AISHWARYA NARESH REGANTI KIRITI BADAM

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Aishwarya Naresh Reganti + Kiriti Badam - 双语对照

Lenny's Podcast: Building AI Products with Aishwarya Naresh Reganti & Kiriti Badam

中英双语完整转录文本 (Bilingual Full Transcript)

[00:00:00] Lenny Rachitsky

English:

We worked on a guest post together. They had this really key insight that building AI products is very different from building non-AI products.

中文翻译:

我们一起合作写了一篇客座文章。他们提出了一个非常关键的见解：构建 AI 产品与构建非 AI 产品有很大的不同。

[00:00:08] Aishwarya Naresh Reganti

English:

Most people tend to ignore the non-determinism. You don't know how the user might behave with your product, and you also don't know how the LLM might respond to that. The second difference is the agency control trade-off. Every time you hand over decision-making capabilities to agentic systems, you're kind of relinquishing some amount of control on your end.

中文翻译:

大多数人往往会忽略“非确定性”（non-determinism）。你不知道用户会如何使用你的产品，你也不知道大语言模型（LLM）会如何回应。第二个区别是“自主性与控制权”（agency control）之间的权衡。每当你将决策能力交给智能体（agentic）系统时，你实际上是在放弃自己的一部分控制权。

[00:00:25] Lenny Rachitsky

English:

This significantly changes the way you should be building product.

中文翻译:

这从根本上改变了你构建产品的方式。

[00:00:28] Kiriti Badam

English:

So we recommend building step-by-step. When you start small, it forces you to think about what is the problem that I'm going to solve. In all this advancements of the AI, one easy, slippery slope is to keep thinking about complexities of the solution and forget the problem that you're trying to solve.

中文翻译:

因此，我们建议循序渐进地构建。当你从小处着手时，它会迫使你思考：我要解决的问题到底是什么？在 AI 的所有这些进步中，一个很容易陷入的误区是：不断思考解决方案的复杂性，却忘记了你最初试图解决的问题。

[00:00:42] Aishwarya Naresh Reganti

English:

It's not about being the first company to have an agent among your competitors. It's about have you built the right flywheels in place so that you can improve over time.

中文翻译:

这不在于你是否是竞争对手中第一家拥有智能体的公司，而在于你是否建立了正确的“飞轮效应”（flywheels），以便能够随着时间的推移不断改进。

[00:00:50] Lenny Rachitsky

English:

What kind of ways of working do you see in companies that build AI products successfully?

中文翻译:

在那些成功构建 AI 产品的公司中，你看到了什么样的协作方式?

[00:00:55] Aishwarya Naresh Reganti

English:

I used to work with the CEO of now Rackspace. He would have this block every day in the morning, which would say catching up with AI 4:00 to 6:00 AM. Leaders have to get back to being hands-on. You must be comfortable with the fact that your intuitions might not be right. And you probably are the dumbest person in the room and you want to learn from everyone.

中文翻译:

我曾与现任 Rackspace 的 CEO 共事。他每天早上都会留出一段固定时间，写着“凌晨 4 点到 6 点跟进 AI 动态”。领导者必须重新亲力亲为。你必须接受你的直觉可能并不正确这一事实。你可能觉得自己是房间里最笨的人，并且渴望向每个人学习。

[00:01:13] Lenny Rachitsky

English:

What do you think the next year of AI is going to look like?

中文翻译:

你认为 AI 的下一年会是什么样子?

[00:01:16] Kiriti Badam

English:

Persistence is extremely valuable. Successful companies right now building in any new area, they are going through the pain of learning this, implementing this and understanding what works and what doesn't work. Pain is the new moat.

中文翻译:

坚持是非常宝贵的。目前在任何新领域取得成功的公司，都在经历学习、实施以及理解哪些行得通、哪些行不通的痛苦过程。“痛苦”就是新的护城河。

[00:01:29] Lenny Rachitsky

English:

Today, my guests are Aishwarya Reganti and Kiriti Badam. Kiriti works on Kodex at OpenAI and has spent the last decade building AI and ML infrastructure at Google and at Kumo. Ash was an early AI researcher at Alexa and Microsoft and has published over 35 research papers. Together, they've led and supported over 50 AI product deployments across companies like Amazon, Databricks, OpenAI, Google, and both startups and large enterprises. Together, they also teach the number one rated AI course on Maven, where they teach product leaders all of the key lessons they've learned about building successful AI products. The goal of this episode is to save you and your team a lot of pain and suffering and wasted time trying to build your AI product. Whether you are already struggling to make your product work or want to avoid that struggle, this episode is for you. If you enjoy this podcast, don't forget to subscribe and follow to your favorite podcasting app or YouTube.

(00:02:22):

It helps tremendously. And if you become an annual subscriber of my newsletter, you get a year free of a ton of incredible products, including a year free of Lovable, Replit, Bold, Gamma, NA, and Linear Dev and Posttalk, Superhuman, Descript, Whisper Flow, Perplexity, Warp, Granola, Magic [inaudible 00:02:38] Mobbin, and Stripe Atlas. Head on over to lennysnewsletter.com and click product pass. With that, I bring you Aishwarya, Reganti, and Kiriti Badam after a short word from our sponsors.

中文翻译:

今天，我的嘉宾是 Aishwarya Reganti 和 Kiriti Badam。Kiriti 目前在 OpenAI 负责 Kodex 项目，过去十年间曾在 Google 和 Kumo 构建 AI 和机器学习（ML）基础设施。Ash 曾是 Alexa 和微软的早期 AI 研究员，发表了 35 多篇研究论文。他们共同领导并支持了 50 多个 AI 产品的部署，涵盖了亚马逊、Databricks、OpenAI、Google 等公司，以及初创企业和大型企业。他们还在 Maven 上共同教授排名第一的 AI 课程，向产品负责人传授构建成功 AI 产品的关键经验。本集的目的是为你和你的团队节省在构建 AI 产品时可能遇到的痛苦、折磨和浪费的时间。无论你是在努力让产品运转，还是想避免这些困难，这一集都非常适合你。如果你喜欢这个播客，别忘了在你的播客应用或 YouTube 上订阅和关注。

(00:02:22):

这对我们帮助很大。如果你成为我时事通讯的年度订阅者，你将免费获得一年的一系列优秀产品，包括 Lovable、Replit、Bold、Gamma、NA、Linear Dev、Posttalk、Superhuman、Descript、Whisper Flow、Perplexity、Warp、Granola、Magic、Mobbin 和 Stripe Atlas。请访问 lennysnewsletter.com 并点击“产品通行证”（product pass）。下面，在听完赞助商的简短介绍后，我们将请出 Aishwarya Reganti 和 Kiriti Badam。

[00:02:49] Lenny Rachitsky (Sponsor Message)

English:

This episode is brought to you by Merge. Product leaders hate building integrations. They're messy. They're slow to build. They're a huge drain on your roadmap, and they're definitely not why you got into product in the first place. Lucky for you, Merge is obsessed with integrations. With a single API, B2B SaaS companies embed Merge into their product and ship 220 plus customer-facing integrations in weeks, not quarters.

(00:03:14):

Think of merge like Plaid, but for everything B2B SaaS. Companies like Mistral AI, Ramp, and Drata use Merge to connect their customers as accounting, HR, ticketing, CRM, and file storage systems to power everything from automatic onboarding to AI-ready data pipelines. Even better, Merge now supports the secure deployment of connectors to AI agents with a new product so that you can safely power AI workflows with real customer data. If your product needs customer data from dozens of systems, Merge is the fastest, safest way to get it. Book and attend a meeting at merge.dev/lenny, and they'll send you a $50 Amazon gift card. That's merge.dev/lenny.

中文翻译:

本集由 Merge 赞助。产品负责人讨厌构建集成。它们很乱，构建速度慢，会大量消耗你的路线图资源，而且这绝对不是你进入产品领域的初衷。幸运的是，Merge 对集成非常痴迷。通过单一 API，B2B SaaS 公司可以将 Merge 嵌入其产品中，并在几周（而非几个季度）内交付 220 多个面向客户的集成。

(00:03:14):

把 Merge 想象成 B2B SaaS 领域的 Plaid。像 Mistral AI、Ramp 和 Drata 这样的公司使用 Merge 来连接客户的会计、人力资源、工单、CRM 和文件存储系统，从而驱动从自动入职到 AI 就绪的数据管道等一切功能。更棒的是，Merge 现在通过新产品支持向 AI 智能体安全部署连接器，这样你就可以安全地利用真实的客户数据驱动 AI 工作流。如果你的产品需要来自数十个系统的客户数据，Merge 是最快、最安全的选择。在 merge.dev/lenny 预约并参加会议，他们会送你一张 50 美元的亚马逊礼品卡。地址是 merge.dev/lenny。

[00:03:59] Lenny Rachitsky (Sponsor Message)

English:

This episode is brought to you by Strella, the customer research platform built for the AI era. Here's the truth about user research. It's never been more important or more painful. Teams want to understand why customers do what they do, but recruiting users running interviews and analyzing insights takes weeks.

(00:04:14):

By the time the results are in, the moment to act has passed. Strella changes that. It's the first platform that uses AI to run and analyze in depth interviews automatically, bringing fast and continuous user research to every team. Strella's AI moderator asks real follow-up questions, probing deeper when answers are vague, and services patterns across hundreds of conversations all in a few hours, not weeks. Product, design, and research teams at companies like Amazon and Duolingo are already using Strella for Figma prototype testing, concept validation, and customer journey research, getting insights overnight instead of waiting for the next sprint. If your team wants to understand customers at the speed you ship products, try Strella. Run your next study at strella.io/lenny. That's S-T-R-E-L-L-A.io/lenny.

中文翻译:

本集由 Strella 赞助，这是为 AI 时代构建的客户研究平台。关于用户研究的真相是：它从未像现在这样重要，也从未像现在这样痛苦。团队想要了解客户行为背后的原因，但招募用户、进行访谈和分析见解需要数周时间。

(00:04:14):

等到结果出来时，行动的最佳时机已经过去了。Strella 改变了这一点。它是第一个利用 AI 自动进行和分析深度访谈的平台，为每个团队带来快速且持续的用户研究。Strella 的 AI 主持人会提出真实的后续问题，在回答模糊时深入追问，并在几小时内（而非几周）从数百场对话中总结出模式。亚马逊和 Duolingo 等公司的产品、设计和研究团队已经在使用 Strella 进行 Figma 原型测试、概念验证和客户旅程研究，一夜之间就能获得见解，而无需等待下一个冲刺周期。如果你的团队想以交付产品的速度了解客户，请尝试 Strella。在 strella.io/lenny 开启你的下一项研究。

[00:05:08] Lenny Rachitsky

English:

Ash and Kiriti, thank you so much for being here and welcome to the podcast.

中文翻译:

Ash 和 Kiriti，非常感谢你们来到这里，欢迎参加我们的播客。

[00:05:13] Aishwarya Naresh Reganti

English:

Thank you, Lenny.

中文翻译:

谢谢你，Lenny。

[00:05:14] Kiriti Badam

English:

Thank you for having us. Super excited for this.

中文翻译:

谢谢你的邀请。非常期待这次交流。

[00:05:16] Lenny Rachitsky

English:

Let me set the stage for the conversation that we're going to have today. So you two have built a bunch of AI products yourself. You've gone deep with a lot of companies who have built AI products, have struggled to build AI products, build AI agents. You also teach a course on building AI products successfully and you're kind of on this mission to just reduce pain and suffering and failure that you constantly see people go through when they're building AI products. So to set a little just foundation for the conversation we're going to have, what are you seeing on the ground within companies trying to build AI products? What's going well? What's not going well?

中文翻译:

让我为今天的对话做个铺垫。你们两位亲自构建过许多 AI 产品。你们深入接触过许多正在构建 AI 产品、在构建过程中挣扎、或者正在构建 AI 智能体的公司。你们还教授一门关于成功构建 AI 产品的课程，你们的使命似乎就是减少人们在构建 AI 产品时经常经历的痛苦、折磨和失败。为了给我们的对话打下基础，你们在那些尝试构建 AI 产品的公司中看到了什么现状？哪些进展顺利？哪些不太顺利？

[00:05:54] Aishwarya Naresh Reganti

English:

I think 2025 has been significantly different than 2024. One, the skepticism has significantly reduced. There were tons of leaders last year who probably thought this would be yet another crypto wave and kind of skeptical to get started. And a lot of the use cases that I saw last year were more of slap chat on your data. And that was calling themselves an AI product. And this year, a ton of companies are really rethinking their user experiences and their workflows and all of that and really understanding that you need to deconstruct and reconstruct your processes in order to build successful AI products. And that's the good stuff. The bad stuff is the execution is still all over the place. Think of it. This is a three-year-old field. There are no playbooks, there are no textbooks. So you really need to figure out as you go. And the AI lifecycle, both pre-deployment and post-deployment is very different as compared to a traditional software lifecycle.

(00:06:57):

And so a lot of old contracts and handoffs between traditional roles, like say PMs and engineers and data folks has now been broken and people are really getting adapted to this new way of working together and kind of owning the same feedback loop in a way. Because previously, I feel like PMs and engineers and all of these folks had their own feedback loops to optimize. And now you need to be probably sitting in the same room. You're probably looking at agent traces together and deciding how your product should behave. So it's a tighter form of collaboration. So companies are still kind of figuring that out. That's kind of what I see in my consulting practice this year.

中文翻译:

我认为 2025 年与 2024 年有很大不同。首先，怀疑态度显著减少了。去年有很多领导者可能认为这只是又一波类似加密货币的浪潮，对开始行动持怀疑态度。去年我看到的很多用例更多是“在你的数据上套个聊天框”，然后就自称是 AI 产品。而今年，大量公司正在真正重新思考他们的用户体验、工作流等等，并真正意识到你需要解构并重构你的流程，才能构建成功的 AI 产品。这是好的方面。不好的方面是执行仍然很混乱。想想看，这是一个只有三年的领域。没有现成的剧本，没有教科书。所以你真的需要边做边摸索。而且 AI 的生命周期，无论是部署前还是部署后，与传统软件生命周期相比都非常不同。

(00:06:57):

因此，传统角色（比如 PM、工程师和数据人员）之间的许多旧契约和交接流程现在都被打破了，人们正在适应这种新的协作方式，在某种程度上共同拥有同一个反馈回路。因为以前，我觉得 PM、工程师和所有这些人都有各自需要优化的反馈回路。而现在，你们可能需要坐在同一个房间里，一起查看智能体的运行轨迹（agent traces），共同决定产品应该如何表现。这是一种更紧密的协作形式。所以公司们还在摸索中。这就是我今年在咨询实践中看到的。

[00:07:37] Lenny Rachitsky

English:

So let me follow that thread. We worked on a guest post together that came out a few months ago. And the thing that stood out to me most that stuck with me most after working on that post is this really key insight that building AI products is very different from building non-AI products. And the thing that you're big on getting across is there's two very big differences. Talk about those two differences.

中文翻译:

让我顺着这个思路继续。我们几个月前合作发表了一篇客座文章。在完成那篇文章后，最让我印象深刻、最挥之不去的是那个关键见解：构建 AI 产品与构建非 AI 产品非常不同。你们非常强调要传达两个巨大的差异。谈谈这两个差异吧。

[00:08:01] Aishwarya Naresh Reganti

English:

Yes. And again, I want to make sure that we drive home the right point. There are tons of similarities of building AI systems and software systems as well, but then there are some things that kind of fundamentally change the way you build software systems versus AI systems. And one of them that most people tend to ignore is the non-determinism. You're pretty much working with a non-deterministic API as compared to traditional software. What does that mean and why does that have to affect us is in traditional software, you pretty much have a very well-mapped decision engine or workflow. Think of something like Booking.com. You have an intention that you want to make a booking in San Francisco for two nights, et cetera. The product has kind of been built so that your intention can be converted into a particular action and you kind of are clicking through a bunch of buttons, options, forms, and all of that, and you finally achieve your intention.

(00:08:59):

But now that layer in AI products has completely been replaced by a very fluid interface, which is mostly natural language, which means the user can literally come up with a ton of ways of saying or communicating their intentions. And that kind of changes a lot of things because now you don't know how your user's going to be here. That's on the input side. And the output is also that you're working with a non-deterministic probabilistic API, which is your LLM. And LLMs are pretty sensitive to prompt phrasings and they're pretty much black boxes. So you don't even know how the output surface will look like. So you don't know how the user might behave with your product, and you also don't know how the LLM might respond to that. So you're now working with an input, output, and a process. You don't understand all the three very well. You're trying to anticipate behavior and build for it.

(00:09:53):

And with agentic systems, this kind of gets even harder. And that's where we talk about the second difference, which is the agency control trade-off. What we mean by that, and I'm kind of shocked so many people don't talk about this. They're extremely obsessed with building autonomous systems, agents that can do work for you. But every time you hand over decision-making capabilities or autonomy to agentic systems, you're kind of relinquishing some amount of control on your end. And when you do that, you want to make sure that your agent has gained your trust or it is reliable enough that you can allow it to make decisions. And that's where we talk about this agency controlled trade-off, which is if you give your AI agent or your AI system, whatever it is, more agency, which is the ability to make decisions, you're also losing some control and you want to make sure that the agent or the AI system has earned that ability or has built up trust over time.

