Andrew Ng: Building Faster with AI

【吴恩达：用AI加速创业】

Summary: Andrew Ng shares insights on accelerating startup success through AI, emphasizing execution speed, concrete ideas, and leveraging AI tools like agentic workflows and coding assistance. He highlights the importance of the application layer for opportunities and rapid prototyping to validate ideas quickly.

摘要： 吴恩达分享如何利用AI加速创业成功，强调执行速度、具体想法以及借助AI工具（如智能体工作流和编程辅助）。他指出应用层蕴含巨大机遇，并提倡快速原型验证。

⏱ Estimated Reading Time: 72 min

📚 四级生词 📚 六级生词 📚 雅思生词 📚 托福生词 📚 专八生词 📚 SAT生词 📚 考研生词 📚 GRE生词 📚 高考生词 📚 其它生词生词

Table of Contents

- 00:05:35 That's actually business already doing this, but it is concr
- 00:05:52 But if someone says, could you build an app, Gmail integrate
- 00:06:10 If you go and tell all your friends, we should use AI to opt
- 00:06:27 We can discover that much more fast, which is what's importa
- 00:06:48 So let's do that quickly.
- 00:07:11 And then after that long process, I think YC sometimes calls
- 00:07:36 It can be surprisingly good mechanism for making decisions.
- 00:07:57 And then one other thing, but many successful starters, at a
- 00:08:18 Just pivot on the dime to pursue a totally different concret
- 00:08:43 Every time you talk to a customer, you totally change your m
- 00:09:05 So when you're building a lot of applications, one of the bi
- 00:09:55 So the speed of engineering is going up rapidly, and the cos
- 00:10:20 The other type of software I do is, write, maintain producti
- 00:10:41 But in terms of building quick and dirty prototypes, we're n
- 00:11:06 And I know I'm not supposed to tell people to write insecure
- 00:11:26 And like a leaking PII, leaking sends data, that is very dam
- 00:11:48 But I think by driving the cost of a proof of concept low en
- 00:12:08 I tend to tell my teams to move fast and be responsible.
- 00:12:33 And then starting on our six, seven months ago, decided to d
- 00:13:01 And the interesting thing is, if you're even half a generati
- 00:13:21 But because the cost of software engineering is going down,
- 00:13:38 Some of you may have heard of Jeff Bezos' terminology of a t
- 00:13:58 Once you build on top of a certain tech stack, you know, set
- 00:14:17 I don't want to over-hype it. We don't do that all the time.
- 00:14:36 Over the last year, a bunch of people advised, have advised
- 00:15:00 When we moved from assemblies, high-level languages like Cob
- 00:15:20 And as coding becomes easier, more people should learn to co
- 00:15:44 And I think I'm pretty little bit ahead of her for most busi
- 00:16:06 And one of my team members knew art history.
- 00:16:33 And I could never have the control that my collaborates are
- 00:16:55 And learning to code, not that you need to write the code yo
- 00:17:19 A lot more of my teams have started to complain that the bot
- 00:17:38 It was this like p.m. product manager at the engineering rat
- 00:17:57 So literally yesterday, one of my teams came to me.
- 00:18:19 I still don't know if this proposed I heard yesterday is a g
- 00:18:41 That helps you get faster as well.

- 00:19:02 And if you're subject matter to expert, this is actually sur
- 00:19:27 When I travel, I often send the hotel lobby.
- 00:19:47 And very respectfully, ask strangers, hey, we're building th
- 00:20:08 Stem prototypes, 100 testers.
- 00:20:30 It depends on how many users you have.
- 00:20:54 I wish we were able to use the first tactic, to make high qu
- 00:21:15 That turns out to be really important.
- 00:21:41 So many people, including non-tech people, have good instinc
- 00:22:00 So there are a lot of people that are really good at marketi
- 00:22:24 Whereas if you need an HR problem, you can find someone that
- 00:22:40 How do you get a voice out to low latency?
- 00:23:00 It feels like if you don't know the right answer, and most o
- 00:23:26 The other reason why I find, staying on top of AI really hop
- 00:24:01 And this creates a lot of new opportunities for startups to
- 00:24:19 But if you get a second building block, like you also know h
- 00:24:39 Get more building breaks, get more building breaks, and very
- 00:25:09 But when I look at the Deep Learning Diode I course catalog,
- 00:25:30 I find that there are many things that matter for startup, n
- 00:25:52 I find that as a, as exactly if I'm judge on the speed and q
- 00:26:10 And if you haven't learned to go to coffee shop and talk to
- 00:26:30 APPLAUSE I have a quick question.
- 00:27:02 And so for a long time, there'll be a lot of things that hum
- 00:27:22 But so people that know how to use AI to get computers to do
- 00:27:42 My question is about the future of compute.
- 00:28:06 So turns out there's one framework you can use for deciding
- 00:28:34 So one of my mental filters is there's certain hype narrativ
- 00:28:59 AI is so powerful soon no one will even have a job anymore.
- 00:29:19 Yes, Jasper, Brandon Trouville, small number of companies go
- 00:29:41 I think we have a lot of room to run still for terrestrial G
- 00:30:02 But what are some of the most dangerous biases or over hyped
- 00:30:36 I find myself not using the term AI safety that much, not be
- 00:30:57 Like using electric electric motor can be used to build a Da
- 00:31:16 It is how you apply it that makes it safer unsafe.
- 00:31:37 I think just one or two days ago, there was a Wall Street Jo
- 00:32:02 And so these high narratives do keep on getting amplified.
- 00:32:26 Whatever great mode product or feature you have can be repli

- 00:32:48 There goes market, channel, competitors, technology, mode.
- 00:33:06 Do you have a channel to get to customers?
- 00:33:28 But enterprise products, sometimes, if you have a, maybe mod
- 00:33:56 But I feel like at this moment in time, the number of opport
- 00:34:25 Although this is important to figure a long way.
- 00:35:03 So which is different from static engineering?
- 00:35:25 Only a small number of startups are lucky enough to have use
- 00:35:45 But it's actually really difficult to get to point where you
- 00:36:10 So for example, if you build a customer service chatbot or o
- 00:36:38 So for example, have a lot of products that build on top of
- 00:36:59 And so the model we use week by week, you know, sometimes ou
- 00:37:19 Switching costs for the orchestration platforms is a little
- 00:37:43 But another score of thought is that there'll be personal tu
- 00:38:00 Like don't think the disruption is here yet.
- 00:38:19 Deep learning data AI slash avatile.
- 00:38:42 But I think what I'm seeing is frankly tons of experimentati
- 00:39:02 The reality is work is complex, right?
- 00:39:28 So I think I think we should all just keep working on it.
- 00:39:53 So I was curious, how do you think, you know, us as AI build
- 00:40:18 I know it sounds simple, but that's actually really hard to
- 00:40:33 So I hope more people will do that.
- 00:40:56 So then everyone learned to code and then they got better.
- 00:41:15 And my question is also about education.
- 00:41:43 I think the knowledge will diffuse.
- 00:42:05 And one of the reasons is there are two gatekeepers and Jord
- 00:42:37 Which thank goodness we shut down where the put in place rea
- 00:43:17 That's always a side for innovation and prevent the diffusio
- 00:43:42 But despite to protect open source, we've been winning, but

---

00:00:00

▷ It's really great to see all of you.

很高兴见到大家。

What I want to do today since this is this build that Startup School is share of you some lessons I've learned about building startups at AI Fund.

今天我想在创业学院分享我在AI基金创办公司时学到的经验。

AI Funds Adventure Studio, and we build an average of about one startup per month.

AI基金冒险工作室平均每月创立一家初创公司。

Because we co-founded startups were in their writing code, talking to customers, design on features, that are in pricing, and so we've done a lot of reps of not just washing other build startups, but actually being in the weeds, building startups with entrepreneurs.

因为我们联合创立的公司涉及写代码、与客户沟通、功能设计和定价，我们不仅观察创业过程，更深入参与其中。

Back to Top

00:00:29

And what I want to do today is share of you some of the lessons I've learned building startups, especially around this changing AI technology and what it enables.

今天我想分享在AI技术变革背景下创业的经验。

And it'll be focused on the theme of speed.

主题将围绕"速度"展开。

So it turns out that, but those of you that want to build a startup, I think a strong predictor for startups also success is an execution speed.

对创业者而言，执行速度是成功的关键预测指标。

Back to Top

00:00:50

And I actually have a lot of respect for the entrepreneurs and executives that can just do things really quickly.

我十分敬佩行动迅速的创业者和高管。

And new AI technology is enabling startups to go much faster.

新AI技术正让创业速度大幅提升。

So what I want to do is share of you some of those best practices, which are frankly changing every two to three months though, to let you get that speed that hopefully lets you have higher odds of success.

我将分享当前的最佳实践（尽管每2-3个月就会更新），帮助提高成功概率。

Back to Top

00:01:11

Before diving to speed, a lot of people ask me, hey Andrew, where are the opportunities for startups?

