EPISODE 942

[INTRODUCTION]

[0:00:00.3] JM: Machine learning is widely understood by the software community, but it is still hard to build a company around machine learning, because there is not easy access to large, unique data sets.

Scale is a platform for training and validating data that is used for machine learning. Most machine learning models are built with supervised learning. Labelled examples are analyzed to understand the mathematical correlations between those labels. The more labelled training examples there are, the more accurate the correlations will be.

Today we have high quality frameworks for writing the models. We have cheap cloud computing for training and deploying the models. The biggest factor that is preventing a wide variety of potential machine learning applications from existing is lack of access to large labeled data sets. Scale gives developers an API for labeling images and sound and natural language and video. Scale uses a platform of scalers, who are labeling the data. These are people that are paid through scale and they will do these tasks at an API request.

Scale is used by self-driving car companies to label data from the self-driving car cameras. It's used by Airbnb, it's used by OpenAI, it’s used by retailers and robotics companies. The scale product is used broadly and at high volume. Scale was started only three years ago and has raised a 100 million dollars at a valuation above 1 billion dollars, making it one of the fastest-growing startup software companies in history.

Alexandr Wang joins the show to discuss how Scale works, the future of machine learning and the future of work. He also describes the complexities of building Scale and how he manages his own psychological state.

[SPONSOR MESSAGE]

© 2019 Software Engineering Daily

[0:02:06.7] JM: Today's show is sponsored by Datadog, a modern full-stack monitoring platform for cloud infrastructure, applications, logs and metrics all in one place. Use Datadog's rich, customizable dashboards to monitor, correlate, visualize and alert on data from disparate devices and cloud back-ends, to have full visibility into performance.

Datadog breaks down the silos within an organization's teams and removes blind spots that could cause potential downtime. With more than 250 integrations, Datadog makes it easy to collaborate together and monitor every layer of their stack within a single platform.

Try monitoring with Datadog today using a free 14-day trial at softwareengineeringdaily.com/datadog and they'll send you a free t-shirt. That's softwareengineeringdaily.com/datadog. You sign up and you get a free t-shirt. Check it out at softwareengineeringdaily.com/datadog.

[INTERVIEW]

[0:03:20.4] JM: Alexandr Wang, welcome to Software Engineering Daily

[0:03:22.4] AW: Thanks for having me.

[0:03:24.1] JM: The core of what you do is data labeling. Data labeling is important for machine learning, because in order to train machine learning models, we need labeled data sets. Explain what happens when a piece of data is sent to the scale API for labeling.

[0:03:43.4] AW: Good question. We receive this data and then first, we run it through a bunch of machine learning algorithms on our own. We go through and pre-process the data, maybe there is some issue in the data that we have to notify the customer about. Most of the time, we pre-process it and we try to do as much with the labeling as possible automatically while managing bias, then it gets sent through to our labeler platform for quality verifications and fixes of any errors that might have been made on the data already.

Then after that process, we send it back to the customer via webhook, or some other API integration and then they take that data, they add it to their data lake or whatnot and then eventually, they retrain their model on it.

[0:04:25.6] JM: Explain how you make sure that the correct labels are being applied.

[0:04:30.2] AW: Yes. This is if for anyone who's worked on data labeling, in a nutshell, this is maybe the hardest problem is managing quality of these labeling procedures. The way that we approach it is twofold. I think we view this as both a product, an engineering problem of how do you ensure quality, as well as an operations problem, of how do you build scalable processes that actually work, right?

From a product and engineering standpoint, we build a lot of systems internally at scale that do a variety of things that help us measure the quality of data, as well as measure the quality of particular kinds of errors. A lot of times what happens in these labeling processes is there's certain kinds of errors that the labelers might be making that they're – they might not know that those are actually errors for example, right? A lot of it is helping build automated systems that will notice when they're making those mistakes, retrain, or give them content or training courses that tell them, “Oh, you're making this error automatically.” Then and then help them get better and better and better over time.

Constant monitoring, quality monitoring automatically and training and retraining is a really big part of it. Then we make sure that all of our quality settings at the end, we have a large amount of QA on top of all this data that samples some small portion of the data to ensure that it's very high-quality. We ensure that all that QA is trained to the highest standards and is looking for absolutely any error that might be in the data.

[0:06:03.1] JM: We talked about this a little bit in the first episode that we did, but I want to recover it because I think it's a problem that has a lot of engineering depth to it. The classic thing that people always talk about with Mechanical Turk is at least I hear this, you need to send a training example to three different Turks and you take the two-out-of-three answer. Because you assume that there's going to be somebody who is going to be just answering whatever, like labeling everything a cat, even though obviously not most pictures are cats. Do you have to have that redundancy, where you have three people that you send each piece of data to?

[0:06:49.0] AW: This is a really good question. That's been the, I guess the standard method for a very long time. I actually personally think that that method only takes you so far. Doing this, we call it consensus, but there's this duplication of labeling and then taking the majority vote. That only works in a small number of kinds of labeling tasks actually. It won't work for – in particular, as machine learning gets more and more advanced, we're doing much more advanced things with machine learning, like object recognition, or object segmentation, or voice transcription, etc. All of these tasks are not well suited to that approach, because if you have three people transcribe an audio snippet, it's very unlikely that two out of three of them are going to be exactly the same. There's always going to be small nuances in between them.