中文翻译:

是的。再次强调，我想确保我们传达了正确的观点。构建 AI 系统和软件系统有很多相似之处，但有些东西从根本上改变了你构建软件系统与 AI 系统的方式。其中一个大多数人倾向于忽略的是“非确定性”。与传统软件相比，你基本上是在处理一个非确定性的 API。这意味着什么，为什么会影响我们？在传统软件中，你通常有一个映射得非常清晰的决策引擎或工作流。想想 Booking.com 之类的网站。你有一个意图，想在旧金山预订两晚住宿等等。产品已经构建好了，你的意图可以转化为特定的动作，你点击一系列按钮、选项、表单，最后实现你的意图。

(00:08:59):

但在 AI 产品中，那一层已经完全被一个非常流动的界面所取代，主要是自然语言，这意味着用户可以用无数种方式来表达或传达他们的意图。这改变了很多事情，因为现在你不知道用户会如何表现。这是在输入端。而在输出端，你面对的是一个非确定性的概率 API，即你的 LLM。LLM 对提示词（prompt）的措辞非常敏感，而且它们基本上是黑匣子。所以你甚至不知道输出界面会是什么样子。你不知道用户会如何与你的产品互动，也不知道 LLM 会如何回应。所以你现在面对的是输入、输出和过程，而你对这三者都不太了解。你是在尝试预测行为并为此构建产品。

(00:09:53):

对于智能体系统，这变得更加困难。这就是我们谈到的第二个区别：自主性与控制权的权衡（agency control trade-off）。令我惊讶的是，竟然没多少人谈论这个。人们极度痴迷于构建自主系统，即能为你工作的智能体。但每当你将决策能力或自主权交给智能体系统时，你就在放弃自己的一部分控制权。当你这样做时，你需要确保你的智能体已经赢得了你的信任，或者它足够可靠，可以让你允许它做决定。这就是我们所说的自主性与控制权的权衡：如果你给 AI 智能体或 AI 系统更多的自主权（即做决定的能力），你也在失去一些控制权，你需要确保该智能体已经赢得了这种能力，或者随着时间的推移建立了信任。

[00:10:49] Lenny Rachitsky

English:

So just to summarize what you're sharing here, essentially, people have been building product, software products for a long time. We're now in a world where the software you're building is one, non-deterministic, can just do things differently. As you said, you go to booking.com, you find a hotel, it's going to be the same experience every time. You'll see different hotels, but it's a predictable experience. With AI, you can't predict that it's going to be the exact same thing, the thing that you plan it to be every time. And then the other is there's this trade-off between agency and control. How much will the AI do for you versus how much should the person still be in charge? And what I'm hearing is the big point here is this significantly changes the way you should be building product. And we're going to talk about the impact on how the product development lifecycle should change as a result.

(00:11:35):

Is there anything else you want to add there before we get into that?

中文翻译:

总结一下你分享的内容：本质上，人们构建软件产品已经很久了。但现在我们处于这样一个世界：你构建的软件首先是非确定性的，它可能会以不同的方式做事。正如你所说，你去 booking.com 找酒店，每次的体验都是一样的。你会看到不同的酒店，但体验是可预测的。而有了 AI，你无法预测它每次都会完全一样，或者完全符

合你的计划。其次是自主性与控制权之间的权衡。AI 应该为你做多少，而人应该在多大程度上保持主导？我听到的大重点是，这显著改变了你构建产品的方式。接下来我们将讨论这如何影响产品开发生命周期的改变。

(00:11:35):

在深入讨论之前，还有什么想补充的吗？

[00:11:39] Kiriti Badam

English:

Yeah, it's definitely one of the key points that this kind of distinction needs to exist in your mind when you're starting to build. For example, think about if your objective is to hike Half Dome in Yosemite. You don't start hiking it every day, but you start training yourself in minor parts and then you slowly improve and then you go to the end goal. I feel like that's extremely similar to what you want to build AI products in the sense that when you don't start with agents with all the tools and all the context that you have in the company in day one and expect it to work or even tinker at that level. You need to be deliberately starting in places where there is minimal impact and more human control so that you have a good grip of what are the current capabilities and what can I do with them and then slowly lean into the more agency and lesser control.

(00:12:29):

So this gives you that confidence that, okay, I can know that, okay, this is the particular problem that I'm facing and the AI can solve this extent of it. And then let me next think through what context I need to bring in, what kind of tools I need to add to this to improve the experience. So I feel like also it's a good and a bad thing in the sense that it's good that you don't have to see the complexity of the outside world of all of this fancy AI agents force and feel like I cannot do that. Everyone is starting from very minimalistic structures and then evolving. And the second part is the bad thing is that as you are trying to build this one click agents into your company, you don't have to be overwhelmed with this complexity. You can slowly graduate.

(00:13:16):

So that's extremely important. And we see this as a repeating pattern over and over.

中文翻译:

是的，这绝对是关键点之一：当你开始构建时，脑子里必须有这种区分。例如，想象你的目标是徒步优胜美地的半圆顶（Half Dome）。你不会第一天就去爬，而是先在一些小的部分训练自己，然后慢慢提高，最后才去挑战终极目标。我觉得构建 AI 产品非常类似，你不能在第一天就给智能体配上公司所有的工具和背景信息，并指望它能正常工作，甚至不能在那个层面上进行修补。你需要有意识地从影响最小、人类控制最多的地方开始，这样你就能很好地掌握当前的能力以及我能用它们做什么，然后慢慢转向更多的自主性和更少的控制。

(00:12:29):

这给了你信心：我知道我面临的是这个特定问题，AI 可以在多大程度上解决它。然后我再思考需要引入什么背景信息，需要添加什么工具来改善体验。所以我觉得这既是好事也是坏事。好事在于，你不需要一开始就面对外部世界那些花哨 AI 智能体的复杂性，并感到“我做不到”。每个人都是从极简结构开始，然后演进的。坏事（或者说挑战）在于，当你试图在公司里构建这种“一键式智能体”时，你不需要被复杂性压垮，你可以慢慢升级。

(00:13:16):

这非常重要。我们看到这是一个不断重复的模式。

[00:13:20] Lenny Rachitsky

English:

Okay. So let's actually follow that because that's a really important component of how you recommend people build AI stuff, AI products, AI agents, all the AI things. So give us an example of what you're talking about here, this idea of starting slow with agency and control and then moving up this rung.

中文翻译:

好，让我们顺着这个话题聊，因为这是你们推荐人们构建 AI 产品、智能体等所有 AI 事物的核心组成部分。请给我们举个例子，说明你所说的“从低自主性和高控制权开始，然后逐步攀升”是什么意思。

[00:13:38] Kiriti Badam

English:

Yeah. For example, a very important or very prevalent application of AI agents is customer support. Imagine you are a company who has a lot of customer support tickets and why even imagine OpenAI is the exact same thing when we were launching products and there was a huge spike of support volume as we launched successful products like Image or GPT-5 and things like that. The kind of questions you get is different. The kind of problems that the customers bring to you is different. So it's not about just dumping all the list of help center articles that you have into the AI agent. You kind of understand what are the things that you can build. And so initially the first step of it would be something like you have your support agents, the human support agents, but you will be suggesting in terms of, okay, this is what the AI thinks that is the right thing to do.

(00:14:33):

And then you get that feedback loop from the humans that, okay, this is actually a good suggestion for me in this particular case and this is a bad suggestion. And then you can go back and understand, okay, this is what the drawbacks are or this is where the blind spots are, and then how do I fix that? And once you get that, you can increase the autonomy to say that, okay, I don't need to suggest to the human. I'll actually show the answer directly to the customer. And then we can actually add more complexity in terms of, okay, I was only answering questions based on health center articles, but now let me add new functionality. I can actually issue refunds to the customers. I can actually raise feature requests with the engineering team and all of these things. So if you start with all of this on day one, it's incredibly hard to control the complexity.

(00:15:19):

So we recommend building step by step and then increasing it.

中文翻译:

好的。例如，AI 智能体一个非常重要且普遍的应用是客户支持。想象你是一家拥有大量客服工单的公司——其实不用想象，OpenAI 在发布 Image 或 GPT-5 等成功产品时就面临完全相同的情况，客服量会激增。你收到的问题类型不同，客户带来的问题也不同。所以，这不仅仅是把所有的帮助中心文章都塞给 AI 智能体。你需要了解你能构建什么。最初的第一步可能是：你仍然有真人客服，但 AI 会提供建议，比如“AI 认为这是正确的做法”。

(00:14:33):

然后你从真人那里获得反馈回路：“在这个特定案例中，这确实是个好建议”，或者“这是个坏建议”。然后你可以回头去理解：缺点在哪里？盲点在哪里？我该如何修复？一旦你掌握了这些，你就可以增加自主性，说：“好吧，我不需要建议给真人了，我会直接向客户展示答案。”接着我们可以增加更多复杂性，比如：“以

前我只是根据帮助中心文章回答问题，现在让我增加新功能，我可以实际为客户办理退款，或者向工程团队提交功能请求”等等。如果你在第一天就尝试做所有这些，控制复杂性将变得极其困难。

(00:15:19):

所以我们建议分步构建，然后逐步增加。

[00:15:23] Lenny Rachitsky

English:

Awesome. And you have a visual actually that we'll share of what this looks like. But just to kind of mirror back what you're describing, this idea of start with high control, low agency, the example you gave is the support agents just kind of giving suggestions, is not able to do anything, the user is in charge. And then as that becomes useful and you are confident it's doing the right sort of work, you give it a little more agency and you kind of pull back on the control the user has. And then if that's starting to go well, then you give it more agency and the user needs less control to control it. Awesome.

中文翻译:

太棒了。你们实际上有一个图表，我们会分享出来展示它的样子。简单复述一下你描述的内容：从高控制、低自主开始，你举的例子是客服智能体只提供建议，不能执行任何操作，由用户（真人客服）主导。当这变得有用，并且你确信它在做正确的工作时，你给它多一点自主权，并收回用户拥有的一部分控制权。如果进展顺利，你就给它更多自主权，用户需要的控制就更少了。太棒了。

[00:16:02] Aishwarya Naresh Reganti

English:

I think the higher level idea here is with AI systems, it's all about behavior calibration. It's incredibly impossible to predict upfront how your system behaves. Now, what do you do about it? You make sure that you don't ruin your customer experience or your end user experience. You keep that as is, but then remove the amount of control that the human has. And there is no single right way of doing it. You can decide how to constrain that autonomy. I mean, a different example of how you could constrain autonomy is pre-authorization use cases. Insurance pre-authorization is a very ripe use case for AI because clinicians spend a lot of time pre-authorizing things like blood tests, MRIs and things like that. And there are some cases which are more of low hanging fruits. For instance, MRIs and block tests, because as soon as you know patient's information, it's easier to approve that and AI could do that versus something like an invasive surgery, et cetera, is more high risk. You don't want to be doing that autonomously.

(00:17:11):

So you can kind of determine which of these use cases should go through that human and the loop layer versus which of the use cases AI can conveniently handle. And then all through this process, you're also logging what the human is doing because you want to build a flywheel that you could use in order to improve your system. So you're essentially not showing the user experience, not eroding trust, at the same time logging what humans would otherwise do so that you can continuously improve your system.

中文翻译:

我认为这里更高层次的理念是：对于 AI 系统，核心在于“行为校准”（behavior calibration）。预先预测系统如何表现几乎是不可能的。那么该怎么办呢？你要确保不破坏客户体验或最终用户体验。保持体验不变，但逐渐减少人类所需的控制量。这没有唯一的正确方法，你可以决定如何约束这种自主性。举个不同的例子，比如预

授权（pre-authorization）用例。保险预授权是 AI 非常成熟的用例，因为临床医生花费大量时间预授权验血、MRI 等项目。有些案例属于“低垂的果实”，例如 MRI 和常规化验，一旦知道患者信息，批准起来相对容易，AI 可以胜任；而像侵入性手术之类的高风险项目，你肯定不想让它自主完成。

(00:17:11):

所以你可以确定哪些用例应该经过“人机回环”（human-in-the-loop）层，哪些用例 AI 可以方便地处理。在整个过程中，你还要记录人类的操作，因为你想建立一个可以用来改进系统的飞轮。所以你本质上是在不损害用户体验、不侵蚀信任的前提下，记录人类在同样情况下会做什么，以便持续改进你的系统。

[00:17:41] Lenny Rachitsky

English:

So let me give you a few more examples of this kind of progression that you recommend. And the reason I'm spending so much time here is this is a really key part of your recommendation to help people build more successful AI products. This idea of start slow with high control and low agency and then build up over time once you've built confidence that it's doing the right sort of work. So a few more examples that you shared in your post that I'll just read. So say you're building a coding assistant, V1 would be just suggest inline completion and boilerplate snippets. V2 would be generate larger blocks like tests or refactors for humans to review. And then V3 is just apply the changes and open PRs autonomously. And then another example is a marketing assistant. So V1 would be draft emails or social copy, just like here's what I would do.

(00:18:26):

V2 is build a multi-step campaign and run the campaign. And V3 is just launch it A/B tested auto-optimize campaigns across channels. Awesome. Yeah. And again, just to summarize where we're at, just to give people the advice we've shared so far. One is just important to understand AI products are different. They're non-deterministic. And you pointed out, and I forgot to actually mirror back this point, both on the input and the output. The user experience is non-deterministic. People will see different things, different outputs, different chat conversations, different maybe UI if it's designing the UI for you. And also the output obviously is going to be non-terministic. So that's a problem and a challenge. And then-

中文翻译:

让我再举几个你们推荐的这种进阶过程的例子。我之所以花这么多时间在这里，是因为这是你们帮助人们构建更成功 AI 产品的核心建议：从高控制、低自主开始，在建立信心确信它能胜任后，再随着时间推移逐步提升。我在你们的文章中读到过几个例子，我来念一下。假设你在构建一个编程助手，V1 版本只是建议行内补全和样板代码片段；V2 版本会生成更大的代码块，如测试或重构，供人类审查；V3 版本则是自主应用更改并开启 PR（拉取请求）。另一个例子是营销助手：V1 是草拟电子邮件或社交文案，就像在说“这是我会做的”；

(00:18:26):

V2 是构建多步骤营销活动并运行；V3 则是直接发布、进行 A/B 测试并跨渠道自动优化营销活动。太棒了。再次总结一下我们目前的进展，给听众一些建议：首先，理解 AI 产品是不同的，它们是非确定性的。你刚才指出（我刚才忘了复述这一点），这种非确定性同时存在于输入和输出端。用户体验是非确定性的——人们会看到不同的东西、不同的输出、不同的聊天对话，如果 AI 为你设计 UI，甚至 UI 也会不同。而且输出显然也是非确定性的。这是一个问题，也是一个挑战。然后——

[00:19:08] Aishwarya Naresh Reganti

English:

I mean, if you think of it's also the most beautiful part of AI, which is, I mean, we are all much more comfortable talking than following a bunch of buttons and all of that. So the bar to using AI products is much lower because you can be as natural as you would be with humans, but that's also the problem, which is there are tons of ways we communicate and you want to make sure that that intent is rightly communicated and the right actions are taken because most of your systems are deterministic and you want to achieve a deterministic outcome, but with non-deterministic technology and that's where it gets a little messy.

中文翻译:

我的意思是，如果你换个角度看，这也是 AI 最美妙的部分。比起点击一堆按钮，我们都更习惯于交谈。所以使用 AI 产品的门槛要低得多，因为你可以像与人交流一样自然。但这也是问题所在：我们沟通的方式有无数种，而你需要确保意图被正确传达并采取正确的行动。因为你的大多数（后端）系统是确定性的，你想通过非确定性的技术实现确定性的结果，这就是事情变得有点棘手的地方。

[00:19:44] Lenny Rachitsky

English:

Awesome. I love the optimistic version of why this is good. Okay. And then the other piece is this idea of this trade-off of autonomy versus control when you're designing a thing. And I imagine what you're seeing is people try to jump to the ideal, like the V3 immediately and that's when they get into trouble both. It's probably a lot harder to build that and it just doesn't work. And then they're just like, "Okay, this is a failure. What are we even doing?"

中文翻译:

太棒了，我喜欢关于这为什么是好事的乐观版本。好，另一部分是设计产品时自主性与控制权之间的权衡。我猜你看到的情况是，人们试图立即跳到理想状态，比如直接做 V3，结果在两方面都遇到了麻烦：构建它可能难得多，而且根本行不通。然后他们就会觉得："好吧，这是一个失败，我们到底在干什么？"

[00:20:08] Kiriti Badam

English:

Exactly. I feel there's a bunch of things that you actually have to get confidence in before you get to V3. And it's easy to get overwhelmed that, oh, my AI agent is doing these things wrong in a hundred different ways and you're not going to actually tabulate all of them and fix it. Even though you've learned how do you deal with the evaluation practices and stuff like that, if you're starting on the wrong spot, you are actually going to have a hard time correcting things from there. And when you start small and when you start with building a very minimalistic version with high human control and low agency, it also forces you to think about what is the problem that I'm going to solve. We use this term called problem first. And to me, it was obvious in the sense that that I do need to think about the problem, but it's incredible how well it resonates with the people that in all this advancements of the AI that we are seeing, one easy, slippery slope is to just keep thinking about complexities of the solution and forget the problem that you're trying to solve.

(00:21:10):

So when you're trying to start at a smaller scale of autonomy, you start to really think about what is the problem that I'm trying to solve and how do I break it down into levels of autonomy that I can build later? So that is incredibly useful and we keep repeating this part and over and over with everyone we talk to.

