Latent Space: Long Live Context Engineering - with Jeff Huber of Chroma

Allessio：

各位好。欢迎来到新演播室的最新一期直播节目。我是 Allessio，Decibel 的 CTO，今天与我一同做客的是来自 SmoI AI 的 Swyx。

Swyx：

嘿，说 “欢迎” 其实有点奇怪 —— 很明显，实际上 Jeff 这几个月来一直在 Chroma 欢迎我们这些访客。不过还是要感谢邀请，能来这儿我很开心。Jeff，你是 Chroma 的创始人兼 CEO 吧？我关注 Chroma 很久了，尤其是在你们还在旧办公室的时候。你们最初是从开源向量数据库做起的，对吗？现在 Chroma 已经成了很多不同项目的首选开源向量数据库，甚至像《Voyager》论文里的项目也用了你们的技术。我其实也不清楚完整的客户名单，但如今你会如何介绍 Chroma 呢？

Jeff Huber：

通常会根据受众调整表述方式，但 Chroma 创立的核心原因其实是：

我们在应用机器学习领域深耕多年，发现搭建 Demo 很容易，但要构建一个能稳定投入生产的系统却极具挑战性。**Demo 和生产环境之间的差距，与其说是“工程问题”，不如说更像“炼金术”**——就像 xkcd（知名漫画网站）上有幅漫画很形象：一个人站在一大堆冒着热气的“垃圾”上，另一个人问“这是你的数据系统？”，他回答“你猜”；对方再问“怎么判断它好不好，又怎么改进？”，他说“**就搅一搅这堆东西，看看会不会变好**”。这种情况本质上是不合理的。我们大概在 2021 到 2022 年左右就一直在讨论这个问题。

与此同时，我们还有一个核心观点。 latent space 是一种非常重要的工具。插一句，我同意无论是“ latent space” 相关的播客内容，还是其背后的技术（注：这里 latent space 刚好是播客的名字，同时也是技术名词），都是被严重低估却至关重要的工具：它对模型的可解释性很有帮助，能让基础模型“看见”自己的数据；我们人类也能通过这个“共享空间”理解系统运作的逻辑。这就是我们创立 Chroma 的起点，也是我们一直想坚持的方向 —— 我们想帮助其他人构建基于 AI 的生产级应用，让从数据到生产的过程更像工程实践，而非炼金术。

而数据库（向量数据库）并非“支线任务”，而是“主线任务”的一部分。在这个过程中我们发现，“搜索”其实是构建 AI 应用的

关键负载之一 —— 它不是唯一的负载，但绝对是至关重要的那种。而且，在你把一件事做到世界一流水平之前，你没有资格去做更多事，这需要极度专注、甚至可以说 “偏执” 的投入。过去几年我们其实一直在做这件事。刚才的介绍可能有点冗长，简单总结一下：**如果现在问大家 “Chroma 是什么”，答案会是 “我们为 AI 应用构建检索引擎，打造面向 AI 的现代搜索基础设施”**，大概是这样的定位。

Swyx：

我想再深入问一点：在你看来，“信息检索”和 “搜索” 是一回事吗？还是说两者略有不同？我想先明确一下术语定义。

Jeff Huber：

我觉得 “面向 AI 的现代搜索基础设施” 这个说法，值得拆解来看。首先，“现代” 是相对于 “传统” 而言的，核心在于 “分布式系统” —— 过去 5 到 10 年里，分布式系统领域出现了很多核心技术组件，而这些组件显然不会出现在更早期的技术中，这是定义 “现代” 的关键。比如 **“存储与计算分离”。Chroma 用 Rust 开发，并且将对象存储作为关键的持久化层和数据层，Chroma 的分布式版本也能适配云环境** —— 这就是 “现代” 的体现。

然后是 “面向 AI” 这一点，我认为它体现在四个不同维度：**第一，用于搜索的工具和技术，与传统搜索系统不同；第二，负载类型，与传统搜索系统不同；第三，开发者群体，与传统搜索系统不同；第四，消费搜索结果的 “用户”，也与传统搜索系统不同。**

传统搜索系统中，人类是完成 “最后一公里” 的角色 —— 你会自己判断哪些搜索结果相关、打开新标签页、总结信息等等，这些都是人类在做。但现在，这个角色变成了语言模型（LLM）。人类最多只能处理十个超链接，而语言模型能处理数量级远大于此的信息 —— 这些差异都会影响系统的设计思路和目标定位。

Allessio：

2023 年的时候，向量数据库领域应该是最热门的赛道之一吧？当时很多公司都很活跃。你们是如何保持专注、聚焦核心目标，而不是一味追求融更多钱、搞大动静的？而且你们推出 Chroma Cloud 花了不少时间 —— 没有为了赶进度就推出一个可能在生产环境中出问题的版本，而是慢慢来。作为创始人，你能给大家一些建议吗？比如在 AI 领域，如何保持耐心？如何坚持自己的愿景，而不被周围的喧嚣干扰？

Jeff Huber:

初创公司的发展路径有很多种，思路也各不相同。一种思路是 “快速试错”—— 比如 lean startup 模式，找到市场信号后就顺着用户需求的 “梯度下降”走。但我对这种模式的质疑是：如果完全遵循这种方法，你最终可能会做出一个 “中学生约会 APP”—— 因为这似乎是用户需求最 “浅显” 的领域；往极端说，AI 领域的 “老虎机”大概就是这种思路的产物。

另一种思路则是:先建立一个非常明确的观点 —— 最好是反共识的观点，或者至少是像 “秘密” 一样未被广泛认可的观点 —— 然后全身心投入，专注于实现这个目标。这两种思路有本质区别，我们一直选择后者。

其实当时 Chroma 的单节点版本表现已经很好了，访问量很大，而且很明显用户想要 “托管服务”—— 我们本可以快速推出一个这样的产品抢占市场，但我们觉得不行：Chroma 想被用户记住的，是我们的开发者体验；我们的品牌，以及品牌所传递的 “匠心”，必须是清晰且出众的。而推出一个 “单节点托管服务”，根本达不到我们心中 “优秀开发者体验” 的标准。