在讨论速度前，很多人问我创业机会在哪里。

So this is what I think of as a AI stack, where at the lowest level are the semiconductor companies, then the clouds are high by scalars built on top of that, a lot of the AI foundation, all the companies build on top of that.

我认为AI技术栈底层是半导体公司，之上是云平台和基础模型。

And even though a lot of PR excitement and hype has been on these technology layers, it turns out that almost by definition, the biggest opportunities have to be at the application layer because we actually need the applications to generate even more revenue so that they can afford to pay the foundation, cloud and semiconductor technology layers.

尽管技术层备受关注，但最大机会必然在应用层，因为应用才能创造足够收入支撑底层技术。

Back to Top

00:01:52

So for whatever reason, media and social media tends not to talk about the application layer as much, but for those of you thinking of building startups, almost by definition, the biggest opportunities have to be there, although of course, the opportunities are all layers of the set.

媒体较少讨论应用层，但对创业者而言这里蕴藏最大机会。

One of the things that's changed a lot over the last year, and in terms of AI tech trends, if you ask me, what's the most important tech trend in AI?

过去一年最重要的AI趋势是什么？

I would say is the rise of a gentigai.

我认为是智能体AI的崛起。

Back to Top

00:02:17

And about a year and a half ago, when I started to go around and give talks to try to convince people that AI agents might be a thing, I did not realize that around last summer, a bunch of marketers would get a hold of this term and use it as a sticker and slap it on everything in sight, which made it almost lose some of its meaning.

一年半前我推广AI智能体概念时，没想到去年夏天营销人员会滥用这个术语。

But I'll share with you from a technical perspective, why I think a gentigai is exciting and important, and also opens up a lot more startup opportunities.

但从技术角度看，智能体AI确实激动人心且能创造新机会。

So it turns out that the way a lot of us use LOMs is to prompt it to have it during an output.

目前多数人通过提示词使用大模型。

Back to Top

00:02:51

▷ And the way we have an LOM output something is as if you're going to a human, or in this case, an AI, and asking it to please type on an essay for you by writing from the first word to the last word, all in one go, whatever, using backspace.

这就像要求人类或AI一次性完成整篇文章。

▷ And humans, we don't do our best writing, being forced to type in this linear order.

人类被迫线性写作时也难以发挥最佳水平。

▷ And in terms of neither does AI, but despite the difficulty of being forced to write in this linear way, our LOMs do surprisingly well.

AI同样如此，但大模型在线性写作中表现惊人。

Back to Top

00:03:18

▷ With a gentigrap, those, we can go to AI system and ask it to please first write an essay online, then do some web research if it needs to, and fetch some web pages to put in their own context, then write the first draft, then read the first draft in critique, and revise it, and so on.

通过智能体，我们可以让AI系统先列提纲、做网络研究、写初稿、自我修订。

▷ And so we end up with this iterative workflow, where your model does some thinking, and some research, does some revision, goes back to do more thinking, and by going around this loop many times, it is slower, but the deliver is a much better work product.

这种迭代工作流虽慢但产出质量更高。

▷ So for a lot of projects, AI fund has worked on everything from pulling out complex compliance documents to medical diagnosis, to reasoning about complex legal documents.

AI基金的项目涉及合规文件、医疗诊断和法律文件分析。

Back to Top

00:03:58

We found that these agentic workflows are really a huge difference within working, versus not working, but a lot of the work that needs to be done, a lot of available businesses to be built still, will be taking workflows, existing only workflows, and figuring out how to implement them into these types of agentic workflows.

智能体工作流效果显著，未来需将现有流程转化为智能体工作流。

So just to update the picture for the AI stack, what has emerged over the last year is a new agentic orchestration layer that helps application builders orchestrate or coordinate a lot of calls to the technology layers underneath.

AI技术栈新增了智能体编排层。

And the good news is, the orchestration layer has made it even easier to build applications, but I think the basic conclusion that the application layer has to be the most valuable layer to stack still holds true.

编排层简化了应用开发，但应用层仍最具价值。

Back to Top

00:04:43

With a bias or focus on the application layer, let me now dive into some of the best practices of learning, but how startups can move faster.

聚焦应用层，我将分享加速创业的最佳实践。

It turns out that at AI fund, we only focus on working on concrete ideas.

AI基金只关注具体想法。

▷ So to me, a concrete idea, a concrete product idea, is one that's specified enough detail that an engineer can go and build it.

具体想法是指工程师能立即执行的明确方案。

▷ So for example, if you say, let's use AI's optimized healthcare assets, that's actually not a concrete idea.

"用AI优化医疗资源"就不是具体想法。

Back to Top

bilibili 00:05:14

▷ It's too vague.

这太模糊。

▷ If you tell me it's a very software, do you use AI's optimized healthcare assets? Different engineers would do totally different things.

不同工程师会做出完全不同产品。

▷ And because it's not concrete, you can't build it quickly and you don't have speed.

不具体就无法快速执行。

▷ In contrast, if you had a concrete idea, like let's write software to let hospitals, let patients with MRI machines laws online to optimize usage, I don't know if this is a good or a bad concrete idea.

相比之下，"开发MRI机器在线预约系统"就是具体想法。

Back to Top

bilibili 00:05:35

That's actually business already doing this, but it is concrete and that means engineers can build it quickly.

虽然已有类似业务，但工程师能快速实现。

If it's a good idea, you find out there's not a good idea, you will find out by having concrete ideas by as you speed.

通过具体想法能快速验证可行性。

Or someone would say, let's use AF email personal productivity.

"用AI提升邮件效率"也不具体。

Too many interpretations of that, that's not concrete.

解释空间太大。

Back to Top

---

bilibili 00:05:52

But if someone says, could you build an app, Gmail integrate the automation that uses, let's use the right prompt, so it's my filter and tag email, is that it's concrete?

而"开发Gmail插件用AI自动分类邮件"就很具体。

I could go build that this afternoon.

这样的方案下午就能开工。

So concrete is by as you speed.

具体化才能加速。

And the deceptive thing for a lot of entrepreneurs is the vague ideas tend to get a lot of crudals.

具有欺骗性的是，模糊想法常获得更多认可。

Back to Top

00:06:10

If you go and tell all your friends, we should use AI to optimize the use of healthcare assets.

如果你告诉朋友"用AI优化医疗资源"。

Everyone will say that's a great idea, but it's actually not a great idea, at least in a sense of being something you can build.

大家都会说好，但这并非可执行的好想法。

It turns out when you're vague, you're almost always right.

模糊时你总是"正确"。

But when you're concrete, you may be right or wrong.

具体化后可能对错分明。

Either way, it's fine.

但两者都好。

Back to Top

00:06:27

We can discover that much more fast, which is what's important for us to start up.

快速验证对创业至关重要。

In terms of executing concrete ideas, I find that AI fun, I ask my team to focus on concrete ideas because a concrete idea gives clear direction, and the team can run really fast, to build it and either validate it, prove it out, or falsify it and conclude it doesn't work.

执行具体想法能为团队提供明确方向，快速验证或否定。

Either way, it's fine.

两种结果都好。

Back to Top

00:06:48

So let's do that quickly.

所以要快速行动。

And it turns out that finally, good concrete ideas usually requires someone, could be you, could be a subject matter expert.

好的具体想法通常需要领域专家长期思考。

Think about a problem for a long time.

长期思考某个问题。

So for example, actually before you're starting Coursera, I spent years thinking about online education, top of the users, holding my own intuitions about what would make a good, a tech platform.

比如创立Coursera前，我花了数年思考在线教育。

Back to Top

00:07:11

And then after that long process, I think YC sometimes calls it wondering the idea maze.

YC称之为"想法迷宫探索"。

But after thinking about it for a long time, you find that the guts of people that thought about this for a long time can be very good about rapidly making decisions.

长期思考后，直觉决策会更准确。

As in, after you've thought about this, thought the customers have done for a long time, if you're also as expert, should I build this feature or that feature, you know, the gut, which is an instantaneous decision, can be actually a surprisingly good proxy.

对客户需求有深刻理解后，瞬时决策反而更可靠。

Back to Top

00:07:36

It can be surprisingly good mechanism for making decisions.

这是很好的决策机制。

And I know I work on AI, you might think I'll say, oh, we need data.

虽然我从事AI，但并非所有决策都需要数据。

And of course, I love data.

我确实重视数据。

The turns out getting data for a lot of starters is just slow mechanism for making decisions.

但对初创公司，收集数据可能太慢。

And subject matter expert with good guts is often a much better mechanism for making a speedy decision.

领域专家的直觉反而能加速决策。

Back to Top

00:07:57

And then one other thing, but many successful starters, at any moment in time, you're pursuing one very clear hypothesis, they're building out and trying to sell the value of all the spot.

成功初创公司通常专注验证一个明确假设。

▷ And a startup doesn't have resources to hedge and try 10 things at the same time.

初创公司资源有限，无法同时尝试多个方向。

▷ So pick one, go for it.

所以专注一个方向。

▷ And if data tells you to use faith in that idea, that's totally fine.

