EPISODE 1094

[INTRODUCTION]

[00:00:00] JM: Deepgram is an end-to-end deep learning platform for speech recognition. Unlike the general purpose APIs from Google and Amazon, Deepgram models are custom-trained for each customer. Whether the customer is a call center, a podcasting company, or a sales department, Deepgram can work with them to build something specific to their use case.

Sound data is incredibly rich. Consider all of the features in a voice recording; volume, intonation, inflection, and once the speech is transcribed, there are many more features that can be discovered from that text transcription.

Scott Stephenson is the CEO of Deepgram, and he joins the show to talk through end-to-end deep learning for speech as well as the dynamics of the Deepgram business and the deployment strategy for working with customers.

If you are interested in sponsoring Software Engineering Daily, send me an email, jeff@softwareengineeringdaily.com. The show reaches 30,000 engineers each day, and if you are interested in reaching those engineers as well, we'd love to hear from you. You can also become a paid subscriber to the show by going to software daily.com and clicking on subscribe. That would help us support the show and it would mean a lot to us. Thank you.

[SPONSOR MESSAGE]

[00:01:33] JM: Scaling a SQL cluster has historically been a difficult task. CockroachDB makes scaling your relational database much easier. CockroachDB is a distributed SQL database that makes it simple to build resilient, scalable applications quickly. CockroachDB is Postgres compatible, giving the same familiar SQL interface that database developers have used for years.

But unlike older databases, scaling with CockroachDB is handled within the database itself so you don't need to manage shards from your client application. Because the data is distributed,

you won't lose data if a machine or data center goes down. CockroachDB is resilient and adaptable to any environment. You can host it on-prem, you can run it in a hybrid cloud and you can even deploy it across multiple clouds.

Some of the world's largest banks and massive online retailers and popular gaming platforms and developers from companies of all sizes trust CockroachDB with their most critical data. Sign up for a free 30-day trial and get a free T-shirt at cockroachlabs.com/sedaily.

Thanks to Cockroach Labs for being a sponsor, and nice work with CockroachDB.

[INTERVIEW]

[00:02:57] JM: Scott Stephenson, welcome to the show.

[00:02:59] SS: Thanks for having me.

[00:03:00] JM: You work on Deepgram. Deepgram is an end-to-end deep learning platform for speech recognition. How does Deepgram compare to the speech recognition platforms that are available on the cloud providers, like the AWS or the GCP platforms?

[00:03:13] SS: Yeah. There are differences. There are real differentiators inside Deepgram. There are key deep technical differences on how the model works, but also the business model of the company. Some key differentiators there are that Deepgram does have a cloud API for speech recognition, but we also operate on-prem. So you can install Deepgram software on your servers. You could put it in your AWS VPC, or Google, or Microsoft, any of those. That's fine. And Deepgram works fully there. No features have been removed for that. This is a key differentiator and being able to be installed on-prem.

But one of the biggest differences is that we use end-to-end deep learning and how we do speech recognition. I know if you looked around sort of at a lot of competitors, they would be mentioning deep learning and using machine learning and speech recognition. It is true that they're using deep learning in speech recognition, but they're not doing it the whole way. They're replacing what's called an acoustic model, which is the part that guesses the phonemes

that people are speaking. Phonemes are the vocabulary of spoken words, like ah, oo, ch, th, all of that.

It's very typical to replace that with deep learning, but then the rest of the chain, which is a hit Markov model with a beam search and a text language model that's associated with that, that's all done in a traditional way. What Deepgram is doing is saying, "No. Audio should come in. Words should come out, and it should all be one deep learning model that can learn from itself." That's what we do and that has a really big differentiation on the accuracy side and the ability to attain just like very, very high-accuracy compared to competitors because you can train custom models for customers.

So the end-to-end deep learning allows you to do that because of its data-driven approach and sort of hands-off. You don't have any speech engineers involved in training that. It's actually a purely data-driven approach where you just label data and then train the model, and then you get like a 30% reduction in your errors. And then doing it again and get like another 30% reduction in your errors. That's one of the key differentiators for us along with scale then, which I know sounds like sort of stupid, you're like, "What? Google, Amazon, others. Why wouldn't they have the best scale? They have all the engineers. They have whatever." Well, their goal in life is a little bit different. They want you to use computation. That's how they make their money.

