。再次，我确信我可以在网上找到有人描述如何做我想做的事情的好教程。但你最近看过互联网吗？前5个结果通常只是一些垃圾内容农场，它们有2008年的有缺陷的代码，仅为了搜索引擎优化而更新，但仍然不起作用。

因此，我直接请求一个语言模型教我如何做我想做的事情。我之前已经和微控制器打过交道，所以我或多或少了解它们的工作方式。但我之前从未使用过Pico W。我只需要一些帮助来开始处理所有的依赖关系，然后我可以弄清楚剩下的部分。

我为新的微控制器编写的第一个“hello world”程序总是让一个LED闪烁。这可以让我测试我是否能够编译并上传代码到设备，是否正确设置了所有的引脚，并且基本上知道我在做什么。所以，让我们只是请求一个闪烁程序。（再次：这在互联网上存在吗？几乎可以肯定。但那样我就得去搜索它了。）一旦我有了这段代码并且运行起来，从这里我就知道该怎么做了。我知道python是如何工作的（信不信由你！）。因此，我可以从那里直接继续编辑东西，把特殊的Micro Python的东西处理掉。

当我遇到另一个需要我特别处理的问题时，我可以再次请求模型帮助我。例如，我接着只是让模型为我编写一个连接到wifi的脚本。

4、代码简化

作为安全研究员，我经常需要处理别人的研究项目，这些项目包含数千行代码，我必须弄懂它们的工作原理以便进行攻击。这听起来并不困难，如果每个人都编写清晰的代码，实际上也确实不该困难，但现实世界并非如此。研究人员通常没有动力发布整洁的代码。所以，人们往往会发布他们能用的任何杂乱代码。（我自己也不例外。）

我无法分享与研究相关的例子，但我可以分享我正在进行的一个个人项目的例子。据说我对康威的生命游戏有种不健康的痴迷。最近，我试图找到一种快速的方法，通过Python评估一些生命游戏模式。有一个很好的C++工具叫golly可以做这个，但我不想将我的Python代码重写为C++。

golly有一个CLI工具正好满足我的需求——我所需要的只是一种正确调用它的方法。这的第一步是简化支持约50种不同命令行选项的C++代码，让它只执行我想要的操作。所以我将所有500行的C++代码输入到LLM中，请求一个更简短的文件来完成相同的任务。

你知道吗？这完美地奏效了。然后，我请求一个围绕C++代码的Python封装器。这同样有效。这是那些令人烦恼的任务之一，如果由我来做，我可能永远不会完成。但现在我可以请求别人帮我完成，我得到的东西比我原来的Python代码快了100倍。

我发现自己经常这样做。这里还有一个例子，我在用Python做同样的事情。

再说一次，这些任务并不复杂。但每次这样做，我都能节省大量时间。这就是我认为大型语言模型如今惊人的原因之一：它们的用途可能不够光鲜，也不会因为说“这是我用大型语言模型简化日常工作的平凡方式”而赢得网络赞誉，但这是实实在在的帮助。

5、处理单调任务

我必须完成的许多任务都是单调无聊的，不需要太多思考，但却必须做。

事实上，我发现自己之所以会拖延任务，往往是因为我知道完成这些任务会感到无聊和痛苦。LLMs极大地减轻了这种痛苦，让我在开始任务时就知道自己只需要解决有趣的问题。因此，我想逐一介绍一些通过请求LLMs帮忙而解决的非常普通的问题。

例如，最近我需要拆解一个用Python 3.9编写的程序。大多数Python反编译器只支持到Python 3.7，但无法在我处理的3.9版本上运行。

反编译实际上并不难，主要是在重建控制流程时避免出错。因此，我没有亲自花时间为几千个操作码和几百行代码进行这种转换，而是让LLM帮我做了。它做得非常好！效果远超我的预期。这里有三个不同的对话，在这些对话中我让模型帮我完成了这项工作。另一个例子是，当我需要将一些非结构化数据转换成结构化格式时。比如，我在做一个项目，需要列出一些书名和作者名。我在网上找到了一些非结构化格式的数据，然后让LLM帮我格式化。最近，我在写一篇关于如何破解某个防护的博客文章，想展示我不得不修改的代码完整差异。于是我粘贴了差异和之前如何将差异转换为HTML的示例，并让LLM按照之前的格式为我生成差异。此外，作为我的工作的一部分，我经常需要为我使用的资源生成引用。谷歌学术很容易引用论文，我可以直接复制粘贴。但引用网页稍显麻烦；我最近开始请求LLM帮我生成引用。（确保这是正确的！）我可以继续举出至少一百个类似的例子。但我想你已经明白了我的意思。

