Long Live Context Engineering - with Jeff Huber of Chroma

57 MIN · YOUTUBE · HTTPS://YOUTU.BE/PIBIZ_BXL_G?SI=FMCUUBBYULON2541
https://youtu.be/pIbIZ_Bxl_g?si=fmCUubByulOn2541

SUMMARY

Jeff, the founder and CEO of Chroma, discusses the evolution of their vector database and its focus on making AI application development more reliable and efficient. He emphasizes the importance of context engineering and memory in AI, advocating for a developer-centric approach that prioritizes quality and thoughtful design.

- *Chroma started from the need to bridge the gap between demo and production in AI applications, focusing on building a reliable retrieval engine.*
- *The company emphasizes modern search infrastructure for AI, utilizing distributed systems and separation of storage and compute.*
- *Jeff discusses the importance of context engineering, which involves optimizing what information is included in the context window for AI models.*
- *He highlights the significance of patience and a strong vision in startup development, advising founders to focus on quality rather than chasing trends.*
- *Chroma aims to enhance developer experience by providing a seamless onboarding process and usage-based billing in its cloud offerings.*
- *The conversation touches on the emerging field of memory in AI, suggesting that effective memory management is crucial for improving AI performance.*
- *Jeff encourages the development of high-quality, labeled datasets for better benchmarking and retrieval system performance.*
- *He reflects on the cultural aspects of building a startup, emphasizing the importance of hiring the right people and maintaining a coherent brand identity.*

[Music] Hey everyone, welcome to the Len Space podcast in the new studio. This is Allesio, partner and CTO of Decible, and I'm joined by Swixs, founder of Small AI. >> Hey, hey, hey. It's weird to say welcome because obviously actually today's guest, Jeff, has welcomed us to Chroma for many months now. Welcome. >> Thanks for having me. Good to be here. Jeff, you're founder CEO of Chroma. I've sort of observed Chroma for a long long time especially back in the the old office and uh you were you originally sort of got your start in the open source vector database right like you sort of you're the open source vector database of choice of of a lot of different projects uh particularly with even even projects like the uh Voyager paper you guys were used in that I don't even know like the the full list but um how do you introduce Chroma today?

>> It's a good question. I mean naturally you always want to kind of take your messaging and make it fit your audience. >> Yeah. >> But I think the reason that Chroma got started is because we had worked for many years in applied machine learning and we'd seen how demos demos were easy to build but building a production

reliable system was incredibly challenging and that the gap between demo and production didn't really feel like engineering. It felt a lot more like alchemy. There's some good like XK XKCD memes about this guy standing on top of a giant steaming pile of garbage and uh the other character asks this is your data system and he's like yes. He's like how do you know how do you know if it's good or how do you make it better?

Oh, you just like stir the pot and then like see if it gets any better. That just seemed intrinsically wrong. And this is back in like 2021 2022 that like we were having these conversations and so that coupled with like a thesis that like latent space was a very important tool. That is a plug. Yes, we >> agree. That is a plug. We need to >> ring the bell. >> Yeah, exactly. Ring the latent space both the podcast but also the technology was a very underrated tool and a very like important tool for interpretability. It's fundamentally how models see their own data. We as humans can kind of you know have that shared space to understand what's going on.

That's where we got started. And so I think that's also where we continue to want to go like what do we want to do? We want to help developers build production applications with AI and want to make the process of going from demo to production feel more like engineering and less like alchemy. Doing a database is like not a side quest. It is a part of the main quest. What we realized along the way was search was really a key workload to how like AI applications were going to get built. It's not the only workload, but it's definitely a really important workload and that you don't earn the right to do more things until you've done one thing at a worldclass level. that requires maniacal and you know kind of uh maniacal focus.

Um and so that's really what we've been doing for the last few years. That was a long kind of rambly introduction but like maybe to sort of land the plane you know if you ask people you know what does Chroma do today? We build a retrieval engine for AI applications. We're working on modern search infrastructure for AI. Um some version of that. >> I'll do a double click on this. Is information retrieval and search the same thing or are they slightly different in your mind? I just wanted to clarify our terminology. Yeah, I think that you know that modern search infrastructure for AI. Yeah, we can maybe unpack that for a couple seconds.

So modern is in contrast to traditional and mostly what that means is like modern distributed systems. So there's a bunch of primitives in building great distributed systems that have come on to the scene in the last 5 10 years that obviously are not in technology that is older than that by definition. Separation read and write separation of storage and compute. Chrome is written in Rust. It's fully multi-tenant. Um we have we use object storage as a key persistence tier and like data layer for chroma uh distributed in chroma cloud as well. So that's the modern piece and then the 4 AI piece actually I think is it matters in four kind of different ways like 4i means four different things like it means number one the tools and technology that you use for search are different than in classic search systems. Number two the workload is different than classic search systems.

Number three, the developer is different than classic search systems. And number four, the people who's the person who's consuming those search results is also different than in classic search systems. Think about like classic search systems like you as the human were doing the last mile of search. You know, you were doingick. Exactly. You're like, uh, like which of these are relevant? Open a new tabs, summarize, blah blah blah blah. You the human were doing that and now it's a language model. Humans can only digest 10 blue links. Language

models can digest orders of magnitude more. All of these things matter and I think influence like how a system is designed and what it's sort of like made for.

>> Back in 2023, I think the Vector DB category was kind of one of the hottest ones and you had Pine Con raise 100 million, you had all these different WVA, you had all these companies. >> Y >> how did you stay focused on like what mattered to you rather than just try to raise a lot of money, make a big splash and it took you a while to release Chroma Cloud 2, which rather than just getting something out that maybe broke once you got to production, you kind of took your time. Yeah.