中文翻译:

没错。我觉得在达到 V3 之前，有很多事情是你必须建立信心的。你很容易感到不知所措，觉得“天哪，我的 AI 智能体有一百种出错的方式”，而你不可能真的把它们全部列出来并一一修复。即使你学会了如何处理评估实践之类的事情，如果你从错误的地方开始，你之后也很难纠正。当你从小处着手，从一个高人类控制、低自主性的极简版本开始时，它也会迫使你思考：我要解决的问题是什么？我们使用“问题优先”（problem first）这个词。对我来说，思考问题是显而易见的，但令人惊讶的是，这个观点在人们心中引起了如此强烈的共鸣。在 AI 的所有进步中，一个很容易陷入的误区就是只顾着思考解决方案的复杂性，而忘记了你试图解决的问题。

(00:21:10):

所以，当你尝试从较小规模的自主性开始时，你会真正开始思考：我要解决的问题是什么？我该如何将其分解为以后可以构建的不同自主等级？这非常有用，我们对每个交谈的人都会反复强调这一点。

[00:21:31] Lenny Rachitsky

English:

And there's so many other benefits to limiting autonomy because there's just danger also of the thing doing too much for you and just messing up your, I don't know, your database, sending out all these emails you never expected. And there's like so many reasons this is a good idea.

中文翻译:

限制自主性还有很多其他好处，因为如果让它为你做太多事，可能会有危险，比如搞乱你的数据库，或者发出你从未预料到的邮件。有很多理由说明这是一个好主意。

[00:21:45] Aishwarya Naresh Reganti

English:

Yep. I recently read this paper from a bunch of folks at UC Berkeley. Basically Matei Zaharia, [inaudible 00:21:54] and the folks at Databricks and it said about 74% or 75% of the enterprises that they had spoken to, their biggest problem was reliability. And that's also why they weren't comfortable deploying products to their end users or building customer facing products because they just weren't sure or they just weren't comfortable doing that and exposing their users to a bunch of these risks. And that's also why they think a lot of AI products today have to do with productivity because it's much low autonomy versus end-to-end agents that would replace workflows. And yeah, I love their work otherwise as well, but I think that's very in line with what at least we are seeing at my startup as well.

中文翻译:

是的。我最近读了加州大学伯克利分校几位学者的一篇论文，主要是 Matei Zaharia 和 Databricks 的团队写的。论文提到，在他们访谈的企业中，约 74% 或 75% 的企业表示最大的问题是可靠性。这也是为什么他们不敢向最终用户部署产品或构建面向客户的产品，因为他们不确定，或者不放心让用户暴露在这些风险中。这也是为什么他们认为当今许多 AI 产品都与生产力有关（因为自主性较低），而不是替换整个工作流的端到端智能体。我很喜欢他们的研究，这与我们在我的初创公司所看到的情况非常吻合。

[00:22:38] Lenny Rachitsky

English:

Okay. Very interesting. There's an episode that'll come out before this conversation where we go deep into another problem that this avoids, which is around prompt injection and jailbreaking and just how big of a risk that is for AI products where it's essentially an unsolved and unsolvable problem potentially. I'm not going to go down that track, but that's a pretty scary conversation we had that'll be out before this conversation.

中文翻译:

好，非常有意思。在这次对话之前会发布一集节目，我们会深入探讨这种方法可以避免的另一个问题，即提示词注入（prompt injection）和越狱（jailbreaking）。对于 AI 产品来说，这基本上是一个尚未解决、甚至可能无法解决的问题，风险巨大。我不想在这里展开讨论，但那是一次相当令人担忧的对话，会在本集之前播出。

[00:23:02] Aishwarya Naresh Reganti

English:

I think that will be a huge problem once systems go mainstream. We're still so busy building AI products that we're not worried about security, but it will be such a huge problem to kind of, especially with this non-deterministic API again. So you're kind of stuck because there are tons of instructions that you could inject within your prompt and then it's going really bad.

中文翻译:

我认为一旦系统进入主流，这将是一个巨大的问题。我们现在还忙于构建 AI 产品，顾不上担心安全，但这将成为一个巨大的挑战，尤其是考虑到这种非确定性的 API。你可能会陷入困境，因为提示词中可以注入无数种指令，然后事情就会变得非常糟糕。

[00:23:28] Lenny Rachitsky

English:

Okay. Let's actually spend a little time here because it's actually really interesting to me and no one's talking about this stuff, which is like the conversation we had is just it's pretty easy to get AI to do stuff it shouldn't do. And there's all these guardrail systems people put in place, but turns out these guardrails aren't actually very good and you can always get around them. And to your point, as agents become more autonomous and robots, it gets pretty scary that you could get AI to do things you shouldn't do.

中文翻译:

好，让我们在这里多花一点时间，因为这对我来说真的很有趣，而且似乎没人谈论这些。就像我们之前聊过的，让 AI 做它不该做的事其实很容易。人们设置了各种护栏系统，但事实证明这些护栏并不怎么好用，总能被绕过。正如你所说，随着智能体变得更加自主，甚至变成机器人，如果你能让 AI 做不该做的事，那确实挺可怕的。

[00:23:54] Kiriti Badam

English:

I think this is definitely a problem, but I feel in the current spectrum of customers adopting AI, the extent to which companies can actually get advantage of AI or improve their processes or streamline the existing processes that they have, I feel it's still in the very early stage. 2025 has been an extremely busy year for AI agents and customers trying to adopt AI, but I feel the penetration is still not as much as you would actually get advantage out of it. So with the right sort of human in the loop points in here, I feel we can actually avoid a bunch of these things and focus more towards streamlining the processes. And I am more on the optimist side in the sense that you need to try and adopt this before actually trying to be only for highlighting the negative aspects of what could go wrong.

(00:24:47):

So I feel like strongly that companies has this adopt this, they definitely ... No company at OpenAI we talk to has never had been the case that, oh, AI cannot help me in this case. It has always been that, oh, there is this set of things that it can optimize for me and then let me see how I can adopt it.

中文翻译:

我认为这确实是个问题，但在目前客户采用 AI 的光谱中，公司实际能从 AI 中获益、改进流程或简化现有流程的程度，我觉得仍处于非常早期的阶段。2025 年对 AI 智能体和尝试采用 AI 的客户来说是极其忙碌的一年，但我感觉渗透率还没有达到能充分发挥其优势的程度。因此，通过设置正确的人机回环点，我觉得我们可以避免很多此类问题，并更多地专注于简化流程。我更倾向于乐观派，即你应该先尝试采用它，而不是只盯着可能出错的负面因素。

(00:24:47):

所以我强烈认为公司应该采用它。在 OpenAI，我们接触过的公司中，从来没有一家说“AI 在这种情况下帮不了我”。他们总是说：“哦，有一系列事情它可以帮我优化，让我看看该如何采用它。”

[00:25:06] Lenny Rachitsky

English:

Sweet. I always like the optimistic perspective. I'm excited for you to listen to this and see what you think because it's really interesting. And to your point, there's a lot of things to focus on. It's one of many things to worry about and think about. Okay, let's get back on track here. So we've shared a bunch of pro-tips and important piece of advice. Let me ask, what other patterns and kind of ways of working do you see in companies that do this well and teams that build AI products successfully? And then just what are the most common pitfalls people fall into? So we could just maybe start with, what are other ways that companies do this well, build AI products successfully?

中文翻译:

太棒了，我一向喜欢乐观的视角。我很期待你们听完那一集后的想法，因为那真的很有趣。正如你所说，有很多事情需要关注，这只是众多需要担心和思考的事情之一。好，让我们回到正轨。我们已经分享了一些专业技巧和重要建议。我想问，在那些做得好的公司和成功构建 AI 产品的团队中，你还看到了哪些模式和协作方式？还有，人们最常掉进的坑有哪些？我们可以先从“公司如何成功构建 AI 产品”的其他方式开始。

[00:25:43] Aishwarya Naresh Reganti

English:

I almost think of it as like a success triangle with three dimensions that's never always technical. Every technology problem is a people problem first. And with companies that we have worked with, it's these three dimensions, like great leaders, good culture and technical prowess. With leaders itself, we work with a lot of companies for their AI transformation, training, strategy and stuff like that. And I feel like a lot of companies, the leaders have built intuitions over 10 or 15 years and they're kind of highly regarded for those intuitions. But now with AI in the picture, those intuitions will have to be relearned and leaders have to be vulnerable to do that. I used to work with the CEO of now Rackspace, Gagan. So he would have this block every day in the morning, which would say catching up with AI 4:00 to 6:00 AM, and he would not have any meetings or anything like that.

(00:26:42):

And that was just his time to pick up on the latest AI podcast or information and all of that. And he would have weekend vibe coding sessions and stuff like that. So I think leaders have to get back to being hands-on. And that's not because they have to be implementing these things, but more of rebuilding their intuitions because you must be comfortable with the fact that your intuitions might not be right and you probably are the dumbest person in the room and you want to learn from everyone. And that I've seen that being a very distinguishing factor of companies that build products which are successful because you're kind of bringing in that top-down approach. It's almost always impossible for it to be bottom-up. You can't have a bunch of engineers go and get buy-in from the leader if they just don't trust in the technology or if they have misaligned expectations about the technology.

(00:27:34):

I've heard from so many folks who are building that our leaders just don't understand the extent to which AI can solve a particular problem or they just vibe code something and assume it's easy to take it to production and you really need to understand the range of what AI can solve today so that you can guide decisions within the company. The second one is the culture itself. And again, I work with enterprises where AI is not their main thing and they need to bring in AI into their processes just because a competitor is doing it. And just because it does make sense because there are use cases that are very ripe. Then along the way, I feel a lot of companies have this culture of FOMO and you will be replaced and those kind of things and people get really afraid. Subject matter experts are such a huge part of building AI products that work because you really need to consult them to understand how your AI is behaving or what the ideal behavior should be.

(00:28:27):

But then I've spoken to a bunch of companies where the subject matter experts just don't want to talk to you because they think their job is being replaced. So I mean, again, this comes from the leader itself. You want to build a culture of empowerment, of augmenting AI into your own workflows so that you can 10X at what you're doing instead of saying that probably you'll be replaced if you don't adopt AI and stuff like that. So that kind of an empowering culture always helps. You want to make your entire organization be in it together and make AI work for you instead of trying to guard their own jobs, et cetera. And with AI, it's also true that it opens up a lot more opportunities than before. So you could have your employees doing a lot more things than before and 10x their productivity. And the third one is the technical part which we talk about.

(00:29:18):

I think folks that are successful are incredibly obsessed about understanding their workflows very well and augmenting parts that could be ripe for AI versus the ones that might need human in the loop somewhere, et cetera. Whenever you're trying to automate some part of a workflow, it's never the case that you could use an AI agent and that will solve your problems. It's always, you probably have a machine learning model that's going to do some part of the job. You have deterministic code doing some part of the job. So you really need to be obsessed with understanding that workflow so you can choose the right tool for the problem instead of being obsessed with the technology itself. And another pattern I see is also folks really understand this idea of working with a non-deterministic API, which is your LLM. And what that means is they also understand the AI development lifecycle looks very different and they iterate pretty quickly, which is can I build something iterate quickly in a way that it doesn't ruin my customer experience at the same time gives me enough amount of data so that I can estimate behavior.

(00:30:31):

So they build that flywheel very quickly. As of today, it's not about being the first company to have an agent among your competitors. It's about, have you built the right flywheels in place so that you can improve over time? When someone comes up to me and says, "We have this one click agent, it's going to be deployed in your system." And then in two or three days, it'll start showing you significant gains. I would almost be skeptical because it's just not possible. And that's not because the models aren't there, but because enterprise data and infrastructure is very messy and you need a bit to ... Even the agent needs a bit to understand how these systems work. There are very messy taxonomies everywhere. People tend to do things like get customer data, we want, get customer data, we do, and these kind of things. And all those functions exist and they're being called and basically there's a lot of tech debt that you need to deal with.

(00:31:23):

So most of the times, if you're obsessed with the problem itself and you understand your workflows very well, you will know how to improve your agents over time instead of just slapping an agent and assuming that it'll work from day one. I probably will go as far to say that if someone's selling you one click-agents, it's pure marketing. You don't want to buy into that. I would rather go with a company that says, "We're going to build this pipeline for you," and that will learn over time and build a flywheel to improve than something that's going to work out of the box. To replace any critical workflow or to build something that can give you significant ROI, it easily takes four to six months of work, even if you have the best data layer and infrastructure layer.

中文翻译:

我几乎把它看作一个由三个维度组成的“成功三角形”，而且它绝不仅仅是技术问题。每个技术问题首先都是人的问题。在我们合作过的公司中，这三个维度是：优秀的领导者、良好的文化和技术实力。关于领导者本身，我们为许多公司提供 AI 转型、培训、战略等方面的咨询。我感觉很多公司的领导者在 10 到 15 年间建立了自己的直觉，并因这些直觉而备受推崇。但现在有了 AI，这些直觉必须重新学习，领导者必须展现出脆弱性（vulnerable）去做到这一点。我曾与现任 Rackspace 的 CEO Gagan 共事。他每天早上都会留出一段固定时间，写着“凌晨 4 点到 6 点跟进 AI 动态”，期间不安排任何会议。

(00:26:42):

那是他了解最新 AI 播客或信息的时间。他还会进行周末“氛围感编程”（vibe coding）之类的活动。所以，我认为领导者必须重新亲力亲为。这并不是说他们必须亲自去实现这些功能，而是为了重建他们的直觉，因为你必须接受你的直觉可能不正确，你可能是房间里最笨的人，并且想向每个人学习。我发现这是区分成功构建产品公司的关键因素，因为这带来了一种自上而下的推动。几乎不可能完全自下而上。如果领导者不信任这项技术，或者对技术有错误的预期，工程师们很难获得领导者的支持。

(00:27:34):

我听过很多开发者抱怨，说领导者根本不理解 AI 能解决特定问题的程度，或者他们只是随便写点代码（vibe code）就以为很容易投入生产。你真的需要了解当今 AI 能解决的问题范围，以便指导公司内的决策。第二个是文化本身。同样，我与一些 AI 并非主业的企业合作，他们引入 AI 只是因为竞争对手在做，或者因为确实有成熟的用例。在这个过程中，我发现很多公司有一种 FOMO（错失恐惧）文化，或者“你会被取代”之类的论调，这让员工感到害怕。领域专家（SME）是构建有效 AI 产品的关键，因为你需要咨询他们来了解 AI 的表现或理想行为应该是怎样的。

(00:28:27):

但我曾与一些公司交谈，那里的领域专家根本不想和你说话，因为他们觉得自己的工作会被取代。所以，这又要回到领导者身上。你想建立一种赋能文化，将 AI 融入工作流，让你所做的事情效率提升 10 倍，而不是说“如果你不采用 AI，你可能会被取代”。这种赋能文化总是有帮助的。你想让整个组织齐心协力，让 AI 为你服务，而不是试图保住自己的饭碗。事实上，AI 开启了比以前更多的机会，你可以让员工做比以前更多的事情，并将生产力提高 10 倍。第三个是我们谈到的技术部分。

(00:29:18):

我认为成功的人都极其痴迷于深入理解他们的工作流，并将适合 AI 的部分进行增强，而将需要人机回环的部分保留。当你尝试自动化工作流的某个部分时，绝不是说用一个 AI 智能体就能解决所有问题。通常是，你可能有一个机器学习模型负责一部分工作，确定性的代码负责另一部分。所以你必须痴迷于理解工作流，以便为问题选择正确的工具，而不是痴迷于技术本身。我看到的另一个模式是，人们真正理解处理非确定性 API（即 LLM）的理念。这意味着他们也理解 AI 开发生命周期看起来非常不同，他们迭代得非常快——即我能否构建一些东西并快速迭代，在不破坏客户体验的同时，给我足够的数据来评估行为。

(00:30:31):

所以他们很快就建立了那个飞轮。在今天，重点不在于成为竞争对手中第一个拥有智能体的公司，而在于你是否建立了正确的飞轮，以便随着时间的推移不断改进。如果有人跑来跟我说："我们有一个一键式智能体，部署到你的系统里，两三天内就能看到显著收益。"我几乎会持怀疑态度，因为这根本不可能。这不是因为模型不行，而是因为企业的数据和基础设施非常混乱，智能体需要时间来理解这些系统是如何运作的。到处都是混乱的分类法（taxonomies）。人们倾向于做类似"获取客户数据_版本1"、"获取客户数据_最终版"之类的事情。所有这些函数都存在并被调用，基本上有很多技术债需要处理。

(00:31:23):

所以大多数时候，如果你痴迷于问题本身并非常了解你的工作流，你就会知道如何随着时间的推移改进你的智能体，而不是仅仅塞进一个智能体并假设它从第一天起就能工作。我甚至敢说，如果有人向你推销"一键式智能体"，那纯粹是营销。你不要买账。我宁愿选择一家说"我们将为你构建这个管道，它会随着时间的推移学习并建立改进飞轮"的公司，而不是一个开箱即用的东西。要替换任何关键工作流或构建能带来显著投资回报率（ROI）的东西，即使拥有最好的数据层和基础设施层，也至少需要四到六个月的工作。

[00:32:05] Lenny Rachitsky

English:

Amazing. There's a lot there that resonates so deeply with other conversations I've been having on this podcast. One is just for a company to be successful at seeing a lot of impact from AI, the founder-CEO has to be deep into it. I had Dan Shipper on the podcast and they work with a bunch of companies helping them adopt AI. And he said that's the number one predictor of success. Is the CEO chatting with ChatGPT, Claude, whatever, many times a day. I love this example you gave with the Rackspace CEO has catch up on AI news in the morning every day. I was imagining he'd be chatting with the chatbot versus reading news.