A, I think that approach is limited. Then B, the question is do you have multiple people look at each piece of data to ensure it's really correct? I think we definitely do. We definitely have multiple people look over every single piece of data to basically ensure maximum correctness. What we do is actually as a system is since we are constantly tracking which kinds of errors various people in the platform are making, we try to have multiple people look at each individual piece of data who have very uncorrelated error, so very different errors that they make, so you have all of your bases covered if it goes through multiple of these people in series. That's actually a really core. That's been really huge for us is ensuring that each piece of data is very tightly quality controlled.

[0:08:22.8] JM: Just to understand a little bit better why the consensus process doesn't even work for more complex machine learning tasks, is it because there's just increasing subjectivity as the tasks get more complex? Instead of just labeling if a thing as a cat or a dog or transcribing a piece of audio, you're actually doing things like describing a scene. If you're describing a scene, you're likely to get three divergent answers, all of which may be correct.

[0:08:54.2] AW: Yeah. There's two things, I think one of them is definitely subjectivity. That's definitely one direction that causes there to be divergence. For example, we did this recent project with OpenAI that they just published all the results about. There, we were helping to fine-tune their language model, GPT2, which would produce a lot of different kinds of text and to rate it on different subjective axes.

In those cases, again because of the subjectivity like you're saying, the best two-out-of-three approach wouldn't have worked very well, because you actually – it was the divergence of the responses meant that you had to quality control some other way. That's definitely only. Then the other thing is that there's actually just a lot of kinds of tasks where even if two people have the same intent in what exactly they're doing, they won't respond with the exact same thing and it's really hard to know how to combine different responses, right?

For example in the case of transcribing audio is very simple, right? Let's say you hear somebody say like, "Because the dog was green, we painted it purple," or something like that. Even the punctuation of is somebody going to add a comma after the first clause is relevant. Then in particular in audio, you'll always have particular words that are hard to make out. I'm sure as a podcaster, you know this pretty well, but you'll always have words that are really hard to make out. Then different people might respond with different things. As the number of choices increases and the number of maybe not subjective but more freeform, the responses, the less that approach works.

[0:10:36.3] JM: Right. I've seen Kubernetes be transcribed into some different things in my podcast transcriptions. I don't blame them. I mean, in this process of the data getting labeled, you actually have the opportunity to train your own machine learning models to do the labeling as accurately, potentially as these scalers are labeling the data. What are the tasks where just by virtue of the amount of data going through your system you will be able to eventually, potentially factor out the scaler and just defer completely to your own internal models for doing data labeling?

[0:11:24.7] AW: Yeah. It's an interesting question. One where I actually – my lean on this subject is more conservative, in the sense that I think it will always be very important to have a human in the loop for – philosophically for machine learning and also more practically, to manage quality of a lot of these models, right?

It's like, invariably, models will make mistakes in somewhat weird ways that are – might seem strange to humans and will impact – part of these models are on the wild somewhere. Say they're on a self-driving car, or they're operating some fraud process, or some loan process, etc. All of these areas where people are deploying machine learning models, you actually really care

about the models consistently getting better and behaving in ways that are predictable to humans, etc. Because of that, I think it's really dangerous to ever let these models go wild in some sense.

You always want there to be humans involved in the process, so that they can constantly keep the models in check and keep them within some operating zone that is reasonable, is well-managed, etc. Again, it's this old-school approach. Because a lot of these machine learning models today, they primarily learn from the humans that help label the data to train them, it's really important that that process stays true to what humans actually expect.

[0:13:01.0] JM: You have 16 APIs at this point, at least judging by your homepage, you probably have some more that you're testing. Your homepage is starting to look something like AWS. Just a few examples, you have video annotation, polygon labeling, audio and speech sentiment analysis. How do you maintain high-quality across all these APIs when there's so much surface area?

[0:13:30.4] AW: Yeah. It's actually a really good question and it's one that especially as we build our products and as we build all of our systems, we think a lot about. If you look at the cloud companies like AWS, GCP, etc., I mean, different developers have different beliefs, but they've generally done a pretty good job at maintaining quality and reliability of the services on these platforms, with some exceptions that I think we the developers like I joke about a lot. As in general, I think they've done a very good job.

If you look at organizationally how that made that happen, AWS, one thing they really focus on was making it so that there were a lot of very independent teams that could work as startups in some sense over their domain of products. There were these platform and infrastructure teams that would build tools, that would help all these teams move faster, or build infrastructure that would help all these teams move faster.

Organizationally, that's the approach that we need to take. I think it's the right one when you're – when your business looks like ours, where you'll need to be able to provide developers and machine learning engineers and data scientists, etc., with a wide variety of different options.

Because in reality, they have all sorts of different kinds of data, they have all sorts of different kinds of machine learning they want to do on top of this data.

It's really important that customers have access to all of these products. Then us knowing that, we view it as okay, how do we setup our engineering and product and design organization to actually scale the number of products effectively? We do take this – we have lots of small teams that are dedicated to each product and have full sovereignty over the product to help make them great.

[0:15:15.2] JM: I was thinking about how – what you are doing what these APIs compares to one of these cloud providers. One way I was thinking that they might differ is the sensitivity of something like S3, or RDS, or load balancing infrastructure, it's a super sensitive infrastructure, super high uptime. Data labeling is in many cases an offline process. It's not on the critical path for a business. I'm wondering if that lower – not that it would lower the quality of service that you would need to guarantee, but in the event that it's going to take 30 minutes to get a piece of data labeled, rather than 20 seconds, or 5 seconds, is that fatal for you? Is the scale API infrastructure on the critical path of your customers? Or is it okay if you have variable latencies?