我完全明白这种任务可能会让人觉得“就这？”但我们要记住，五年前这些模型几乎不能连贯地写出一个段落，更别说为你解决整个问题了。

6、使每个用户都成为“高级用户”

如果你曾经看过比你不那么熟练的人使用某个工具，这可能会有点让人难受。他们可能会花费几分钟甚至几小时的时间去完成一个本可以通过某种宏或在处理特定任务时巧妙使用并行应用程序来自动化的任务。

然而，学习执行这些操作所需的技巧需要时间，并且具有挑战性。

例如，我最近试图编写一个Python程序来处理来自Apple Lisa键盘的输入。我在线找到了一个人用C语言编写的相关代码，其中包含许多像#define KEYNAME key_code这样的语句，我想将它们转换为一个Python字典，将整数代码映射到对应的字符串。

我是一个Emacs用户。我知道如何在Emacs中解决这个问题，这甚至并不难。这是我刚记录的一些关键操作，可以达到这个效果：

C-h C-s #def [enter] M-f [delete] C-d M-f C-[space] M-f C-w C-a C-y : " M-f ", C-g C-] } C-[{

尽管这对我几乎是自然的，但到目前为止，我已经花了超过一半的生活时间在Emacs上变得足够熟练，以至于这成为了自然反应。但你知道现在我连接了LLM到我的编辑器，我会怎么做吗？C-h C-h 请将这些#define重写为{keycode: string, ...}的字典格式。

然后，文本就在我眼前被重写了！

正是在这样的情况下，我认为LLMs对非专家的潜在效用甚至高于专家。这个模型为每个人提高了起点，如果你之前完全不会做，现在突然能做很多事情。

7、作为API参考

真正的程序员想要理解某个工具的工作原理时会阅读参考手册。但我是个懒惰的程序员；我更愿意直接得到答案。因此，现在我向语言模型提问。

当我向人们展示这些例子时，有些人会变得有些防御性，他们说：“LLM没有做任何你不能用已有工具完成的事情！”你知道吗？他们说得对。但是，用搜索引擎能做的事，用一本关于该主题的实体书也能做；用一本实体书能做的事，通过阅读源代码也能做。

然而，每一种方式比前一种都简单。当事情变得更简单时，你会更频繁地做，而且方式上也会有所不同。

这就是我问“哪个$命令可以传递所有剩余参数”并获得答案的例子。（紧接着是另一个“我该如何使用这个东西”的问题！）这实际上是我最常用LLMs的方法之一。我之所以不能给你展示更多这样的例子，是因为我在Emacs和我的shell中都内置了查询LLMs的工具。因此，当我想要做这些事情的90%的时间，我甚至不需要离开我的编辑器。

8、搜索难以找到的内容

在互联网上搜索内容曾经是一项需要学习的技能。你想在查询中包含哪些特定的词汇？它们应该是复数还是单数？过去时？你希望避免在页面上出现哪些词汇？我是想要X和Y，还是X或Y？

现在情况已经不同了。我想不起上次我在Google中使用OR的时间。我也想不起上次我使用减号(-)来移除结果子集的时间。在大多数情况下，今天你只需要写下你想要找的内容，搜索引擎就会为你找到。

但搜索引擎仍然不是100%的自然语言查询。它仍然有点像你在玩反向危险边缘游戏，试图使用答案中会有的关键词而不是问题。这是一项我认为我们几乎都忘记了我们学过的技能。

对于今天的一些简单任务（随着时间的推移会越来越多），语言模型只是更好。我可以直接输入“所以我知道+对应于__add__，但是是什么”，它会告诉我答案是__inv__。这是用标准搜索引擎很难搜索到的东西。是的，我知道有方法可以问，这样我就能找到答案。可能如果我输入“python文档元类"add"”，我可以搜索页面上的并得到答案。但你知道还有什么有效吗？只要问LLM你的问题。

这样做任何一次只节省几十秒的时间，但当你在解决某个编码任务的过程中，已经试图同时记住一百万件事时，能够将你试图解决的问题倾倒出来并得到一个连贯的答案是令人惊讶的。

这并不是说他们今天在这方面已经完美。语言模型只有在线上被足够频繁地重复时才知道事情。"足够频繁"是什么意思取决于模型，所以我确实需要花一些精力思考我是应该询问模型还是询问互联网。但模型只会变得更好。