如果数据否定它，完全没问题。

Back to Top

---

bilibili 00:08:18

▷ Just pivot on the dime to pursue a totally different concrete idea.

立即转向新具体想法。

▷ So that's what often feels like an AI fund.

这就是AI基金的工作方式。

▷ We're pursuing one thing doggily with determination until the world tells us we were wrong, then change and pursue a totally different thing with equal determination and equal dogginess.

我们专注一个方向直到被证明错误，然后以同样决心转向新方向。

▷ And one of the pads in our scene, if every piece of new data calls you to the pivot, it probably means you're starting off from two week of base of knowledge.

如果每个新数据都让你改变方向，可能说明知识储备不足。

Back to Top

---

bilibili 00:08:43

Every time you talk to a customer, you totally change your mind.

每次与客户交流都彻底改变想法时。

Part of it means you don't know enough about that sector yet to have a really high quality concrete idea and finding someone to start about a subject for longer may get drawn to better power.

意味着需要更深入了解该领域才能形成高质量具体想法。

In order to go faster, the other thing often think about is the built feedback loop, which is rapidly changing when it comes to how we build with AI coding assistance.

要加速，还需建立快速反馈循环，AI编程辅助正改变这一过程。

Back to Top

bilibili 00:09:05

So when you're building a lot of applications, one of the biggest risks is customer acceptance, a lot of startups struggle, not because we can't build whatever we want to build, but because we build something and it turns out, no, we cares.

许多创业失败源于产品无人问津，而非技术实现。

And so for a lot of the way I build startups, especially applications, less so deep tech, less so technology startups, but definitely application startups is often built software, so this is an engineering toss, and then we will get feedback from users and just a product management toss, and then we'll go back, basically use the feedback, we'll tweak our views on what to build, go back to write more software, and we go around this loop many, many times, iterate toward product market fit.

我们通过"开发-用户反馈-调整"的循环迭代实现产品市场匹配。

And it turns out that with AI coding assistance, which Andre talked about as well, rapid engineering is becoming possible in a way that just was not possible, it's becoming much more feasible.

AI编程辅助使快速工程成为可能。

Back to Top

00:09:55

▷ So the speed of engineering is going up rapidly, and the cost of engineering is also going down rapidly.

工程速度提升，成本下降。

▷ This changes the mechanisms by which we drive startups around this loop.

这改变了创业迭代的方式。

▷ When I think about the software I did I do, I maybe put into two major buckets.

我将软件开发分为两类。

▷ Sometimes I've built quick and dirty prototypes to test an idea, you say, building new customers service chat box, building AI to process legal documents, whatever, but the quick and dirty prototype to see if we think it works.

快速原型验证想法可行性。

Back to Top

00:10:20

▷ The other type of software I do is, write, maintain production software, maintain legacy software, but these massive production ready go bases.

另一类是生产级软件开发维护。

▷ Depending on which analyst report you trust, it's been hard to find very rigorous data on this.

不同分析报告数据差异较大。

You know, when writing production quality code, maybe we're 30% to 50% faster with AI assistance, hard to find a rigorous number, maybe, these falls to be.

AI辅助使生产代码效率提升30-50%。

Back to Top

00:10:41

But in terms of building quick and dirty prototypes, we're not 50% faster.

但在原型开发中，效率提升不止50%。

I think we're easily 10 times faster, maybe much more than 10 times faster.

可能达到10倍甚至更高。

And there are a few reasons for this.

原因有几个。

When you're building standalone prototypes, this less integration with legacy software infrastructure legacy data needed.

原型开发较少依赖旧系统集成。

Also, the requirements for reliability, even scalability, even security are much lower.

对可靠性、扩展性和安全性要求较低。

Back to Top

00:11:06

And I know I'm not supposed to tell people to write insecure code, right?

虽然不该鼓励写不安全代码。

Feels like the wrong thing to say.

这么说不太合适。

▷ But I routinely go to my team and say, good, hey, write insecure code, because if this software is only gonna run on your laptop, and if you don't plan to maliciously hack your own laptop, it's fine to have insecure code, right?

但我常对团队说：在原型阶段可以写不安全代码，如果只在你自己的笔记本上运行。

▷ But of course, after it seems you're working, please do make a secure before you ship it to someone else.

当然正式发布前必须确保安全。

Back to Top

---

bilibili 00:11:26

▷ And like a leaking PII, leaking sends data, that is very damaging.

泄露个人数据会造成严重损害。

▷ So before you ship it, make a secure and scalable, but they were just assessing it, it's fine.

所以发布前要做好安全加固。

▷ And so I find increasingly, SOTAS will systematically pursue innovations by building 20 prototypes to see what works, right?

因此我们常通过构建20个原型来系统化探索创新。

▷ Because I know that there's some ads in AI, a lot of proof of concepts don't make into production.

虽然很多概念验证不会进入生产环境。

Back to Top

00:11:48

But I think by driving the cost of a proof of concept low enough, it's actually fine, it lots of proof of concepts, don't see the light of day.

但我认为只要将概念验证的成本压得足够低，即使很多POC最终未能落地也是可以接受的。

And I know that the mantra, move fast and break things, got a bad rep, because it broke things.

我知道"快速行动、打破常规"这句口号因实际破坏性而声誉不佳。

And some teams took away from this that you should not move fast, but I think that's a mistake.

一些团队因此认为不该追求速度，但我觉得这是误解。

Back to Top

00:12:08

I tend to tell my teams to move fast and be responsible.

我通常告诉团队要"快速行动且负责任"。

And I think they actually found ways to move really quickly while still being responsible.

事实上他们已找到了既快速推进又保持责任感的方法。

And in terms of the AI-assisted coding landscape, I think was it three, four years ago, code autocomplete, right? Popularized by GitHub copilot.

在AI辅助编程领域，三四年前代码自动补全功能（如GitHub Copilot）开始普及。

And then there was a cursor, wind-served generation of AI-enabled IDEs, we didn't use wind-served cursor quite a lot.

随后出现以Cursor为代表的第二代AI集成开发环境，但我们并未频繁使用Cursor。

Back to Top

00:12:33

▷ And then starting on our six, seven months ago, decided to do this new generation of agent coding assistance, including, like you're using O-Fe a lot for coding, cloud coding is fantastic.

六七个月前我们开始采用新一代智能编码助手，比如频繁使用的O-Fe，云端编程体验极佳。

▷ Since quad-4 release is become, and I was beginning a few months, I may use something different, but the tools evolving really rapidly.

自Quad-4版本发布后，虽然我几个月前可能用其他工具，但这类工具迭代速度惊人。

▷ But I think cloud code X, this is a new generation of highly-agent coding assistance that is making developer productivity keep on growing.

我认为Cloud Code X这类新一代高智能编码助手正在持续提升开发效率。

Back to Top

00:13:01

▷ And the interesting thing is, if you're even half a generation or one generation behind, is actually makes a big difference compared to if you're on top of the latest tools.

有趣的是，哪怕落后半代或一代工具，其效率差距会非常明显。

▷ And the final idea of thinking really different approaches to software engineering now, compared to even three or six months ago.

现在的软件工程方法论与三个月或半年前已截然不同。

▷ One surprising thing is, we're used to thinking of code as this really valuable artifact, which is so hard to create.

令人惊讶的是，我们曾将代码视为极难创造的高价值产物。

Back to Top

00:13:21

But because the cost of software engineering is going down, code is much less of a valuable artifact as it used to.

但随着开发成本降低，代码的价值属性已大幅减弱。

Some on Teams where, with completely rebuilt the code base, three times the last month, right?

有些团队甚至上个月就彻底重构了三次代码库。

Because it's not that hard anymore to just completely rebuild the code base.

因为现在彻底重写代码库已非难事。

Pick a new data schema is fine, because the cost of doing that has plummeted.

选择新数据架构也很轻松，因其成本已骤降。

Back to Top

00:13:38

Some of you may have heard of Jeff Bezos' terminology of a two-way door versus a one-way door.

你们可能听过贝索斯提出的"双向门"与"单向门"决策理论。

A two-way door's decision they can make. If you change your mind, come back out, you know, reverse it relatively cheaply.

"双向门"决策可低成本撤销。

Whereas the one-way door is, you make a decision and you change your mind is very costly, very difficult to reverse.

而"单向门"决策的变更代价极高。

So choosing the software architecture of your tech stack, used to be a one-way door.

过去选择技术栈的软件架构属于"单向门"决策。

Back to Top

00:13:58

Once you build on top of a certain tech stack, you know, set the data schema really hard to change it.

一旦基于某技术栈开发，数据架构就难以变更。

So that used to be a one-way door.

因此过去这是"单向门"。

I don't want to say it's totally a two-way door, but I find that my team were more often built on a certain tech stack a week later, change your mind.

虽不能说完全变成"双向门"，但我的团队常在一周后就更换技术栈。

Let's throw the code base away and redo it from scratch on the new tech stack.

直接废弃旧代码库，用新栈重写。

Back to Top

00:14:17

I don't want to over-hype it. We don't do that all the time.