[0:16:20.3] AW: It's a really good question. I think in general with as the world of infrastructure, infrastructure loosely used changes, different infrastructure products will require different – well, different dimensions upon which they have to be extremely good, right? A simple example of this is you're right, a lot of these AWS infrastructure, it's very important that they have very high uptime, they have low-latency, but those are their key metrics. Then if you think about something like Stripe, it's still very important that it has very high uptime. The latency matters a lot less than some of these AWS services, but then it also matters that they have low fraud rates and that they have a great developer experience so it's easy to integrate.

It’s just like, each infrastructure product is slightly different characteristics that are required. You're right, which is the fact that for data labeling, latency it turns out matters a lot less, because there's an expectation that and this is what happens in practice, right? That because there's this active labeling process where humans will go through the data, that could take longer.

That being said, there's other performance dimensions upon which we have to be extremely good. One of those being data quality. I think it's interesting question in terms of whether or not Scale is on the critical path. There's one way to look at it which is okay, if we label data a little bit slower and then the new model trains up a little bit slower, that might not be the end of the world for the customer. If we provide poor quality data and then the model retrains on that data and then starts behaving poorly when it's launched into production, that's very bad, right?

At least the current way a lot of these machine learning pipelines work, the critical thing is actually data quality and bias, much more so than the performance – the latency, or op time of our service. That means we take it much more seriously, that we take this quality aspect much more seriously, then I think a lot of these other services might.

[0:18:31.7] JM: How do you gracefully degrade?

[0:18:33.8] AW: It's a really good question. Oftentimes, if you think about services, you want – whenever you have issues to first of all, overly communicate with your customers. If you notice that maybe something's going wrong, and so you won't be able to deliver as high-quality, or you won't be able to deliver as much data as originally expected, that these both in an infrastructure level, as well as a human level, surprisingly, you notify the customer that that thing is happening.

Then other times, what we do, or one thing we've built in is there's some efficient frontier of quality and latency, right? If we have more time to do stuff, we'll be able to do it at higher quality. If we have less time, we'll be able to do it at lower quality. We give our customers some amount of trade-off there, which is okay, you have the option of – you can get all your data sooner, it might be lower quality, or you wait, you'll get higher quality data.

I mean, one thing we notice is that quality is king in general when it comes to data. Then sometimes, you have other needs. Like say, you have a deadline and you need to improve the model. It really varies. That's really one thing is in general infrastructure as you degrade, in particular where these decision – like right now, because of this latency stuff, it's okay if our customers or developer makes a decision that you give optionality.

[SPONSOR MESSAGE]

[0:20:05.6] JM: Feature flagging makes it easy for your team to quickly change the way that your product works. CloudBees Rollout lets you manage feature flags easily. When you have a solution to manage feature flags at scale, you're empowered to continuously and intelligently roll out changes as soon as they are code complete on any platform, even mobile.

You can decouple development from code releases for real-time change control, you can rollback only the changes that you don't want, or keep them around. You can toggle features, you can use multivariate flags for A/B testing and you can remove misbehaving features with a kill switch. All of this is part of the feature flag platform that is CloudBees Rollout.

Visit softwareengineeringdaily.com/cloudbees and try a free 14-day trial and experience how CloudBees Rollout can help you with every release. Visit softwareengineeringdaily.com/cloudbees to get a free trial. CloudBees Rollout is trusted by large users, such as Zendesk and jet.com. Try it out today at softwareengineeringdaily.com/cloudbees.

[INTERVIEW CONTINUED]

[0:21:29.4] JM: One of the reasons it's important to have this graceful degradation in mind is this is not scaling the Uber workforce. Because in the Uber workforce, there's a lot of people who can drive cars. I mean, they can play this supply-and-demand game, where you ramp up brand advertising if you need more drivers. Eventually, that's going to get you more drivers. Or you offer them better deals and you start to take a – you start to take a hit to your margins, but you can scale up the workforce very quickly.

With the scale API, it's a little bit different because training somebody to label polygons, for example, that might be a little bit more tricky, or at least non-standardized than driving. If you move up the stack, it gets more and more specialized. For example, you could imagine scale for I don't know, labeling tumors, right? I mean, we already have this conversation in the public, Nexus, that the radiologist is going to be out of work. You and I both know that that is glossing over the subtlety of what will actually happen to radiologists. Maybe radiology gets turned into more of an API request with a human in the loop. What's interesting about thinking about that is it makes me wonder if there are white collar data labeling jobs in the future.

[0:23:06.1] AW: It's an interesting question. I mean, in some sense, there are definitely – I mean, for example if you're an engineer, there's definitely processes that you do that one day if you're a believer in machine learning, could be significantly automated. Code reviews for example, is a clear task that it requires a lot of semantics. Right now, we're not suited to solve it. Eventually, we're going to have the technology that can help automate that. We will have technology that will – you'll have models that understand what are parts of this code that you would likely flag and you could have some tool that is more automated.

Or for example, even writing lots of boilerplate code. Software engineering is one of – I mean, I think most people think it was a very – high judgment job that might be slow to be automated. Even a lot of these more repetitive things that you do as a part of it, you definitely can imagine a good amount of automation in the future using machine learning. I definitely think it's machine learning and "AI" is very profound in the sense that it has the ability to just speed everything up, speed up all of these decision-making processes that humans do in the long-term, once as technology improves, etc.

Again, code reviews probably not going to be automated any time soon, but maybe one day. It's it's much more – like you're saying, it's much more nuanced than like, "Oh, it's just going to be fully automated at one point." It's not a one-zero problem, right? It's not like, you're going to go from doing all the time to just fully automate. It's going to go pretty slowly from okay, you're doing it without any assistance, then you have a little bit of assistance to help make it cognitively easier and help guide you and help you not make mistakes and also help your load with stuff that stuff that you wouldn't otherwise be able to get to. Then slowly over time, it'll just become more and more efficient. There's a slow adaptation to more machine learning in the world, I think.