>> Can you maybe give people advice on in the AI space how to be patient as a founder and how to have your own vision that you follow versus kind of like following the noise around you? >> There are different ways to build a startup and so you know different schools of thought here. So one school of thought certainly is like the find signal and kind of follow the gradient descent of what people want sort of lean startup style. My critique of that would be that if you follow that methodology, you will probably end up building a gating app for middle schoolers because that just seems to be like the lowest base take of what humans want to some degree. The slot machine would be the AI equivalent of that versus, you know, the other way to build a startup is to have a very strong view, presumably a contrarian view or at least a view that seems like a secret and then to just be maniacally focused on that thing. You know, they're different structure of like, okay, Chroma's single node is like doing really well, getting a bunch of traffic. Clearly, having a hosted service is the thing people want. Like, we could just spend uh we could very quickly get a product in the market. But we felt like, no, really what we want Chroma to be known for is our developer experience. It's like we want our brand to be we want Chroma's brand and the craft expressed in our brand to be extremely wellnown and we felt like by offering a single node product as a service like it was not going to meet our bar of like what great developer experience could and should look like.

Yeah. We made the decision of like no we're going to like build the thing that we think is right which was really challenging. Um, it took a long time and obviously I'm incredibly proud that it exists today and that it's like serving hundreds of thousands of developers and they love it but it was hard to get there. >> When you're building the team, how do you message that? If I go back maybe like a year and a half ago, you know, I could join Chroma, I could join all these different companies.

>> How do you keep the vision clear to people when on the outside you have, oh, I'll just use PG Vector or like, you know, whatever else the thing of the day is. Do you feel like that helps you bring people that are more aligned with the vision versus more of the missionary type on just joining this company before it's hot and maybe yeah any learning that you have from recruiting early on? >> The upstream version of Conway's law like you ship your or chart is you ship your culture cuz I think your org chart is downstream of your company's culture.

We've always placed an extremely high premium on that on the people that we actually have here on the team. Um, I think that the slope of our future growth is entirely dependent on the people that are here in this office. And, you know, that could mean going back to zero. That could mean, you know, linear growth, that could

mean all kinds of versions of like hyperlinear growth, exponential growth, hockey stick growth. And so, yeah, we've just really decided to hire very slowly and be really picky.

And I don't know, I mean, you know, the future will determine whether or not that was the right decision. But I think having worked on a few startups before, like that was something that I really cared about was like I just want to work with people that I love working with and like want to be shoulder-to-shoulder with in the trenches and I think can independently execute on the level of like craft and quality that like we owe developers. And so that was how we chose to do it. We'll talk about standard connation on all the other fun stuff towards the end, but we'll we'll focus on Chroma. I always want to put like some headline numbers up front. So, I'm just trying to do a better job of like giving people the brain dump on what they should know about Chroma.

>> 5 million monthly downloads is is what I have on Pi >> and 21,000 GitHub stars. Anything else people should know? Like that's like the typical sales call like headline stuff like that, you know? >> Yeah. Um yeah, 21,000 GitHub stars, 5 billion plus monthly downloads. Um, I've looked at the number recently. I think it's like over 60 or 70 million alltime downloads now. >> For many years running, Chrome has been the number one used project broadly, but also within communities like Lang Chain, Llama Index.

>> Okay, cool. Fair enough. Yeah, I think like when you say single node Chroma, like I think you're describing the core difference between like what Chroma cloud has been and I think we're releasing this in in line with like your GA and Chroma cloud. >> Uh, yes. So like what should people know about Chroma cloud and like how you've developed this experience from from the start like you you you mentioned separation of storage and compute like what does that >> 100%. Chroma is known for its developer experience. I don't know that we were the first to do this. I think we were with Chroma you just pip install chroma and then you can use it.

>> It's just like in memory >> like I think the first >> you can persist. >> It could be the first database to ever be pip installable. Um >> any SQLite wrapper is pip installable technically you know. No, SQLite was not like PIP installable even to this day. I don't think >> you probably have a deeper dive in knowledge of this. I'm just speculating myself. >> Yeah. So that that led to like a very seamless onboarding experience for new users because you could just run a command and then you could use it. We did all the work to make sure that like regardless of the deployment target or architecture that you're running it on, like it would just work. In the early days, we had people do really good stuff like run it on Arduinos and Power PC architectures and like really esoteric stuff, but like we would like go the extra mile to like make sure that it worked everywhere and just it just always worked. So that was Chroma single node. So going back to like the developer experience we wanted to have in a cloud product like we thought that in the same way that you could run pivots and like not have to think about it, you to learn a bunch of abstractions, you don't have to like spend a bunch of time learning which this really complicated API. That same story had to be true for the cloud. And so what that meant is like having a version of the product where you have to be forced to think about like how many nodes you want or how to size those nodes or how what your sharding strategy should be or your backup strategy or your data ting strategy or I could go on like that just wasn't wasn't good enough. It needed to be like zero config, zero knobs to tune.

It should just be always fast, always very cost- effective, and always fresh without you having to do or think about anything regardless of how your traffic goes up and down and how your data scale goes up and down. That was sort of the the motivating criteria. It also like usage based billing. That was really important because that just is like so fair. We only charge you for the minimal slice of compute that you use and like nothing more, which not all serless databases can claim, but it is true inside of Chroma that we like truly only charge you for the narrow slice of what you use. And so like that was the criteria that we entered kind of the design criteria process >> which is you know de facto you're also building a serverless compute platform.

>> Yeah you have to no exactly that motivated the design of chroma distributed. Chroma distributed is also a part of the same monor repo that's open source Apache 2 and then the control and data plane are both fully open source Apache 2 and then Chroma cloud uses Chroma distributed to run a service and that service you can sign up create a database and load in data in under 30 seconds and this is a time of filming people get like five bucks of free credits which is actually enough to load in like a 100,000 documents and query it 100,000 times which obviously for a lot of use cases actually might mean they use for free for years, which is fine. And to get there, we had to do kind of all the all the hard work.

>> Yeah. I think every blog should basically have semantic indexing. So like, you know, host your personal blog on Chroma, you know, like we're not >> Yeah. I mean, you know, the the mission of organizing the world's information remains unsolved. >> Yeah. Yeah. You have one of your usual cryptic tweets and you text you tweeted context engineering a couple months ago. What was it? Uh, April. I I think everybody now is talking about context engineering. Can you give the canonical definition for you and then how um Chroma plays into it and then we'll talk about all the different pieces of it. I think it's something that's incredibly important when like a new market is emerging is abstractions and the primitives that you use to reason about that thing. And AI, I think like in part of its hype, has also had a lot of primitives and abstractions that have gotten thrown around and have led to a lot of developers not actually be able to think critically about what is this thing, how do I put it together, what problems can I solve, what matters, where should I spend my time. For example, the term rag. We never use the term rag. Like I hate the term rack.