For Deepgram, we try to make it fast. It's more than an order of magnitude faster. And so you can save a ton on your total cost of ownership especially if you go on-prem. Those are the big differentiators for us.

[00:05:45] JM: On the comparison to the cloud provider, you really think that there's a misalignment of incentives there?

[00:05:51] SS: Well, yeah. I mean, from a product producer perspective, for sure, for the cloud companies. What they really want you to do is if you're Google, they want you to buy ads from them. That's their number one goal. And the number two goal is to get you using GCP, because they actually can make money that way. Amazon, it's a similar story. They want you to buy consumer-packaged goods. That's number one. And number two is they want you to use AWS. That's how they make money. All these other products on top of it are just dressing in order to

get you to do that. That also means that they like partner with Deepgram. So they know that their own product is lacking in many areas. And so they still would use Deepgram on their platforms in order to get people on to their cloud instances so they can still make money that way.

[00:06:32] JM: Interesting. So I'd love to know more about those partnerships. But first, I think we should talk a little bit more about what Deepgram actually is. If I build a Deepgram model, what does that actually mean? Give me an example of what a Deepgram model is and what it's going to do for me.

[00:06:47] SS: Sure. I can tell you some of the guts inside. Deep learning models are built element. I like to think of them that way maybe because I'm a physicist, but whatever. They have these elemental pieces like a convolutional neural network or a fully connected layer or a recurrent neural network or an attention-based mechanism inside. You basically choose from a few of these constituents parts. You choose some numbers, like how wide they are, how deep they are, how many layers you have, that kind of thing. But then after that point, you are hands-off and you're saying, "Okay, I'm going to feed you data model. I'm going to feed the model data," and then the model is going to make guesses and make horrible guesses at first, by the way. But then it's going to get slapped on the wrist and say like, "No. That was all very wrong. But if you move in this direction slightly, if you tune your parameters slightly in this direction, you'll be a little bit better." So you do that over and over and over, and like millions and millions and millions of times. And the model crawls itself to the way where its parameters are set up in the right spot to be able to have a high accuracy.

The way our models work is fully that way. It isn't just like a piece of it is like that. All of it is like that. In order to get a custom model, what you do is take the general model, which was trained that way and then you do what's called transfer learning on it. This is a little like cloning somebody's brain's state. You're like, "Hey, I've got your brain and I'm going to clone it, and then now I'm going to have you experience the world in a different way compared to what the general model was doing."

The general model was like listening to podcasts, audiobooks, phone calls, like all sorts of stuff. It was pulled in many directions and it has to be like okay at all those things. But it's not allowed

to focus on any specific one. When you train a custom model, you're usually doing that. You're picking like two out of all of those, or one out of all of those and saying, "Hey! Focus on this thing and get really good at it."

Typically, for us, what this means is a customer, they're a large brand that has a call center and the call center is sales or support or both and then we're making like a custom model for their sales or a custom model for their support and then they're using it at really large scale. The process is just get raw data from them, convert it into training data and then train the model using that training data and then host that model on the scalable API, whether it's on-prem or in the cloud.

[00:09:04] JM: There's a domain specificity here. If an enterprise comes to you, you're talking to that enterprise and getting their custom data as supposed to training on whatever off-the-shelf general purpose data that would be available to some other trained model.

[00:09:22] SS: Yeah, the important thing here is speech data is really hard to get right. It's really hard to source many variance of it. I like to think of it like, "Hey, there's a landscape," and the landscape is just super dark. It's like you're playing Zelda or something and everything is dark and you haven't mapped out the underground cave yet, but it's huge. It's a labyrinth. What you can do is light up certain areas by going and getting data like that. It means people have a certain age with certain gender and certain background and speaking certain words in a certain acoustic environment and all of that. You start adding all those things up, and man, there're a lot of different possibilities.