中文翻译:

太棒了。这里有很多内容与我在这档播客中进行的其他对话产生了深刻共鸣。其中一点是，一家公司要想在 AI 方面看到巨大成效，创始人兼 CEO 必须深入参与。我曾邀请 Dan Shipper 上节目，他们帮助很多公司采用 AI。他说成功的头号预测指标就是：CEO 是否每天多次与 ChatGPT、Claude 等工具聊天。我非常喜欢你举的 Rackspace CEO 每天早上跟进 AI 新闻的例子。我之前还以为他是在和聊天机器人聊天，而不是读新闻。

[00:32:42] Aishwarya Naresh Reganti

English:

With the kind of information you have as of today, you could just ... I mean, you want to choose the right channels as well because everybody has an opinion. So whose opinion do you want to bank on? I feel like having that good quality set of people that you're listening to really makes sense. So he just has a list of two or three sources that he always looks at. And then he comes back with a bunch of questions and bounces it around with a bunch of AI experts to see what they think about it. And I was part of that group, so I kind of know-

中文翻译:

就目前的信息量而言，你真的需要选择正确的渠道，因为每个人都有自己的看法。那么你想信任谁的观点呢？我觉得拥有一组高质量的倾听对象非常有意义。所以他只有两三个固定的信息源。然后他会带着一堆问题回来，和一群 AI 专家交流，看看他们的看法。我当时就在那个专家组里，所以我知道——

[00:33:11] Lenny Rachitsky

English:

I love that.

中文翻译:

我喜欢这个做法。

[00:33:13] Aishwarya Naresh Reganti

English:

... about the questions that he comes up with.

中文翻译:

……关于他提出的那些问题。

[00:33:13] Lenny Rachitsky

English:

That's cool.

中文翻译:

这很酷。

[00:33:15] Aishwarya Naresh Reganti

English:

It's pretty cool. I was like, "Why are you doing so much?" And then he says, "It trickles down into a bunch of decisions that we take."

中文翻译:

确实很酷。我曾问他：“你为什么要花这么多精力？”他说：“这会渗透到我们做出的许多决策中。”

[00:33:21] Lenny Rachitsky

English:

Okay. Let me talk about another topic that's very ... It's been a hot topic on this podcast. It was a hot topic on Twitter for a while, evals. A lot of people are obsessed with evals, think they're the solution to a lot of problems in AI. A lot of people think they're overrated that you don't need evals. You can just feel the vibes and you'll be all right. What's your take on evals? How far does that take people in solving a lot of the problems that you talk about?

中文翻译:

好。让我谈谈另一个话题，这也是本播客的一个热门话题，在 Twitter 上也火了一阵子，那就是“评估”（evals）。很多人痴迷于评估，认为它们是解决 AI 许多问题的良药。也有很多人认为评估被高估了，你不需要评估，只要“凭感觉”（feel the vibes）就行。你们对评估怎么看？它在多大程度上能帮助人们解决你们提到的那些问题？

[00:33:47] Kiriti Badam

English:

In terms of what is going on in the community, I feel there's just this false dichotomy of this either evals is going to solve everything or online monitoring or production monitoring is going to solve everything. And I find no reason to trust one of the extremes in the sense that I will entirely bank my application on this or like that to solve the thing. So if you take a step back, think of what are evals. Evals are basically your trusted product thinking or your knowledge about the product that is going into this set of data sets that you're going to build in the sense that this is what matters to me. This is the kind of problems that my agent should not do and let me build a list of datasets so that I'm going to do well on those. And in terms of production monitoring, what you're doing there is you're deploying your application and then you're having some sort of key metrics that actually communicate back to you on how customers are using your product.

(00:34:47):

You could be deploying any agent and if the customer is giving a thumbs up for your interaction, you better want to know that. So that is what production monitoring is going to do. And this production monitoring has existed for products for a long time, just that now with the AI agents, you need to be monitoring a lot more granularity. It's not just the customer always giving you explicit feedback, but there is many implicit feedback that you can get. For example, in ChatGPT, if you are liking the answer, you can actually give a thumbs up. Or if you don't like the answer, sometimes customers don't give you thumbs down, but actually regenerate the answer. So that is a clear indication that the initial answer that regenerator is not meeting the customer's expectation. So these are the kind of implicit signals you always need to think about.

(00:35:35):

And that spectrum has been increasing in terms of production monitoring. Now let's come back to the initial topic of like, okay, is it evals or is it production monitoring? What does it matter? So I feel, again, we go back to this problem first approach of what is it that you're trying to build. You're trying to build a reliable application for your customers that's not going to do a bad thing. It's always going to do the right thing. Or if it is doing a wrong thing, you're basically alerted very quickly. So I break this down into two parts. One is nobody goes into deploying an application without actually just testing that. This testing could be wipes or this testing could be, "Okay, I have this 10 questions that it should not go wrong no matter what changes I make, and let me build this and let's call this an evaluation dataset." Now, let's say you build this, you deployed this, and then you figured, "Okay, now I need to understand whether it's doing the right thing or not."

(00:36:32):

So if you're a high throughput or a high transaction customer, you cannot practically sit and evaluate all the traces. You need some indication to understand what are the things that I should look at. And this is where production monitoring comes into the picture that you cannot predict the base in which your agent could be doing wrong, but all of these other implicit signals and explicit signals, those are going to communicate back to you what are the traces that you need to look at. And that is where production monitoring helps. And once you get this kind of traces, you need to examine what are the failure patterns that you're seeing in these different types of interactions. And is there something that I really care about that should not happen? And if that kind of failure modes are happening, then I need to think about building an evaluation dataset for it.

(00:37:20):

And okay, let's say I built an evaluation dataset for my agent trying to offer refunds where explicitly I have configured it not to. So I built this evaluation dataset and then I made my changes in tools or prompts or whatever, and then I deployed the second version of the product. Now there is no guarantee that this is the only problem that you're going to see. You still need production monitoring to actually catch different kinds of problems that you might encounter. So I feel evals are important, production monitoring is important, but this notion of only one of them is going to solve things for you that is completely dismissible in my opinion.

中文翻译:

关于社区里正在发生的事情，我觉得存在一种虚假的二元对立：要么认为评估能解决一切，要么认为在线监控或生产监控能解决一切。我找不到理由去信任任何一个极端，即完全把应用寄托在其中一个上面。退一步想，评估是什么？评估本质上是你对产品的可信思考或知识，并将其转化为一组数据集，表达出“这对我很重要”、“这是我的智能体不该出的错”，然后建立数据集确保在这些方面表现良好。而生产监控是指你部署应用后，通过某些关键指标了解客户如何使用你的产品。

(00:34:47):

你部署任何智能体，如果客户对交互点赞，你肯定想知道。这就是生产监控的作用。生产监控在产品领域已经存在很久了，只是现在有了 AI 智能体，你需要监控得更细致。不仅是客户提供的显式反馈，还有很多隐式反馈。例如在 ChatGPT 中，如果你喜欢答案，可以点赞；如果你不喜欢，有时客户不点踩，而是直接点击“重新生成”。这清楚地表明初始答案不符合客户预期。这些是你始终需要思考的隐式信号。

(00:35:35):

生产监控的范畴一直在扩大。回到最初的话题：是评估还是生产监控？这重要吗？我觉得，还是要回到“问题优先”的方法：你到底想构建什么？你想为客户构建一个可靠的应用，它不会做坏事，总是做正确的事；或者如果它做错了，你能很快收到警报。我把它分为两部分。第一，没有人会在不测试的情况下部署应用。这种测试可以是“凭感觉”，也可以是“我有这 10 个问题，无论我怎么改，它们都不能出错，让我为此建立一个评估数据集”。假设你建立了这个，部署了它，然后你发现：“好吧，现在我需要知道它是否在做正确的事。”

(00:36:32):

如果你是一个高吞吐量或高交易量的客户，你不可能实际坐下来评估所有的运行轨迹（traces）。你需要一些指标来了解我应该看哪些部分。这就是生产监控发挥作用的地方——你无法预测智能体出错的所有方式，但所有这些隐式和显式信号会告诉你哪些轨迹需要查看。这就是生产监控的帮助所在。一旦你获得了这些轨迹，你需要检查在不同类型的交互中看到了哪些失败模式。是否有我非常在意、绝对不该发生的事情？如果发生了这类失败模式，那么我就需要考虑为此建立一个评估数据集。

(00:37:20):

比如，我为我的智能体建立了一个评估数据集，测试它是否会在我明确配置不退款的情况下尝试提供退款。我建立了这个评估集，修改了工具或提示词，然后部署了产品的第二个版本。现在，并不能保证这是你会遇到的

唯一问题。你仍然需要生产监控来捕捉可能遇到的其他各种问题。所以我觉得评估很重要，生产监控也很重要，但认为其中任何一个能解决所有问题的想法，在我看来都是不可取的。

[00:37:58] Lenny Rachitsky

English:

All right. A very reasonable answer. And the point here isn't, it's not just as simple as do both. It's more that there are different things to catch and one approach won't catch all the things you need to be paying attention to.

中文翻译:

好，非常合理的回答。这里的重点不仅仅是简单的“两者都做”，而是有不同的东西需要捕捉，单一的方法无法捕捉到所有你需要关注的事情。

[00:38:11] Aishwarya Naresh Reganti

English:

Exactly.

中文翻译:

没错。

[00:38:12] Lenny Rachitsky

English:

Awesome.

中文翻译:

太棒了。

[00:38:13] Aishwarya Naresh Reganti

English:

I want to take two steps back and kind of talk about how much weight the term evals has had to take in the second half of 2025 because you go meet a data labeling company and they tell you our experts are writing evals and then you have all of these folks saying that PMs should be writing evals, they're the new PRDs. And then you have folks saying that evals is pretty much everything, which is the feedback loop you're supposed to be building to improve your products. Now, step back as a beginner and kind of think what are evals? Why is everyone saying evals? And these are actually different parts of the process and nobody is wrong in the sense that yes, these are evals, but when a data labeling company is telling you that our experts are writing evals, they're actually referring to error analysis or experts just leading notes on what should be right.

(00:39:02):

Lawyers and doctors write evals, that doesn't mean they're building LLM judges or they're building this entire feedback loop. And when you say that a PM should be writing evals, doesn't mean they have to write an LLM judge that's good enough for production. I think there are also very prescriptive ways of doing this and plus one to KD, which is you cannot predict upfront if you need to be building an LLM judge versus you need to be using implicit signals from production monitoring, et cetera. I think Martin Fowler at some point had this term called semantic diffusion back in the 2000s, which kind of means that someone comes up with a term, everybody starts butchering it with their own definitions and then you kind of lose the actual definition of it. That is what is happening to evals or agents or any word in AI as of today, everybody kind of sees a different side to it, I guess.

(00:39:54):

But if you make a bunch of practitioners sit together and ask them, "Is it important to build an actionable feedback loop for AI products?" I think all of them will agree. Now, how you do that really depends on your application itself. When you go to complex use cases, it's incredibly hard to build LLM judges because you see a lot of emerging patterns. If you built a judge that would test for verbosity or something like that, it turns out that you're seeing newer patterns that your LLM judge is not able to catch, and then you just end up building too many evals. And at that point, it just makes sense to look at your user signals, fix them, check if you have regressed and move on instead of actually building these judges. So it all depends. I think one statement that every ML practitioner will tell you is it really depends on the context. Don't be obsessed with prescriptions they're going to change.

中文翻译:

我想退后两步，谈谈“评估”（evals）这个词在 2025 年下半年承担了多少重量。你去见一家数据标注公司，他们会告诉你“我们的专家正在编写评估”；然后你听到有人说“PM 应该编写评估，评估是新的 PRD（产品需求文档）”；还有人说“评估几乎就是一切，它是你为了改进产品而应该建立的反馈回路”。现在，作为一个初学者退一步想：评估到底是什么？为什么每个人都在说评估？这些实际上是流程中不同的部分，没有人是错的，它们确实都是评估。但当数据标注公司说专家在写评估时，他们实际上是指错误分析或专家关于什么是正确结果的注释。

(00:39:02):

律师和医生也会写评估，但这并不意味着他们在构建“LLM 裁判”（LLM judges）或整个反馈回路。当你说 PM 应该写评估时，并不意味着他们必须写出一个足以投入生产的 LLM 裁判。我认为做这件事有很多规范的方法，我非常同意 KD（Kiriti）的观点：你无法预先预测你是需要构建一个 LLM 裁判，还是需要利用来自生产监控的隐式信号。Martin Fowler 在 2000 年代曾提出过一个词叫“语义扩散”（semantic diffusion），意思是有人发明了一个词，然后每个人都开始用自己的定义去曲解它，最后这个词原本的定义就丢失了。这就是目前发生在评估、智能体或 AI 领域任何词汇上的情况，我想每个人看到的都是它的一面。

(00:39:54):

但如果你让一群从业者坐在一起问：“为 AI 产品建立一个可操作的反馈回路重要吗？”我想所有人都会同意。至于如何实现，则完全取决于你的应用本身。在复杂的用例中，构建 LLM 裁判极其困难，因为你会看到很多新兴模式。如果你构建了一个测试“啰嗦程度”的裁判，结果发现出现了 LLM 裁判无法捕捉的新模式，最后你只会陷入构建过多评估的泥潭。到那时，查看用户信号、修复它们、检查是否退化然后继续前进，比实际构建这些裁判更有意义。所以这取决于具体情况。我想每个机器学习从业者都会告诉你的一句话是：这真的取决于上下文（context）。不要痴迷于那些一成不变的规范，它们是会变的。

[00:40:45] Lenny Rachitsky

English:

That's such an important point, this idea that, especially that evals just means many things to different people now. It's just a term for so many things. And it's complicated to just talk about evals when you see it as the stuff data labeling companies are giving you and things PMR, right? And there's also benchmarks. People call benchmarks a little bit evals. It's like-

中文翻译:

这是一个非常重要的观点，即“评估”现在对不同的人意味着很多不同的东西。它成了许多事物的代名词。当你把它看作数据标注公司提供的东西，或者 PM 写的东西时，仅仅谈论“评估”就变得很复杂。还有基准测试（benchmarks），人们有时也把基准测试称为评估。这就像——

[00:41:03] Aishwarya Naresh Reganti

English:

I recently spoke to a client who told me, "We do evals." And I was like, "Okay, can you show me your dataset?" And said, "No, we just checked LM Arena and Artificial Analysis. These are independent benchmarks and we know that this model is the right one for our use case." And I'm like, "You're not doing evals. That's not evals. Those are model evals."

中文翻译:

我最近和一个客户交谈，他告诉我：“我们做评估。”我说：“好，能给我看看你的数据集吗？”他说：“不，我们只是查看了 LM Arena 和 Artificial Analysis。这些是独立的基准测试，我们知道这个模型适合我们的用例。”我说：“你这不是在做评估。那不是评估，那是模型评估（model evals）。”

[00:41:20] Lenny Rachitsky

English:

But it makes sense. The word, it could be used in that context. I get why people think that, but yeah, now it's just confusing it even more.

中文翻译:

但这也有道理。这个词确实可以用在那个语境下。我明白人们为什么会那样想，但没错，这让事情变得更加混乱了。

[00:41:26] Aishwarya Naresh Reganti

English:

Yep.

中文翻译:

是的。

[00:41:27] Lenny Rachitsky

English:

Just one more line of questioning here that I think that's on my mind is the reason this became kind of a big debate is Cloud Code. The head of Cloud Code, Boris, was like, "Nah, we don't do evals on Cloud Code. It's all vibes." What can you share, Kiriti, on Kodex and the Kodex team, how you approach evals?

中文翻译:

关于这个话题我还有一个想问的。这件事之所以变成大辩论，是因为 Cursor（注：原文误为 Cloud Code，实指 Cursor 代码编辑器）。Cursor 的负责人 Boris 曾说："不，我们在 Cursor 不做评估，全凭感觉。"Kiriti，关于 Kodex 以及 Kodex 团队，你能分享一下你们是如何处理评估的吗？

[00:41:44] Kiriti Badam

English:

So Kodex, we have this balanced approach of you need to have evals and you need to definitely listen to your customers. And I think Alex has been on your podcast recently and he's been talking about how you're extremely focused on building the right product. And a big part of it is basically listening to your customers. And coding agents are extremely unique compared to agents for other domains in the sense that these are actually built for customizability and these are built for engineers. So coding agent is not a product which is going to solve these top five workflows or top six workflows or whatever. It's meant to be customizable in multi different ways. And the implication of that is that your product is going to be used in different integrations and different kinds of tools and different kinds of things. So it gets really hard to build an evaluation dataset for all kinds of interactions that your customers are going to use your product for.

(00:42:38):

With that said, you also need to understand that, okay, if I'm going to make a change, it's at least not going to damage something that is really core to the product. So we have evaluations for doing that, butt the same time we take extreme care on understanding how the customers are using it. For example, we built this code review product recently and it has been gaining extreme amount of traction. And I feel like many, many bugs in OpenAI as well as even our external customers are getting caught with this. And now let's say if I'm making a model change to the code review or a different kinds of RL mechanism that I trained with it, and now if I'm going to deploy it, I definitely do want A/B test and identify whether it's actually finding the right mistakes and how are users reacting to it? And sometimes if users do get annoyed by your incorrect code regis, they go to the extent of just switching off the product.