所以我们决定：要做就做 “全栈式” 的云服务，这个过程确实极具挑战性，花了很长时间，也遇到了很多困难。但现在，Chroma Cloud

已经稳定运行了，服务着数十万开发者，他们也很喜欢 —— 虽然过程艰难，但结果是好的。

Allessio:

那在组建团队时，你们是如何传递这种愿景的？比如回到一年半以前，有人可能会在 “加入 Chroma” 和 “加入其他公司” 之间犹豫 —— 当时市场上还有 PG Vector 这类热门选择。你如何向团队清晰传递愿景，从而吸引那些真正认同愿景、有使命感的人，而不是只追求短期利益的人？在早期招聘中，你们有什么经验可以分享吗？

Jeff Huber:

康威定律有个 “上游版本”：你输出的组织架构，就是你输出的文化；而组织架构其实是公司文化的 “下游产物”。我们一直非常重视团队成员的选择 —— 办公室里这些人的能力和理念，直接决定了公司未来的成长曲线：可能回到原点，可能线性增长，也可能是指数级、爆发式增长。

所以我们决定 “慢招聘”，并且非常挑剔。未来才能证明这个决定是否正确，但根据我之前几次创业的经验，我最在意的是：我只想和

“愿意一起并肩作战” 的人共事 —— 能在 “战壕里” 一起扛，并且能独立交付符合开发者期待的 “高质量成果”。这就是我们招聘的核心标准。

Swyx：

后面我们会聊些轻松的话题，但现在还是聚焦 Chroma。我一直想先了解一些 “头条级” 数据 —— 这样能让大家快速抓住 Chroma 的核心信息。我了解到你们的月下载量有 500 万次，GitHub 星标有 2.1 万，还有其他值得一提的关键数据吗？就是那种 “销售演示” 里会重点强调的核心指标。

Jeff Huber：

GitHub 星标是 2 万左右，月下载量其实已经超过 500 万了 —— 最近数据更新，现在大概是 600 到 700 万之间。多年来，Chroma 可能不仅是（向量数据库领域）使用最广泛的项目之一，在 “生产链” 中，我们的索引使用量也很大。

Swyx:

好的，明白了。我觉得你提到的 “单节点版本”，应该就是 Chroma Cloud 和之前版本的核心区别吧？而且我听说这次聊天的时间，刚好和 Chroma Cloud 的正式发布时间差不多，对吗？那大家应该了解 Chroma Cloud 的哪些信息？你们从一开始是如何设计它的用户体验的？比如你之前提到的 “存储与计算分离”，具体是指什么？

Jeff Huber:

完全正确。Chroma 的 “开发者体验” 是出了名的 —— 我觉得我们可能是第一个做到 “零门槛上手” 的向量数据库：你只需要安装，然后就能用，早期甚至可以像 “内存数据库” 一样用。而且我记得，Chroma 可能是第一个支持 “pip 安装”的持久化向量数据库。

Swyx:

从技术上来说，SQLite 的包也能通过 pip 安装吧？

Jeff Huber:

SQLite 可没法做到 “pip 安装后直接用” —— 体验完全不一样。正是这种 “无缝上手” 的体验，让新用户能快速入门：只需要敲一个命令，就能开始用。我们做了很多工作，确保无论部署目标的架构是什么，用户都能顺利启动 —— 早期甚至有人在 PowerPC 架构或者受限环境里运行 Chroma，我们会额外花功夫确保它在各种环境下都能工作。这就是 Chroma 单节点版本的特点，核心还是围绕 “开发者体验” 。

我们想做云服务（Chroma Cloud），也是因为希望它能像单节点版本一样：5 秒内就能启动，不用分心去学习复杂的 “计算 API” —— 那种接口对用户来说太复杂了。

所以 **Chroma Cloud 的设计目标是：用户不用被迫思考 “需要多少个节点” “节点规格怎么选” “分片策略（sharding strategy）是什么” “备份策略（backup strategy）是什么” “数据共享策略（data sharing strategy）是什么” —— 这些都不用管**。如果只是 “能用”，那还不够；它必须是 “零配置、零参数”，而且无论用户的流量起伏、数据量增减，它都能 “始终快速、始终低成本、始终保持数据最新”，用户什么都不用做、不用想。这就是我们的核心设计准则。

另外，“按需计费”也很重要：**我们只按用户实际使用的“最小计算单元”收费，**不多收一分钱——不是所有向量数据库都能做到这一点，但 Chroma Cloud 确实是这样。所以这些都是我们在设计时的核心考量。

Swyx：

这么说的话，你们其实也在构建一个“计算服务平台”？

Jeff Huber：

你说对了——这种“分布式设计”的动机，和 Chroma Distributed 是一致的，而 Chroma Distributed 本身也是开源的，基于 Apache 2.0 协议。Chroma 的控制平面和数据平面也都是基于 Apache 2.0 协议开源的。

而 Chroma Cloud 就是用 Chroma Distributed 来提供服务的：用户注册账号、创建数据库、加载数据，整个过程不到 3 秒。录制这期节目时，我们还提供 5 美元的免费 credits——这笔钱其实足够加载 10 万个文档、执行 10 万次查询了，很多用户甚至可能好几年都用不到付费额度，这对我们来说也没问题。要实现这些，我们确实做了很多硬核工作。

Swyx：

我觉得每个博客都应该有 “语义索引”（semantic indexing）功能 —— 比如把个人博客的数据存在 Chroma 里，对吧？

Jeff Huber：

“信息组织”（organizing information）这个使命至今还没完全实现。

Allessio：

是啊。

你平时总发一些有点 “晦涩” 的推文，比如几个月前 —— 大概是四月份？—— 你提到了 “context engineering”。现在大家都在讨论这个概念了，你能给它一个 “权威定义” 吗？然后再说说 Chroma 在其中扮演的角色，之后我们再展开聊其他相关内容。

Jeff Huber：

我认为这一点非常重要：当一个新市场出现时，用来 “理解这个领域” 的核心概念和术语至关重要。AI 领域其实已经有很多被滥用的

概念和术语，导致很多开发者无法清晰思考：“这到底是什么？我该怎么组合这些技术？能解决什么问题？重点在哪？该把时间花在哪？”