>> Yeah, I killed the rag track partially because of your influence. >> Thank you. Thank you. A, it's just retrieval first of all. like retrieval event generation are three concepts put together into one thing like that's just really confusing and of course rag got known now as is branded as like you know oh you're just using single dense vector search and that's what rag is it's also dumb I think one of the reasons I was really excited about the term I mean obviously AI engineering which you did a ton of work for like context engineering is in some ways a subset of AI engineering like what is it it's a it's a high status job context engineering is the job of figuring out what should be in the context window any given LM generation step and there's both an inner loop which is setting up the in you know what should be in the context window this time and there's the outer loop which is how do you get better over time at filling the context window with only the relevant information and we recently released a technical report about context rock which goes sort of in detail in depth about how the performance of LLM is not invariant to how many tokens you use as you use more and more tokens the model can pay attention to less and then also can reason sort of less effectively. I think this really motivates the problem. You know, context rot implies the need for context engineering. And I guess like why I'm really excited about the meme and you know, I I got maybe both lucky uh to some degree that you know, called it back

in April. This is going to be a big meme is that it elevates the job to it. It clearly describes the job and it elevates the status of the job. This is what frankly most AI startups any AI startup that you know of that you think of today that's doing very well like what are they fundamentally good at?

What is the one thing that they're good at? It is context engineering. >> Particularly, I would feel like the the the a lot of pieces I've read, a lot of it focuses on agents versus non- aent stuff. Like the context engineering is more relevant for agents. Do you make that distinction at all or you're just looking at context engineering generally? No, I mean there's interesting agent implications of like you know agent learning can you know can agents kind of learn from their interactions which maybe are less relevant and like static sort of knowledge based corpuses chat your documents obviously then again like you know I think you can make the argument that even like chat your document use cases like should get better with more interactions I don't draw a distinction between agent and non- agent I don't actually know what agent means still but again primitives abractions words they matter I don't know like what does agent I don't know.

>> Well, there's many definitions out there. I've taken a stab. >> Most terms that can mean anything are just a vehicle for people's hopes and fears. >> Yeah. >> Um I think you know agent is the same thing >> for sure. >> Well, maybe we'll try to be more concise or precise about context engineering so that it doesn't uh it actually means something and you know people can actually use it to to do stuff. One thing I definitely will call out for context engineering or context rot in general is I think that there's been a lot of marketing around needle in a haststack where every frontier model now comes out with like completely green perfect charts of like full utilization across you know 1 million tokens. I'm wondering what you guys' uh takes are all on on that kind of marketing and Yeah. Yeah.

>> So maybe to back up a little bit. The way that we came to work on this research was we were looking actually at agent learning. So we were very curious like could you give agents access to like prior successes or prior failures and if you did would that help boost agent performance? So we specifically looking at a couple different data sets uh sweep bench inclusive and you we started seeing interesting patterns where like on sort of multi-turn agent interactions where you're giving it the whole conversation window like the number of tokens explodes extremely quickly and instructions that were clearly in there like were being ignored and were not being enacted upon and we're like oh that that clearly is a problem. We've now felt the pain. sort of a meme amongst people in the know that like this was true and like I think also you know some of the research community's reaction to the context technical report is like yeah we know and you know that's fine but nobody else knew and like kind of nice if like you can actually teach builders what is possible today versus what is not possible today I don't blame the labs I mean building models is so insanely competitive everybody invariably is like picking the benchmarks that they want to do the best on they're training around those are also the ones that you know find their way into their marketing you know, most people are not motivated to come out and say, "Here are all the ways that our thing is great, and here are the ways that our thing is not great."

You know, I don't know. I can have I have some sympathy for, you know, why this was not reported on. But yeah, I mean, there was there was this bit of like this sort of implication where like, oh, look, our model is perfect on this task, needle in a haststack. Therefore, the context window you can use for whatever you want.

There was an implication there. And well, I hope that that is true someday. That is not the case today.

>> Yeah. Yeah. will uh send people at least on the YouTube video we'll put this chart which is kind of your figure one of the context rot report. It seems like Sonnet 4 is the best in terms of area under curve is how I think about it. Then Quinn wow and then GPC 41 and Gemini Flash are are uh degrade a lot quicker in terms of the context length. >> Yep. I don't have much commentary. That is what we found for this particular task. Again, how that translates people's actual experience and real world you know tasks is entirely different. I mean there is a certain amount of love that developers have for Claude and like maybe those two things are correlated. Yeah, I think it shows here if if this is this is true, that's that's a big explanation for why >> you follow my instructions, you know, like is a clear baseline uh you know thing people want.

>> I don't think it's super answered here, but I have a theory also that reasoning models are better at context utilization because they can loop back. Normal autogressive models, they just kind of go left to right, but uh reasoning models in theory, they can loop back and look for things that they needed connections for that they may not have paid attention to in the initial pass. There's a paper today that showed I think maybe the opposite. But >> really, >> I I'll send to you later.

>> Yeah, that'd be fascinating to figure out. >> There papers every day. I thought the best thing was that you did not try to sell something. You're just like, "Hey, this thing is broken. Kind of sucks." How do you think about problems that you want to solve versus research that you do to highlight some of the problems and then hoping that other people will participate? like does everything that you talk about is on the Chroma road map basically or are you just advising people hey this is bad work around it but don't ask us to fix it kind of going back what I said a moment ago like Chroma's broad mandate is to make the process of building a applications more like engineering and less like alchemy um and so you know this pretty broad tent but we're a small team and we can only focus on so many things we've chosen to focus very much on one thing for now and so I don't think that I don't have the hubris to think that we can ourselves solve this stuff conclusively for a very dynamic and large emerging industry. I think it does take a community. It does take like a rising tide of people all working together. We intentionally wanted to like make very clear that like we do not have any like commercial motivations in this research. You know, we do not posit any solutions. We don't tell people to use Chroma. It's just here's the here's the problem.