(00:43:36):

So those are the signals that you want to look at and make sure that your new changes are doing the right thing. And it's extremely hard for us to think of these kind of scenarios beforehand and develop evaluation data sets for it. So I feel like there's a bit of both. There's a lot of vibes and there's a lot of customer feedback and we are super active on the social media to understand if anybody's having certain types of problems and quickly fix that. So I feel it's a ... How do I put this? It's like a domain of things that you do here.

中文翻译:

在 Kodex，我们采取一种平衡的方法：既需要评估，也绝对需要倾听客户。我想 Alex 最近也上过你的播客，他谈到了如何极度专注于构建正确的产品。其中很大一部分就是倾听客户。编程智能体与其他领域的智能体相比非常独特，因为它们是为可定制性而生的，是为工程师构建的。编程智能体不是一个只解决前五个或前六个工作流的产品，它旨在以多种不同方式进行定制。这意味着你的产品将被用于不同的集成、不同的工具和不同的场景。因此，要为客户使用产品的所有交互类型构建评估数据集是非常困难的。

(00:42:38):

话虽如此，你也需要确保：如果我要做一个更改，它至少不会破坏产品的核心功能。所以我们有相应的评估来确保这一点，但同时我们也非常关注客户的使用情况。例如，我们最近构建了一个代码审查（code review）产品，它获得了巨大的关注。我觉得 OpenAI 内部以及外部客户的许多 bug 都是通过它发现的。现在，假设我要对代码审查进行模型更改，或者使用不同的强化学习（RL）机制进行训练，当我准备部署它时，我肯定想进行

A/B 测试，确定它是否真的发现了正确的错误，以及用户的反应如何。有时，如果用户被你错误的建议惹恼了，他们甚至会直接关掉产品。

(00:43:36):

所以你需要关注这些信号，确保你的新更改在做正确的事。我们很难预先想到所有这些场景并开发评估数据集。所以我觉得两者兼而有之：有很多“凭感觉”，也有很多客户反馈。我们在社交媒体上非常活跃，以了解是否有人遇到了特定类型的问题并迅速修复。我觉得这……怎么说呢？这就像是你在这里做的一系列事情的集合。

[00:44:08] Lenny Rachitsky

English:

That makes so much sense. Okay. What I'm hearing, Codex, pro evals, but it's not enough.

中文翻译:

非常有道理。我听到的是，Codex 支持评估，但仅有评估是不够的。

[00:44:13] Kiriti Badam

English:

Yes.

中文翻译:

是的。

[00:44:13] Lenny Rachitsky

English:

But also just watch customer behavior and feedback. And also there's some vibes just like, is this feeling good? As I'm using it, generating great code that I'm excited about that I think is great.

中文翻译:

还要观察客户行为和反馈。同时也有一些“感觉”，比如：这用起来感觉好吗？它生成的代码是否让我感到兴奋，是否让我觉得很棒？

[00:44:24] Kiriti Badam

English:

I don't think if anybody's coming and seeing that I have this concrete set of evals that I can bet my life on and then I don't need to think about anything else, it's not going to work. And every new model that you're going to launch, we get together as a team and test different things. Each person is concentrating on something else. And we have this list of hard problems that we have and we throw that to the model and see how well they're progressing. So it's like custom evals for each engineer, you would say, and just understand what the product is doing in its new model.

中文翻译:

我不认为如果有人说“我有一套坚实的评估，我可以拿命担保，之后就什么都不用想了”，这行得通。每当我们要发布新模型时，团队都会聚在一起测试不同的东西。每个人关注点不同。我们有一份难题列表，我们会把这些题扔给模型，看它们的进展如何。你可以说，这就像是每个工程师的“自定义评估”，只是为了了解产品在新模型下的表现。

[00:44:58] Lenny Rachitsky (Sponsor Message)

English:

If you're a founder, the hardest part of starting a company isn't having the idea, it's scaling the business without getting buried in back office work. That's where Brex comes in. Brex is the intelligent finance platform for founders. With Brex, you get high limit corporate cards, easy banking, high yield treasury, plus a team of AI agents that handle manual finance tasks for you. They'll do all the stuff that you don't want to do, like file your expenses, scour transactions for waste, and run reports all according to your rules. With Brex's AI agents, you can move faster while staying in full control. One in three startups in the United States already runs on Brex. You can too at brex.com.

中文翻译:

如果你是一位创始人，创业最难的部分不是构思，而是在不被后台琐事淹没的情况下扩展业务。这就是 Brex 发挥作用的地方。Brex 是为创始人打造的智能财务平台。通过 Brex，你可以获得高额度的企业卡、便捷的银行业务、高收益的国库账户，以及一支为你处理手动财务任务的 AI 智能体团队。它们会处理所有你不想做的事情，比如报销费用、搜寻浪费的交易、根据你的规则运行报告。有了 Brex 的 AI 智能体，你可以跑得更快，同时保持完全控制。美国三分之一的初创公司已经在使用 Brex。你也可以在 brex.com 加入他们。

[00:45:43] Lenny Rachitsky

English:

We've been talking for almost an hour already, and we haven't even covered your extremely powerful software development workflow for building AI products that you two developed that you teach in your course, that you basically combined all the stuff we've been talking about into a step-by-step approach to building AI products. You call it the continuous calibration, continuous development framework. Let's pull up a visual to show people what the heck we're talking about, and then just walk us through what this is, how this works, how teams can shift the way they build their AI products to this approach to help them avoid a lot of pain and suffering.

中文翻译:

我们已经聊了快一个小时了，还没聊到你们为构建 AI 产品开发的那个极其强大的软件开发工作流。你们在课程中教授这个工作流，基本上把我们刚才聊的所有内容结合成了一个构建 AI 产品的分步方法。你们称之为“持续校准、持续开发”（CCCD）框架。让我们展示一张图表，告诉大家我们到底在说什么，然后请带我们了解一下这是什么、它是如何运作的，以及团队如何将构建 AI 产品的方式转向这种方法，以避免大量的痛苦和折磨。

[00:46:18] Aishwarya Naresh Reganti

English:

Before we go about explaining the life cycle, a quick story on why Kiriti and I came up with this is because there are tons of companies that we keep talking to that have the pressure from their competitors because they're all building agents. We should be building agents that are entirely autonomous. And I did end up working with a few customers where we built these end-to-end agents. And turns out that because you start off at a place where you don't know how the user might interact with your system and what kind of responses or actions the AI might come up with, it's really hard to fix problems when you have this really huge workflow, which is taking four or five steps, making tons of decisions. You just end up debugging so much and then kind of hot fixing to the point where at a time we were building for a customer support use case, which is the example that we give in the newsletter as well.

(00:47:13):

And we had to shut down the product because we were doing so many hot fixes and there was no way we could count all the emerging problems that were coming up. And there's also quite some news online. Recently, I think Air Canada had this thing where one of their agents predicted or hallucinated a policy for a refund, which was not part of their original playbook, and they had to go by it because legal stuff. And there have been a ton of really scary incidents. And that's where the idea comes from. How can you build so that you don't lose customer trust and you don't end up, or your agent or AI system doesn't end up making decisions that are super dangerous to the company itself. At the same time, build a flywheel so that you can improve your product as you go. And that's where we came up with this idea of continuous calibration, continuous development.

(00:48:08):

The idea is pretty simple, which is we have this right side of the loop, which is continuous development, where you scope capability and curate data, essentially get a data set of what your expected inputs are and what your expected outputs should be looking at. This is a very good exercise before you start building any AI product because many times you figure out that a lot of the folks within the team are just not aligned on how the product should behave. And that's where your PMs can really give in a lot more information and your subject matter experts as well. So you have this data set that you know your AI product should be doing really well on. It's not comprehensive, but it lets you get started. And then you set up the application and then design the right kind of evaluation metrics. And I intentionally use the term evaluation metrics, although we say evals because I just want to be very specific in what it is because evaluation is a process, evaluation metrics are dimensions that you want to focus on during the process.

(00:49:07):

And then you go about deploying, run your evaluation metrics. And the second part is the continuous calibration, which is the part where you understand what behavior you hadn't expected in the beginning, right? Because when you start the development process, you have this data set that you're optimizing for, but more often than not, you realize that that data set is not comprehensive enough because users start behaving with your systems in ways that you did not predict. And that's where you want to do the calibration piece. I've deployed my system. Now I see that there are patterns that I did not really expect and your evaluation metrics should give you some insight into those patterns, but sometimes you figure out that those metrics were also not enough and you probably have new error patterns that you have not thought about. And that's where you analyze your behavior, spot error patterns.

(00:49:59):

You apply fixes for issues that you see, but you also design newer evaluation metrics to figure out that they are emerging patterns. And that doesn't mean you should always design evaluation metrics. There are some errors that you can just fix and not really come back to because they're very spot errors. For instance, there's a tool calling error just because your tool wasn't defined well and stuff like that. You can just fix it and move on. And this is pretty much how an AI product lifecycle would look like. But what we specifically also mention is while you're going through these iterations, try to think of lower agency iterations in the beginning and higher control iterations. What that means is constrain the number of decisions your AI systems can make and make sure that they're humans in the loop and then increase that over time because you're kind of building a flywheel of behavior and you're understanding what kind of use cases are coming in or how your users are using the system.

(00:50:59):

And one example I think we give in the newsletter itself is the customer support. This is a nice image that kind of shows how you can think of agency and control as two dimensions. And each of your versions keep on increasing the agency or the ability of your AI system to make decisions and lower the control as you go. And one example that we give is that of the customer support agent, where you can break it down into three versions. The first version is just routing, which is your agent able to classify and route a particular ticket to the right department? And sometimes when you read this, you probably think, is it so hard to just do routing? Why can't an agent easily do that? And when you go to enterprises, routing itself can be a super complex problem. Any retail company, any popular retail company that you can think of has hierarchical taxonomies.

(00:51:52):

Most of the times the taxonomies are incredibly messy. I have worked in use cases where you probably have taxonomy that says some kind of hierarchy and then that says shoes and then women's shoes and men's shoes all at the same layer where idea you should be having shoes and then women's shoes and men's shoes should be subclasses. And then you're like, okay, fine. I could just merge that. And you go further and you see that there's also another section on the shoes that says for women and for men, and it's just not aggregated. It's not fixed for some reason. So if an agent kind of sees this kind of a taxonomy, what is it supposed to do? Where is it supposed to route? And a lot of the times we are not aware of these problems until you actually go about building something and understanding it.

(00:52:39):

And when these kind of problems, real human agents see these kind of problems, they know what to check next. Maybe they realize that the node that says for women and for men that's under shoes was last updated in 2019, which means that it's just a dead node that's lying there and not being used. So they kind of know that, okay, we're supposed to be looking at a different node and stuff like that. And I'm not saying agents cannot understand this or models are not capable enough to understand this, but there are really weird rules within enterprises that are not documented anywhere. And you want to make sure that the agents have all of that context instead of just throwing the problem at that.

(00:53:17):

Yeah. Coming back to the versions we had, routing was one where you have really high control because even if your agent routes to the wrong department, humans can take control and undo those actions. And along the way, you also figure out that you probably are dealing with a ton of data issues that you need to fix and make sure that your data layer is good enough for the agent to function. We do is what we said of a Copilot, which is now that you've figured out routing works fine after a few iterations and you've fixed all of your data issues, you could go to the next step, which is, can my agent provide suggestions based on some standard operating procedures that we have for the customer support agent? And it could just generate a draft that the human can make changes to. And when you do this, you're also logging human behavior, which means that how much of this draft was used by the customer support agent or what was omitted. So you're actually getting error analysis for free when you do this because you're literally logging everything that the user is doing that you could then build back into your flywheel.

(00:54:22):

And then we say, post that, once you've figured out that those drafts look good and most of the times maybe humans are not making too many changes, they're using these drafts as is. That's when you want to go to your end-to-end resolution assistant that could draft a resolution that could solve the ticket as well. And those are the stages of agency where you start with low agency and then you go up high. We also have this really nice table that we put together, which is what do you do at each version and what you learn that can enable you to go to the next step and what information do you get that you can feed into the loop, right? When you're just doing your routing, you have better quality routing data, you also know what kind of prompts you need to be building to improve the routing system.

(00:55:09):

Essentially, you're figuring out your structure for context engineering and building that flywheel that you want. And while I go through this, I want to also be very clear that two things. One is when you build with CCCD in mind, it doesn't mean that you've fixed the problem all for one. It's possible that you've probably gone through V3 and you see a new distribution of data that you never previously imagined, but this is just one way to lower your risk, which is you get enough information about how users behave with your system before going to a point of complete autonomy. And the second thing is you're also kind of building this implicit logging system. A lot of people come and tell us that, "Oh, wait, there are evals. Why do you need something like this? " The issue with just building a bunch of evaluation metrics and then having them in production is evaluation metrics catch only the errors that you're already aware of, but there can be a lot of emerging patterns that you understand only after you put things in production.

(00:56:17):

So for those emerging patterns, you're kind of creating a low risk kind of a framework so that you could understand user behavior and not really be in a position where there are tons of errors and you're trying to fix all of them at once. And this is not the only way to do it. There are tons of different ways. You want to decide how you constrain your autonomy. It could be based on the number of actions that the agent is taking, which is what we do in this example. It could be based on topic. There's just some domains where it's pretty high risk to make a system completely autonomous for certain decisions, but for some other topics, it's okay to make them completely autonomous and depending on the complexity of the problem. And that's where you really want your product managers, your engineers and subject matter experts to align on how to build this system and continuously improve it.

(00:57:10):

The idea is just behavior calibration and not losing user trust as you do that behavior calibration, I guess.

中文翻译:

在解释生命周期之前，先讲个小故事。Kiriti 和我之所以想到这个框架，是因为我们一直在和大量公司交谈，他们面临着来自竞争对手的压力，因为大家都在构建智能体。他们觉得：“我们也应该构建完全自主的智能体。”我确实曾与一些客户合作构建过这种端到端智能体。结果发现，因为你从一个完全不知道用户会如何与系统交互、也不知道 AI 会给出什么回应或动作的地方开始，当你的工作流非常庞大（包含四五个步骤、要做无数决策）时，修复问题变得极其困难。你最后会陷入无休止的调试和紧急修复（hot fixing）中。我们当时在为一个客户支持用例构建产品（这也是我们在时事通讯中举的例子）。

(00:47:13):

最后我们不得不关闭那个产品，因为紧急修复太多了，我们根本无法统计不断冒出来的新问题。网上也有不少类似新闻。最近，加拿大航空的一个智能体“幻觉”出了一个退款政策，而这并不在他们的原始手册中，但由于法律原因他们不得不执行。发生了很多这类可怕的事件。这就是这个想法的来源：你如何构建产品，既不失去客户信任，又不让你的智能体或 AI 系统做出对公司极其危险的决策？同时，还要建立一个飞轮，以便随做随改。这就是我们提出“持续校准、持续开发”（CCCD）框架的原因。

(00:48:08):

这个理念很简单：右边的环是“持续开发”，你在这里界定能力范围并整理数据，本质上是建立一个数据集，明确预期的输入是什么，预期的输出应该是什么。在构建任何 AI 产品之前，这都是一个非常好的练习，因为很

多时候你会发现团队内部对产品应该如何表现根本没有达成一致。这时 PM 和领域专家可以提供大量信息。这样你就拥有了一个 AI 产品应该表现良好的数据集。它不全面，但能让你起步。然后你搭建应用，设计合适的“评估指标”（evaluation metrics）。我特意用“评估指标”这个词，虽然我们也说评估，但我想更具体一点，因为评估是一个过程，而评估指标是你在过程中想要关注的维度。

(00:49:07):

然后你进行部署，运行评估指标。第二部分是“持续校准”，即你理解那些最初没预料到的行为。因为当你开始开发时，你针对某个数据集进行优化，但往往你会发现该数据集不够全面，因为用户会以你未预测到的方式使用系统。这就是你需要做校准的地方。我部署了系统，现在我看到了意料之外的模式，评估指标应该能给你一些见解，但有时你会发现指标也不够，出现了你没想到的新错误模式。这时你需要分析行为，找出错误模式。

(00:49:59):

你针对看到的问题进行修复，同时设计新的评估指标来捕捉这些新兴模式。这并不意味着你总是要设计评估指标，有些错误你可以直接修复而不用回头再看，因为它们是偶发错误。例如，仅仅因为工具定义得不好而导致的工具调用错误，修复它然后继续就行。这基本上就是 AI 产品生命周期的样子。但我们特别提到的是，在这些迭代过程中，尝试在开始时采用“低自主性、高控制”的迭代。这意味着限制 AI 系统可以做出的决策数量，确保有人机回环，然后随着时间的推移增加自主性，因为你正在建立一个行为飞轮，并且正在了解有哪些用例进来，或者用户是如何使用系统的。

(00:50:59):

我们在时事通讯中举的一个例子是客户支持。这张图展示了如何将自主性和控制权视为两个维度。随着版本的更迭，你不断增加 AI 系统的自主性（决策能力），并降低控制权。在客户支持智能体的例子中，可以分为三个版本。V1 只是“路由”（routing）：智能体能否将特定的工单分类并路由到正确的部门？有时你读到这里会想：路由有那么难吗？为什么智能体不能轻易做到？但在企业中，路由本身可能是一个极其复杂的问题。任何你能想到的流行零售公司都有层级化的分类法。

(00:51:52):