比如 “RAG”这个术语 —— 我们从来不用，我甚至反感这个词。

Swyx:

对，我现在也尽量不用 “RAG” 了，部分也是受你的影响。

Jeff Huber:

谢谢。首先，**“RAG” 本质上是 “检索（Retrieval）” “增强（Augmented）” “生成（Generation）” 三个概念的拼凑，非常容易混淆；其次，现在 “RAG” 已经被等同于 “只用单模态稠密向量搜索（single dense vector search）”，这也很荒谬**。

而 “context engineering（上下文工程）” 这个术语 —— 显然，你在 “AI 工程”领域做了很多工作，而 “上下文工程” 在某种程度上是 “AI 工程” 的一个子集。它的核心定义是：在 LLM 的每一次生成步骤中，确定 “上下文窗口（context window）里应该包含哪些信息” 的工作。

这个工作包含两个循环：“内循环（inner loop）” 是 “即时筛选” —— 比如你明确知道这次生成需要哪些上下文；“外循环（outer loop）” 则是 “长期优化” —— 如何通过持续迭代，让上下文窗口里始终只包含 “相关信息”。

我们最近发布了一份关于 “context rot” 的报告，其中详细说明：LLM 的性能并非 “token 数量无关” —— **随着 token 数量增加，模型的注意力会分散，推理能力也会随之减弱**。而这恰恰凸显了 “上下文工程” 的必要性 —— 我对这个术语感到兴奋，也很庆幸自己在四月就预判它会成为一个重要概念，因为它不仅清晰定义了这项工作的核心，还提升了这项工作的 “地位”。

坦白说，现在大多数做得好的 AI 初创公司或 AI 原生应用，它们真正擅长的核心能力是什么？其实就是 “上下文工程”，而非其他工程领域。

Swyx:

我感觉我读过的很多资料都在聚焦 “智能体（agents）” 和 “非智能体” 的区别，似乎 “上下文工程” 对智能体更重要。你会做这种区分吗？还是说你只关注 “上下文” 本身，不刻意区分场景？

Jeff Huber:

智能体确实有一些特殊场景，比如 “智能体学习（agent learning）” —— 智能体能否从交互中学习？这和 “静态知识库” 比如文档检索的场景不太一样。但即便在文档检索这类场景中，难道不应该通过更多交互来优化效果吗？

我其实不刻意区分 “智能体” 和 “非智能体” —— 因为我至今仍不清楚 “智能体” 到底指什么。不过还是要强调，术语很重要，定义模糊的术语毫无意义。

Swyx：

确实，“智能体” 的定义太多了，得先理清才行。

Jeff Huber：

大多数定义模糊的术语，其实都是人们表达期望与担忧的 “载体”。“智能体” 肯定也是这样。

Swyx：

那我们还是尽量把 “上下文工程” 的定义说得更简洁、更精准，让它真正有实际意义，能指导人们做事。我想特别提一点：围绕 “上下文工程” 和 “上下文腐化（context rot）”，现在有很多营销宣传 —— 比如 Haystack，还有很多前沿模型发布时，都会放出 “在 100 万 token 范围内效果完美、利用率 100%” 的图表。你们对这种营销方式怎么看？

Jeff Huber：

或许我可以先说说我们是怎么开始研究 “上下文腐化” 的 —— 最初其实是为了研究 “智能体学习”：我们很好奇，如果让智能体获取 “过往成功或失败案例”，能不能提升它的性能？于是我们用几个数据集做了基准测试，结果发现了一个有趣的规律：**在多轮交互场景中，对话窗口的 token 数量会极速膨胀，而原本清晰包含在上下文里的指令，会被模型忽略、不执行** —— 这显然是个大问题。

其实圈内人早就知道这个痛点，甚至已经成了一种 “共识梗”。后来研究界对我们那份 “上下文报告” 的反应也是 “早就知道了” —— 但问题是，圈外人并不知道。如果能让开发者清楚 “现在哪些能做到、哪些做不到”，其实是件好事。

我其实也理解模型实验室（labs）的处境：模型研发竞争太激烈了，大家自然会挑选“自己表现最好的基准测试”，围绕这些测试训练模型，然后把结果放进营销材料里。**很少有人会主动站出来说“我们的模型在这方面很强，但在那方面不行”**——我能理解他们为什么不公开这些信息，但问题在于，现在有种“虚假宣传”的倾向：“我们的模型在这个任务上表现完美，所以你可以随便用上下文”——这种暗示是不对的。未来或许能实现，但现在绝对做不到。

Swyx：

我会让团队把你们报告里的“图 1”放在 YouTube 视频里——那张图显示，在曲线覆盖范围上，GPT-4 的表现最好，其次是 Claude，而 GPT-4o 的性能随上下文长度增加，衰减得快多了。

Jeff Huber：

我对此没有太多额外评论——这只是我们在特定任务中观察到的结果。至于这和人们在真实场景中的体验有多大关联，就是另一回事了。我猜开发者对 Claude 的好感度比较高，或许这两者之间有关联。

Swyx：

是啊，如果这个结果是真的，就能很好地解释为什么（有些模型）不遵循指令了。

Jeff Huber:

这至少能提供一个“基准参考”—— 如果你在开发中遇到类似问题，可以先对照这个结果排查。

Swyx：

虽然这个问题还没完全解决，但我有个猜想：“推理型模型” 的上下文利用率可能更好，因为它们能 “回头看”；而普通的 “自回归模型” 只能 “向左或向右” 生成文本，没法在初始处理后再回头找需要关联的信息。

Jeff Huber：

这有数据支持吗？我倒觉得可能相反，不过我会去查证一下。

Swyx:

要是能弄清楚这点就有意思了。

Allessio:

现在每天都有新论文。我觉得你们这份报告最棒的地方是不推销产品——你们只是客观指出这个问题存在，有点棘手。你是怎么看待“要解决的问题”和“用于揭示问题的研究”之间的关系的？比如，你们会不会先通过研究指出问题，再希望其他人一起参与解决？你们聊到的所有问题，都会纳入 Chroma 的路标吗？还是说只是“提醒大家这个问题不好解决，先规避，但别指望我们来修复”？