The world right now from a speech data perspective, there is not very much of it actually. There needs to be like two orders of magnitude more in order to make the models really, really, really good. But nevertheless, what you do is you have an operation that continues to do that over time. But the reason I'm bringing that up is like every speech company has the same datasets that you can just buy. Everybody goes to the same places and buys the same ones. You train your general model on it, but it's still not that good basically.

So what you have to do is actually start labeling your own and go out and build that repertoire. That might be crawling the web. It might be all sorts of ways to do it. But first, specifically for a

customer, like nobody has that customer data. There is no way for the model to have experienced it before. But now you just allow the model to experience it on-prem or in the cloud, and then now it gets like significantly better on that type of audio.

[00:10:53] JM: As I understand, your work in particle physics actually related to your work with Deepgram, just the fact that there are these very noisy high-dimensional spaces and analyzing them in particle physics realm or in audio realm, have some similarities. How did the two fields relate to each other?

[00:11:16] SS: Yes. This is really interesting, and that many really, hard to difficult problems in the world actually have like a similar form. They have really messy data generally in an unstructured format. And then you have to come up with some kind of way to get some intelligence in there, like label the data, like be able to make some kind a determination one way or the other. Is this the kind of thing we want or not, that kind of thing.

Once you have that, then you can start to think about making statistical models, or machine learning models, or deep learning models in order to learn how to tell the difference between the two. This is exactly what I was doing in particle physics. We had a deep underground dark matter detector. It was 2 miles underground. It was very much like a James Bond layer with yellow railing and cranes and everything. It was right next to the world's tallest dam in a tunnel that was bored-out by a tunnel boring machine. It was crazy.

But what we're doing is just running away from cosmic radiation, deep underground, because if you don't know it, there is radiation everywhere. The food that you eat, you, you're radioactive. Like everything around us is radioactive. Cosmic radiation is raining down. What you're trying to do is run away from that by going underground. But it just so happens to be that like rocks and other stuff are radioactive too. It's just not nearly as much as what's on the surface of the earth. But nevertheless, you go underground, and what we created was like a tub of liquid xenon, which is a really good detector material to give off light when a particle flies inside and then bounces off from something.

If you have a bounce happen inside the detector, it will give off light and it will give off charge. The detectors that we used in order to sense those charges were called photomultiplier tubes.

They’re analog devices, and each one of them has a waveform as an output. So you're tracking them, and there're hundreds of them. So you’re like in real-time tracking them and at the few nano-second level every sample and trying to figure out like is there something interesting happening in the detector right now or not? If so, we’ll save the data, because you can't save all of the data. There's like way too much. Anyway, so that's the first step.

Then the next step is, “Okay, if it was somewhat interesting. Now, is it really interesting? Was it actually dark matter or was it some background event?” that kind of thing. That form, you're talking about unstructured waveform data where you have to have like machine learning in real-time figure out if anything interesting is going on. And then after the fact, like analyze it for a deeper understanding. That is exactly what you're doing in speech. The way that we came into it was kind of just for fun. We wanted to make a backup copy of our lives. We’re like, “Hey, wouldn't it be cool if we had a backup copy of our lives? The best way to do it would be through audio. So let's start doing that.”

Once we got the like a few hundred hours of audio that way, we’re like, “Whoa! The discoverability problem is a real problem here. How do you understand what's inside all the audio? I’m not going to go back and listen to my life again. That's crazy.” And that's what really got the wheels turning on that, and then we started Deepgram from it.

[00:14:04] JM: The number of applications for audio, for very accurate audio models, it's kind mind-boggling. I mean, you could use it to transcribe dialogues. You could use it in real-time for conference work. Before we get in a little deeper into the engineering, let's talk about some of the high-level use cases. Tell me about some of the stranger use cases or the very specific domain-specific use cases that you've seen for these audio learning models.

[00:14:36] SS: We typically see usage in two large areas here, which is like you’re a large enterprise that has a call center and a data science team. If you check off those boxes, then your data science team probably looks at your call center data as like a very high-value data set, but they just don't have a way to get access to it. And what I mean is audio recordings are like dark data. How do you convert that into something that is more structured so your team could build like NLP models to understand it, or that sort of thing? Anyway, that's one side of it.