大多数时候，这些分类法极其混乱。我处理过一些用例，分类法在同一层级上既有“鞋类”，又有“女鞋”和“男鞋”，而理想情况下应该是“鞋类”作为父类，“女鞋”和“男鞋”作为子类。你可能会想，好吧，我可以合并它们。但你进一步看，发现还有一个关于鞋子的部分写着“适合女性”和“适合男性”，由于某种原因它们没有被聚合或修复。如果智能体看到这种分类法，它该怎么办？它该路由到哪里？很多时候，直到你实际构建并理解它，你才会意识到这些问题。

(00:52:39):

当真人客服遇到这类问题时，他们知道下一步该检查什么。也许他们意识到“鞋类”下那个“适合女性”的节点最后一次更新是在 2019 年，这意味着它只是一个废弃节点。所以他们知道应该看另一个节点。我不是说智能体无法理解这些，或者模型能力不够，而是企业内部有很多没有记录在案的奇怪规则。你需要确保智能体拥有所有这些背景信息，而不是直接把问题扔给它。

(00:53:17):

回到版本划分：V1 路由具有极高的控制权，因为即使智能体路由错了部门，人类也可以接管并撤销动作。在这个过程中，你还会发现并修复大量数据问题，确保数据层足以支持智能体运行。V2 是我们所说的“副驾驶”（Copilot）：既然你已经通过几次迭代确定路由运行良好并修复了数据问题，你可以进入下一步：智能体能否根据现有的标准操作程序（SOP）提供建议？它可以生成一个草稿，由真人进行修改。这样做时，你也在记录人类行为，即草稿中有多少被采用了，哪些被删除了。这相当于免费获得了错误分析，因为你记录了用户的所有操作，并可以将其反馈回飞轮。

(00:54:22):

然后我们说，在那之后，一旦你发现草稿看起来不错，且大多数时候人类不需要做太多修改，直接使用草稿，那就是你进入 V3 “端到端解决助手”的时候，它可以直接起草并解决工单。这就是自主性的阶段：从低到高。我们还整理了一个表格，说明每个版本要做什么、学到了什么（以便进入下一步），以及获得了哪些可以反馈到循环中的信息。当你做路由时，你获得了更高质量的路由数据，也知道需要构建什么样的提示词来改进路由系统。

(00:55:09):

本质上，你是在确定“上下文工程”（context engineering）的结构并建立你想要的飞轮。在讲这些时，我想明确两点。第一，当你带着 CCCD 理念构建时，并不意味着你一次性解决了所有问题。你可能到了 V3 还会看到以前从未想象过的新数据分布，但这只是降低风险的一种方式，即在进入完全自主之前，获得足够多关于用户如何与系统交互的信息。第二，你也在建立这种“隐式记录系统”。很多人问：“已经有评估了，为什么还需要这个？”仅仅在生产环境中建立一堆评估指标的问题在于，评估指标只能捕捉你已经意识到的错误，但有很多新兴模式只有在投入生产后才能理解。

(00:56:17):

所以对于这些新兴模式，你是在创建一个低风险的框架，以便理解用户行为，而不是处于一种错误成堆、试图一次性修复所有问题的境地。这并不是唯一的方法，有很多不同的方式。你可以根据智能体采取的动作数量来约束自主性（如本例所示），也可以根据主题来约束。在某些领域，让系统对某些决策完全自主是高风险的，但对另一些主题则是可以接受的，这取决于问题的复杂性。这就是你需要产品经理、工程师和领域专家达成一致的地方：如何构建这个系统并持续改进它。

(00:57:10):

核心理念就是“行为校准”，并且在进行校准的过程中不失去用户的信任。

[00:57:17] Lenny Rachitsky

English:

We'll link folks to this actual post if they want to go really deep. You basically go through all of these steps by step, a bunch of examples. And the idea here is, as you said, that the reason, everything about what you're describing here is about making it continuous and iterative and kind of moving along this progression of higher autonomy, less control. And this idea of even calling continuous calibration, continuous development is communicating it's this kind of iterative process. And just to be clear, this naming is kind of ode to CI/CD, continuous integration, continuous deployment suite. And the idea here is that this is the version of that for AI where instead of just integrating into unit tests and deploying constantly, it's running evals, looking at results, iterating on the metrics you're watching, figuring out where it's breaking and iterating on that. Awesome. Okay.

(00:58:08):

So again, we'll point people to this post if they want to go deeper. That was a great overview. Is there anything else before we go into different topic around this framework specifically that you think is important for people to know?

中文翻译:

如果大家想深入了解，我们会提供这篇文章的链接。你们在文中详细介绍了所有步骤和大量例子。这里的核心理念如你所说，是让过程变得持续且迭代，并沿着“更高自主性、更少控制”的路径前进。甚至将它命名为“持续校准、持续开发”也是为了传达这种迭代过程。明确一下，这个命名是对 CI/CD（持续集成、持续部署）的致敬。这里的理念是，这是 AI 版本的 CI/CD：不是简单地集成到单元测试并不断部署，而是运行评估、查看结果、迭代你关注的指标、找出故障点并进行迭代。太棒了。

(00:58:08):

再次强调，想深入了解的可以去看那篇文章。这是一个很棒的概述。在转入下一个话题之前，关于这个框架，还有什么你们认为重要的信息要告诉大家吗？

[00:58:18] Aishwarya Naresh Reganti

English:

I think one of the most common questions we get is, how do I know if I need to go to the next stage or if this is calibrated enough? There's not really a rule book you can follow, but it's all about minimizing surprise, which means let's say you're calibrating every one or two days and you figure out that you're not seeing new data distribution patterns, your users have been pretty consistent with how they're behaving with the system. Then the amount of information you gain is kind of very low and that's when you know you can actually go to the next stage. And it's all about the wipes at that point, do you know you're ready, you're not receiving any new information. But also it really helps to understand that sometimes there are events that could completely mess up the calibration of your system. An example is GPT-4o doesn't exist anymore, or it's going to be deprecated in APIs as well.

(00:59:16):

So most companies that were using 4o should switch to 5 and 5 has very different properties. So that's where your calibration's off again. You want to go back and do this process again. Sometimes users start behaving with systems also differently over time or user behavior evolves. Even with consumer products, you don't talk to ChatGPT the same way you were talking, say, two years ago, just because you know the capabilities have increased so much. And also just people get excited when these systems can solve one task, they want to try it out on other tasks as well. We built this system for underwriters at some point. Underwriting is a painful task. There are agreements that are like loan applications are like 30 or 40 pages, and the idea for this bank was to build a system that could help underwriters pick policies and information about the bank so that they could approve loans.

(01:00:15):

And for a good three or four months, everybody was pretty impressed with the system. We had underwriters actually report gains in terms of how much time they were spending, et cetera. And first three months, we realized that they were so excited with the product that they started asking very deep questions that we never anticipated. They would just throw the entire application document at the system and go, "For a case that looks like this, what did previous underwriters do? " And for a user, that just seems like a natural extension of what they were doing, but the building behind it should significantly change. Now, you need to understand what does for a case like this mean in the context of the loan itself? Is it referring to people of a particular income range or is it referring to people in a particular geo and stuff like that?

(01:00:58):

And then you need to pick up historical documents, analyze those documents, and then tell them, "Okay, this is what it looks like," versus just saying that there's a policy X, Y, and Z, and you want to look up that policy. So something that might seem very natural to an end user might be very hard to build as a product builder, and you see that user behavior also evolves over time, and that's when you know that you want to go back and recalibrate.

中文翻译:

我认为我们收到的最常见问题之一是：我怎么知道是否该进入下一个阶段，或者是否已经校准得足够好了？这并没有现成的规则手册，但核心在于"最小化意外"（minimizing surprise）。这意味着，假设你每一两天校准一次，你发现没有看到新的数据分布模式，用户对系统的行为非常一致。那么你获得的新信息量就很低了，这时你就知道可以进入下一个阶段了。到那时，更多是凭感觉——你知道你准备好了，因为没有新信息进来了。但同时，理解某些事件可能会彻底打乱系统的校准也很有帮助。例如，GPT-4o 不再存在了，或者 API 即将弃用。

(00:59:16):

所以大多数使用 4o 的公司必须转向 5（注：此处为假设性例子），而 5 具有非常不同的属性。这时你的校准又失效了，你需要重新走一遍这个过程。有时用户随时间推移使用系统的方式也会改变，或者用户行为在演进。即使是消费级产品，你现在和 ChatGPT 说话的方式肯定和两年前不一样，因为你知道它的能力提升了。而且当人们发现系统能解决一个任务时会很兴奋，想尝试其他任务。我们曾为核保师（underwriters）构建过一个系统。核保是一项痛苦的任务，贷款申请协议通常有 30 到 40 页。这家银行的想法是构建一个系统，帮助核保师提取政策和银行信息，以便批准贷款。

(01:00:15):

在最初的三四个月里，大家对这个系统印象非常深刻。核保师确实报告了在节省时间等方面的收益。但在前三个月，我们意识到他们对产品太兴奋了，开始问一些我们从未预料到的深度问题。他们会把整个申请文件扔给系统说："对于像这样的案例，以前的核保师是怎么做的？"对于用户来说，这似乎是他们工作的自然延伸，但背后的构建逻辑必须发生重大变化。现在，你需要理解在贷款背景下，"像这样的案例"意味着什么？是指特定收入范围的人，还是特定地理区域的人？

(01:00:58):

然后你需要调取历史文件，分析这些文件，然后告诉他们："好的，情况是这样的"，而不是仅仅说"有政策 X、Y 和 Z，你去查查"。所以，对最终用户来说非常自然的事情，对产品构建者来说可能非常难以实现。你会看到用户行为随时间演进，这时你就知道需要回去重新校准了。

[01:01:24] Lenny Rachitsky

English:

What do you think is overhyped in the AI space right now? And even more importantly, what do you think is under-hyped?

中文翻译:

你认为目前 AI 领域有哪些东西被过度炒作（overhyped）的？更重要的是，你认为有哪些东西是被低估（under-hyped）的？

[01:01:34] Kiriti Badam

English:

As I said, super optimistic in different things that are going in AI. So I wouldn't say overhyped, but I feel kind of misunderstood is the concept of multi-agents. People have this notion of, "I have this incredibly complex problem. Now I'm going to break it down into, hey, you are this agent. Take care of this. You're this agent. Take care of this." And now if I somehow connect all of these agents, they think they're the agent utopia and it's never the case that there are incredibly successful multi-agent systems that are built. There's no doubt about that. But I feel a lot of it comes in terms of how are you limiting the ways in which the system can go off tracks. And for example, if you're building a supervisor agent and there are subagents that actually do the work for the super agent, supervisor agent, that is a very successful pattern.

(01:02:24):

But coming with this notion of I'm going to divide the responsibilities based on functionality and somehow expect all of that to work together in some sort of gossip protocol, that is extremely misunderstood that you could do that. I don't think current ways of building and current model capabilities are right there in terms of building those kind of applications. I feel that is kind of misunderstood than overrated. Underrated, I feel it's hard to probably believe, but I still feel coding agents are underrated in the sense that I feel like you can go on Twitter and you can go on Reddit and you see a lot of chatter about coding agents, but talking to an engineer in any random company, especially outside of Bay Area, you can see the amount of impact this coding agents can create and the penetration is very low. So I feel like 2025 and 2026 is going to be an incredible year for optimizing all of these processes.

(01:03:25):

And I feel that is going to be creating a lot of value with AI.

中文翻译:

正如我所说，我对 AI 的各种进展非常乐观。所以我不会说“过度炒作”，但我觉得“多智能体”（multi-agents）的概念被误解了。人们有这样一种想法：“我有一个极其复杂的问题，现在我要把它分解：嘿，你是这个智能体，负责这个；你是那个智能体，负责那个。”他们觉得如果把这些智能体连接起来，就能实现智能体乌托邦。虽然确实有非常成功的多智能体系统被构建出来，这一点毋庸置疑，但我认为关键在于你如何限制系统“脱轨”的方式。例如，如果你构建一个主管智能体（supervisor agent），由子智能体实际为主管智能体工作，这是一个非常成功的模式。

(01:02:24):

但如果抱着“我要根据功能划分职责，并指望它们通过某种闲聊协议（gossip protocol）协同工作”的想法，那是极大的误解。我不认为目前的构建方式和模型能力已经达到了可以构建这类应用的水平。我觉得这更多是被误解了，而不是被高估。至于被低估的，虽然可能很难相信，但我仍然觉得编程智能体（coding agents）被低估了。虽然你在 Twitter 或 Reddit 上能看到很多关于编程智能体的讨论，但如果你去和湾区以外任何一家普通公司的工程师交谈，你会发现编程智能体能产生的巨大影响，而目前的渗透率还非常低。所以我认为 2025 年和 2026 年将是优化所有这些流程的绝佳年份。

(01:03:25):

我觉得这将通过 AI 创造巨大的价值。

[01:03:28] Lenny Rachitsky

English:

That's really interesting on that first point. So the idea there is you'll probably be more successful building and using an agent that is able to do its own sub-agent splitting of work versus a bunch of, say, Codex agents. Will you do this task, you do that task?

中文翻译:

第一个观点很有意思。所以你的意思是，构建并使用一个能够自己拆分子任务的智能体，可能比让一群（比如 Codex）智能体各司其职（你做这个，他做那个）更容易成功?

[01:03:44] Kiriti Badam

English:

You can have agents to do these things and you as a human can orchestrate it or you can have one larger agent that is going to orchestrate all of these things, but letting the agents communicate in terms of peer-to-peer kind of protocol, and then especially doing this in a customer support kind of use case is incredibly hard to control what kind of agent is replying to your customer because you need to shift your guardrails everywhere and things like that.

中文翻译:

你可以让智能体做这些事，然后由你作为人类来编排；或者由一个更大的智能体来编排所有这些事。但如果让智能体以点对点（P2P）协议的方式自由通信，尤其是在客户支持这类用例中，将极其难以控制是哪个智能体在回复客户，因为你需要到处移动你的护栏等等。

[01:04:08] Lenny Rachitsky

English:

Yeah. Okay. Great picks. Okay. Ash, what do you got?

中文翻译:

明白。很好的见解。Ash，你呢?

[01:04:12] Aishwarya Naresh Reganti

English:

Can I say evals? Will I be canceled?

中文翻译:

我能说是“评估”吗? 我会因此被“取消”关注吗?

[01:04:15] Lenny Rachitsky

English:

In which category? Which bucket do they go?

中文翻译:

在哪一类? 属于哪个桶?

[01:04:18] Aishwarya Naresh Reganti

English:

Overrated.

中文翻译:

被高估的。

[01:04:20] Lenny Rachitsky

English:

Overrated. Okay, go for it. We won't let you get canceled.

中文翻译:

被高估。好，说吧，我们不会让你被取消的。

[01:04:22] Aishwarya Naresh Reganti

English:

Just kidding. I think evals are misunderstood. They are important, folks. I'm not saying they're not important, but I think just this, I'm going to keep jumping across tools and going to pick up and learn if new tool is overrated. I still am old school and feel like you would really need to be obsessed with the business problem you're trying to solve. AI is only a tool. I try to think of it that way. Of course, you need to be learning about the latest and greatest, but don't be so obsessed with just building so quickly. Building is really cheap today. Design is more expensive, really thinking about your product, what you're going to build. Is it going to really solve a pain point? Is what is way more valuable today? And it will only become more true in the near future. So really obsessing about your problem and design is underrated and just rote building is overrated, I guess.

中文翻译:

开个玩笑。我认为评估被误解了。各位，评估很重要，我不是说它不重要。但我认为，仅仅不断在各种工具之间跳跃、追求学习每一个新工具，这是被高估的。我还是比较传统，觉得你真的需要痴迷于你试图解决的业务问题。AI 只是一个工具，我试着这样去思考。当然，你需要学习最新最好的技术，但不要仅仅痴迷于快速构建。今天构建的成本非常低，设计反而更贵——真正思考你的产品，你要构建什么，它是否真的能解决痛点？这在今天更有价值，而且在不久的将来会变得更加正确。所以，真正痴迷于问题和设计是被低估的，而仅仅是机械地构建（rote building）是被高估的。

[01:05:15] Lenny Rachitsky

English:

Awesome. Okay. Similar sort of question. From a product point of view, what do you think the next year of AI is going to look like? Give us a vision of where you think things are going to go by, say by the end of 2026.

中文翻译:

太棒了。类似的问题：从产品的角度来看，你认为 AI 的下一年会是什么样子？给我们一个愿景，你认为到 2026 年底事情会发展到什么程度？

[01:05:30] Kiriti Badam

English:

Yeah, I feel there's a lot of promise in terms of this background agents are proactive agents who is ... They're going to basically understand your workflow even more. If you think of where is AI failing to create value today, it's mainly about not understanding the context. And the reason that it's not understanding the context is it's not plugged into the right places where actual work is happening. And as you do more of this, you can give the agent more of context and then it start to see the world around you and understand what are the set of metrics that you're optimizing for or what are the kind of activities that you're trying to do. It is a very easy extension from there to actually gain more out of it and then let the agent prompt you back. We already do this in terms of ChatGPT pulse, which kind of gives you this daily update of things you might care about.

(01:06:20):

And it's very nice to actually have that jog your brain up in terms of, "Oh, this is something that I haven't thought about. Maybe this is good." And now when you extend this to more complex tasks, like a coding agent, which says that, "Okay, I have fixed five of your linear tickets and here are the patches. Just to review them at the start of your day." So I feel that is going to be extremely useful. And I see that as a strong direction in which products are going to build in 2026.