Jeff Huber:

之前我提到过，Chroma 的核心使命是“**让 AI 应用的构建过程更像工程，而非炼金术**”——这个使命很宽泛，但我们团队规模不大，只能聚焦少数事情。我没有那种“我们能独自彻底解决这个领域所有问题”的狂妄——这个新兴行业变化太快、规模太大，需要整个社区共同努力，需要更多人协作。

所以我们在做研究时，特意明确了“不涉及任何商业化”：我们不提出解决方案，也不暗示“必须用 Chroma 才能解决”——我们

只是把问题摆出来。可能有人会觉得“这太消极了”，但也可能是“积极的耐心”——不管怎样，要做的工作还有很多。

有意思的是，模型实验室其实并不关心这些问题，而且越来越不关心：现在模型市场的核心似乎是“消费者端”，“服务开发者”只是次要目标——既然是次要目标，他们就没动力花功夫教开发者“怎么用 AI 构建产品”。

Swyx：

“次要目标”的意思是，他们真的没动力做“帮助开发者”这种基础工作？

Jeff Huber：

对，就是没动力。而如果你是一家 SaaS 公司或消费级公司，“AI 原生产品的核心能力”是你的“护城河”，你也不会公开“怎么做”。

所以这就形成了一个天然空白：**真正有动力“为开发者指明 AI 构建方向”的人太少了。**我们觉得填补这个空白对我们来说是件好事，所以就这么做了。

Swyx：

我想反驳一下 “消费者端优先” 的说法 —— 比如 Anthropic，他们不是在做 Claude 的 “记忆功能”，还向所有人开放吗？虽然可能有点 “过度曝光”，但我觉得他们肯定很在意 “记忆利用率” —— 毕竟 “上下文利用率” 和 “上下文工程” 对消费者端也很重要，哪怕他们只关注消费者、不关心开发者。

Jeff Huber：

没错，但有两个问题：第一，现在 “记忆功能” 的实际效果到底怎么样？第二，就算效果好，他们会公开研究成果吗？绝对不会 —— 这是他们的秘密，怎么可能公开？

所以我觉得，真正 “有动力且愿意教开发者用 AI 构建实用产品” 的公司很少，而我们恰好有这个动力。

Allessio：

那你们能把这种影响力扩大到 “像 Haystack 一样”，甚至倒逼模型提供商改进吗？

Jeff Huber:

没有任何方法能“倒逼”任何人做什么。我们在做这份研究时也考虑过：或许可以把它设计成一个“正式基准测试”，让大家能轻松复现——我们已经开源了代码，所以如果是大型模型公司的人看到，他们可以拿自己未发布的模型来跑这些测试。

对我来说，**一个“能完美关注并推理 3000 个 token”的 6 万 token 上下文窗口模型，比一个“有 500 万 token 窗口但性能拉胯”的模型有价值得多**——作为开发者，前者对我的帮助远大于后者。我当然希望模型提供商会重视这个方向，围绕它训练模型、评估性能，并且把结果告诉开发者——如果能这样就太好了。

Allessio:

你觉得这个问题未来会变好吗？另外，你们怎么判断“哪些问题该自己解决，哪些问题该等模型提供商解决”？毕竟你之前说过，他们可能不会公开解决方案，只会把好功能留在内部。

Jeff Huber:

我没说 “他们不会改进”，只是说 “他们不会公开方法”。

Allessio:

但如果他们不公开，就意味着 “模型 API 无法实现这些功能，只有他们内部能用” —— 我是这个意思。

Jeff Huber:

预测 “什么会变好、什么不会” 风险很高，我觉得没必要猜。

Swyx:

希望不用猜，工程师们会解决的。

Jeff Huber:

希望全人类都能解决。

Swyx:

对我来说，“上下文工程” 领域的 “方法论发展” 也很有意思。比如 Nvidia 的 Lance Martin 写过一篇很棒的博客，总结了各种

“上下文筛选方法”；你们在纽约也举办了第一届线下 meetup，我们之后在旧金山也会办一场。我很好奇，你在这个领域看到了什么？比如谁在做有意思的工作？目前最核心的争议点是什么？

Jeff Huber:

现在还很早，很多人其实还没做什么实质性工作 —— **大部分人还是在 “把所有内容都塞进上下文窗口”，这种做法很普遍**。还有人用 “Caching” 来优化，这确实能提升速度、降低成本，但根本没解决 “上下文腐化” 的核心问题。

目前还没有太多 “最佳实践”，但我能看到一些早期模式。这个问题本质上很简单：假设有 N 个候选文本块，上下文窗口里只有 Y 个位置，你需要把 1 万个、10 万个甚至 100 万个候选块 “筛选到 20 个关键块” —— 这个 “筛选过程” 其实是很多行业的经典问题，但现在大家用什么工具解决它，还说不准。不过我观察到了几个趋势：

第一个趋势是 “多级检索（multi-stage retrieval）” —— 很多人称之为 “第一阶段检索”：用向量搜索、全文搜索、元数据过滤等多种信号，先把 1 万个候选块筛选到 300 个。就像我们之前说的，语言模型不用处理 “十个超链接”，它能 “暴力处理” 更多

内容 —— 所以现在很多人会用语言模型做 “重排序”，把 300 个块再筛选到 30 个。

这种方式的成本效益其实比很多人想的高 —— 我听说有人自己部署模型，输入 token 的成本能低到 “100 万 token 只要 1 美分”，输出 token 的成本几乎为零，因为任务很简单。

Swyx：

这些是专门的 “重排序模型” 吧？不是普通的 LLMs？

Jeff Huber：

不，就是用普通语言模型做重排序。当然也有 “专用重排序模型” —— 它们体积更小、速度更快，成本自然也更低。但我看到的趋势是：“会写提示词的应用开发者，现在开始用提示词做重排序”。