并非夸大其词，我们并非总是如此。

There are still costs to redoing that.

重写仍有成本。

I find my team is often rethinking what is a one-way door and what's now a two-way door because the cost of software engineering is so much lower now.

由于开发成本降低，团队常需重新评估决策类型归属。

And maybe going a little bit beyond software engineering, I feel like it's actually a good time to impart everyone to build AI.

超越软件工程范畴，现在正是普及AI构建的大好时机。

Back to Top

00:14:36

Over the last year, a bunch of people advised, have advised others not to learn to code on the browser AI will automate it.

过去一年很多人鼓吹"无需学编程，AI将自动化一切"。

I think we'll look back on this as some of the worst career advice ever given.

这终将被视为最糟糕的职业建议。

Because as better tools make software engineering easier, more people should do it, not fewer.

因为工具越强大，越该有更多人参与开发。

So when many decades ago, the world moved from punch cards to keyboard and terminal that made coding easier.

就像数十年前从打孔卡转向键盘终端提升了编程便利性。

Back to Top

00:15:00

When we moved from assemblies, high-level languages like Cobalt, there actually people arguing back then that now we have Cobalt, we don't need programmers anymore.

当从汇编转向COBOL等高级语言时，也有人声称不再需要程序员。

That people actually don't pay for this to that effect.

这种论调显然错误。

But of course, there was wrong and program languages made easier to code and more people learn to code.

实际上编程语言简化后，更多人可以学习编程。

Check sentences, IDs, IDs, AI coding assistance.

检查语句、IDE、AI编程辅助。

Back to Top

---

bilibili 00:15:20

And as coding becomes easier, more people should learn to code.

编程越简单，越该全民学习。

I have a controversial opinion, which is, I think it's actually time for everyone of every job role to learn to code.

我的争议性观点是：所有岗位都该学编程。

And in fact, on my team, my CFO, my head of talent, my recruiters, my friend desk, a person, all of them knew how to code.

我的团队中CFO、人才主管、招聘专员、前台都懂编程。

And I should see all of them performing better at all of their job functions because they can code.

编程能力确实提升了他们的工作表现。

Back to Top

bilibili 00:15:44

And I think I'm pretty little bit ahead of her for most businesses not there yet.

虽然大多数企业尚未达到这种程度。

But in the future, I think when part everyone to code, a lot of people can be more productive.

但我认为未来全民编程将大幅提升生产力。

I want to share with you one lesson I learned as well on why we should have people learn to do this, which is when a teacher in Genese VIF everyone on Coursera, we needed to generate background art like this using my journey.

以Coursera课程背景图制作为例：当需要MidJourney生成艺术图时，

Back to Top

bilibili 00:16:06

And one of my team members knew art history.

团队中懂艺术史的成员

And so he could prompt my journey with the genre, the palette, the artistic inspiration, had a very good control over the images he generated.

能通过流派、色调等提示词精准控制产出。

So we end up using all of Tommy's generated images.

因此我们最终全部采用他的作品。

Whereas in Contrast, I don't know art history.

而我不懂艺术史，

And so when I prompt your image generation, I could write, please make pretty pictures of robots for me.

只能笼统提示"生成机器人美图"，

Back to Top

00:16:33

And I could never have the control that my collaborates are good.

无法达到他的控制精度。

And so I could generate as good images as he could.

自然无法产出同等质量作品。

And I think with computers, one of the most important skills of the future is the ability to tell a computer exactly what you want so they'll do it for you.

未来核心技能是精准向计算机表达需求。

And there will be people that have that deeper understanding of computers that will be able to command the computer to get the outcome you want.

对计算机理解更深者将能更好驾驭它们。

Back to Top

00:16:55

And learning to code, not that you need to write the code yourself. C or AI to code for you seems like it will remain the best way to do that for a long time.

学习编程（未必亲自写码）仍是掌握该技能的最佳途径。

▷ With software engineering becoming much faster, the other interesting dynamic I'm seeing is that the product management work, getting user feedback, deciding what features to build, that is increasingly the bottleneck.

随着开发加速，产品管理和用户反馈反而成为瓶颈。

▷ And so I'm seeing very interesting dynamics in multiple teams over the last year.

过去一年观察到有趣现象：

Back to Top

00:17:19

▷ A lot more of my teams have started to complain that the bottleneck on product engineering and design because the engineers have gotten so much faster.

由于工程师效率提升，更多团队开始抱怨产品设计环节拖累。

▷ Some machine trends I'm seeing.

观察到几个趋势：

▷ Three, four, five years ago, Silicon Valley used to have these slightly suspicious rules of thumb, but nonetheless, rules of thumb will have 1 p.m. to four engineers or 1 p.m. to seven engineers.

三五年前硅谷流行1名产品经理配4-7名工程师的经验法则。

Back to Top

00:17:38

▷ It was this like p.m. product manager at the engineering ratio, right?

即产品经理与工程师的比例。

▷ I think we're great in the salt, but those typical are the 1 p.m. to six, seven engineers.

虽然比例各异，但通常是1:6或1:7。

▷ And with engineers becoming much faster, I don't see product management work designing what to build, becoming faster at the same speed that engineering is.

如今工程师效率提升后，产品管理效率并未同步增长。

▷ I'm seeing this ratio shift.

该比例正在变化。

Back to Top

00:17:57

▷ So literally yesterday, one of my teams came to me.

比如昨天就有团队提议

▷ And for the first time, when we're planning a hit conference project, this team proposed to me not have 1 p.m. to four engineers, but have 1 p.m. to 0.5 engineers.

将标准比例改为1名产品经理配0.5名工程师。

▷ So the team I should propose to me, I still know that it's a good idea.

虽然不确定是否合理，

▷ For the first time in my life, I saw your managers propose to me, having twice as many p.m. as engineers was a very interesting dynamic.

但产品经理数量反超工程师的现象值得关注。

Back to Top

00:18:19

I still don't know if this proposed I heard yesterday is a good idea, but I think it's a sign of where the world's going.

这或许预示着行业方向。

And I find that as a p.m.s, they can code or engineers with some product instincts often end up doing better.

懂编程的产品经理或具产品思维的工程师表现更优。

The other thing that found important for startup leaders is because engineering is going so fast, if you're good tactics, we're getting rapid feedback to shape your perspective on what to build faster.

初创领导者需善用快速反馈调整产品方向。

Back to Top

00:18:41

That helps you get faster as well.

这能加速决策循环。

So I'm going to go through a portfolio of tactics for getting product feedback to keep shaping what you will decide to build.

以下是一套获取产品反馈的方法：

And we're going to go through a list of the faster, maybe less accurate, the slower one-acquired tactics.

按速度从快到慢排序。

So the fastest tactic for getting feedback is, look for the product yourself and just go by your gut.

最快但欠精准：凭直觉自查产品。

Back to Top

00:19:02

▷ And if you're subject matter to expert, this is actually surprisingly good.

领域专家用此法效果不错。

▷ If you know what you're doing. Lubit's slower is go ask the friends of teammates to get feedback to play your product and get feedback.

次快：邀请队友朋友试用反馈。

▷ Lubit's slower is ask the attend strangers for feedback.

更慢：收集陌生人反馈。

▷ It turns out when I built products, one of the most important skills I think I learned was how to send to the coffee shop, how to sit in there.

我常驻咖啡厅或酒店大堂，

Back to Top

00:19:27

▷ When I travel, I often send the hotel lobby.

礼貌邀请路人试用产品。

▷ It turns out, learn to spot places in high-foot traffic and very respectfully, grab strangers and ask them for feedback on whatever I'm building.

高人流区域是理想测试场。

▷ This used to be easier, was less known.

以前这样做更容易。

▷ When people recognize you, it's a little bit more awkward.

现在被认出时会尴尬。

I found that I've actually sat with teams at the hotel lobby, very high-foot traffic.

但酒店大堂测试法仍有效。

Back to Top

---

bilibili 00:19:47

And very respectfully, ask strangers, hey, we're building the same, do you mind taking a look?

礼貌请求："我们在开发这个，能请您看看吗？"

Oh, and I actually learned in the coffee shop that a lot of people working, a lot of people don't want to be working.

咖啡厅里很多人其实想暂时逃离工作。

So we give them, excuse me, distracted. They're very happy to do that too.

因此很乐意协助测试。

But I've actually made tons of product decisions in the hotel lobby and the coffee shop collaborators, just like that.

我在这些场所做出了无数产品决策。

Back to Top

---

bilibili 00:20:08

Stem prototypes, 100 testers.

进阶方法：百人规模原型测试。

If you have access, the logic goes to users.

若有条件直接接触用户。

Stem prototypes are more users.

更大规模测试。

And these get to be slow and slow tactics.

速度依次递减。

And I know Silicon Valley, we like to talk about A, B testing.

虽然硅谷推崇A/B测试，

Of course, I do a ton of A, B testing.

我也大量使用，

But contrary to what many people think, A, B testing is not one of the slowest tactics in my menu, because it's just slow to ship it.