>> It's implied. Um, listen, we weren't sad that that was maybe maybe it may be a positive indication, you know, but like still there's still reasons around SP, you know, speed and cost regardless, I think. But there's just a lot of work to do. And I think that like it's interesting where like the labs don't really care and they're not motivated to care. Increasingly as the market to be to be a good LM provider, the main market seems to be consumer. You're just not that motivated to like help developers >> as a secondary concern. as a secondary concern. You're just like not that motivated really to do the leg work to like help developers learn how to build stuff.

>> And then like if you're a SAS company or you're a consumer company building with AI, you're you know AI native company like this is your like this is your secret sauce. You're not going to market how to do stuff. And so like I think there's just like a there's a natural empty space which is people that are actually have the motivations

to like help show the way for how developers can build with AI. Like they're just there's not a lot of obvious people who are like obviously investing their time and energy in that. But I think that is obviously a good thing for us to do and so that's kind of how I thought about it.

>> Just a bit of push back on the consumer thing like you say labs and you know don't you think like opening I building memory into chatbt and making available to literally everybody probably too much in your face I would argue but like they would really care to make the memory utilization good. I think context utilization context engineering is important for them too even if they're only building for consumer and don't care about developers. >> Yeah. How good is it today is obviously one important question. Um, but we'll skip that one. Like even if that's the case, are they actually going to publish those findings?

>> No. Never. >> Exactly. It's alpha, right? Why would you give away your secrets? >> Yeah. Yeah. >> And so I think there's just like very few companies that actually are like in the position where like they have the incentive and they really care about like trying to teach developers how to build useful stuff with AI. >> And so I think that we have that incentive. But do you think you could get this to grow to the point of being the next needle in a haststack and then forcing the models providers to actually be good at it?

>> There's no path to forcing anybody to do anything. And so uh we thought about that when we were kind of putting this together. We're like oh maybe we should like sort of formulate this as a formal benchmark that you can make it very easy to like we did open source all the code. So like you could you know if you're watching this and you're from a large model company you can do this. You can take your new model you haven't released yet and you can run you know these numbers on it. And you know, I would rather have a model that has a 60,000 context token context window that is able to perfectly pay attention to and perfectly reason over those 60,000 tokens than a model that's like 5 million tokens. Like just as a developer, the former is like so much more valuable to me than the latter. I certainly hope that model providers do like pick this up as a thing that they care about and that they train around and that they, you know, evaluate their progress on and they communicate to developers as well. That would be great.

>> Do you think this will get better lessons as well? How do you decide which of the because you know you're basically saying yeah >> the models will not learn this. It's going to be a a trick on top of it that you won't get access to. >> I'm not saying that. >> Well, but when you're saying that they will not publish how to do it, well, it means that the model API will not be able to do it, but they will have something chat GBT that will be able to do it.

>> I see. >> Yeah. It's very risky to bet what's going to be better lesson versus what is not. I don't think I'll hazard a guess. Hopefully not AI engineers. >> Yeah. Hopefully not all of humanity. I don't know, you know. Yeah, >> to me also an interesting discipline developing just around context engineering. Um, Lance Martin from Lang Chain did a really nice blog post with like all the different separations and then you in New York you had you hosted your your first meetup. We're going to do one here in San Francisco as well.

But I'm just kind of curious like what are you seeing in the in the fields like who's doing interesting work? What

are the top debates? That kind of stuff. >> I think this is still early. I mean a lot of people are doing nothing. A lot of people are just still yeeting everything into the context window. That is very popular. >> Yeah. >> And you know they're using context caching and that certainly helps but like their cost and speed but like isn't helping the context raw problem at all.

And so yeah I don't I don't know that there's lots of best practices in place yet. I mean I'll highlight a few. So the problem fundamentally is quite simple. It's you know you have n number of sort of candidate chunks and you have y spots available and you have to do the process to curate and call down from 10,000 or 100,000 or a million candidate chunks which 20 matter right now. >> Yeah, for this exact step >> that optimization problem is not a new problem to many applications and industries. sort of a classic um a classic problem and of course like what tools people use to solve that problem again I think it's still very early um it's hard to say but a few patterns that I've seen so one pattern is to use what a lot of people call first stage retrieval to do a big call down so that's would be using signals like vector search like full text search like metadata filtering metadata search and others to go from let's say 10,000 down to 300 like we were saying a moment ago like you don't have to give an LLM 10 blue links, you can brute force a lot more. And so using an LLM as a reranker and brute forcing from 300 down to 30, I've seen now emerge a lot. Like a lot of people are doing this and it actually is like way more cost effective than I think a lot of people realize. I've heard of people that are running models themselves that are getting like a penny per million input tokens >> and like the output token cost is basically zero because it's like a you know the simplest. These are dedicated reanker models, right? Not full LM.

>> No, these are LLMs. >> Okay. >> They're just using LM as reankers. >> Okay. >> And of course, there are also dedicated reanker models that by definition are going to be so like cheaper because they're much smaller and faster because they're much smaller. But like what I've seen emerge is like application developers who already know how to prompt are now applying that tool to reranking. And I think that like this is going to be the dominant paradigm. I actually think that like probably purposebuilt reankers will go away in the same way that like >> purpose-built they'll still exist right like if if you're if you're at extreme scale extreme cost yes you'll care to optimize that and the same way that if you're running with hardware right like you're just going to use a CPU or GPU unless you absolutely have to have an ASIC or an FPGA and I think the same thing is true about like reankers where like as LM become 100 a thousand times faster 100 a thousand times cheaper that like people are just going to use LMS for reankers and that actually like brute forcing information curation is going to become extremely extremely popular. Now today the prospect of running 300 parallel LLM calls even if it's not very expensive you know the tail latency on any one of those like 300 LM calls API availability like it's all still really bad and so like there are good reasons to not do that today in a production application but those will also go away over time. So those patterns I think I've seen emerge that are that that's a that is a new thing that I think I've only seen start to really become popular in the last few months and by popular I mean like popular in like the leading tip of the spear but I think will become a very very dominant paradigm.