我觉得这会成为主流范式 —— 未来 “专用重排序模型” 可能会消失，就像 “专用工具” 的命运一样：只有在 “极致规模、极致成本优化” 的场景下，你才需要它。就像硬件选择：除非必须用 ASIC 或 FPGA，否则大家只会用 CPU 或 GPU。重排序也是如此 ——

如果语言模型的速度提升 100 倍、1000 倍，成本降低 100 倍、1000 倍，人们就会直接用语言模型做重排序，“暴力检索” 也会变得非常流行。

不过现在在生产环境中这么做还不现实：就算成本不高，跑 300 次并行语言模型调用，“尾延迟（tail latency）” 和 API 可用性都是大问题。但这些问题会随着时间解决 —— 这是我近几个月看到的新趋势，目前还只是 “行业前沿”，但未来肯定会成为主流。

Swyx：

是啊，我们在 “代码索引（code indexing）” 领域也看到过类似情况 —— 你刚才聊的内容其实适用于所有 “上下文场景”，但代码显然是一种特殊的上下文和语料库。我们之前有几期节目，请了相关的技术人员来聊，他们说 “不会对代码库做嵌入（embed）或索引，只会提供工具，用工具做代码搜索”。我一直在想：这会不会成为 “智能体” 的核心检索范式？比如构建智能体时，让它调用另一个 “带递归重排序和总结功能的智能体”，或者调用 “带工具的智能体”，而不是把所有功能塞进一个智能体里？你对这个有看法吗？当然，“智能体” 的定义本来就模糊，我只是随口问问。

Jeff Huber:

我们先拆解一下 “索引（indexing）” —— 它本质上是一种 “权衡（tradeoff）”：当你为数据建立索引时，是用 “写入性能” 换取 “查询性能” —— 数据摄入会变慢，但查询会快很多。当数据集变大时，这种权衡就更明显了：如果你的代码库只有 15 个文件，可能不需要索引；但如果要搜索某个项目的所有开源依赖（比如 VS Code 或 Cursor 编辑器里的 node_modules 文件夹），直接搜索会很慢 —— 这时候建立索引，用写入性能换查询性能，就是合理的选择。

所以 “索引” 的本质就是这样，没什么神秘的。**现在大家提到 “嵌入（embedding）”，首先想到的是 “相似度匹配”，但 “嵌入” 其实是个通用概念，指 “信息表示”** —— 有很多工具可以做嵌入，其中 “倒排索引” 就是个非常实用的工具，而且被严重低估了。我们在 Chroma 里其实也做了相关工作：Chroma Distributed 原生支持 Elasticsearch，你可以在 Chroma 里直接做倒排索引搜索 —— 因为我们认为这是 “混合检索” 的重要工具。

另外，针对你提到的 “代码场景”，我们还为 Chroma 加了一个功能：“索引分支” —— 你可以基于现有索引，在 100 毫秒内创建一个副本，成本只要几美分；之后你只需要把 “变更内容” 应用

到新索引上就行。这样一来，任何 “逻辑上会变化的语料库”，都能轻松处理。

Swyx：

所以核心成果是 “极快的重新索引（reindexing）”？

Jeff Huber：

没错。现在你可以为 “每个代码提交（commit）” 建一个索引，也可以搜索 “不同分支” 或 “发布标签” 的内容 —— 任何 “版本化的语料库”，现在都能轻松、低成本地搜索。这就是我们对 “倒排索引、嵌入” 的思考 —— 这个领域还在快速发展，任何声称 “已经找到答案” 的人，你都不该信。

Allessio：

你说 “代码嵌入（code embedding）被低估了”，为什么这么认为？

Jeff Huber: （24:02）

大多数人只是把 “在互联网数据上训练的通用嵌入模型” 拿来处理代码 —— 对某些场景有用，但能覆盖所有场景吗？我觉得不行。

换个角度想：这些不同的技术组件，本质上都是为了 “找信号（find signal）”。全文搜索在 “查询者熟悉数据” 时表现很好：比如我想在 Google Drive 里找 “包含所有投资人信息的表格”，直接搜股权表就行 —— 因为我知道我的表格叫这个名字，全文搜索在这种场景下完美适配，因为我是 “自己数据的专家”。

但如果是 “不熟悉数据的人” 找这个文件，他可能会搜 “包含所有投资人名单的表格” —— 这时候 “嵌入空间（embedding space）” 就能匹配到，因为它能理解语义。

所以还是那句话：**不同工具适用于不同场景，关键看 “谁在写查询” “对数据有多熟悉”** —— 组合使用这些工具，才能找到最合适的方案。

我猜现在 “代码搜索” 场景中，85%-90% 的查询用 “倒排索引” 就能满足需求 ——Google 搜索、GitHub 搜索显然都是以倒排索引为核心的。但如果再结合 “嵌入”，可能能提升 5%-15% 的效果。现在有些技术领先的团队，已经把 “嵌入” 纳入他们的 “代

码检索 / 代码搜索栈” 了。

当然，没必要为了用而用—— 如果能带来实际收益，才值得投入。对想做到行业顶尖、抢占市场、提供最好服务的公司来说，这就是用 AI 构建优秀软件的意义：80% 的效果很容易达到，但从 80% 到 100% 的每一步提升，都需要付出大量努力 —— 而每提升 1%，都是更好服务用户的资本。

Allessio：

你怎么看 “开发者体验” 和 “智能体体验” 的关系？比如，是否应该重新格式化代码，让它更容易嵌入，然后针对这种格式训练模型？你更倾向于哪种思路？

Jeff Huber：

我见过一种在某些场景下很有效的工具：不是直接对代码做嵌入，而是先用语言模型生成代码功能的自然语言描述，然后要么只嵌入这个描述，要么把描述和代码放在一起，要么分别嵌入到不同的向量索引里—— 这属于 “文本块重写” 的大范畴。

这个思路其实和 “索引” 的逻辑相关：在 “写入 / 摄入阶段”，你能提取的结构化信息越多、做的预处理越多，后续的查询任务就越容易，效果也越好。比如，你能提取的元数据越多，就越该在摄入时提取；能做的文本块重写越多，就越该在摄入时做。