但它其实很耗时。

Back to Top

00:20:30

It depends on how many users you have.

且依赖用户基数。

So the other thing is, as you use anything but the first tactic, some teams will look at the data and make a decision.

使用非直觉方法时，团队常直接依据数据决策。

But the missing piece is, when I AB test something, I don't just use the result of AB test to pick product A or product B.

但A/B测试的价值不止于选择方案。

My team will often sit down and look carefully to the data to hone our instincts, to speed up, to improve the rate.

我们通过分析数据打磨决策直觉。

Back to Top

00:20:54

▷ I wish we were able to use the first tactic, to make high quality decision.

虽然理想状态是用直觉做高质量决策。

▷ You're often sit down and think, gee, I thought, this product name will work better than their product name.

比如曾认为某个命名更优，

▷ Clearly, my mental model that uses wrong, so we sit down and think, to update our mental model, using all of that data, to improve the quality of our guts on how to make product decision faster.

但数据证明原认知错误，此时需要修正思维模型。

Back to Top

00:21:15

▷ That turns out to be really important.

这过程至关重要。

▷ All right. So talk about concrete ideas, speed up engineering, speed up product feedback.

总结：加速工程与反馈循环。

▷ This is one last thing we'll touch on, which is a scene that, understanding AI, actually makes you go faster.

最后强调：理解AI能加速发展。

And here's why.

原因如下：

As an AI person, maybe I'm biased to be pro AI, but I want to share you why.

作为AI从业者或许带偏见，但事实如此。

So it turns out that when it comes to mature technology, like mobile, many people have had smartphones for a long time, we kind of know what a mobile app can do, right?

成熟技术（如移动应用）的潜力已被充分认知。

Back to Top

00:21:41

So many people, including non-tech people, have good instincts about what a mobile app can do.

非技术人员也了解移动应用的边界。

If you look at mature job roles, they sales marketing, HR legal, they're all really important, are really difficult.

成熟岗位如销售、HR等虽重要，

But you know, there are enough marketers that have done marketing for long enough, and the marketing tactics haven't changed that much in the last year.

但你知道，有足够多的营销人员从事营销工作足够长时间，而且营销策略在过去一年里没有太大变化。

Back to Top

00:22:00

So there are a lot of people that are really good at marketing, and it's really important, really hard, but that knowledge is relatively diffused because the knowledge of how to do HR, like it hasn't changed dramatically, you know, in the last six months.

所以有很多人非常擅长营销，这非常重要，也非常困难，但这些知识相对分散，因为像人力资源这样的知识在过去六个月里并没有发生巨大变化。

But AI is the emerging technology, and so the knowledge of how to do AI really well is not widespread.

但AI是新兴技术，因此如何真正做好AI的知识并不普及。

And so teams that actually get it, they understand AI do have an advantage over teams that don't.

因此，真正理解AI的团队确实比不理解的团队有优势。

Back to Top

00:22:24

Whereas if you need an HR problem, you can find someone that knows how to do it well, probably.

而如果你需要解决人力资源问题，你很可能可以找到知道如何做好的人。

But if an AI problem, knowing how to actually do that, could put your head of other companies.

但如果是AI问题，知道如何真正解决它，可能会让你领先于其他公司。

So things like what accuracy can you get for a customer service chat box?

比如，你能为客服聊天机器人获得多高的准确率?

You know, should you problem a fine team using a journey workflow?

你知道，你是否应该用一个旅程工作流来优化团队?

Back to Top

00:22:40

How do you get a voice out to low latency?

如何实现低延迟的语音输出?

There are a lot of these decisions that if you make the right technical decision, you can solve the problem in a couple days, they make the wrong technical decision, you could chase a blind alley for three months, right?

有很多这样的决策，如果你做出正确的技术决策，你可以在几天内解决问题，如果做出错误的技术决策，你可能会在死胡同里浪费三个月，对吧?

And one thing I'm a bit surprised by, it turns out if you have two possible architecture decisions, it's one bit of information.

让我有点惊讶的是，如果你有两个可能的架构决策，这实际上是一个比特的信息。

Back to Top

00:23:00

It feels like if you don't know the right answer, and most of your choices slow, right?

感觉如果你不知道正确答案，大多数选择都会让你慢下来，对吧?

One bit, you know, try both of you, it feels like one bit of information that most buy you a two X speed up.

一个比特，你知道，尝试两者，感觉像一个比特的信息可以让你提速两倍。

And I think in some theoretical sense, that is true.

我认为在某种理论意义上，这是对的。

But what I see in practice, if you flip the wrong bit, you're not twice as slow, you spend like 10 times longer, chasing a blind alley, which is why I think going into this right technical adjustment, it really makes sense that I was going so much faster.

但我在实践中看到的是，如果你选错了比特，你不是慢两倍，而是可能多花十倍时间在死胡同里，这就是为什么我认为正确的技术调整能让我快这么多。

Back to Top

00:23:26

The other reason why I find, staying on top of AI really hopeful for startups is over the last two years, we have just had a ton of wonderful Genei tools or Genei building blocks, right?

我发现紧跟AI对初创企业非常有希望的另一个原因是，过去两年我们有了大量出色的Genei工具或Genei构建模块，对吧?

Partial list, but prompting, AgenteGwerpFills, Eval's, Godrails, RAD, Voice-AT, AcingProWing, loss of ETL, embedding, fine tuning, GraphDB, how to integrate a computer, use MCB Reasing Models.

部分列表包括提示、AgenteGwerpFills、Eval's、Godrails、RAD、Voice-AT、AcingProWing、ETL损失、嵌入、微调、GraphDB、如何集成计算机、使用MCB Reasing模型。

There's a long and wonderful list of building blocks that you can quickly combine to build software that no one on the planet could have built, even a year ago.

有一长串出色的构建模块，你可以快速组合它们来构建一年前地球上没人能构建的软件。

Back to Top

00:24:01

And this creates a lot of new opportunities for startups to build new things.

这为初创企业创造了许多构建新事物的机会。

So when I learned about these building blocks, this is actually a picture that I had in mind.

所以当我了解这些构建模块时，我脑海中实际上浮现出这样的画面。

If you own one building block, like you have a basic white building block, you know, you can build some cool stuff, maybe you know how to prompt, see a one building block, you know, you build some amazing stuff.

如果你有一个构建模块，比如一个基本的白色构建模块，你可以构建一些很酷的东西，也许你知道如何提示，看到一个构建模块，你可以构建一些很棒的东西。

Back to Top

00:24:19

But if you get a second building block, like you also know how to build chat box, see a white Lego break and a black Lego break, you can build something more interesting.

但如果你有第二个构建模块，比如你也知道如何构建聊天机器人，看到一个白色乐高块和一个黑色乐高块，你可以构建更有趣的东西。

If you acquire a blue building break as well, you can build something even more interesting.

如果你再获得一个蓝色乐高块，你可以构建更有趣的东西。

Get few red building breaks, maybe a little yellow one, more interesting.

再拿几个红色乐高块，也许还有一个小黄色块，更有趣。

Back to Top

00:24:39

Get more building breaks, get more building breaks, and very rapidly, the number of things you can combine them into, grows, kind of, carbonate, or, or growth exponentially.

拿更多的乐高块，拿更多的乐高块，很快，你能组合的东西的数量会以某种方式呈指数增长。

And so knowing all these wonderful building blocks, let's combine them in much richer combination.

因此，了解所有这些出色的构建模块，让我们以更丰富的方式组合它们。

One thing that Deep Learning Diode does, I actually take a lot of Deep Learning Diode short causes myself, you know, to, to this work of great part, work of, I think, like, finish all the leading AI companies in the world, and so the, and, and, and try to hand out building blocks.

Deep Learning Diode做的一件事，我自己也参加了许多Deep Learning Diode短期课程，你知道，为了完成这项重要工作，我认为，完成世界上所有领先AI公司的工作，然后尝试分发构建模块。

Back to Top

00:25:09

But when I look at the Deep Learning Diode I course catalog, this is actually what I see.

但当我查看Deep Learning Diode课程目录时，这就是我看到的。

And whenever I take these courses to learn these building blocks, I feel like I'm getting new things that can combine to form, kind of, combine to form the exponentially more software applications that were not possible just one or two years ago.

每当我参加这些课程学习这些构建模块时，我感觉我正在获得可以组合成指数级更多软件应用的新东西，这些应用在一两年前是不可能的。

So just to wrap up this my last slide, then you'll take questions if, if you'll have any.

所以总结一下我的最后一页幻灯片，然后如果有问题的话，你们可以提问。

Back to Top

00:25:30

I find that there are many things that matter for startup, not just speed, but when I look at the startups that AI fund is building, I find that the management team's ability to execute as speed is highly correlated with its odds of success.

我发现对初创企业来说重要的东西很多，不仅仅是速度，但当我观察AI基金投资的初创企业时，我发现管理团队以速度执行的能力与其成功几率高度相关。

And some things with learning to get your speed is, you know, work on concrete ideas.