[0:25:11.2] JM: I want to know more about managing a supply and demand pool like this. There's a variety of ways you can deal with it. I've read about how Mechanical Turk handles it. I've read about how data labeling shops in China, obviously there's a lot of machine learning going on in China as well and they have a lot of data labeling to do. You've got competitors, like CrowdFlower, although I guess CrowdFlower I think is Mechanical Turk under the hood, but I'm

sure you have some competitors. What are the places where you can differentiate in the process of managing that labor pool?

[0:25:57.7] AW: Yeah. Again, this is something we think a lot about. I think we very much approach this whole problem as a product and engineering problem, versus an operations problem. I mean, first of all, that philosophy impacts you a lot, especially – and it compounds over time as you build various things. In particular for example, we've always taken the approach that the labelers are not actively managed by a supervisor, or anything like that, which is how you would approach it if you were more operationally focused. They're managed automatically through systems that are able to constantly monitor their quality and give them feedback in cases where their quality is degrading, or the way they can improve.

Much are able to actually, much better and more fine-grainly help guide a labeler than if somebody was 10% of the time over their shoulder watching their work. A lot of it is taking this approach, which is if you were to build a product suite that were to aid labelers in getting trained up and becoming more efficient, and understanding how well they're doing, etc. You were to start from that point how sophisticated and how efficient could you make the whole process?

We've definitely seen, especially over time, like we've been working on this for three years. First of all, our gains have compounded, right? The product gains that we get continue to snowball and give us more and more infrastructure that lets us build better and better and more efficient processes. Also, it's the gap between that and if you were to solve this as a purely operational problem continues to widen.

It's a pretty good question. I think it might not be obvious. If you really think about it, there's a lot in terms of a human management, or in a purely operationally managed process that is very inefficient. If you're able to replace the small things that in a human management process, like providing feedback, or setting expectations, or understanding how well people are performing, etc. If you're able to productize a lot of those functions, then you can produce a much more efficient, much more scalable system.

[0:28:22.4] JM: What you said there about the product and engineering focus, rather than the operational focus, like when I think about Mechanical Turk versus Scale, I sometimes think

about what is the corollary to the Fiverr and Upwork marketplace? What I'll say about that is Fiverr and Upwork, I am fascinated by these platforms. I am a power user of them. I've been able to basically – I don't know if I would be able to build this business in the same way if I didn't have these knowledge work gig economies. It's quite interesting that you never really hear people talk about the gig economy in terms of Fiverr and Upwork and other online work platforms. It's always the manual labor, or the Uber, or the tremendous freedom of the gig economy has definitely extended to knowledge work, which I think the corporate world hasn't fully grappled with yet, but that's a different topic.

In my interaction with Fiverr, you occasionally do have these issues where there's debatable quality, especially on creative work. Think about creative work, it has a lot of subjectivity. Like you asked for a website design, somebody gives you a website design and you're like, “That sucks.” This person was – they had a bunch of five-star ratings and they produced bad design work. I want to refund.

Then you have this additional subjectivity problem of the customer support center. Then Fiverr not only has to institutionalize how they deal with subjectivity across the core platform, the product engineering question, but then they have this deeply operational problem of how the customer service department like A, scales and B, is standardized in its practices. It just sounds like a complete headache. How have you managed to avoid that?

[0:30:26.6] AW: It's a good question. Yeah. I mean, again, the devil is always in the details, right? There's always nuance when it comes to the edge cases and how do you scale these things up, is a really good question. I think there's a couple a couple insights here that make things as more and more feasible. As a general rule, it's there's a couple simple principles that help.

First is you want to measure anything that you want to improve, right? It's simple enough. An organization's always say it, like make what you measure truism. For cases like this, I mean, part of the challenge with the way that you framed the problem, which is oh, you have this subjectivity in creative work and then you have – and then you have to figure out how the customer support agent is going to have to deal with it. A lot of it is you're framing it – again, as this very subjective problem, which makes it very challenging to operationalize against, right?

Now I'm not saying – we don't do website design, so I fully appreciate that website designers – There's like, yeah, yeah. There's a layer of shrew inherent subjectivity that is difficult. As much as possible, you want to build measurement into things that you need to operate against, right? This is one reason why we invest so much into quality measurement systems, right? Because it can't simply just be a case that we produce some data for a customer and then they come to us and just say, "Oh, we think this data sucks." Then we have to say, "Oh, okay. How does it suck? Etc."

It's always about, what is the very measurably, what is the magnitude of the issues? What is the actual magnitude of the errors? Then how do we build different processes that can support dealing with different kinds of errors and pushing all those error rates down, is one way to look at it. One approach is measurement as much as possible is really critical. Another is that and this is a general rule of building infrastructure, right? You always hold yourself to a higher standard than the customer holds you to. It's simple, right? We have some internal SLA, which is higher than our external SLA, because we need to really make sure we don't breach the external SLA. Simple stuff like that that builds this operational capability and this project mindset around how do you build with just very robust systems.

[0:32:57.8] JM: You're building this system for people to label data and have a job. At the same time, you're building a company where people work. There's a lot of changes to the way that labor knowledge work is done these days. I'd like your predictions on how work – I'm more interested in the corporate side of things, but I am also interested in the in the knowledge work gig economy side of things, but particularly your experience.