或者，每当我遇到随机崩溃时，我会将模型转储我所看到的并要求解释，就像我在这里做的那样，当我只是输入zsh没有找到匹配的“远程通配符传输问题”。或者，作为一个完全独立的例子，我去年在写一篇博客文章时，想要第一个字母大写，让其余的文字环绕它，就像我在这句话中所做的那样。现在这被称为下沉式大写字母。但我不知道这个。我只知道我想要的效

果，所以问语言模型“我想让它看起来像一本华丽的书，文字围绕O”，它给了我我想要的东西：这个任务是另一个属于“我只是因为LLMs才做的”类别——我不会认为花大量时间弄清楚如何做是值得的。但因为我可以直接问模型，我就这样做了，它让我的帖子看起来更好一些。

9、解决一次性任务

有两种类型的程序。首先，你有一些你想要正确完成的程序；它们将会存在一段时间，因为你需要维护它们好几年，所以代码的清晰性很重要。然后，你有那些只存在大约25秒的程序；它们将帮助你完成某些任务，然后立即被丢弃。

在这些情况下，我根本不关心代码的质量，而且程序是完全独立的，我现在几乎专门使用LLMs来为我编写这些程序。

请注意：大多数这些情况再次出现，你会看着它们说“就这？”。但就像我之前说的，我每天只有那么多小时来处理一个项目。如果我能节省编写一次性使用的程序的时间和精力，我会选择这样做。

可能最常见的例子是帮助我生成一些图表，以可视化我作为某些研究实验结果生成的一些数据。我有几十个这样的例子。可能接近一百而不是零。它们看起来基本上都一样，这里只是一个例子：或者，另一个类似的例子，当我有一种格式的数据并想将其转换为另一种格式的数据时。通常这是我只需要做一次的事情，一旦完成，我就会扔掉生成的脚本。但我还可以给你举出一千个其他例子。通常情况下，当我有一个足够简单的脚本想要编写时，我会直接请求LLM整体编写。例如，这里我请求LLM为我编写一个脚本，让它大声读出我的论文，这样我可以确保它们没有愚蠢的语法问题。在许多情况下，当我不太清楚我想要什么时，我也会从请求模型提供一些初始代码开始，然后从那里进行迭代。例如，这里有一个一次性任务，我只是需要快速处理一些数据。在2022年，我会花两分钟用python编写这个，然后等待几个小时因为它只运行一次——优化它所需的时间会比python程序运行的时间还要长。但现在呢？你敢打赌我会花同样的两分钟请求用Rust代码为我处理数据。或者这里是另一个例子，我请求模型为我下载一个数据集，并对其进行一些初始处理。这对我来说容易做吗？可能是。但这不是我想要考虑的任务；我想要考虑的是我将要用数据集做的研究。消除分心非常有价值，不仅仅是节省几分钟的时间。

另一次，我编写了一个程序，这样我就可以用小立方体3D打印一些像素化的图像。为此，我想将PNG转换为STL文件；但这不是项目的重点。这只是沿途必须发生的事情。所以我请求LLM为我解决这个问题。或者，作为另一个例子，我最近想使用Docker Compose来设置一个新项目。我遇到了一些问题，只想让它运行起来，然后我会弄清楚出了什么问题。所以我只是来回几次，我所做的就是复制一个又一个错误消息，直到它最终给我一个有效的解决方案。我还会发现自己在很多情况下首先请求一个完整的解决方案，然后请求如何修改它的提示。在这次对话中，我首先请求一个解析HTML的程序，然后请求API参考或其他改进方式的提示。最近我一直在尝试做一些电子产品相关的事情，我有一个在Arduino上运行的C程序，但我希望它在树莓派Pico上以MicroPython运行。这个转换过程没有什么有趣的；它只需要完成。所以我没有亲自完成工作，只是请语言模型为我做。对于另一个项目，我需要用一些花哨的

ML模型在一些交互循环中分类一些图像。我本可以自己编写，或者我可以直接请求模型为我做。

10、解释事物给我听

我最近开始对电子学产生了兴趣。我年轻的时候做过一些电子项目，并在大学期间上过几门相关课程。但现在我想进行实际的电子项目，我发现有许多我不了解的细节，这让开始任何项目都变得困难。

我可以去读一本关于实用电子学的书。我可能真的会在某个时候这么做，以便彻底理解这个主题。但我并不想把我的时间花在感觉自己在学习上。我从事电子学的部分原因是想从整天的阅读和写作中抽身而出。