中文翻译:

是的，我觉得“后台智能体”或“主动智能体”（proactive agents）非常有前景。它们将更深入地理解你的工作流。如果你思考 AI 今天在哪些方面未能创造价值，主要是因为它不理解上下文。而不理解上下文的原因是它没有接入实际工作发生的正确位置。随着你做得更多，你可以给智能体更多背景信息，然后它开始观察你周围的世界，理解你正在优化的指标集或你正在进行的活动。从那里很容易延伸出更多的价值，让智能体反过来提示你。我们已经在 ChatGPT pulse 中看到了这一点，它会给你每日更新你可能感兴趣的事情。

(01:06:20):

这能很好地激活你的大脑：“哦，这是我没想到的，也许这不错。”现在，当你把它扩展到更复杂的任务时，比如一个编程智能体说：“我已经修复了你的五个 Linear 工单，这是补丁，你今天开始工作时审查一下就行。”我觉得这将非常有用。我认为这是 2026 年产品构建的一个强有力方向。

[01:06:44] Lenny Rachitsky

English:

That so cool. So essentially agents anticipating what you want to do and getting ahead of you and I've solved these problems for you or I think this is going to crash your site. Maybe you should fix this thing right here or I see the spike here and let's refactor our database. Amazing. What a world. Okay. Ash, what do you got?

中文翻译:

太酷了。本质上是智能体预测你想做什么并走在你前面：“我已经为你解决了这些问题”，或者“我觉得这会让你的网站崩溃，也许你应该修复这里”，或者“我看到这里的流量激增，让我们重构一下数据库”。太神奇了，真是个奇妙的世界。Ash，你呢?

[01:07:04] Aishwarya Naresh Reganti

English:

I'm all in for multimodal experiences in 2026. I think we have done quite some progress in 2025, and not just in terms of generation, but also understanding. Until now, I think LLMs have been our most commonly used modules, but as humans, we are multimodal creatures, I would say. Language is probably one of our last forms of evolution. As the three of us are talking, I think we're constantly getting so many signals. I'm like, "Oh, Lenny's nodding his head, so probably I would go in this direction or Lenny's bored, so let me stop talking." So there's a chain of thought behind your chain of thought and you're constantly altering it with language that dimension of expression is not explored as well. So if we could build better multimodal experiences that would get us closer to human-like conversation richness. And you will also, just given the kind of models, there's a bunch of boring tasks as well, which are ripe for AI.

(01:08:04):

If multimodal understanding gets better, there are so many handwritten documents and really messy PDFs that cannot be passed even by the best of the models as of today. And if it's possible, there'll be so much data that we can tap into.

中文翻译:

我非常看好 2026 年的多模态（multimodal）体验。我认为我们在 2025 年已经取得了很大进展，不仅在生成方面，还在理解方面。到目前为止，LLM 是我们最常用的模块，但作为人类，我们是多模态生物。语言可能是我们进化的最后形式之一。当我们三个人交谈时，我们不断接收到很多信号。我会想：“哦，Lenny 在点头，所以我可能应该往这个方向说”，或者“Lenny 听烦了，让我闭嘴吧”。在你的“思维链”背后还有另一层思维链，你不断地根据语言以外的维度调整它，而这些表达维度目前还没有被很好地探索。所以，如果我们能构建更好的多模态体验，就能让我们更接近人类对话的丰富程度。此外，考虑到目前的模型，还有一堆无聊的任务非常适合 AI。

(01:08:04):

如果多模态理解变得更好，目前即使是最好的模型也无法解析的大量手写文档和极其混乱的 PDF 就能被处理了。如果这能实现，我们将能挖掘出海量的数据。

[01:08:21] Lenny Rachitsky

English:

Awesome. I just saw Demis from DeepMind, AI, Google, whatever they call the whole org, talking about this where he thinks that's going to be a big part of where they're going, combining the image model work, the LLM, and also their world model stuff, Genie, I think is what it's called. Yes. So that's going to be a wild, wild time. Okay. Last question. If someone wants to just get better at building AI products, what's just maybe one skill or maybe two skills that you think they should lean into and develop?

中文翻译:

太棒了。我刚看到 DeepMind（或者说 Google AI 部门）的 Demis 谈到这一点，他认为这将是他们未来的重要组成部分：结合图像模型、LLM 以及他们的世界模型（好像叫 Genie）。那将是一个疯狂的时代。好，最后一个问题：如果有人想提高构建 AI 产品的能力，你认为他们应该重点培养哪一两个技能?

[01:08:52] Aishwarya Naresh Reganti

English:

I think we did cover a bunch of best practices for AI products, which is start small, try to get your iteration going well and build a flywheel and all of that. But again, if you kind of look at it at a 10,000 feet level for anybody building today, like I was saying, implementation is going to be ridiculously cheap in the next few years. So really nail down your design, your judgment, your taste and all of that. And in general, if you're building a career as well, I feel for the past few years, your former years, say the first two, three years of building your career is always focused on execution, mechanics and all of that. And now we have AI that could help you ramp pretty quickly and post that. I mean, after a few years, I think everybody's job becomes about your taste, your judgment and kind of what is uniquely you.

(01:09:49):

I think nail down on that part and try to figure out how you can bring in that kind of a perspective. It doesn't have to mean that you should be significantly old, have years of experience. We recently hired someone and we use this very popular app for tracking our tasks and we've been using it for years and we pay a high subscription fee for it. And this guy just came with his own vibe coded app to the meeting. He onboarded us to all of it and he's like, "Okay, let's start using this." And I think that kind of agency and that kind of ownership to really rethink experiences is what will set people apart. And I'm not being blind to the fact that vibe coded apps have high maintenance costs. And maybe as we scale as a company, we have to replace it or we have to think of better approaches.

(01:10:36):

But given that we are a small size company now and just ... I was really shocked because I never thought of it. If you've been used to working in a certain way, you associate a cost with building. And I feel like folks who grew up in this age have a much lower cost associated in their mind. They just don't mind building something and going ahead with it. And they're also very enthusiastic to try out new tools. That's also probably why AI products have this retention problem because everybody's so excited about trying out these new tools and all of that. But essentially having the agency and ownership, and I think it's also the going to be the end of the busy work era. You can't be sitting in a corner doing something that doesn't move the needle for a company. You really need to be thinking about end-to-end workflows, how you can bring in more impact.

(01:11:26):

I think all of that will be super important.

中文翻译:

我认为我们确实涵盖了很多 AI 产品的最佳实践，比如从小处着手、保持良好迭代、建立飞轮等等。但如果从一万英尺的高度来看，正如我所说，未来几年实施（implementation）的成本将低得离谱。所以，真正磨练你的设计、判断力、品味等等。总的来说，如果你在规划职业生涯，过去几年，职业生涯的早期（比如前两三年）总是专注于执行、机械操作等。现在 AI 可以帮你快速上手。在那之后，我认为每个人的工作都会变成关于你的品味、判断力以及你独特的特质。

(01:09:49):

我认为要在这部分下功夫，思考如何带来那种独特的视角。这并不意味着你必须年纪很大或有多年经验。我们最近雇了一个人，我们一直用一个非常流行的任务追踪应用，用了好几年，付着高昂的订阅费。结果这家伙带着他自己“凭感觉编写”（vibe coded）的应用来开会，向我们演示了所有功能，然后说：“好，让我们开始用这个吧。”我认为这种真正重新思考体验的自主性和主人翁精神（ownership）是让人脱颖而出的关键。我并非无视“凭感觉编写”的应用维护成本很高，也许随着公司规模扩大，我们需要更换它或思考更好的方法。

(01:10:36):

但考虑到我们现在是一家小公司……我真的很震惊，因为我从未想过。如果你习惯了某种工作方式，你会把构建与高成本联系起来。而我觉得在这个时代成长起来的人，脑子里对构建的成本预期要低得多。他们不介意随手构建一个东西并付诸实践。他们也非常热衷于尝试新工具。这可能也是为什么 AI 产品存在留存问题，因为每个人都太兴奋于尝试新工具了。但本质上，拥有自主性和主人翁精神，以及——我认为这也将是“瞎忙活”（busy work）时代的终结。你不能再躲在角落里做一些对公司毫无贡献的事情。你真的需要思考端到端的工作流，思考如何带来更大的影响。

(01:11:26):

我认为所有这些都将非常重要。

[01:11:28] Lenny Rachitsky

English:

That reminds me, I just had Jason Lemkit on the podcast. He's very smart on sales, go to market, run Saster, and he replaced his whole sales team with agents. He had 10 salespeople and then he was 1.2 and 20 agents. And one of the agents, it was just tracking everyone's updates to Salesforce and kind of updating it automatically for them based on their calls. And one of the salespeople was like, "Okay, I quit." And it turned out he wasn't really doing anything. He was just sitting around and he's like, "Okay, this will catch me. I got to get out of here. So to your point about, it'll be harder to sit around and twiddle your thumbs, I think is really right.

中文翻译:

这让我想起我刚邀请了 Jason Lemkin 上节目。他在销售、进入市场（GTM）方面非常聪明，经营着 SaaStr。他用智能体替换了整个销售团队。他原本有 10 名销售人员，后来变成了 1.2 个人加 20 个智能体。其中一个智能体只是追踪每个人在 Salesforce 上的更新，并根据他们的通话自动更新。其中一个销售人员说："好吧，我辞职。"结果发现他其实什么都没干，只是闲坐着，他想："好吧，这玩意儿会发现我的，我得赶紧走。"所以，关于以后很难再闲坐着无所事事，我觉得你说得很对。

[01:12:07] Kiriti Badam

English:

Yeah. I think to add on to that, I feel like persistence is also something that is extremely valuable, especially given that anybody who wants to build something, the information is at your fingertips even more than the past decade. You can learn anything overnight and become that sort of Ironman kind of approach. So I feel like having that persistence and going through the pain of learning this, implementing this and understanding what works and what doesn't work. And as you are going through this pain of developing multiple approaches and then solving the problem, I feel that is going to be the real moat as an individual. I like to call it pain is the new moat, but I feel that is exactly super useful to actually have this in, especially in building these AI products.

中文翻译:

是的。我想补充一点，我觉得"坚持"也是极其宝贵的，尤其是考虑到现在任何想构建东西的人，信息都比过去十年更容易获得。你可以一夜之间学会任何东西，变成那种"钢铁侠"式的人物。所以我觉得拥有那种坚持，经历学习、实施、理解哪些行得通、哪些行不通的痛苦过程。当你经历开发多种方法并最终解决问题的痛苦时，我觉得这将成为你作为个人的真正护城河。我喜欢称之为"痛苦是新的护城河"，我觉得这在构建 AI 产品时特别有用。

[01:12:56] Lenny Rachitsky

English:

Say more about this. I love this concept. Pain is the new moat. Is there more there?

中文翻译:

多聊聊这个。我喜欢这个概念，"痛苦是新的护城河"。还有更多见解吗?

[01:13:00] Kiriti Badam

English:

Yeah, I feel as a company, I mean, successful companies right now building in any new area, they are successful not because they're first to the market or they have this fancy feature that more customers are liking it. They went through the pain of understanding what are the set of non-negotiable things and trade them off exactly with what are the features or what are the model capabilities that they can use to solve that problem. This is not a straightforward process. There's no textbook to do this or there's no straightforward way or a known credit path to be here. So a lot of this pain I was talking about is just going through this iteration of like, "Okay, let's try this and if this doesn't work, let's try this." And that kind of knowledge that you built across the organization or across your own lived experiences, I feel that pain is what translates into the moat of the company. This could be a product of evals or something that you built. And I feel that is going to be the game changer.

中文翻译:

是的。我觉得作为一家公司，目前在任何新领域取得成功的公司，其成功并非因为他们是第一个进入市场的，或者他们拥有更多客户喜欢的花哨功能。而是因为他们经历了痛苦的过程，理解了哪些是不可妥协的原则，并精确地在功能或模型能力之间进行权衡，以解决问题。这不是一个简单的过程。没有教科书教你怎么做，也没有现成的路径。所以我说的很多痛苦就是经历这种迭代：“好吧，让我们试试这个，如果不行，再试试那个。”这种在整个组织中建立的、或者通过你个人生活经验积累的知识，这种痛苦最终转化为了公司的护城河。这可能是评估的产物，也可能是你构建的某种东西。我觉得这将是游戏规则的改变者。

[01:13:59] Lenny Rachitsky

English:

That is awesome. It's like turning a coal into diamond.

中文翻译:

太棒了。这就像把煤炭变成钻石。

[01:14:03] Kiriti Badam

English:

Yes.

中文翻译:

是的。

[01:14:04] Lenny Rachitsky

English:

I feel like we've done a great job helping people avoid some of the biggest issues people consistently run into building AI products. We covered so many of the pitfalls and the ways to actually do it correctly. Before we get to our very exciting lightning round, is there anything else that you wanted to share? Anything else you want to leave listeners with?

中文翻译:

我觉得我们做得很好，帮助人们避开了构建 AI 产品时经常遇到的一些大问题。我们讨论了这么多陷阱以及正确的方法。在进入令人兴奋的闪电轮（lightning round）之前，还有什么想分享的吗？有什么想留给听众的吗？

[01:14:25] Aishwarya Naresh Reganti

English:

Be obsessed with your customers. Be obsessed with the problem. AI is just a tool and try to make sure that you're really understanding your workflows. 80% of so called AI engineers, AIPMs spend their time actually understanding their workflows very well. They're not building the fanciest and the most cool models or workflows around it. They're actually in the weeds understanding their customer's behavior and data. And whenever a software engineer who's never done AI before, here's the term, look at your data. I think it's a huge revelation to them, but it's always been the case. You need to go there, look at your data, understand your users, and that's going to be a huge differentiator.

中文翻译:

痴迷于你的客户。痴迷于问题。AI 只是一个工具，确保你真正理解你的工作流。80% 的所谓 AI 工程师和 AI PM 实际上把时间花在深入理解工作流上。他们并不是在构建最花哨、最酷的模型或工作流，而是在一线理解客户的行为和数据。每当一个从未接触过 AI 的软件工程师听到“看看你的数据”这个词时，我觉得对他们来说都是巨大的启示，但事实一向如此。你需要深入其中，查看数据，理解用户，这将成为巨大的差异化优势。

[01:15:09] Lenny Rachitsky

English:

That's a great way to close it. The AI isn't the answer. It's a tool to solve the problem. With that, we have reached our very exciting lightning round. I've got five questions for both of you. Are you ready?

中文翻译:

这是一个很好的总结。AI 不是答案，它是解决问题的工具。那么，我们进入了非常令人兴奋的闪电轮。我为你们准备了五个问题。准备好了吗?

[01:15:22] Aishwarya Naresh Reganti

English:

Yay. Yes.

中文翻译:

耶，准备好了。

[01:15:24] Lenny Rachitsky

English:

All right. So you can both answer them. You can pick one which you want to answer. Either way, up to you. What are two or three books you find yourself recommending most to other people?

中文翻译:

好。你们可以都回答，也可以选一个回答。随你们。你们最常向别人推荐的两三本书是什么?

[01:15:32] Aishwarya Naresh Reganti

English:

For me, it's this book called When Breath Becomes Air, Lenny. It was written by Paul Kalanithi. I think he was an Indian original neurosurgeon who was diagnosed with lung cancer at 31 or 32. And the whole book is his memoir and just is written after he was diagnosed. And it's really beautiful, especially because I read it during COVID and all we ever wanted to do during COVID is stay alive. There are a bunch of really nice quotes within the book as well, but I remember one of them, he was kind of arguing against a very popular quote by Socrates, which is, "The unexamined life is not worth living," or something like that, which means you really need to be thinking about your choices, you need to understand your values, your mission and all of that. And Paul says, "If the unexamined life is not worth living, was the unlived life worth examining?" Which means are you spending so much time just understanding your mission and purpose that you've forgotten to live?

(01:16:32):

And I think everybody who's staying in the AI era and building and continuously going through the space of reinventing themselves need to take a pause and live for a bit, I guess. They need to stop evaling life too much.

中文翻译:

对我来说，是《当呼吸化为空气》（When Breath Becomes Air）。作者是 Paul Kalanithi。他是一位印度裔神经外科医生，在 31 或 32 岁时被诊断出肺癌。整本书是他在确诊后写的备忘录。它非常美，特别是因为我在疫情期间读了它，而那时我们唯一想做的就是活下去。书里有很多精彩的语录，我记得其中一段，他反驳了苏格拉底的名言"未经审视的人生不值得过"。苏格拉底的意思是你需要思考你的选择、价值观和使命。而 Paul 说："如果未经审视的人生不值得过，那么未经体验的人生值得审视吗？"这意味着，你是否花了太多时间去理解使命和目标，以至于忘记了去生活？

(01:16:32):

我认为在 AI 时代不断构建、不断重塑自我的每个人，都需要停下来生活一段时间。他们需要停止对生活进行过度的"评估"。

[01:16:46] Lenny Rachitsky

English:

I was going to say that. That's where my mind went. You got to write some evals for your life. Oh my God, we've gone too far.

中文翻译:

我正想说呢。我也想到这儿了。你得给你的生活写点"评估"。天哪，我们中毒太深了。

[01:16:52] Aishwarya Naresh Reganti

English:

Yep. Yeah.

中文翻译:

是的。

[01:16:53] Lenny Rachitsky

English:

Beautiful.

中文翻译:

太美了。

[01:16:53] Aishwarya Naresh Reganti

English:

That's my favorite book.