>> Yeah, we've we've also covered a little bit on especially on the code indexing side of this the house. So everything we've been talking about applies to all kinds of context. I think code is obviously a special kind of context and corpus that you want to index. We've had a couple of episodes the cloud code guys and the the client guys talk about they don't embed or they don't index your codebase. They just give tools and use the use the tools to code search. And I've often thought about whether or not like this should be the primary context

retrieval paradigm where when you build an agent you effectively call out uh to another agent with all these sort of recursive reankers and summarizers or another agent with tools. Y >> um or do you sort of glom them on to a single agent? I don't know if you have an opinion obviously because agent is very illdefined but I'll just put it out there pull that apart. So you know indexing by definition is a trade-off like when you index data you're trading right time performance for query time performance. You're making it slower to ingest data but much faster to query data which obviously scales as data sets get larger. And so like if you're only gpping very small you know 15 file code bases you probably don't have to index it and that's okay. If you want to search all of the open source dependencies of that project, you all have done this before in VS Code or cursor, right? You like run a search over like the node modules folder. It takes a really long time to run that search. That's a lot of data. Like to make that indexed and sort of you got make that trade-off of right time performance or create time performance.

Like that's what that's what indexing is like. Like just like demystify it. What is this, right? Like that's what it is. You know, embeddings are known for semantic similarity today. Embeddings is just a generic concept of like information compression. There's actually like many tools you can use embeddings for. I think embeddings for code are still extremely early and underrated, but Reax is obviously an incredibly valuable tool. And you know, we've actually worked on now inside of Chroma, both single load and distributed. We support Reax search natively. So you can do Reax search inside of Chroma because we've seen that as like a very powerful tool for code search. It's great. And we build indexes to make red search go fast at large data volumes. On the coding use case that you mentioned, another use case that another feature we added to Chroma is the ability to do forking. So you can take an existing index and you can create a copy of that index in under 100 milliseconds for pennies. And in so doing, you then can just apply the diff for what files changed to the new index.

So any like corpus of data that's logically changing. >> So very fast reindexing is >> yeah basically the result. But now you can like have an index for like different each commit. So if you want to search different commits, search different branches or different release tags like any corpus that's kind of logically versioned, you now can search all those versions very easily and very cheaply and cost effectively. And so yeah, I think that you know that's kind of how I sort of think about like reax and indexing and embeddings. I mean yeah the needle continues to move here. I think that anybody who claims to have the answer, you just like shouldn't listen to them.

When you say that code embeddings are underrated, what do you think that is? >> Most people just take generic embedding models that are trained on the internet >> and they try to use them for code >> and like it works okay for some use cases, but does it work great for all use cases? I don't know. Another way to think about these different primitives and what they're useful for. Fundamentally, we're trying to find signal. Text search works really well.

Lexical search, text search works really well when the person who's writing the query knows the data. If I want to search my Google Drive, I just for the spreadsheet that has all my investors, I'm just going to type in cap table because I know there's a spreadsheet in my Google Drive called Cap Table full text search. Great. It's perfect. I'm a subject matter expert in my data. Now, if you wanted to find that file and you didn't know that I had a

spreadsheet called cap table, you're going to type in the spreadsheet that has the list of all the investors. And of course, in embedding space, in semantic space, that's going to match. And so I think again these are just like different tools and it depends on like who's writing the queries. It depends on what expertise they have in the data like what blend of those tools is going to be the right fit. My guess is that like for code today it's something like 90% of queries or 85% of queries can be satisfactory run with reax. Rejax is obviously like the dominant pattern used by Google code search, GitHub code search, but you maybe can get like 15% or 10% or 5% improvement by also using embeddings. Very sophisticated teams also use embeddings for code as a part of their code retrieval code search stack. And uh you know, you shouldn't assume they just enjoy spending money on things unnecessarily. They're getting some either eating out some benefit there. And of course, like for companies that want to be like top of their game and want to like, you know, corner their market and want to serve their users the best, this is kind of what it means to build great software with AI. 80% is quite easy, but getting from 80% to 100% is where all the work is. And like, you know, each point of improvement like is a point on the board and is a point that like I think users care about and is a point that you can use to yeah, fundamentally just like serve your users better. Do you have any thoughts on the developer experience versus agent experience? Like this is another case where well we should maybe reformat and rewrite the code in a way that it's easier to embed and then train models there. Where are you on that spectrum?

>> Yeah, I mean one tool that I've seen work well for some use cases is instead of just embedding the code, you first have an LLM generate like a natural language description of like what this code is doing. And either you embed like just the natural language description or you embed that and the code or you embed them separately and you put them into like separate you know vector search indexes. Chunk rewriting is kind of like the broad category for like what that is. Again this is like the idea here is like it's related to indexing which is as much structured information as you can put into your write or your ingestion pipeline you should. So all of the metadata you can extract do it at ingestion. all of the chunk rewriting you can do it at ingestion. If you really invest in like trying to extract as much signal and kind of pre-bake a bunch of the signals at the ingestion side, I think it makes the downstream query task like much easier. But also, you know, just cuz we're here like it's worth saying like people should be creating small golden data sets of what queries they want to work and what chunks should return and then like they can quantitatively evaluate what matters. Maybe you don't need to do a lot of fancy stuff for your application.

It's entirely possible that again just using reax or just using vector search depending on the use case that's maybe all you need. I guess again anybody who's claiming to know the answer you should the first thing you should ask is let me see your data and then if they don't have any data then you have your answer already. >> I'll uh give a plug to a talk that you gave at the conference uh how to look at your data. Yes, looking at your data is important having having golden data sets. So these are all like good practices that I feel like somebody should put into like a little pamphlet.

Call it the ten commandments of AI engineering or something. >> Okay, you might do that. Thou shall look at your data. >> We're about to move on to memory, but like I want to sort of leave space for like, you know, any other threads that you feel like you always want to get on the soap box about that. Yeah, that >> that's dangerous. That's really dangerous thing to ask. Um, >> I have one to to key off of because I think u I didn't I didn't know I didn't know where to insert this in the conversation but we were kind of skirting near it that I'm

trying to explore which is you know uh I think you had this rant about RA and G where the original transformer was sort of like an encoder decoder architecture >> then GBT turns most transformers into decoder only but then we're also encoding with all the u um embedding models as encoder only models. So in some sense we sort of decoupled the transformer into first we encode everything with the encoder only model put it into a vector database like chroma and chroma also does other stuff but you know um then then we decode with uh the LLMs and I just think it's like a very interesting meta learning about the overall architecture like it is stepping out of just the model to models and system >> and I'm curious if you have any reflections on that or if you have any modifications to what I just said.