Shifting your experience to the idea of building a company, how do you want your company to look in five years or 10 years, in terms of how much work is taking place at the office? Do you want it to be remote? How do you want the work to proceed? Do you want people working shorter work days? Give me your perspective for what an ideal work structure looks like as your company evolves?

[0:34:05.4] AW: Yeah. It's really interesting. It's something that I think is – you're exactly right, which is right now we're in this – we're in somewhat of an upheaval, because there's this "old

way of doing things, just cannot be the most efficient." There has to be something better and I think we're – there's a lot of frankly experimentation to understand what the right approach is.

I think there's a couple things that are really – that are important to keep in mind. The first, which is maybe the overall insight is that talent is pretty evenly distributed around the world. That is certainly true. It is certainly true that it's not – not all the great engineers work in the Bay Area, in Seattle, in New York. There are great engineers literally everywhere, everywhere where there's Internet and there's access to computers. You're lying yourself you don't think that, right? It's really important to recognize first, that talent is evenly distributed. Then a lot of people don't necessarily want to live in the Bay Area. If you take a very talent first approach, then you need some structure to support a global engineering, or whatever other craft, workforce.

The other thing though that is actually – that is tricky that people talk less about is that culture and norms around how work gets done are very difficult to communicate, like extremely, extremely difficult to codify and communicate efficiently across, especially globally, but also even with people you're in the office every day with, it's not particularly easy. That's maybe the main challenge or the main problem, right?

Certainly, you could have a global engineering force. The real question is how efficiently are you communicating the culture and the work norms to all these people around the world? How are you doing that effectively? I think that's maybe the – if you're to say, what's the blocker to every organization going fully remote right now? I think that is maybe the core blocker. Because organizations are defined by their culture. If the way that you build your team means that you can only communicate the cultures, or culture ends up being more of a soup, or more closer to the average culture, then that really hurts you.

I do think some people are doing really interesting things to solve this. GitLab, they have an employee handbook, literally anybody on the Internet can go look at it, which is really incredible. They just write everything down. Even then, I think, I don't think that's perfect. I don't think that fully solves the problem, because there's a lot that's implicit interest how people interact that is relatively nuanced.

I don't have a solution. I think the approach that we take that I think works is you have – you always have some Nexus, or some relatively concentrated areas where there's a lot of people, where culture is uniform, well-communicated, and then you support people who work remotely, work all around the world and you figure out how to communicate that culture to them. The simplest way is they just – it turns out they spend some amount of time in the office, or in one of these nexuses and they will absorb it and grok it and meet the people they work with, etc. Then it's more stable.

[0:37:33.3] JM: It's worth pointing out that we're doing this interview in person. We could have just as easily done it over VoIP. You and I both know that there are unspoken communication points. Eye contact, am I distracted, how is my body language when I'm reacting to a question. These things are proof points of a culture. Do people feel comfortable on average? Are people actually focused on their work? These are things that you can't totally convey with as much bandwidth over the Internet. It doesn't matter how many words are in your employee handbook. I'm sorry. Employee handbooks don't get communicated through osmosis. You have to actually read it.

Compiling code takes time and reading an employee handbook is compiling a lot of code for your brain. I think it's a lossy process. I'm fascinated by GitLab, but I am – we'll see, right? I think the jury is very much still out on whether that experiment will work. I think a lot of it comes down to what is the tooling that we need? I'm very glad GitLab is running this experiment. I think it's beautiful. I'm a huge fan of SEED. We'll see. Is Slack good enough? Is Zoom good enough?

[0:39:02.0] AW: Yeah. Again, I think it's great that these experiments are being run, right? Because they're until you actually run the experiment, you'd never know. I do think – I'm really hopeful that – Yeah, I'm definitely hopeful that GitLab continues to be as wildly successful as they have been. I'm also hopeful that more and more companies try different slight variations, right? That we figure out what the exact – maybe not the exact right, but what a very reasonable mode ends up looking like.

It's really exciting, because there's new tools coming online every day. There's more and more information that people just absorb, right? It's really interesting. One thing that's been crazy, right? If I just think about the book *Zero to One*. When *Zero to One* first came out, all of the

ideas I think were quite fresh, they were really new, they're really interesting. At this point, most of the ideas from *Zero to One* have basically enter the lexicon of startup people, right? Now, all the ideas seem trite. It's pretty crazy, right? Just in the, I don't know, it's published maybe four or five years ago. In those four or five years, all those ideas have become operating standards, or truisms to most people in the community. Even the culture of Silicon Valley and the ability for us to learn move so, so quickly. I'm very optimistic that we'll be able to grok this remote work problem.

[0:40:29.5] JM: Do you really think that book has been fully internalized? I think that book has a lot more depth that has not been fully realized by Silicon Valley, particularly the more controversial areas of it, like scapegoating, right? People don't really talk about that aspect of the book. There's the scapegoating idea. Arguably, Elon Musk was the PayPal scapegoat, right? People don't really talk about that.

[0:40:55.9] AW: To be clear, yes, there's definitely ideas in there that are not – that people don't – people have not fully internalized. A lot of the ones have really been, I think – have really – I mean, a lot I just certainly have become very commonplace, in a way that I think that five years ago when you read it for the first time, the big ideas, the things that you would read about it when you first read it, or when you read it when it came out that would seem shocking or interesting are very different from the ones that he would read now, that when reading the book you would find shocking or interesting.

[0:41:27.4] JM: That I agree with. Although I wonder, is that a product of just being a – because that's so true of all the good books you read. I'm re-reading *Hard Thing About Hard Thing*. I read that book – I've read it a couple times in the past. I feel like I'm reading a totally new book.