另外值得一提的是：**开发者应该建立 “小型黄金数据集（small golden data sets）” —— 明确哪些查询需要返回哪些文本块，然后用它来量化评估效果。**或许你的应用根本不需要复杂方案，只用倒排索引或向量搜索就够了。但再次强调：任何声称 “知道答案” 的人，你首先该问 “能看看你的数据吗？” —— 如果他们拿不出数据，答案就很明显了。

Swyx：

我想推荐一下你在会议上的演讲 ——“如何分析数据（how to look at your data）”，“看数据” 和 “有黄金数据集” 都是非常好的实践。这些内容其实可以整理成一本小手册。

接下来我们要聊 “记忆（memory）”，但在这之前，你有没有什么 “一直想公开吐槽或强调” 的话题？趁现在可以畅所欲言。

Jeff Huber:

这可是个 “危险” 的任务。

Swyx:

我倒有个想聊的 —— 之前一直没找到机会插入对话，现在刚好快聊到了：我记得你有篇关于 “编码器 - 解码器架构” 的文章。原始 Transformer 模型是 “编码器 - 解码器架构”，后来 GPT 把它改成了 “仅解码器（decoder-only）” 架构；但现在的嵌入模型（embedding models）又都是 “仅编码器（encoder-only）” 架构。

从某种意义上说，我们相当于把 Transformer 拆成了两部分：先用 “仅编码器模型” 把所有数据编码（encode），存入 Chroma 这类向量数据库；再用语言模型做解码（decode）。我觉得这是 “机器学习系统架构” 的一个很有意思的演变 —— 跳出了 “单一模型” 的思维，转向 “模型 + 系统”。你对这个演变有什么看法？或者我刚才的描述有需要修正的地方吗？

Jeff Huber:

我觉得这里有个直觉是对的：我们现在做事情的方式还很 “粗糙”，5 到 10 年后回头看，会觉得现在像 “穴居人” 一样原始。比如，为什么我们要先编码成向量，再转回自然语言？为什么不直接把 “嵌入向量” 传给模型？本质上，我们其实是在 latent space 里来回转换，对吧？

Swyx:

对，这和你之前提到的 latent space 是一致的。

Jeff Huber:

关于 “未来检索技术”，我觉得有几件事可能会成真：第一，数据会 “一直留在 latent space”，不再转回自然语言；第二，“每生成一次就检索一次”的模式会改变 —— 这其实已经开始有变化了，很令人兴奋，但长期以来，我们一直是检索一次，然后生成一串 token，为什么不能 “按需检索（retrieve as needed）” 呢？

几周前有篇论文，讲的是使用检索工具” —— 模型在 “理解意图、处理文本” 的过程中，会主动去搜索。还有 “检索增强语言模型”，这些都是相关方向。

Swyx：

对，“检索增强语言模型” 是更早的论文了。

Jeff Huber：

是啊，有很多这类论文，但都没火起来。

这些模型大多有个问题：要么 “检索器（retriever）” 是固定的，要么 “语言模型（LM）” 是固定的，而且语料库不能动态变化——— 大多数开发者不想处理这种僵化的开发体验。

Swyx：

我觉得如果 “收益足够高”，我们肯定会用，但模型实验室不想让我们这么做 —— 它们的影响力太大了。

Jeff Huber：

而且，在这个行业里，“做这种事得不到认可” —— 没人会因为 “解决了开发者的检索体验问题” 而获得声望。但 “持续检索（continual

retrieval）“ 这个方向，未来肯定会有新进展；留在嵌入空间也会是个有趣的趋势。另外，关于 “GPU 内存分页”，其实有更高效的方式 —— 这是 5 到 10 年后的长期方向了。但我相信，未来回头看时，我们现在的做法会显得 “极其粗糙”。

Swyx：

是啊，我们现在用 “纯语言” 解决多模态问题，虽然已经是很大的成就了，但和 “原本设想的方式” 其实差得很远。

Allessio：

你之前说 “记忆（memory）是上下文工程的成果之一”，还发过一条随性的推文说 “别把 AI 记忆搞得太复杂”。你怎么理解记忆？它是不是上下文工程的另一个未被关联的价值点？

Jeff Huber

“记忆” 是个很好的术语，因为它对大众来说 “易于理解” —— 但本质上，我们还是在 “把语言模型拟人化”。我们人类知道自己如何使用记忆：有些人擅长用记忆学习任务，然后灵活适应新环境；我们也希望 “能和 AI 坐在一起，花 10 分钟或几小时教它做事，之后它就能像人一样可靠地完成” —— 这个愿景非常有吸引力，我相

信未来会实现。

但 **“记忆” 的底层是什么？其实还是上下文工程—— 核心都是如何把正确的信息放进上下文窗口。** “记忆” 是上下文工程带来的成果，或许还有其他技术也能带来 “记忆”，比如用强化学习（RL）通过数据微调模型—— 我不是说只有调整上下文才能实现记忆，但上下文工程肯定是关键工具。

Allessio:

你觉得 “基于对话提炼隐含偏好的记忆合成”，和 “基于提示词筛选相关记忆的记忆检索”，这两者有很大区别吗？

Jeff Huber:

我觉得它们的数据来源是一致的—— 告诉你如何更好检索的反馈信号，同样能告诉你该记住什么。所以这不是两个不同的问题，而是同一个问题的两面。

Swyx:

我现在更纠结的是 “记忆的结构”—— 比如现在有很多类比，像 “长

期记忆”“短期记忆”，还有人尝试用 “睡眠（sleep）” 来类比 AI 的 “记忆整理”。你觉得 AI 一定需要 “批量收集周期” 或 “垃圾回收周期” 吗？比如 “让语言模型‘休眠’时整理记忆”—— 但我们是在 “用人类的记忆模式类比 AI”，AI 的工作方式可能根本不是这样。你有没有看到过 “真正有效的 AI 记忆方案”？

Jeff Huber：

还是回到之前的话题：当我们开始为 AI 创造“新概念、新缩写”时，我会有点担心 —— 比如突然出现 “10 种 AI 记忆类型” 的信息图，你会忍不住想 “这些真的有区别吗？还是只是为了让人觉得‘厉害’？”