>> I think there's some intuition there which is like the way we do things today is very crude and will feel very caveman in five or 10 years. >> You know, why aren't we just why are we going back to natural language? Why aren't we just like passing the embeddings like directly to the models who are just going to functionally like reput space, right? >> Yeah. They have a very thin embedding uh layer. Yeah. >> Yeah. So, I think like there's a few things that I think might be true about retrieval systems of the future. So, like number one, they just stay in lat space. they don't go back to natural language. Number two, instead of doing like this is actually starting out to change, which is really exciting, but like for the longest time, we've done one retrieval per generation.

>> Okay, >> you retrieve and then you stream out a number of tokens. Like why are we not >> continually retrieving? Yeah. >> As we need to, Greg, >> don't call it that. Um, but there was a paper or a paper in a in a you know, maybe like a GitHub that came out a few weeks ago. I think it was called unfortunately ragar one where they like teach uh deepcar1 you know kind of give it the tool of how to retrieve and so like kind of in its internal chain of thought and it's infant compute it's actually like searching >> there's also retrieval augmented language models I think this is an older paper >> yeah yeah there's a bunch of you know realm and retro and it's kind of a long history here um so I think that you know >> somehow not that popular I don't know why >> somehow not that popular well there a lot of those have the problem where like either the retriever or the language model has to be frozen and then like the corpus can't change which most developers don't want to like deal with the developer experience around.

>> I would say like we would do it if if the gains were that high >> or >> the labs don't want you to do it. I don't know about Yeah, >> the labs have a huge amount of influence. >> Labs have a huge amount of influence. I think it's also just like you don't get you don't get points on the board by doing that well. You just like don't no one cares. The status games don't don't reward you for solving their problem. So yeah, so broadly continual retrieval I think will be interesting to see come out of the scene. Number one. Number two, staying in embedding space will be very interesting. And then yeah, there's some interesting stuff also about kind of like GPUs and how you're kind of like paging information into memory on GPUs.

that I think can be done like much more efficiently. Um, and this is more like five or 10 years in the future we're kind of thinking about. But yeah, I think I think when we look back and think this was like like hilariously crude the way we do things today. >> Maybe maybe not. You know, we're solving IMO uh challenges with just language, you know. >> Yeah, it's great. >> I'm still working on the implications of that. Like it's it's still a huge achievement, but also very different than how I thought we would do things.

You said that memory is the benefit of context engineering. I think there's you had a rant on Twitter about stop making memory for AI so complicated. How do you think about memory and is what are like maybe the other benefits of context engineering that maybe people are not connecting together? I think memory is a good term. It is very legible to a wide population. Again, this is sort of just continuing the anthropomorphization of LLMs. You know, we ourselves understand how we are, we as humans use memory. We're very good at well some of us are very good at using memory to learn how to do tasks and then those learnings being like flexible to name environments and you know the idea of being able to like take an AI sit down next to an AI and then instruct it for 10 minutes or a few hours and kind of just like tell it what you want it to do and it does something and you say hey actually do this next time the same you would with a human at the end of that 10 minutes at the end of those few hours the AI is able to do it now and the same level of reliability that a human could do it like is incredibly attractive and exciting vision. And I think that that will happen. And I think that memory again is like the memory is the term that like everybody can understand like we all understand. Our moms all understand. And and and the benefits of memory are also very appealing and very attractive. But what is memory under the hood? It's still just context engineering, I think, which is the domain of how do you put the right information into the context window. And so yeah, I think of memory as the benefit. context engineering is the tool that gives you that benefit and there may be stuff as well. I mean maybe there's some version of memory where it's like oh you're actually like using RL to improve the model through data scene and so I'm not suggesting that like only changing context is the only tool which you know gives you great performance on tasks but I think it's a very important part. Do you see a big difference between synthesizing the memory which is like based on this conversation what is the implicit preference? Yeah, that's one side and then there's the other side which is based on this prompt what are the memories that I should put in.

>> I think they will be all fed by the same data. So the same feedback signals that tell you how to retrieve better will also tell you what to remember better. So I don't think they're actually different problems. I think they're the same problem. >> To me, the the thing I'm wrestling with a little more is just um what are the structures of memory? That makes sense. So, there's like obviously all these analogies with like long-term memory, short-term memory, let us trying to coin something around sleep. I do think that there there definitely should be some sort of batch collection cycle, maybe sort of garbage collection cycle where where it's like where the LM is sleeping. But I don't know what makes sense. like we're making all these analogies based on what we think how we think humans work, >> but maybe AI doesn't work the same way.

>> Yeah, >> I'm curious about uh anything you see that's working. >> Yeah, I always again, you know, as a through line of this conversation, I always get a little bit nervous when we start creating new concepts and new acronyms for things and then all of a sudden there's, you know, info charts that are like here are the 10 types of memory and you're like why? These are actually >> if you squint they're all the same thing like do they have do they have to be different? You know, like >> you have to blow the people's minds. No, I I don't think you do. I don't know.

You got you got to resist the slot machine. The slot and the slot machine. Um, compaction has always been a useful concept in >> even in databases >> in databases on your computer. We all remember running defrag on our Windows machines in 1998 and uh you know so yeah again >> some of us not old enough to do that. >> I I am >> not at this table. Um and uh yeah so ob obviously offline processing is helpful and I think that is also

helpful in this case and as we were talking about before like what is the goal of indexing? The goal of indexing is to like trade right time performance for query time performance. Compaction is another tool in the toolbox of like sort of right time performance. You're you're reingesting data.

>> It's not indexing but actually it is indexing. >> It's sort of reindexing. Yeah. You're taking data like oh maybe those two data points should be merged. Maybe they should be split. Maybe they should be like rewritten. Maybe there's new metadata we could extract from those. like let's look at the signal of how our applications performing. Let's try to figure out like are we remembering the right things or not? Like the idea that there's going to be like a lot of offline compute and inference under the hood that helps make AI systems continuously self-improve is a sure bet.