[0:41:42.5] AW: Yeah. I mean, there's some function here that's really hard to understand, right? It's a function of where your head is at and the experiences that you've groked and had in the meantime. Then those shape your comprehension of the words that you're reading off the page. It's hard to say. It's always hard to say.

[0:42:00.3] JM: Under that same rubric of reflection and things seeming different at different times, you were at Quora for what was it? A year and a half?

[0:42:09.6] AW: Yeah, roughly.

[0:42:10.9] JM: I think, you at Quora starting when the company was probably two and a half or three-years-old, right? Or somewhere around then?

[0:42:19.3] AW: Yeah. I'm trying to remember. Something in that range. Maybe three, three or four-years-old.

[0:42:26.5] JM: It's interesting to think about the fact that you probably were at Quora around a period where you are now at with your company. I bet you still feel like, “Oh, my gosh. This is crazy. The wheels are coming off here. I'm having trouble managing my psychology here. This person's about to quit. This person's joining and they're asking for way more compensation.” Then if you imagine yourself joining Quora, probably back in the day you're like, “Ah! I'm at a new job and it's great and this company so well-run. Oh, waffles in the morning and the omelets at night.” Isn't that interesting to reflect on how probably how intact the company felt when you joined, relative to I bet how you feel right now?

[0:43:12.4] AW: It's a very valid point, right? Which is when I joined Quora, again it was actually – it was even smaller in terms of head count than Scale is now. You're right, intact is the right word. It felt intact, it felt stable, it felt like most – a lot of things had been figured out. Yeah, if I think about Scale, certainly from my perspective I feel nothing's been figured out. We have a lot more work to be doing. We have so many things that we could be doing so much better.

Actually, the way I think about this actually is it's actually about the context that you gain. You made this comment earlier about how the communication of values, for example is very lossy, right? I have this belief in general about most things that – most things are really tricky to understand. The communication of them is almost always really lossy. When you first join a company, or just over early on in the company, you never really know what are all the nuances, or all the hard things, etc. That means that you think everything's intact, you think that there's – that things are good in general.

Then as you learn more and as you gain more context and you really understand all the problems that be, you gain a really deep appreciation, which is like, “Oh, my God. Things can be – There's so much potential. Things could be a lot better.” I think about over the time – over my time at Quora even and as I learned more and more, right? I had more and more confidence and more intuition about various things that could be done better. I understood more and more about how – I shouldn't say, basically how much opportunity there was. You really grok that over time as you understand the creepy-crawlies, right? Then at Scale, this benefit which is I've known all the creepy-crawlies, or I've gotten to know all the real – the nuances over time and it benefits me in understanding how we could be doing better.

[0:45:12.0] JM: Yeah, I've always felt Quora is super underestimated. One of the things that made me feel that way early on was I felt the company had a pretty distinct approach to engineering. I can't quite put my finger on why it's distinct, but I'm sure you have your ideas. What are the places where you disagree, or diverge with how engineering is done at Quora? Things that you have implemented at Scale where you're like, “We're not doing unit testing as much as Quora. We're throwing that out the window. Too many tests at Quora. Too much work.”

[0:45:46.2] AW: That's a really good question, actually. How you do engineering is deeply tied to the overall culture and style of the company. One thing that I think is certainly very different is that Quora as a product is relatively stable, in the sense that they'll launch a few large new products a year, but for the most part a lot of the optimizations are happening either under the hood or in small parts of Quora.

There's this view at Quora, which I think is fair given their product has worked to date, where it’s like this large complex machine and then – you take a holistic view and you have a more consensus-based view on how you tweak the machine, right? I think that philosophy, I really – especially based on what we're working on at Scale, whereas you mentioned where you have a lot of different products, there's a lot of surface area and there's a lot of the surface area on with our customers, their surface area with a lot of the algorithms use, surface area with the labeler side.

Because of all this – all the surface area. It's important that I think at Scale, that we just move really fast on making product decisions and it's more – it's much less consensus-base as more okay, there's a thousand flowers blooming or whatnot.

It's okay if a lot of people are running and potentially even different directions, as long as we're making changes and moving very quickly. Then this trickles down into some things about how software engineering ends up manifesting, which is overall the move-fast, break things culture, which I'm sure you've heard a lot about. I do think it's you shouldn't need to invest that much in testing.

You should take an approach where it's more okay for things to break and that you – the overall system is the engineers plus the code, right? That whole ecosystem will adapt to things. As long as you have a fast clock speed and a fast iteration speed, that's the thing that dominates. That was maybe one really core thing that I took away from my time at Quora.

Another thing that I think they did they did that I disagree with, but not – again, this is just – I think this is a different strokes thing, Quora invested a lot into data and invested and had very good hygiene and understanding of all the data on the platform and everything that was happening. What I noticed is the full emphasis on data had a cost, which is people operate less through intuition and the gut feelings that people had if it wasn't backed up in the data was very hard to figure out to justify those things.

I do actually think that especially today, in today's fast-moving world, you need to be able to act in situations where you might just have intuition, or you might just have a gut feeling. I think that – being able to do that, I think is really important, especially if you want to launch a bunch of new products over time, or you want to try crazy new things all the time. That's another thing that I think – I think there's a lot of organizations that operating this very data-driven manner, or I think Facebook does as well for example. I think that hurts in this ability to make to make good qualitative decisions.