学习如何提速的一些方法是，你知道，专注于具体的想法。

It's going to be good concrete ideas.

这将是好的具体想法。

Back to Top

00:25:52

I find that as a, as exactly if I'm judge on the speed and quality of my decisions, both do matter, the speed absolutely matters.

我发现，如果以决策的速度和质量来评判我，两者都很重要，速度绝对重要。

Rapid entering with AI coding assistance makes you go much faster, but that shifts the bottleneck to getting user feedback on the product decisions.

借助AI编码辅助快速进入可以让你更快，但这会将瓶颈转移到获取用户对产品决策的反馈上。

And so having a portfolio of tactics to go get rapid feedback.

因此，拥有一套快速获取反馈的策略组合。

Back to Top

00:26:10

And if you haven't learned to go to coffee shop and talk to strangers, it's not easy, but then just be respectful, right?

如果你还没有学会去咖啡店和陌生人交谈，这并不容易，但要保持尊重，对吧?

And just be respectful of people.

尊重他人。

That's actually a very valuable skill for entrepreneurs to have, I think.

我认为这对企业家来说是非常宝贵的技能。

And I think also, say let's help by the AI technology by as you speed.

我还认为，让我们通过AI技术来帮助你提速。

All right, with that, let me thank you very much.

好的，说到这里，非常感谢大家。

Back to Top

00:26:30

APPLAUSE I have a quick question.

（掌声）我有一个快速的问题。

As AI advances, do you think it's more important for humans to develop the tools or learn how to use the tools better?

随着AI的发展，你认为人类更重要的是开发工具还是学习如何更好地使用工具?

Like how can we position ourselves to remain essential in a world where, you know, intelligence is becoming democratized?

比如，在智力正在民主化的世界中，我们如何定位自己以保持重要性?

I feel like AGI has been over-hyped.

我觉得AGI被过度炒作了。

Back to Top

00:27:02

And so for a long time, there'll be a lot of things that humans can do that AI cannot.

因此在很长一段时间内，人类能做而AI不能做的事情还有很多。

And I think in the future, the people that are most powerful are the people that can make computers do exactly what you want to do.

我认为未来最强大的人是那些能让计算机完全按照你的意愿行事的人。

And so I think, staying on top of the tools, some of us will build tools sometimes, but there will be a lot of other tools that others will build that we can just use.

因此我认为，紧跟工具的发展，我们中有些人会偶尔构建工具，但还有很多其他人构建的工具我们可以直接使用。

Back to Top

00:27:22

But so people that know how to use AI to get computers to do what you want to do will be much more powerful.

但知道如何使用AI让计算机做你想做事情的人会强大得多。

Not where about people running out of things to do, but people that can use AI will be much more powerful than people that don't.

不是说人们无事可做，而是能使用AI的人会比不能使用的人强大得多。

Hey, so well, first of all, thank you so much.

嘿，首先非常感谢你。

I have a huge respect for you and I think that you are true inspiration for a lot of us.

我非常尊重你，我认为你是我们很多人的真正灵感来源。

Back to Top

00:27:42

My question is about the future of compute.

我的问题是关于计算的未来。

So as we move towards more powerful AI, where do you think that compute is setting?

随着我们迈向更强大的AI，你认为计算的发展方向是什么？

I mean, we see people saying, let's ship GPUs to space, some people talking about nuclear power data centers.

我的意思是，我们看到有人说要把GPU送到太空，还有人谈论核动力数据中心。

What do you think about it?

你怎么看？

There's some kind of debate in what I was going to say in response to the last question about kind of AGI, about maybe I answered this end a little bit of the last question.

关于AGI的最后一个问题，我本来要说的有一些争论，也许我在回答上一个问题时已经提到了一点。

Back to Top

00:28:06

So turns out there's one framework you can use for deciding what's hype and what's not hype.

事实证明，有一个框架可以用来判断什么是炒作，什么不是炒作。

I think over the last two years, there's been a handful of companies that hyped up certain things, but promotional PR fundraising influence purposes.

我认为过去两年有几家公司为了宣传、公关和筹款目的炒作了一些事情。

And because AI was so new, handful of companies got away with saying almost anything without anyone fact checking them because the technology was not understood.

因为AI太新了，少数公司几乎可以说什么都行，没有人去核实，因为技术还不被理解。

Back to Top

00:28:34

So one of my mental filters is there's certain hype narratives that make these businesses look more powerful, that's been amplified.

所以我的一种思维过滤器是，某些炒作叙事让这些企业看起来更强大，这种叙事被放大了。

And so for example, this idea that AI is so powerful, we might accidentally to human extinction.

比如，这种认为AI如此强大以至于我们可能意外导致人类灭绝的想法。

That's just ridiculous, but it is a hype narrative that makes certain businesses look more powerful and it got round up and actually helps certain businesses fundraising goals.

这很荒谬，但这是一个让某些企业看起来更强大的炒作叙事，它被传播开来，实际上帮助了某些企业的筹款目标。

Back to Top

00:28:59

AI is so powerful soon no one will even have a job anymore.

AI如此强大，很快就没有人会有工作了。

Just not true, right?

根本不是真的，对吧?

But again, that made these business look more powerful, got hyped up.

但同样，这让这些企业看起来更强大，被炒作起来。

Or we are so powerful, so when the high narrative, we're so powerful that by training a new model, we will casually wipe out thousands of startups.

或者我们如此强大，以至于在高叙事下，我们如此强大，通过训练一个新模型，我们可以轻松消灭数千家初创企业。

That's just not true.

这根本不是真的。

Back to Top

00:29:19

Yes, Jasper, Brandon Trouville, small number of companies got wiped out, but it's not that easy to casually wipe out thousands of startups.

是的，Jasper、Brandon Trouville，少数公司被消灭了，但轻松消灭数千家初创企业并不那么容易。

AI needs so much electricity, only nuclear power is good enough for that.

AI需要如此多的电力，只有核能才能满足。

That wind, solar, stuff, this is not true.

风能、太阳能这些东西，这不是真的。

So I think a lot of this GPUs in space, I don't know, is like, go for it.

所以我认为很多关于GPU在太空的说法，我不知道，就像，去做吧。

Back to Top

00:29:41

I think we have a lot of room to run still for terrestrial GPUs.

我认为我们在地面GPU上还有很大的发展空间。

Yeah, but I think some of these hype narratives have been amplified that I think are a distortion of what actually will be done.

是的，但我认为其中一些炒作叙事被放大了，我认为这是对实际会发生的事情的扭曲。

There's a lot of hype in AI and how, and nobody's really certain about how we're going to be building the future with it.

AI有很多炒作，没有人真正确定我们将如何用它来构建未来。

Back to Top

00:30:02

But what are some of the most dangerous biases or over hyped narratives that you've seen people talk about or get poisoned by that they end up running with that we should try to avoid or be more aware of and allow us to have a more realistic view as we are building this future.

但你见过人们谈论或被毒害的最危险的偏见或过度炒作的叙事有哪些，他们最终接受了这些叙事，我们应该尽量避免或更加注意，让我们在构建这个未来时有一个更现实的看法。

So I think the dangerous AI narrative has been over hyped.

所以我认为危险的AI叙事被过度炒作了。

AI is a fantastic tool, but like any other powerful tool, like electricity, lots of ways to use it for beneficial purposes, also some ways to use it in harmful ways.

AI是一个很棒的工具，但像其他强大工具一样，比如电力，有很多方式可以用于有益目的，也有一些方式可以用于有害目的。

Back to Top

00:30:36

▷ I find myself not using the term AI safety that much, not because I think we should build dangerous things, but because I think safety is not a function of technology.

我发现自己不太使用“AI安全”这个词，不是因为我认为我们应该构建危险的东西，而是因为我认为安全不是技术的功能。

▷ It's a function of how we apply it.

它是我们如何应用它的功能。

▷ So like electric motor, you know, you can't, the maker of electric motor can't guarantee that no one will ever use it from unsafe downstream tasks.

就像电动机，你知道，电动机制造商不能保证没有人会将其用于不安全的下游任务。

Back to Top

00:30:57

▷ Like using electric electric motor can be used to build a Dallas machine, electric vehicle can be used to build a smart bomb, but the electric motor manufacturer can't control how we use downstream.

比如电动机可以用来建造达拉斯机器，电动汽车可以用来建造智能炸弹，但电动机制造商无法控制我们如何在下游使用它。

▷ So safety is not a function of electric motor as a function of how you apply it.

所以安全不是电动机的功能，而是你如何应用它的功能。

▷ And I think the same thing for AI, AI is neither safe nor unsafe.

我认为AI也是如此，AI既不是安全的，也不是不安全的。

Back to Top

00:31:16

▷ It is how you apply it that makes it safer unsafe.

是你如何应用它决定了它是更安全还是更不安全。

▷ So instead of thinking about AI safety, I often think about responsible AI because it is how we use it responsibly, hopefully, or irresponsibly the determines that are not what we build with AI technology as it being harmful or beneficial.