我们得抵制这种 “制造概念” 的冲动。不过 “压缩（compaction）” 这个概念，在数据库领域其实一直很有用 —— 比如你电脑里的数据库，大家应该都记得在 Windows 98 系统里运行 “磁盘碎片整理（defrag）” 吧？

Swyx：

我还没老到经历那个时代。

Jeff Huber:

我也没那么老，哈哈。显然，“离线处理” 是有帮助的，在 AI 记忆场景中也一样。就像我们之前聊的“索引”—— 索引的目的是“用写入性能换查询性能”，而 “压缩” 是写入阶段优化的另一个工具：你重新摄入数据，合并相似数据点、拆分冗余数据、重写描述、提取新元数据…… 再结合数据表现信号，判断该记住什么、不该记住什么。

所以，**AI 系统通过后台离线计算和推理实现持续自我优化，这个趋势是肯定的**。

Allessio:

我们之前聊到的 “休眠时计算”，其中一个方向是 “预计算答案（precomputing answers）” —— 比如根据已有数据，预测用户可能会问的问题，提前计算好答案。你觉得 Chroma 在这方面可以做些什么？

Jeff Huber:

三个月前我们发布了一份技术报告，标题是《生成式基准测试

（Generative Benchmarking）》。核心思路是："黄金数据集（golden data set）" 非常有用 —— 它包含 "查询列表" 和 "每个查询对应的正确文本块列表"。有了它，你就能量化评估："这种检索策略在这些查询上能命中 80% 的正确文本块，换个嵌入模型就能命中 90%—— 后者更好"。当然，做工程决策时还要考虑成本、速度、可靠性等因素，但至少你能 "量化衡量系统变化的效果" 了。

我们发现，很多开发者 "有数据、有文本块、有答案，但没有查询（queries）" —— 所以这份报告专门讲了 "如何教语言模型从文本块生成优质查询"，也就是 "文本块 - 查询对（chunk-query pairs）"。你可以让人类手动标注，但人类标注不一致且效率低，QA（问答）标注尤其难。所以我们通过技术报告证明了 "用语言模型生成查询" 的可行性 —— 这种生成问答对（QA pairs）的方法，对检索系统的基准测试非常有效。而且，"黄金数据集" 在很多情况下也能用来做微调—— 这个工具被严重低估了。

Swyx：

对，我完全同意。虽然 "上下文报告（context report）" 关注度很高，但对我来说，"生成式基准测试" 才是更重要的突破 —— 我之前从没见过这个概念，而且更多人能把它用到自己的场景中。相比之下，"上下文腐化" 更多是模型层面的问题，我们除了做好上下

文工程，没太多办法；但 “生成式基准测试” 是你自己就能做的事 —— 生成查询、建立数据集，然后自然就会用上那些最佳实践。

Jeff Huber:

没错，这是团队很出色的一项工作。我在 “应用机器学习开发者工具” 领域做了 10 年，**发现 “高质量小标签数据集” 的回报极高 —— 所有人都觉得需要 100 万个样本，但其实几百个高质量样本就非常有用了**。我经常跟客户说，周四晚上把团队叫到会议室，点个披萨，开个几小时的 ‘数据标注派对’ —— 这样就能搞定初始数据集了。

Swyx:

这不属于 “看数据” 的范畴，而是 “标注数据”，对吧？之前说 “看数据” 可能太笼统了。我同意你的观点 ——“读” 和 “写”（指 “看数据” 和 “标注数据”）都很重要。既然说到这，我得纠正一下之前的话：之前说 “信息检索标准” 不对，我真正想说的是 —— 你有个很酷的个人经历，就是你的 “机械腿（cyborg leg）”，如果你见到 Jeff 本人，可以问问这件事。不知道 “机械腿” 的经历，有没有什么经验能用到 Chroma 的发展上？

Jeff Huber:

太多了，数都数不清。虽然有点老套，但自我反思、坦诚面对自己确实很难。不过，把人生看作短暂的 vapor（水汽），只做自己真正热爱的事、和热爱的人共事、服务热爱的客户—— 这个原则非常有用，甚至可以印在钱上。或许有更快赚 50 亿美元的方法，但回顾我过去的经历，我发现做决策时的权衡都围绕着 “和谁共事” “服务谁” “为技术骄傲” —— 可能是年龄增长的原因吧，我现在越来越想做自己能做到的最好的工作，而且希望这份工作能被很多人看到、使用—— 毕竟做出好东西却没人用，和没做出好东西没什么区别，“影响力” 和 “质量” 同样重要。

Swyx:

这个话题如果敏感可以跳过：你的这些想法，有没有受到宗教（比如基督教）的影响？我问这个是因为，你是硅谷里公开、积极表达宗教信仰的少数人之一，我很想了解这一点 —— 你刚才的话里隐约有这种倾向，所以想深入问问，宗教如何影响你对影响力和决策的看法？

Jeff Huber:

我觉得现代社会越来越 “虚无主义” 了 —— “没什么重要的” “一切都很荒谬（absurdist）” “一切都要快” “一切都为了权力”，

Swyx:

一切都像喜剧。

Jeff Huber:

对，“一切都像喜剧” —— 所以，**能遇到对‘人类繁荣（flourishing for humanity）’有真正信念的人，非常难得；能遇到愿意为实现这个信念牺牲很多，甚至启动自己有生之年可能看不到完成的项目的人，更难得**。过去，启动需要几个世纪才能完成的项目是很常见的事，但现在不是了。

Swyx:

比如巴塞罗那的 “Sagrada Família”，大概 300 年前开始建，明年应该能完工。

Jeff Huber:

我见过它在建的样子，等完工了一定要去看看。很多这类长期项目的发起者，现在已经被遗忘了。**硅谷其实有很多“宗教”，比如“AGI”就是一种宗教：它有“邪恶问题（problem of evil）”——我们的智能不够；有“解决方案”——发展 AGI；有“基督再临（second coming）”——奇点（singularity）会到来，拯救人类；因为有了无限智能，所有问题都会解决，我们会永远幸福，甚至战胜死亡。**所以硅谷从不缺“宗教”，我觉得“宗教是守恒的”——你没法消灭它，只是“崇拜的对象”不同。但我对存在不到 5 年的宗教总是很怀疑。