[SPONSOR MESSAGE]

[0:49:26.0] JM: Looking for a job is painful. If you are in software and you have the skill set needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace that connects highly qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies. Access is exclusive and you can apply to find a job through Vettery by going to veterry.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters, so that you only get job opportunities that appeal to you. No more of those recruiters sending you blind messages that say they are looking for a java rock star with 35 years of experience, who's willing to relocate to Antarctica. We all know that there is a better way to find a job.

Check out veterry.com/sedaily and get a $300 signup bonus, if you accept a job through Vettery. Vettery is changing the way people get hired and the way that people hire. Check out veterry.com/sedaily and get a $300 signup bonus if you accept a job through Vettery. That's That's V-E-T-T-E-R-Y.com/sedaily. Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:51:15.7] JM: When we last spoke, we talked a lot about self-driving. I could be wrong about this, but my sense is that investing in self-driving, data labeling infrastructure really took your company to the next level. You built tools that were desperately needed by Waymo, Uber, Zoox, whatever, name your self-driving car company that uses your data labeling platform. Essentially, those tools if I understand correctly, were very well-designed, beautiful experiences for the data labelers, that allowed them the expressivity that they needed to efficiently label data, which is a profound insight and was a gigantic market depth for you.

It's also very interesting, because the number of customers is obviously somewhat small. You can probably count them on two hands. Well, maybe, maybe more at this point. I guess, there's

drone companies and whatnot. By volume, obviously it's the self-driving car companies, there's not too many of those. Just the depth and the volume of information is so juicy. It really took your company to the next level. Give me your perspective on where we are in self-driving car development.

[0:52:31.6] AW: Yeah. It's not an uncommon question in today's world. Fundamentally, I think here's overall where we're at, right? I think we're at a state where we know technologically where we need to get to. In some sense, the full engineering problem has been mapped out, right? Every year, the systems are getting better and better and better. Not only are they getting more data, but we're being more novel in our approaches, we're getting better sensors, we're getting better compute, etc.

Every year, we're getting closer and closer to this where we know from an engineering perspective we need to get to. Unfortunately, and this is just a reality in the industry that I actually think is fine. There were some very aggressive estimates. If you asked five people in self-driving when they think it'll happen, you'll get five different answers. The ones that were there earliest get publicized, right? That obviously means if they're the ones that are there the earliest, then maybe they're the least likely to be true. I think there was a lot of optimism that maybe incorrectly seeped into the public understanding of self-driving.

The reality is we know what we need to do to improve the technology. Some of the smartest people in the world are working on the problem. It will get there certainly within due time and I do think within the next couple of years, there will be deployments that are meaningful. I mean, maybe we'll all fly to Phoenix and you'll be able to see like okay, this stuff is actually really legitimate. The Waymo cars for example, actually do drive quite well. Despite what different articles will say, I mean, fundamentally they drive quite well.

I mean, these things always follow hype cycles, right? We're at the valley after the peak, right? We're at the point where, "Oh, maybe you could be extremely pessimistic." The reality is it'll happen. It only keeps getting better. It's just more of a – it's less one of those kinds of problems that you could set a deadline. If you're nearing the deadline and it's not perfect, you could just say – let's ship it and let's just patch the holes. It is just a different problem that you need to – by the time you're shipping it, just feel pretty good about everything that you have.

It just doesn't really follow that engineering paradigm. That it's famously popularized by Steve Jobs or whatnot, where you just set a date and you get it done. If it's not done by then, you patch all the holes and just get it out there. You just can't take that approach in this particular problem.

[0:55:10.0] JM: It's definitely not a problem where you want intuition to be guiding your product development.

[0:55:15.5] AW: Yeah. Yeah, for sure. A lot of the stuff that I think people thought would never happen are actually happening and are very exciting, right? I think for example, there was always this belief that LiDARs would be incredibly expensive. Well, it turns out that now there's manufacturers, like Hesai in China, that have made significant strides in the manufacturing process and manufacturing efficiency. If I had to bet, I do think in the future that LiDARs will be inexpensive enough, or yeah, inexpensive enough where it's actually, it's no longer this glaring price tag of a $100,000 or whatnot to put it on a vehicle.

I think the same will happen for specialized compute for self-driving. I think the same thing will happen for everything else. The forces of the marker are definitely pushing us to a world where it's much more feasible and so much less outlandish than it was in the past. Overall summary, still very optimistic on the technology, again knowing the exact state of where all the components are, where perception is, planning, prediction, etc. We're very close, and so it'll just be a couple more edge cycles. Now, it could be large number of edge cycles, but it is just – it is an engineering problem left. It's not some mythical unicorn that we have to go find.

[0:56:36.1] JM: What's the hardest problem you're working on at Scale right now?

[0:56:39.3] AW: There's a number of things that are hard. Maybe the most challenging, or interesting technical problem is one around how do you properly measure and combat bias in data? Bias in data and bias in labels, right? It's a pretty hard problem, because when we say data is biased, right, it's always based on some semantic subjective thing that humans grok from the data, right?

The famous example of AI bias is around race used in a lot of image recognition, or face recognition algorithms. In advance, it would have been opaque, to a computer, to the machine itself, opaque to a human. It's very hard to know that that bias exists in the data set. Unless, you're already looking for that bias, right? What that means is that bias has this unknown-unknowns problem, where there could be different kinds of bias in the data that unless you've built a system that is able to identify these issues, would just be impossible to identify until after the fact, right?

This is something that we think a lot about on behalf of our customers, on behalf of the whole machine learning community. We definitely think we have the ability, because again we're seeing all the data. We have the ability to actually build a solution for the whole community on this problem and combat it effectively, but it's a tricky problem because it's hard to design the exact solution that will work 100% of the time.