这就是LLMs发挥出色的地方。它们可能不如世界上最出色的专家那样知识渊博，但成千上万的人可能知道我可能会提出的任何电子问题的答案。这意味着语言模型很可能也知道答案。它乐于为我提供所有问题的答案，使我可以享受乐趣而不必纠结于细节。虽然我完全可以通过在互联网上搜索来找到答案，但在忙碌一整天后，简单地让模型为我完成这些工作的便利性使我感到非常放松。

这里是一些例子，展示了我如何询问语言模型有关电子学中事物工作原理的基本问题。这些答案完美吗？谁知道呢。但你知道它们比什么更好吗？比我什么都不知道要好。

11、解决具有已知解决方案的任务

几乎所有事情都已经有人做过了。你想做的事情几乎没有什么是真正新颖的。语言模型特别擅长提供它们之前见过的事物的解决方案。

在最近的一个项目中，我需要提升一些Python代码的性能。因此，我（1）请求LLM将其重写为C语言，然后（2）请求它构建一个接口，以便我能从Python调用C代码。

这些任务都不是“难”的。将Python转换为C是我确信自己能在一两小时内完成的。尽管我不完全知道Python到C的API如何工作，但我相信我可以通过阅读文档来了解。但如果需要我自己来做，我永远不会去做。它不是关键路径上的一部分，我宁愿等待计算机解决问题，也不愿花时间加速那些我不经常需要运行的任务。

但是，将Python转换为C主要是简单程序的技术过程，而且有一个标准的Python到C的调用约定。所以，我直接请求LLM来帮我完成。

从那以后，我开始期待这是我可以做到的事情，几乎在任何时候，当我需要一些高速的代码时，我会用Python描述我想要的内容，并请求生成优化的C代码。其他时候，我做同样的事情，但如果我认为比较Rust输出和C输出的正确性更容易的话，我会请求Rust输出。或者，作为另一个例子，用multiprocessing库并行化Python函数并不困难。你需要编写一些样板代码，基本上它就会为你完成。但编写代码有点痛苦，会妨碍你想要完成的实际工作。现在，每当我需要做这个时，我只会请求一个LLM来帮我。还有很多时候，当我尝试测试某个API时，

我最初会编写一个curl请求来开始。一旦它开始工作，我想以编程方式重复任务，我就会将它转换为Python。以前，我通常会做些非常丑陋的事情，直接调用os.popen()运行curl命令，但这并不理想。更好的方法是将其转换为Python的requests库；但这需要时间，所以我不会这么做。但现在我可以简单地请求一个LLM帮我完成，并在更短的时间内获得一个更干净的程序。对于即将到来的一个项目，我可能会在这里讨论，我需要了解人们通常用作简单无线电发射器的是哪些东西。因为我真正想要的是大多数人的答案，LLM是一个完美的选择！

12、修复常见错误

在2022年之前，当我遇到某些流行工具或库的错误信息时，我通常会采取以下步骤：

1. 复制错误信息。
2. 将其粘贴到Google搜索。
3. 点击最顶部的Stack Overflow链接。
4. 确认问题是否与我遇到的一致；如果不是，返回第2步。
5. 如果无效，返回第2步，更换搜索词，祈祷等。

因为，坦白说，通常出问题的工具与我最终想要解决的任务差距很大，我其实并不太关心如何使其工作，我只需要它能工作。2024年现在这个流程是怎样的？

1. 复制错误信息。
2. 向LLM询问：“我该如何修复这个错误？（错误）”
3. 如果无效，反馈“那不起作用”。

我没有任何对话记录来展示这些例子。（或者说，我找了一个小时也没有找到。）但这实际上有一个很好的原因：我已经将它整合进我的工作流程中了。

我是一个Emacs用户。我设置了我的环境，每当我运行一个程序并且它以非零状态码结束时（意味着有错误发生），它会自动调用最新最快的LLM，并要求它解释答案，同时请求一个可以直接应用来修复代码中bug的补丁。

现今的模型大多数情况下还不足以在这项任务上胜过我，但它们正在逐渐进步。偶尔，当LLM修复了一个我知道如果自己追踪可能非常困难的错误时，我会感到非常惊喜，尤其是当错误只是因为某处的小笔误。

13、以及无数其他事情

我上面提到的所有对话只占我过去一年与LLMs互动总数的不到2%。我没有提供其他链接的原因并不是因为这些是模型失败的案例（尽管有很多这样的案例），而是因为：（1）许多互动重复了我已经提到的那些模式，或（2）它们不容易解释清楚发生了什么，也不容易让你自己看到它们为何对我有用。