所以与其思考AI安全，我经常思考负责任的AI，因为是我们如何负责任地使用它，或者不负责任地使用它，决定了我们用AI技术构建的东西是有害还是有益。

▷ And I feel like sometimes that the really weird corner cases they get high top of the news.

有时那些极其罕见的极端案例会登上新闻头条。

Back to Top

00:31:37

▷ I think just one or two days ago, there was a Wall Street Journal article about AI losing control of AI or something.

就在一两天前，《华尔街日报》还刊登了一篇关于AI失控的文章。

▷ And I feel like that article took a corner case experiments run in the lab and sensationalized it in a way that I think was really disproportionate relative to the lab experiment those being run.

我认为那篇文章将实验室里的极端案例实验夸大其词，与实验本身的实际情况严重不符。

▷ And unfortunately, technology is hard enough to understand that many people don't know better.

遗憾的是，技术本身已足够复杂，导致许多人难以辨别真伪。

Back to Top

00:32:02

And so these high narratives do keep on getting amplified.

因此这些夸张的说法不断被放大。

And I feel like this has been used as a weapon against open source software as well, which is really unfortunate.

更不幸的是，这还被当作攻击开源软件的武器。

Thank you for a work.

感谢你的工作。

I think your impact is remarkable.

我认为你的影响力非凡。

My question is, as aspiring founders, how should we be thinking about business in the world where anything can be disrupted in a day?

我的问题是，作为有抱负的创业者，在这个一切可能瞬间被颠覆的时代，我们该如何思考商业?

Back to Top

00:32:26

Whatever great mode product or feature you have can be replicated with vibe code in competitors in basically hours.

任何优秀的产品或功能都可能被竞争对手在几小时内通过代码复制。

It turns out when you start a business, there are a lot of things to worry about.

事实上创业时需要担忧的事情很多。

The number thing I worry about is are you building a product that users love?

但我最关心的是：你是否在打造用户热爱的产品?

It turns out that when you build a business, there are lots of things to think about.

经营企业时需要考虑诸多因素。

Back to Top

00:32:48

▷ There goes market, channel, competitors, technology, mode.

比如市场、渠道、竞争对手、技术和模式。

▷ All that is important.

这些都重要。

▷ But if I were to have a singular focus on one thing, it is, are you building a product that users really want?

但如果只聚焦一点，那就是：你是否在打造用户真正需要的产品?

▷ Until you solve that, it's very difficult to build a valuable business.

若未解决这个问题，就很难建立有价值的业务。

▷ After you solve that, the other questions do come to play.

解决之后，其他问题才会显现。

Back to Top

00:33:06

▷ Do you have a channel to get to customers?

是否有触达客户的渠道?

▷ What is pricing?

如何定价?

Long term, what is your mode?

长期模式是什么?

I find that modes tend to be over-hyped.

我发现模式常被过度炒作。

Actually, I find that more businesses tend to start up with a product and then evolve eventually into a mode.

实际上更多企业先有产品，再逐步形成模式。

But consumer products brand is somewhat more defensible.

但消费品品牌更具防御性。

And if you have a lot of momentum, it comes harder to catch you.

若势头强劲，竞争对手更难追赶。

Back to Top

00:33:28

But enterprise products, sometimes, if you have a, maybe mode is more of a consideration of the channels that are hard to get to enterprises.

而企业级产品中，模式可能更关乎难以触达企业的渠道。

So I think, sorry, when AI fund looks at businesses, we actually wind up doing a fairly complex analysis of these factors and writing to the six-page narrative memo to analyze it before we decide whether or not to go see it or not.

因此AI基金评估项目时，我们会进行复杂分析并撰写六页备忘录，再决定是否投资。

And I think all of these things are important.

这些因素都很关键。

Back to Top

bilibili 00:33:56

▷ But I feel like at this moment in time, the number of opportunities may amount to stuff that is possible that no one's built yet in the world seems much greater than the number of people with the skills to build them.

但当前未被开发的机遇远多于有能力实现的人才。

▷ So definitely at the application layer, that feels like there's a lot of white space for new things you can build that no one else seems to be working on.

在应用层显然存在大量空白领域。

▷ And I would say, you know, focus on building a product that people want that people love and then figure out the rest of it along the way.

建议先专注打造人们喜爱产品，其余问题逐步解决。

Back to Top

bilibili 00:34:25

▷ Although this is important to figure a long way.

尽管长期规划也很重要。

▷ Hi, Professor.

教授您好。

▷ Thank you for your wonderful speech.

感谢精彩演讲。

▷ I'm an underwear researcher from Stanford.

我是斯坦福大学人机交互研究员。

▷ And I think your metaphor in your speech is very interesting.

您演讲中的比喻很有趣。

▷ You said the current AI tools are like bricks and can be built upon accumulation.

您说当前AI工具像砖块可堆叠累积。

▷ However, so far it is difficult to see the accumulative functional expansion of the integration of AI tools because they open a line on the stacking of functions based on intent distribution and are accompanied by dynamic problems of tokens and time overhead.

但目前难见AI工具整合的功能累积扩展，因其基于意图分布的函数堆叠会引发代币和时间开销的动态问题。

Back to Top

---

bilibili 00:35:03

▷ So which is different from static engineering?

这与静态工程不同。

▷ So what do you think will be the perspective of a possible agent to accumulation effect in the future?

您认为未来智能体如何实现累积效应?

▷ But hey, just some quick remarks to that, right?

对此我有几点简要看法。

▷ You mentioned agent, oh, I'm token cost.

您提到智能体和代币成本。

▷ My most common advice to developers is to first approximation.

我给开发者的首要建议是：

▷ Just don't worry about how much tokens cost.

初期不必担心代币成本。

Back to Top

00:35:25

Only a small number of startups are lucky enough to have users use so much of your product that the cost of tokens becomes a problem.

极少数幸运的初创企业才会遇到用户激增导致代币成本问题。

It can become a problem.

这确实可能成为问题。

I've definitely been on a bunch of teams where the cost users like your product and we started to look at all, right?

我参与过多个团队因产品受欢迎而开始关注代币成本。

Genii bills and it was definitely climbing in a way that really became a problem.

账单金额攀升确实成了问题。

Back to Top

00:35:45

But it's actually really difficult to get to point where your token usage costs other problem.

但达到这种规模本身就很困难。

And for the teams I'm on where we were lucky enough that users made our token cost a problem, we often had entry solutions to then bend the curves and bring it back down through prompting, fine tuning, USDSPite, optimize or whatever.

遇到这种情况时，我们通常能用提示工程、微调等方法优化成本。

And then what I'm seeing is that I'm seeing a lot of agent workflows that actually integrate a lot of different steps.

我看到许多智能体工作流整合了多步骤。

Back to Top

00:36:10

So for example, if you build a customer service chatbot or often have to use prompting, maybe optimize some of the results through your DSPite, the emails, the God Rails, maybe the customer service chatbot needs ragged apart the way to get information to feedback to the user.

例如客服聊天机器人需结合提示工程、邮件处理等环节。

So I actually do see these things grow.

这些确实在发展。

But one tip for many of you as well is I will often architect my software to make switching between different building block providers relatively easy.

建议设计软件时确保能轻松切换基础模块。

Back to Top

00:36:38

So for example, have a lot of products that build on top of LOMs, but sometimes they point to specific product and ask me which OM are we using.

比如我们许多产品基于LOMs，但具体使用哪个模型常需切换。

I honestly don't know because we'll build up eVals and where this new model that's released will quickly run eVals to see if the new model is better than the O1.

我们通过快速评估新模型性能来决定是否切换。

And then you'll just switch to the new model of the new model that's better on eVals.

若评估更优就立即切换。

Back to Top

00:36:59

And so the model we use week by week, you know, sometimes our engines will change it without even bothering to tell me because eVals showed the new model works better.

有时工程团队会根据评估结果直接更换模型。

So it turns out that switching costs for foundation models is relatively low and we often architect our software.

因此基础模型的切换成本较低。

Oh, AICS, this open source thing that my friends that I worked on to make switching easier.

我们开发的AICS开源工具能简化切换流程。

Back to Top

00:37:19

Switching costs for the orchestration platforms is a little bit harder, but I find that preserving that flexibility in your choice of building blocks often less you go faster, even as you're building one more thing on top of each other.

编排平台的切换较难，但保持模块选择灵活性可加速开发。

So hope that helps.

希望这些建议有帮助。

Thank you so much.

非常感谢。

In the world of education in AI, there are two paradigms mostly.

AI教育主要有两种范式。

So one is AI can make teachers more productive or automating grading and automating homeworks.

一是AI提升教师效率，如自动批改作业。

Back to Top

00:37:43

But another score of thought is that there'll be personal tutors for every student.

二是为每个学生提供AI导师。

So every student can have one tutor that gets feedback from an AI and gets personal questions from them.

每位学生都能获得个性化辅导。

So how do you see these two paradigms converge?

您认为这两种范式如何融合?

And how would education look like in the next five years?