>> What part of the sleeptime compute thing that we talked about was precomputing answers. So based on the data that you have, what are likely questions >> that the person is going to ask and then can you premputee those things? Mhm. >> How do you think about that in terms of Chroma? >> We released a technical report maybe 3 months ago. The title is generative benchmarking. And the idea there is like well having a golden data set is really powerful. Having a what a golden data set is is you have a list of queries and you have a list of chunks that those queries should result in. And now you can say okay this retrieval strategy gives me for these queries gives me 80% of those chunks. Whereas if I change the embedding model now I get 90% of those chunks. that is better and then you also need to consider cost and speed and API reliability and other factors obviously when making good engineering decisions but like you can measure now like changes to your system and so what we noticed was like developers had the data they had the chunks they had the answers but they didn't have the queries we did a whole technical report around how do you teach an LLM to write good queries from chunks because you you want like chunk query pairs and so if you have the chunks you need the queries Okay, we can have a human do some manual annotation obviously, but humans are inconsistent and lazy and you know QA is hard and so can we teach an LLM how to do that and uh so we sort of did a whole techical report and proved a strategy for doing that well. So I think generating QA pairs is really important for benchmarking a retrieval system golden data set frankly is also the same data set that you would use to fine-tune in many cases and so yeah there's definitely something like very underrated there. Yeah, I'll throw a plus one on that. I think as much attention as the context rock paper is getting, I feel like generative benchmarking was a bigger aha moment for me just because I I actually never came across concept before. And I think like actually more people will apply it to their own personal situations. Whereas context ro is just generally like yeah don't trust the models that much but there's not much you can do about it except do better context >> engineering. Yeah. Yes. Yes. uh whereas generate benchmarking you're like yeah generate your your your evals and and you know part of that is going you're going to need uh the data sets and it'll sort of fall you into the place of all the bit best practices that everyone advocates for. Um so yeah it's a very nice piece of work.

>> I think having worked in applied machine learning developer tools now for 10 years like the returns to a very high quality small label data set are so high. >> Everybody thinks you have to have like a million examples or whatever. No, actually just like a couple hundred even like high quality examples is extremely beneficial. Yeah. And customers all the time I say, "Hey, what you should do is say to your team Thursday night. We're all going to be in the conference room.

We're ordering pizza and we're just going to have a data labeling party for a few hours and that's that's all it takes to bootstrap this. >> Google does this. Open does this. Anthropic does this. You're not above doing this." Great, you know? Right. >> Yeah. Exactly. >> Yeah. >> Yeah. Look at your data. It's again it's what matters. label. Maybe should classify that as label your data, not look at cuz look at seems a bit too >> I agree with that. Yeah, there's some more >> view only, >> right? I agree with that. Yeah. Yeah.

Read and write. >> Read and write. While you mentioned it, I should correct myself. It wasn't standard cognition. It was standard cyborg. >> My favorite fact about you is you're also a cyborg with your with your leg that people if you see Jeff in person, you should ask him about it. Or maybe not. Maybe don't. I don't know. >> I don't care. >> Don't care. Uh standard cyborg, Mighty Hive, and know it. What were those lessons there that you're applying to Chroma?

>> Yeah, more more than I can count. Um, I mean, it's a bit of a cliche. Um, and it's very hard to be self-reflective and honest with yourself about a lot of this stuff, but I think viewing your life as being very short and kind of a, you know, a vapor in the wind and therefore like only doing the work that you absolutely love doing and only doing work that you doing that work with people that you love spending time with and serving customers that you love serving is a very useful like north star. Um, and you know, it may not be the north star to like print a ton of money in some sense. There may be faster ways to scam people into making $5 million or whatever. Um, so, but if I reflect on, and I'm happy to go more detail obviously, but if I reflect on like my prior experiences, like I was always making trade-offs. I was making trade-offs with like the people that I was working with or I was making trade-offs with the customer that I was serving. I was making trade-offs with like the technology and like how proud I was of it. And maybe it's sort of like an age thing, I don't know. But like, you know, the older that I get, I just more and more want to do the best work that I can. And I want that work to not just be great work, but I also want to be seen by the most number of people because ultimately that is what impact looks like. You know, impact is not inventing something great and then nobody using it. Like impact is inventing something great, as many people using as possible.

>> Is any of that uh you know, and we can skip this question if it's sensitive, but like is any of that guided by religion, by Christianity? Uh I and I only asked this because I think you're one of a growing number of openly outwardly positively religious people in the valley and I think that it's kind of what I want to explore. You know I'm not I'm not like that religious myself but I just kind of like how does that inform how you view your impact your you know your choices that there was a little bit of of that in what you just said but I wanted to sort of tease that out more. I think increasingly modern society is nihilist. Nothing matters. It's uh absurdist, right?

Everything is a farce. Everything is power. >> Everything's a comedy. >> Everything's a comedy meme. Yeah. >> Yeah. Exactly. And so like it's very rare and I'm not saying that I always am the living exemplar of this, but like it's very rare to meet people that have genuine conviction about what flourishing for humanity looks like. And it's very rare to meet people that are like actually willing to sacrifice a lot to like make that happen and to start things that like they may not actually see complete in their lifetimes. Like it used to be common place that people would start projects that would take centuries to complete. Yeah.

>> And you know that's like less and less the case. >> The comes to mind is the Saga Familia in Barcelona which uh I think was started like 300 years ago and it's completing next year. >> Yeah. >> I've seen it in construction, but I can't wait to see it completed as well. >> Yeah. I'm sure the places are booked out already. >> Yeah. >> Yeah. >> And so, you know, it's it's common. There actually, you know, a lot of like religions in Silicon Valley. I think AGI is also a religion. It has a problem of evil. We don't have enough intelligence.

It has a solution, a deosex machina. It has the second coming of Christ that AGI the singularity is going to come. It's going to save humanity because we will now have infinite and free intelligence. Therefore, all of our problems will be solved and uh you know we will live in sort of like the palm of grace for all eternity. It's going to solve death, right? And so like I think that like religion still exists in Silicon Valley.