我完全预期未来我使用这些模型的频率会持续增长。作为一个参考，我在2024年通过网页界面进行的LLM查询比2023年增加了30%——而且我甚至无法计算API查询的增加，但我猜测至少增加了2到3倍。

评价LLMs的能力，而非它们的局限

我在面试求职者时获得的最佳建议之一是，根据他们能做什么而不是不能做什么来评价他们。

我怀疑我可以问你一些简单的问题，这可能会让你显得不够称职。比如极端的例子：世界上有十亿人讲普通话；我连数到十都做不到。如果有人给我一份小学水平的普通话考试，我肯定会考得很糟糕。

即便在计算机科学领域内，也有我完全不了解的领域。我关于如何构建SQL的知识仅限于如何编写有效的SELECT语句。这——字面上——是我唯一知道如何编写的语句。

因此，当我看到人们在网络上争论说LLMs只是炒作，因为“它们甚至不能做XXX”时，我真的感到困惑。这里的XX可能是：

- ... 计算句子中的单词数！
- ... 写一首每个单词都以字母“a”开头的诗
- ... 乘以两位数
- ... 从列表中随机选择一个元素

因为你上次真正需要做这些事情，并且真诚地认为LLM是合适的工具是什么时候？

我不会因为我们不能在头脑中分割64位整数而认为人类完全无用——这对计算机来说是极其简单的任务——我也不认为因为你可以构造出LLMs无法解决的任务就应该抛弃LLMs。显然这很容易——关键是你能找到它们能够提供价值的任务吗？

程序员已经很清楚，某些事物对不同的目的可能有用。想要编写操作系统？也许你应该使用C而不是Python。没有人会说“看Python多么愚蠢，你甚至不能强制变量对齐到32字节边界！”这只是在错误的抽象级别上。语言模型也是一样。它们在非常高的抽象级别上操作；你不能期待它们解决甚至最简单的程序都能解决的任务。但你可以期望它们解决不同种类的任务。

结论

撰写这篇文章的初衷有两个。首先，正如我在文章开头所述，我想证明LLMs已经为我提供了大量的价值。此外，我注意到很多人表示喜欢使用LLMs的想法，但不知道它们能如何帮助自己。因此，如果你是这些人之一，希望通过我的使用案例能看到一些示例。

因为至少对我来说，LLMs能做很多事。它们不能做所有事情，甚至不能做大多数事情。但当前的模型，正如它们现在存在的样子，为我提供了可观的价值。

在展示这些示例后，我经常收到的一种反驳是：“但这些任务很简单！任何计算机科学的本科生都能学会如何做到！”确实，本科生可以通过几小时的搜索来告诉我如何正确地诊断CUDA错误以及可以重新安装哪些包。本科生可以通过几小时的工作重写那个程序为C语言。本科生可以通过几小时的学习相关教科书来教我任何我想知道的关于那个主题的内容。不幸的是，我没有一个随时可以回答我任何问题的神奇本科生。但我有语言模型。所以，虽然语言模型还不够好，不能解决我作为程序员工作中的有趣部分，而且当前模型只能解决简单的任务。

但五年前，LLM最好的能做的就是写出看起来像英语的段落。当它们能从一个句子到下一个形成连贯的思想时，我们都感到惊讶。它们的实际用途几乎为零。然而，今天，它们已经提高了我在项目中编程方面的生产力至少50%，并消除了足够多的繁琐工作，让我能够创建一些我否则永远不会尝试的东西。

因此，当人们说“LLMs只是炒作”和“所有LLM都没有为任何人提供实际价值”时，很明显他们是错误的，因为它们为我提供了价值。现在，也许我是个例外。也许我是唯一找到了让这些模型有用的方法的人。

我只能代表我自己。

但考虑到LLMs显著提高了我的生产力——一个在使用LLM之前已经有20年编程经验的人——我相信还有其他人也能从使用AI中受益。

原文链接：https://nicholas.carlini.com/writing/2024/how-i-use-ai.html#intro

素材来源官方媒体/网络新闻

对了，喜欢就别忘了点赞、收藏、转发支持一下！期待在评论区听到您对AI观点和看法!

往期回顾

[1、福布斯年度 "AI 50"完整名单，为何这些公司能在人工智能赛道上独树一帜

[2、对话欧洲版支付宝"Klarna"创始人兼CEO：我用一个AI替换700 人客服团体，为公司带来4000万美元的利润

[3、李飞飞领衔Stanford HAI发布全新500页2024人工智能报告，显示超过30%的工作岗位或被AI取代