未来五年的教育会怎样发展?

I think everyone feels like a change is coming in at a tech.

大家都感受到技术带来的变革。

Back to Top

00:38:00

▷ Like don't think the disruption is here yet.

但颠覆尚未完全到来。

▷ I think a lot of people are experimenting at different things.

目前有许多探索性尝试。

▷ So Coursera has Coursera coach, which I should wear as really well.

比如Coursera的AI助教。

▷ Deep learning data is more focused on teaching AI.

深度学习数据侧重AI教学。

▷ Also, some built-in chatbos.

还有内置聊天机器人。

▷ A lot of teams experience of water grading.

许多团队尝试自动评分。

▷ Oh, there's an avatile of me on the Deep Learning Data website you can talk to if you want.

深度学习数据网站上有我的虚拟形象可对话。

Back to Top

00:38:19

▷ Deep learning data AI slash avatile.

网址是deeplearning.ai/avatar。

▷ And then the thing for some things like language learning with, you know, speak, do a lingo, that has become clearer some of the ways that I would transform it.

像语言学习应用已显现明确转型路径。

But the broader educational landscape, the exact ways that AI would transform it, I see a lot of experimentation.

但整体教育领域的AI转型仍在实验阶段。

I think what key relearning which I've been doing some work with is doing is very promising for K-12 education.

我认为关键重学习在K12教育中很有前景。

Back to Top

00:38:42

But I think what I'm seeing is frankly tons of experimentation.

目前看到的是大量实验。

But the final end state is still not clear.

最终形态尚不明确。

I do think education will be hyper-personalized.

教育终将高度个性化。

But that workflow is an avatile.

但实现路径可能是虚拟形象。

It's a text chatbot.

或文本聊天机器人。

What's the workflow?

具体形式呢?

I think I feel like the hype from a couple of years ago that with AI soon, and we're all so easy, that was hype.

几年前"AI将让教育变简单"的说法过于乐观。

Back to Top

00:39:02

▷ The reality is work is complex, right?

教育工作本质复杂。

▷ Teachers, students, people do really complex workflows.

师生互动流程非常复杂。

▷ And for the next decade, we'll be looking at the work that needs to be done and figuring out how to map it to agent workflows.

未来十年需将其映射到智能体工作流。

▷ And education is one of the sectors where this mapping is still underway, but it's not your mature enough to deploying where the end state is clear.

教育领域的映射仍在进行，尚未成熟。

Back to Top

00:39:28

▷ So I think I think we should all just keep working on it.

我们需要持续探索。

▷ All right, all right, thank you so much, Andrew.

好的，非常感谢吴恩达。

▷ Thank you.

谢谢。

Hey, my question is, I think AI has a lot of great potential for good, but there's also a lot of potential for bad consequences as well, such as exacerbate an economic inequality and things like that.

AI虽潜力巨大，但也可能加剧经济不平等等负面影响。

And I think a lot of our startups here, while they'll be doing a lot of great things, will also be just five or a true of their product, be contributing to some of those negative consequences.

在座初创企业的产品可能无意中助长这些负面效应。

Back to Top

---

bilibili 00:39:53

So I was curious, how do you think, you know, us as AI builders should kind of balance our product building with also the potential societal downsides of some AI products?

AI开发者应如何平衡产品开发与社会责任?

And essentially, how can we both move fast and be responsible as you mentioned in your talk?

如何实现快速且负责任的发展?

Looking at your heart and if fundamentally what you're building, if you don't think it'll make people writ large better off, don't do it, right?

扪心自问：若产品不能造福大众，就不要做。

Back to Top

---

bilibili 00:40:18

I know it sounds simple, but that's actually really hard to do in the moment.

这看似简单实则困难。

▷ But AI fun, we've killed multiple projects, not on financial grounds, but on ethical grounds, where they're multiple projects.

AI基金曾因伦理问题叫停多个经济可行的项目。

▷ We looked at the economic cases, very solid, but we said, you know what, we don't want this to exist in the world and we just killed them on that basis.

这些项目经济前景良好，但我们选择终止。

Back to Top

---

bilibili 00:40:33

▷ So I hope more people will do that.

希望更多人能这样做。

▷ And then I worry about bringing everyone with us.

我还担心如何普惠大众。

▷ One interesting thing I'm seeing is people and also some draw growth that are not engineering are much more productive if they know AI than if they don't.

懂AI的非技术人员生产力显著提升。

▷ And so for example, on my marketing team, my marketers, they know how to code.

比如我的营销团队因掌握编程技能。

▷ Frankly, they were running circles around the ones that don't.

工作效率远超不懂编程的同行。

Back to Top

00:40:56

So then everyone learned to code and then they got better.

后来全员学习编程后整体提升。

But I feel like trying to bring everyone with us to make sure everyone is empowered to build with AI.

应确保所有人都能使用AI工具。

That'll be an important part of what all of us do, I think.

这是我们共同的责任。

I'm one of your big fans and thank you for your online courses.

我是您的忠实粉丝，感谢您的在线课程。

Your courses make the deep learning like much more accessible to the world.

它们让深度学习更易被大众掌握。

Back to Top

00:41:15

And my question is also about education.

我的问题也关于教育。

As AI becomes more powerful and widespread, there's seem to be a growing gap between what can it actually do and what people perceive it.

随着AI能力增强，其实际能力与公众认知的差距日益扩大。

So what do you think about like is it important to educate the general public about deep learning stuff and not only like educate those technical people and make people understand more what really what AI really do and how it works?

是否应向大众普及深度学习知识，而不仅限于技术人员？

Back to Top

00:41:43

I think the knowledge will diffuse.

知识终将普及。

Steve and I AI, we want to empower everyone to build with AI.

我们致力于让所有人都能使用AI。

So we're working on it, many of us are working on it.

许多人正为此努力。

I'll just tell you what I think is the main thing.

关键点在于：

I think there are maybe two dangers.

存在两大风险。

One is if you don't bring people with us fast enough, I hope we'll solve that.

一是知识普及速度不足，希望我们能解决。

There's one other danger, which is, it turns out that if you look at the mobile ecosystem, mobile phones, it's actually not that interesting.

二是类似移动生态的垄断风险。

Back to Top

00:42:05

And one of the reasons is there are two gatekeepers and Jordan and iOS and then let's say let you do certain things, you're not allowed to try certain things on mobile.

就像iOS和安卓限制创新。

And I think this, you know, campus innovators, these dangers of AI have been used by certain businesses that are trying to shut down open source because there are a number of businesses that would love to be a gatekeeper to large scale foundation models.

某些企业正以AI风险为由打压开源，试图垄断基础模型。

So I think, I think, I think, I think, danger is supposed to fall as dangerous of AI in order to get regulators to pass laws like the proposal SP 1047 in California.

所以我认为，我认为，我认为，我认为，为了促使监管机构通过像加州SP 1047提案这样的法律，AI的危险性被夸大了。

Back to Top

00:42:37

Which thank goodness we shut down where the put in place really burdens some regulatory requirements that don't make anyone safer but would make it really difficult for tears to release open source and open weight software.

谢天谢地我们阻止了这项提案，因为它会实施一些真正繁重的监管要求，这些要求不会让任何人更安全，但会让开源和开放权重软件的发布变得非常困难。

So one of the dangers to inequality as well is if these regulatory, you know, awful regulatory approaches, and I've been in the room where some of these businesses set stuff to regulators that was just not true.

所以不平等的一个危险也是这些糟糕的监管方式，我曾亲眼目睹一些企业向监管机构陈述不实信息。

So I think that some of these arguments, the danger is if these regulatory proposals succeed and end up siphoning regulations, leaving us with a small number of gatekeepers where everyone needs the permission of a small number of companies to fine-tune them all from a certain way.

所以我认为这些论点中，危险在于如果这些监管提案成功并最终导致监管集中，我们将只剩下少数守门人，所有人都需要少数公司的许可才能以某种方式微调模型。

Back to Top

bilibili 00:43:17

That's always a side for innovation and prevent the diffusion of this information to let lots of startups build, whether they want responsibly, whether they free the innovate.

这总是会阻碍创新，并阻止信息的传播，使许多初创公司无法建设，无论他们是想负责任地发展还是自由创新。

So I think so long as we prevent this line of attack on open source open weight models from succeeding and would make good progress with a fitness still there.

所以我认为只要我们阻止对开源开放权重模型的这类攻击取得成功，并在保持适当性的同时取得良好进展。

Then I think eventually we get to the diffusion of knowledge and we can hopefully then bring everyone with us.

那么我认为最终我们会实现知识的传播，并有望带动所有人一起进步。

Back to Top

bilibili 00:43:42

But despite to protect open source, we've been winning, but the fight is still on and we still have to keep up that work to protect open source.

但在保护开源方面，我们一直在取得胜利，但斗争仍在继续，我们仍需继续努力保护开源。

Thank you all very much.

非常感谢大家。

It's a wonderful stream I will.

这是一次精彩的直播。

Thank you.

谢谢。

Back to Top