[0:58:15.4] JM: We're nearing the end of our time. I want to just talk a little bit about your psychology and managing yourself through building this company, because this is – what you've done is – I think it's – I don't know if it's unprecedented. I think it's unprecedented, just the velocity at which you've built this company. I mean, you just raised a 100 million dollars, series C, you're three years into the company. I love your approach, like the speed and the bias towards just moving quickly, deferring to human intuition. I think it's pragmatic. I think it's realistic. I think it's the company that you need to build if you actually want to build a large engineering workforce.

I think engineers are losing their patience with the big, old, dusty, mechanistic, heavily unit tested engineering process. I think engineers want to have fun and I think they want to work somewhere that's inspiring. Building that company is taxing. I know you must be thinking deeply about how much sleep do I get? Do I drink coffee? Do I get exercise? Do I take weekends? Do I take vacations? Tell me how you're managing that process.

[0:59:36.6] AW: I'm very lucky, which is that me personally, I'm a very forward-looking person. I've always dug various situations, but I'm always very – even when bad stuff happens, right? I'm always very forward focused in how do we – what's the potential in the future? How do we achieve it? Etc. I think very luckily, I think my emotional state has been stable, which is good

obviously. Then when I think about taking care of myself, managing myself, etc., I think the test that I always apply is right now do I feel continuing to push hard on Scale and do the next X things that I think are really important? Is that deeply energizing for me.

Realistically, there were times in Scale's history where it wasn't energizing for me, right? Or it's like, "Oh, I just want to play video games instead, or I just want to sleep instead." I think once you have this feeling that it's like, "Oh, the work you're doing is not energizing, that's a big thing to fix or it's a big bug," because these things – in general, humans they follow these positive feedback loops, right?

Either the work is energizing and that the success or failure of that work makes you more excited to do the next thing and the next thing and the next thing. You go on in this positive virtuous cycle. Or it goes the other way, where it's the work you do is not energizing and so you do it worse, and so the results are bad and then you feel even worse and then you get even less energized. Then eventually, you're really hating what you're doing.

The way think about this a lot of the time is how do you short-circuit whenever you're not feeling energized and deeply debug and understand what is it about – what you're about to do that's so dreadful, or so unpleasant. Then either just not doing those things and doing other stuff that energizes you, or coming to some conclusion about okay, it actually was this thing, and so that thing, I actually thought about now more and I'm okay with it or whatnot.

I mean, I'll give a really simple example, which is there were – this is I think of actually a really common thing that happens to founders is in the middle of Scale, there's a point where I think a lot of the people and organizational stuff was just – was really deeply non-energizing. It was like, I had to deal with it and it was – it was not the most fun thing. I knew I wasn't that great at it, etc. Then I think two things happen. First, I did less of it, which helped.

Then the second thing is I realized that okay, the thing about this that actually is really exciting is A, watching people get better and really improve. Then B, spending more time and work with people who inspire me. That was the reframe I needed and then I'm like okay, so actually I can focus myself on how do I improve people? How do I help them understand what they can be doing better, etc.? That was a very helpful refrain for me. Those are my 5 cents on the matter.

[1:02:56.0] JM: Great. Okay, last question. Tell me your most far-out idea about how the world will be different in 10 years.

[1:03:04.2] AW: I guess I'm more boring than some other people who might –

[1:03:07.3] JM: I know even your boring one is going to be good.

[1:03:10.1] AW: If we think through what we we're talking about before with the white collar labelers, etc. I mean, I do think there is – there's a very real scenario that in a decade, most of the important processes are run by supervised, learned machine learning algorithms. I think that's far more plausible than other machine learning algorithms taking over the world. There will be this very real scenario in the future, where most people their job is to in some sense, garden or observe, or manage an algorithm, right?

If you think about all the judgments that we make every day, if instead you're like a warden, or a gardener or whatnot, just slowly tending and pushing the model in various directions to help it get better and better at whatever you used to do, in the limit this gets to a point where the model is not blocked by the things that you are. It's executing at a 100X, a 1000X.

What you can do and interacts with other models, etc. You're tending to it. There's this real potential for a huge amount of just maybe efficiency gain, or just the world getting all the things that are holding us back right now, or not holding us back in the future, maybe discovering drugs, where it will just be this very fast, easy process in the future. Diagnosing diseases will be this instant process.

There will be all these things that just become easy and instant. Then all of a sudden, there's a challenge which is like, okay, what are the thing – at that point, what are the hard things, right? That's really hard to say. It's hard to say, imagine everything happens at a 100 times the clock speed is today. Then what's the things that humans, that their clocks be there thinking about. This maybe the crowd approach. I don't know if somebody has written a sci-fi on this, but somebody should.

[1:05:17.3] JM: Read [inaudible 1:05:17.1] books.

[1:05:18.7] AW: Oh, yeah?

[1:05:19.9] JM: I think so. I mean, sci-fi, or reality, depending on how you look at it. Alexandr, it's been awesome talking to you. I really enjoyed this.

[1:05:27.1] AW: Yeah. Thanks for having me again.

[END OF INTERVIEW]

[1:05:37.9] JM: As a programmer, you think in objects. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers in the cloud era.

Millions of developers use MongoDB to power the world's most innovative products and services, from cryptocurrency to online gaming, IoT and more. Try out MongoDB today with Atlas, the global cloud database service that runs on AWS, Azure and Google Cloud. Configure, deploy and connect to your database in just a few minutes.

Check it out at MongoDB.com/Atlas. That's MongoDB.com/Atlas. Thank you to MongoDB for being a sponsor of Software Engineering Daily.

[END]