I think that it's like you know there's a conservation of religion. You kind of can't get rid of it. Yeah. >> Um but >> the god gene >> Yeah. Yeah. I mean, you know, people have different terms for this, but uh like I think that I'm always skeptical of religions that haven't been around for more than 5 years. Put it that way. >> Yeah. This survivorship bias. Anyway, I I I do think like you are one of the more prominent ones that I that I know of and uh I think you you guys are a force for good and uh I like to encourage more of that. I don't know that you know people should believe in something bigger than themselves and build for uh plant trees under which they will not sit. Um it's Am I mangling the quote? Is that is that actually a biblical quote?

>> I don't think it's biblical quote, but I like that quote. That's a good one. So yeah, plus one. >> Like I think society is really collapsed when like you just live for yourself. That that that really is true. >> Agreed. >> Who does your design? Because uh all of your swag is great, your office looks great, the website looks great, the docs look great. How much of that is your input? How much of that do you have somebody who just gets it? And how important is that to like making the brand >> part of the culture?

>> I think all value, you know, again going back to the Conway's law thing, like you ship your org chart, you ship what you care about as a founder in some sense and like I do care deeply about this aspect of what we do. And so I think it does it does it does come from me in some sense. Um I can't take at all credit for everything we've done. We've had the opportunity to work with some like really talented designers and we're hiring as well for that. So if people, you know, are listening to this and want to apply, please do. I think I mean it's cliche to uh decri um Patrick Collison quotes, but he does seem to be one of the like most sort of public embodiers of this idea that like how you do I'm not sure if this is a direct quote from him to be clear. This is more of just a broad aism, but like how you do one thing is how you do everything. and just ensuring that there's a consistent experience of what we're doing where like you said if you come to our office like it feels intentional and thoughtful if you go to our website it feels intentional and thoughtful use our API it feels intentional and thoughtful if you go through interview process it feels intentional and purposeful I think that's so easy to lose it's just so easy to like lose that and in some ways the only way that you keep that is by insisting on that that standard remain.

And I think that that is like one of the main things that like I can do really for the company like as a leader. It's sort of cringe to say, but like you do kind of have to be like the curator of taste. It's not that I have to stamp

everything that goes out the door before it does, but at a minimum companies, you know, maybe it's not even like downhill in quality. It's not sort of legible that any one thing is bad or worse, but it's like more like people just have their own expressions of what good looks like and like you know they turn that up to 11 and then like the brand becomes incoherent. Like what does this thing mean and like what do they stand for?

Again, this not longer a single voice. Yeah. I don't again I'm not claiming that I'm good at perfect at this or good at this. We certainly we we we wake up we wake up every day and we try. you have a lot of uh it's very powerful that the the skill you have to convey like straightforward principles and values and thoughtfulness I think in in everything that you do like yeah I you know you know you know I've I've been impressed with your work for a while >> thank you >> anything we're missing you're hiring designers any other roles that you have open that you want people to to apply for >> if you're a great product designer um that wants to work on developer tools I think we have one of the most kind of unique opportunities at Chroma If you are interested in extending the kind of research that we do, that's also an interesting opportunity. We're always also hiring like very talented engineers that want to work with other people that are very passionate about kind of low-level distributed systems and in some ways solving all the hard problems so that application developers don't have to.

>> When you say that, can you double click on low-level distributed systems? People always say this and then like okay, Rust like like you know like Linux kernel, what are we talking here? Yeah, I mean like that maybe like you know a useful encapsulation of this is like if you care deeply about things like Rust or deterministic simulation testing or >> raft paxos >> TA plus consensus >> TA plus really >> um >> wow >> you know if you just keep I'm saying these these are like proxies for you would like you would like the work that we do here. I just really want to tease out the hiring message, but also I um part of my goal is also to try to identify who are what what is the type of engineer that people that startups are really trying to hire and they cannot get >> because the better we can identify this thing I can you know maybe create like some kind of branding around it create an event and like get these like there's a there's a supply side and a demand side and they can't find each other and that's why I put AI engineer together was that that was that was part of it but then this distributed systems person which like I have heard heard from you and like a hundred other startups.

>> What is the skill set? What are they called? What do they do? And part of that is like part of that is cloud engineering because a lot of times you're just dealing with AWS. >> Sure. >> A lot of that a lot of times you're just dealing with I don't know debugging network calls and and uh consistency things if you're doing replication or whatever. >> Um where do they go? What do they do? Yeah. Yeah. Like but they don't use TA plus at work, you know.

>> Probably not. Yeah. I mean last year I started like the SF systems group. >> Yes. the reading group. >> Um, yeah, there's like presentations and the point of that was like let's bring let's create a meeting place >> for people that like care about this topic because like there wasn't really a place in the Bay Area for people to do that. >> Um, so that continues to go now and continues to run which is great. I mean to be clear we have a lot of people in the team who are extremely good at this and so like it's not that we have zero it's that we have six or seven um 120 but yeah we yeah it's not that we want more but we are in some ways like I feel like our product road map is very obvious and we know exactly what we need to build for the next even like 18 months but quality is always

a limiting function quality and focus are always limiting functions and like well yes I will always make my land acknowledgement to mythical mammoth month eventually like >> it's good more people >> you do you kind of do need more people because you need more focus like you need more people to care deeply about the work that they do I think AI is certainly an accelerant as helpful the reason that our team is still very small today relative to many of our competitors is because like I think we've really embraced like those tools >> your cursor shop cloud code windsurf >> people use whatever they want >> yeah so I think all of those tools get some usage internally so far uh we've still not found that really any AI coding tools are particularly good at rust though. Um I think not sure why that is other than the obvious there's just like not that many examples of great Russ on the internet >> and so um you know >> yeah you you would think that you know Russ errors would be help you debug it itself >> right >> you would think >> apparently not okay >> I have zero experience in that in that front >> uh I have I've contributed three things to the rest SDK of temporal and that was my total experience with Rust but I I think it's definitely on the rise. It's It's Zigg, it's Rust, >> and I I don't know if there's a third.

Cool languages. >> I think ghost accounts. >> Golang. Yeah, >> ghost accounts. >> If you're in in those in that in that bucket, uh, reach out to Jeff. But, uh, otherwise, I think we're good. >> Thanks for coming on. >> Thanks for having me, guys. Good to see you. >> Thank you. [Music]