中文翻译:

那是我最喜欢的书。

[01:16:55] Kiriti Badam

English:

I like more of science fiction books. So I really like this 3 Body problem series. It's like a three book series. It has elements of grander than science fiction, life outside earth and how it impacts human decision making process. And it also has elements of geopolitics and how much important or valuable abstract science is to human progress. And then when that gets stopped, it's not noticeable in everyday life, but it can cause devastating effects. So I feel like AI helping in these areas, for example, is going to be extremely crucial. And that book is a nice example of what could happen otherwise.

中文翻译:

我更喜欢科幻小说。我非常喜欢《三体》系列。它有三本书，包含了超越科幻的宏大元素、地外生命以及它如何影响人类的决策过程。它还有地缘政治的元素，以及抽象科学对人类进步是多么重要和宝贵。当科学进步停止时，日常生活中可能察觉不到，但它会产生毁灭性的影响。我觉得 AI 在这些领域的帮助将是至关重要的。那本书是一个很好的反面教材，展示了如果不是这样会发生什么。

[01:17:35] Lenny Rachitsky

English:

Completely agree. Absolutely. Love. Might be my favorite sci-fi book except, or series even, and it's three. I have to read of all three, by the way. I find that it only got really good about one and a half books in. So if anyone's tried it and like, "What the heck is going on here?" Just keep reading and get to the middle of the second one and then it gets mind-blowing.

中文翻译:

完全同意。绝对的热爱。这可能是我最喜欢的科幻书甚至系列。顺便说一下，必须读完三本。我发现直到读到一本书半的时候才真正变得精彩。所以如果有人尝试读了却觉得“这到底在讲什么？”，请坚持读下去，读到第二本中间，你就会感到震撼。

[01:17:52] Kiriti Badam

English:

Yes.

中文翻译:

是的。

[01:17:54] Lenny Rachitsky

English:

If you love sci-fi and you're in AI, you got to read this book called A Fire Upon the Deep by Vernon Vinge. Check it out. It's incredible. I saw Noah Smith on his newsletter recommend this book and there's sequels to it, but this is the one that's so incredible. And it's actually, it turns out it's about AGI and super intelligence and all these things, and it's just so epic. And no one's heard of it.

中文翻译:

如果你喜欢科幻且从事 AI 行业，你一定要读读弗诺・文奇（Vernon Vinge）的《深渊上的火》（A Fire Upon the Deep）。去看看吧，太不可思议了。我看到 Noah Smith 在他的时事通讯里推荐过这本书。它有续集，但这本最精彩。事实证明，它讲的是 AGI、超级智能以及所有这些东西，非常宏大。而且好像没什么人听说过它。

[01:18:19] Kiriti Badam

English:

Thank you.

中文翻译:

谢谢推荐。

[01:18:20] Lenny Rachitsky

English:

There you go. I'm giving you one back. Okay, next question. What's a favorite recent movie or TV show that you've really enjoyed?

中文翻译:

不客气，回赠你一个。好，下一个问题：最近有什么喜欢的电影或电视节目吗？

[01:18:26] Aishwarya Naresh Reganti

English:

I started rewatching Silicon Valley and I think it's so true. It's so timeless. Everything is repeating all over again. Anybody who's watched it a few years ago should start rewatching it and you'll see that it's eerily similar to everything that's happening right now with the AI wave.

中文翻译:

我开始重温《硅谷》（Silicon Valley），我觉得它太真实了，永不过时。一切都在重演。几年前看过的人应该再看一遍，你会发现它与现在 AI 浪潮中发生的一切惊人地相似。

[01:18:41] Lenny Rachitsky

English:

That's a good idea to rewatch it. I love that their whole business was like an algorithm to compress, like a compression algorithm. It's like maybe a precursor to LLMs in some small way. No, I get it. All right, Kiriti, what you got?

中文翻译:

重温是个好主意。我喜欢他们整个业务就像一个压缩算法。在某种程度上，它可能是 LLM 的先驱。我懂。Kiriti，你呢?

[01:18:54] Kiriti Badam

English:

I'm going to drag this and say lot a movie or a TV show, but there's this game I picked up recently called Expedition 33. It has nothing to do with AI, but it's an incredibly well-made game in terms of the gameplay or the movie and the story and the music. It's been amazing.

中文翻译:

我要跑个题，不说电影或电视，我最近在玩一款叫《远征 33》（Expedition 33）的游戏。它与 AI 无关，但在玩法、剧情、故事和音乐方面都制作得极其精良。非常棒。

[01:19:10] Lenny Rachitsky

English:

I love that you have time to play games. That's a great sign. I love that. Someone OpenAI, I'm just imagining you're ... There's nothing else going on except just coding and having meetings.

中文翻译:

我很高兴你还有时间玩游戏。这是一个好迹象。我喜欢这个。作为 OpenAI 的员工，我原以为你除了写代码和开会什么都不干。

[01:19:20] Kiriti Badam

English:

Yeah, it has been incredibly hard to find time for that.

中文翻译:

是的，找时间玩游戏确实极其困难。

[01:19:22] Lenny Rachitsky

English:

That's good. That's a good sign. I'm happy to hear this. Okay. What's a favorite product that you've recently discovered that you really love?

中文翻译:

很好，这是个好兆头。很高兴听到这个。好，最近发现并非常喜欢的某个产品是什么？

[01:19:28] Aishwarya Naresh Reganti

English:

For me, it's Whisper Flow. I think I've been using it quite a bit and I didn't know I needed it so much. The best part is it's a conceptual transcription tool, which means if you go to Codex and start using Whisper Flow, it starts identifying variables and all of that. And it's so seamless in terms of transcription to instruction. You could say something like, "I'm so excited today. Add three exclamation marks," and it seamlessly switches. It adds those three exclamation marks instead of writing add three exclamation marks. And I think it's pretty cool. If you're not using it, you should try it.

中文翻译:

对我来说是 Whisper Flow。我最近用得挺多，以前不知道我竟然这么需要它。最棒的是它是一个概念性的转录工具，这意味着如果你在 Codex 中使用 Whisper Flow，它会开始识别变量之类的。它在转录到指令之间非常无缝。你可以说："我今天太兴奋了，加三个感叹号"，它会无缝切换，直接加上三个感叹号，而不是把"加三个感叹号"这几个字写出来。我觉得非常酷。如果你还没用过，应该试试。

[01:20:03] Lenny Rachitsky

English:

I'll do a plug. Get Whisper Flow for free for an entire year for a year for free by becoming an annual subscriber of my newsletter.

中文翻译:

我来插播个广告：成为我时事通讯的年度订阅者，即可免费获得一整年的 Whisper Flow。

[01:20:12] Aishwarya Naresh Reganti

English:

That's how I got access to it, Lenny.

中文翻译:

Lenny，我就是这么拿到的。

[01:20:14] Lenny Rachitsky

English:

There we go. I think I pitched this deal. I think people don't truly understand how incredible this is. They're like, "No way this is real. It's real." And 18 other products, lennysproductpass.com, check it out. Moving on. Kiriti.

中文翻译:

这就对了。我谈下了这个优惠。我觉得人们还没真正意识到这有多棒。他们会想："不可能吧，是真的吗？"是真的。还有其他 18 款产品，请访问 lennysproductpass.com 查看。继续，Kiriti。

[01:20:28] Kiriti Badam

English:

Awesome. I actually am a stickler for productivity. I keep experimenting new CLI tools and things which can make me faster. So I feel like a Raycast has been amazing. I've discovered all this new shortcuts that you can use to open different things, type in shortcut commands and things like that. And Caffeinate is another thing that I've recently discovered from my teammates. It helps you prevent Mac from sleeping so you can run this really long Codex task for four or five hours locally, let it build the thing and then you can wake up and be like, "Okay, this is good. I like this."

中文翻译:

太棒了。我其实是个生产力狂人。我不断尝试新的 CLI（命令行）工具和能让我变快的东西。我觉得 Raycast 非常棒，我发现了所有这些可以用来打开不同东西、输入快捷命令的新快捷键。Caffeinate 是我最近从队友那里发现的另一个好东西。它能防止 Mac 进入睡眠状态，这样你就可以在本地运行四五个小时的超长 Codex 任务，让它去构建，然后你醒来后会觉得：“好，这很棒，我喜欢。”

[01:21:02] Lenny Rachitsky

English:

That hilarious, that combo. Codex and Caffeinate. You guys need to use it, build that yourself, an OpenAI version of that, or the Codex agent should just keep your Mac from sleeping. That's so funny. By the way, Raycast, also part of Lenny's product pass. One year for your Raycast. Amazing. Yeah.

中文翻译:

这个组合太搞笑了，Codex 加 Caffeinate。你们应该自己做一个 OpenAI 版本的，或者让 Codex 智能体直接防止 Mac 睡眠。太逗了。顺便说一下，Raycast 也是 Lenny 产品通行证的一部分，免费用一年。太棒了。

[01:21:20] Aishwarya Naresh Reganti

English:

Lenny didn't tell us these folks. Yes. These are actually our favorite products.

中文翻译:

Lenny 没提前告诉我们这些。是的，这些真的是我们最喜欢的产品。

[01:21:25] Lenny Rachitsky

English:

These are just two of 19 products. No Caffeinate though. I don't know if that's even paid. Okay, let's keep going. Do you have a favorite life motto that you find yourself coming back to in work or in life?

中文翻译:

这只是 19 款产品中的两款。不过没有 Caffeinate，我甚至不知道它是不是付费的。好，继续。你们有没有什么在工作或生活中经常想起的人生格言？

[01:21:35] Aishwarya Naresh Reganti

English:

For me, I think this is one my dad told me when I was a kid and it's always stuck, which is they told it couldn't be done, but the fool didn't know it, so he did it anyway. I think be foolish enough to believe that you can do anything if you put your heart to it, especially now because you have so much data at your hand that could be pointing towards the fact that you probably will be unsuccessful. How many podcasts made it to more than a thousand subscribers or how many companies hit more than one million ARR? And there's always data to show you that you won't be successful, but sometimes just be foolish and go ahead with it.

中文翻译:

对我来说，是我父亲在我小时候告诉我的一句话，一直记在心里：“他们说这事办不成，但那个傻瓜不知道，所以他还是把它办成了。”我认为要足够“愚蠢”地相信只要用心就能做成任何事。特别是在现在，你手头有这么多数据，可能都在指向你可能会失败的事实。有多少播客能超过一千个订阅？有多少公司能达到一百万 ARR（年度经常性收入）？总有数据表明你不会成功，但有时只需保持愚蠢并勇往直前。

[01:22:12] Lenny Rachitsky

English:

That's great. Yeah.

中文翻译:

太棒了。

[01:22:13] Kiriti Badam

English:

For me, I am more of an overthinker. So I really like this quote from Steve Jobs that you can only connect the dots looking backwards. So a lot of the times there are numerous choices and you don't really know the optimal one to pick, but life works in ways that you can actually see back and be like, "Oh, these are actually beautiful in terms of how our transition." So I feel like that is extremely useful in keep moving forward, keep experimenting.

中文翻译:

对我来说，我有点想太多。所以我非常喜欢史蒂夫・乔布斯的那句话：“你只有在回顾过去时才能将点滴联系起来。”很多时候有无数选择，你真的不知道哪个是最佳选择，但生活总能让你在回头看时感叹：“噢，这些转变其实很美妙。”我觉得这对于保持前进、不断尝试非常有用。

[01:22:39] Lenny Rachitsky

English:

Final question. Whenever I have two guests on the podcast at once, I like to ask this question. What's something that you admire about the other person?

中文翻译:

最后一个问题。每当我同时邀请两位嘉宾时，我喜欢问这个问题：你最钦佩对方的一点是什么？

[01:22:48] Aishwarya Naresh Reganti

English:

I think with Kiriti, he's pretty calm and very grounded and he's always been my sounding board. I can throw a ton of ideas at him and he always comes up with, he's able to anticipate the kind of issues that might land into. And he's extremely kind and lets his work speak instead of actually doing a lot of talking, I guess. But if I had to pick one, I think he's the most incredible husband.

中文翻译:

关于 Kiriti，他非常冷静且脚踏实地，一直是我的“共鸣板”。我可以向他抛出无数想法，他总能预见到可能遇到的问题。他非常善良，让工作成果说话，而不是夸夸其谈。但如果非要选一个，我认为他是一个不可思议的丈夫。

[01:23:20] Lenny Rachitsky

English:

Reveal. Little did people know.

中文翻译:

大揭秘。大家之前都不知道。

[01:23:25] Aishwarya Naresh Reganti

English:

We've been married for four years and been the most beautiful four years of my life.

中文翻译:

我们结婚四年了，这是我生命中最美好的四年。

[01:23:31] Lenny Rachitsky

English:

Wow. Okay. How do you follow that?

中文翻译:

哇。好，这你该怎么接?

[01:23:34] Kiriti Badam

English:

Yeah, it's super hard to follow that. I would say I am extremely privileged in terms of working with really smart people in great companies in the Silicon Valley. And I feel the unique thing that stands with Aishwarya across like any other smart folks I've worked on is she has this really amazing knack of teaching and explaining something in a very understandable and easy to comprehend way. And that combined with persistence is super useful, especially in this fast-moving AI world that we are in the sense that there's so many new things coming up. It feels overwhelming, but when I hear her talk about, this is how you make sense of this entire thing, this is where it plugs in. I feel like, oh, that is so simple. I can also do that. So she empowers a lot of people by simplifying things and explaining things in the most understandable way.

(01:24:25):

So I feel that is an incredible quality.

中文翻译:

是的，这确实很难接。我想说，能在硅谷的一流公司与这么多聪明人共事，我感到非常荣幸。我觉得 Aishwarya 与我合作过的其他聪明人相比，最独特的一点是她有一种惊人的天赋，能以一种非常易懂、易于理解的方式来教学和解释事物。这种能力结合她的坚持，在目前快速发展的 AI 世界中极其有用。新事物层出不穷，让人感到不知所措，但当我听她讲解“这就是你如何理解整件事，这就是它的切入点”时，我会觉得：“噢，原来这么简单，我也能做。”她通过简化事物并以最易懂的方式解释，赋予了很多人力量。

(01:24:25):

我觉得这是一种了不起的品质。

[01:24:27] Lenny Rachitsky

English:

Amazing. How sweet. I got to do this all the time. I need more guest to do it. That was great. Okay. Final questions. Where can folks find stuff that you're working on, find you online, share your course link, and then just how can listeners be useful to you?

中文翻译:

太棒了，真贴心。我以后得经常这么问，我需要更多嘉宾这么做。太精彩了。好，最后的问题：大家可以在哪里找到你们正在做的事情？在哪里关注你们？分享一下你们的课程链接，以及听众可以如何帮助你们？

[01:24:41] Aishwarya Naresh Reganti

English:

I write a lot on LinkedIn. So if you want to listen to pragmatists who've been in the weeds, working on AI products and what they're seeing, you can follow my work. We also have a GitHub repository with about 20K stars, and that repository is all about good resources for learning AI. It's completely free. And if you like what we spoke today, we also run a super popular course. We leave a link to it on building enterprise AI products. And the course is a lot about unlearning mindsets and following a problem-first approach instead of a tool-first or a hype-first approach. So you can check that out as well. And if you don't want to do the course, we write a lot, we give out a lot of free resources, we have free sessions, so make sure you follow our work.

中文翻译:

我在 LinkedIn 上写了很多东西。如果你想听听那些在一线构建 AI 产品的务实者的见解，可以关注我。我们还有一个拥有约 2 万星的 GitHub 仓库，里面全是学习 AI 的优质资源，完全免费。如果你喜欢我们今天的谈话，我们还开设了一门非常受欢迎的课程，关于构建企业级 AI 产品，我们会留下链接。这门课程主要是关于打破旧思维，遵循“问题优先”而非“工具优先”或“热度优先”的方法。你们也可以去看看。如果你不想上课，我们也写了很多文章，提供很多免费资源和免费课程，请务必关注我们的动态。

[01:25:27] Kiriti Badam

English:

Yeah, I would also add that you can also find me on LinkedIn. I don't write a lot, I guess, but I'm super all excited to just talk to any complex product that you're building. And if you have thoughts on how you can use coding agents to make your life better or however the problems that you're seeing, always my DMs are open and we can have a great discussion.

中文翻译:

是的，我也想补充一下，你也可以在 LinkedIn 上找到我。我写得不多，但我非常乐意讨论你正在构建的任何复杂产品。如果你对如何使用编程智能体改善生活有想法，或者对你看到的问题有见解，我的私信随时敞开，我们可以进行深入讨论。

[01:25:47] Lenny Rachitsky

English:

Awesome. Well, Kiriti and Ash, thank you so much for being here.

中文翻译:

太棒了。Kiriti 和 Ash，非常感谢你们来到这里。

[01:25:52] Kiriti Badam

English:

Thank you so much.

中文翻译:

非常感谢。

[01:25:53] Aishwarya Naresh Reganti

English:

Thank you, Lenny. This was so much fun.

中文翻译:

谢谢你，Lenny。这次谈话非常愉快。

[01:25:54] Lenny Rachitsky

English:

So much fun. Bye, everyone.

(01:25:58):

Thank you so much for listening. If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify, or your favorite podcast app. Also, please consider giving us a rating or leaving a review as that really helps other listeners find the podcast. You can find all past episodes or learn more about the show at lennyspodcast.com. See you in the next episode.

中文翻译:

非常愉快。大家再见。

(01:25:58):

非常感谢大家的收听。如果你觉得本集有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客应用上订阅本节目。此外，请考虑给我们评分或留下评论，这能极大地帮助其他听众发现这个播客。你可以在 lennyspodcast.com 找到所有往期节目或了解更多信息。下期节目再见。