Swyx：

这么说的话，其实是“幸存者偏差”——能留下来的都是经过时间考验的。不管怎样，我觉得你是公开表达信仰的代表人物之一，而且你们在做积极的事，我希望能有更多这样的人。人应该相信比自己更宏大的事物，并为自己希望后代生活的世界而努力——我可能把这句话说乱了，这其实是圣经里的话吗？

Jeff Huber：

我觉得不是圣经里的，但这句话很好。

Swyx：

同意。我觉得只为自己而活的社会是会崩塌的，这一点很对。

Allessio：

Chroma 的设计是谁负责的？你们的周边产品、办公室、官网、品牌标识都特别棒。这里面有多少是你的想法？有专门的人负责把握设计方向吗？这种 “设计感” 对 “品牌融入文化” 有多重要？

Jeff Huber：

回到 “康威定律”：你输出的组织架构，就是你输出的文化—— 作为创始人，你应该关心品牌设计，我确实很在意这方面。

这方面的想法确实大多来自我，但不能全归功于我 —— 我们有几位非常有才华的设计师，现在也在招聘设计师。虽然引用 Patrick Collison 的话有点老套，但他确实是 ‘如何做事’ 反映 ‘如何做人’ 这个理念的公开践行者。我要说明的是，这不是针对我个人的说法，而是更宽泛的格言：“**你做一件事的方式，就是你做所有事的方式**”。

确保所有体验的一致性很重要：就像你说的，来我们办公室，会觉得有设计感、有思考；上我们官网，会觉得有设计感、有思考；用我们的 API，会觉得有设计感、有思考；甚至我们的发布流程，都让人觉得有意图、有目的。

要保持这种 “一致性” 其实很难，唯一的办法就是坚持高标准。**作为领导者，我能为公司做的重要事情之一，就是做‘品味的守护者’** —— 所有对外输出的内容，我都会先审核，至少在公司目前的规模下是这样。

如果不这样做，品牌质量不会急剧下降，也不会有明显的糟糕之处，但会慢慢变得混乱：不同的人对好设计有不同理解，每个人都放大自己的风格，最后品牌会失去统一的声音——这个品牌到底代表什么？它的立场是什么？
我不是说我擅长或完美做到这一点，但我们每天都在努力。

Swyx：

你很擅长用简单的原则传递价值观和理念，这是一种很强大的能力。我关注你的工作很久了，一直很佩服这一点。

Jeff Huber：

谢谢。

Allessio：

还有什么我们没聊到的吗？你刚才说在招设计师，还有其他开放岗位想让大家知道吗？

Jeff Huber：

如果是优秀的产品设计师，而且想做开发者工具，Chroma 会是个很独特的机会；如果有人想拓展我们正在做的研究方向，这也是个好机会；我们还一直在招优秀的工程师—— 尤其是对底层分布式系统有热情，愿意解决硬核问题，让应用开发者不用再面对这些问题的人。

Swyx：

你说 “底层分布式系统”，能再具体点吗？大家总说招分布式系统工程师，但具体需要会什么？比如会 Rust、懂 Linux 内核？我们到底在聊什么技能？

Jeff Huber：

可以用几个关键词来概括：如果你非常关心 “Rust” “确定性模拟测试” “Raft 协议” “Apache Pulsar” “Apache Kafka” —— 这

些词对你来说是熟悉的信号，那你应该会喜欢我们的工作。

Swyx：

我想把招聘需求说得更清楚些，同时也是为了识别目标人群：现在行业里 “分布式系统工程师” 的供需双方经常找不到彼此，我做 “AI 工程师聚会” 的部分原因就是想解决这个问题。但 “分布式系统工程师” 到底是什么样的？技能栈是什么？该叫他们什么？他们做什么工作？有些 “分布式系统工作” 其实是 “云工程（cloud engineering）”，比如处理 AWS（亚马逊云服务）；有些是 “调试网络请求、解决一致性问题（比如数据复制）” —— 但他们可能根本不用 Apache Pulsar。

Jeff Huber：

现在可能不用，但未来可能会用。去年我发起了一个 “分布式系统读书会（systems reading group）”，还有分享会（presentations）。发起这个读书会的目的，就是为关心分布式系统的人创造一个交流场所—— 因为湾区之前没有这样的地方。现在这个读书会还在办，发展得很好。

要说明的是：我们团队里已经有不少分布式系统专家了—— 有 6、7 个，现在想扩到 10、12 个。我们的产品路线图很清晰，未来 18

个月要做什么，都很明确。“质量” 永远是 “瓶颈”，“质量” 和 “专注度” 决定了我们的速度。我最终还是要 “感谢物理限制（physical constraints）” ——

“需要更多人，因为需要更多专注度” —— 需要更多人 “深度关心自己的工作”。AI 确实是 “加速器”，这也是我们团队 “比很多竞争对手小，但能跟上进度” 的原因 —— 我们很擅长用 AI 工具。

Swyx：

你们也用 GitHub Copilot 这类工具吧?

Jeff Huber：

我们团队每个人都在用各种 AI 工具。但目前，所有 AI 代码工具在 “Rust 开发” 上都不太好用 —— 不知道为什么，很可能是因为网上高质量的 Rust 代码示例太少了。

Swyx：

本以为 “Rust 的报错信息足够详细，AI 能帮着调试”，显然不是这样。我在 Rust 上的经验很少，只给 “Rust 开源项目 Portals 的 decay 模块” 贡献过三次代码。但 Rust 肯定在崛起，除了 Rust，还有第三种 “酷语言” 吗?

Jeff Huber：

我觉得 Go 算一个，还有 Gleam。

Swyx：

对，Gleam 也不错。如果是 “Rust/Go/Gleam 开发者”，可以联系 Jeff；其他方面我们差不多聊完了。

Allessio：

感谢你的到来。

Jeff Huber：

谢谢你们的邀请，很高兴见到你们。谢谢。