The goals that they're trying to achieve there are to train their agents better. So like their call center agents.

It's really difficult to train call-center agents because there are so many of them and the QA team that you have in order to review their calls and figure out what's going well or what's not, or that sort of thing, it's really small. Your QA team is really small. They don't have a huge budget for it. So they try to figure out, "Hey, is there some way that we can augment this with machines?" So we could find the calls that are the most interesting and then feed those to the QA team, and then the QA team will review those, rather than randomly, which is like what the QA team would normally do. Just randomly sample calls, which is like crazy. But anyway, that's how it is done in most of these companies now.

So it's like doing that sort of thing, being able to rank the most interesting calls and then have your QA team listen to it for training purposes. Then there's also an angle here of like what are my customers calling in about? What are they saying? Like, all of that. Again, y ou could have your QA team listen to it, but they're listening to like less than a percent of your audio. So if you want good statistics here, then you get as much as you possibly can and you can have 100% coverage if you're having a machine listen to them all and categorize them. That's part of it.

There's also a really serious like security compliance and fraud angle here where it's like you're constantly monitoring to make sure that there aren't account breaches or that people have permission, like your agents are always asking for permission to do the things that they're doing, that sort of thing. That's one side. That's the enterprise side. Then the other side is like voice platforms. Think about like meeting company, large cloud call center software, like large communication APIs like Twilio, or Nexmo, bought by Vonage recently, and things like that. Anyway, those types of things. They're building products for communication and they need like more repertoire there. They need more features so they can say, "Hey, yeah. We're communications, but were also AI, and you can build on top of us in order to build a fully functioning app." So, both of those.

[SPONSOR MESSAGE]

[00:17:10] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW CONTINUED]

[00:18:46] JM: The specific models that get built, you've got this domain-specific data. You've got maye thousands and thousands of call center calls that have already been recorded. Do you want to only use those calls, or do you want to augment with additional calls or with podcasts data? Tell me that what you would want in terms of data for one particular model.

[00:19:12] SS: Generally, there's a hierarchy. What you're doing is starting out super general and then you're transfer learning into something that's more specific and then transfer learning again into something that's like it even more specific. For instance, you might have a general model that's listening to everything; audiobooks, podcasts, YouTube videos, just like everything, right? That's your base. And then maybe you transfer learn a finance-specific model out of it, and it cares more about like earnings calls and phone calls from finance companies and things like that, right? But then you're a finance company saying like, "Hey, we need to figure out if people are trying to defraud us, if our agents are trained properly. We also want to do some real-time earnings call transcription and like sell that as a product or something."

Then you want to make specific models for that. You would say, “Hey, I want a model that's really good at the phone calls and I want one that’s really good at earnings calls,” and that sort of thing. And then we again break off and transfer learn into those domains. Basically, what you're doing is necking down the data that the model is allowed to focus on at the very most recent time. Even though it’s seen all of it in the past, you’re sort of allowing it to focus in. It's a lot like college student going to school, taking all these different courses and whatnot. But then they get their first job and then they specialize on that first job. It's a very similar process.

[00:20:26] JM: Tell me about the infrastructure that you use for building a specific model. What frameworks do you use and cloud providers?

[00:20:34] SS: Sure, yeah. Interestingly, at Deepgram, we don't use many cloud providers for our internal model training. There are several reasons for that. One that a lot of people would think of is cost, but that's actually not the number one reason. The number one reason is the size of the dataset that we have and the ability for the GPUs while they’re training to access that dataset. And then the other angle here is for the GPUs to communicate with each other, because our models are large and our dataset is very large. So there's a ton of inter-process communication that has to happen. And speech models don't train like in hours, or days, or even weeks. They train in like months and years. So you have to have a super reliable system that is having like high inter-process communication going on. Basically, it's a high-performance supercomputer that you have to use in order to build these speech recognition models. That's for the general model, by the way.

For the general model, that's true, because there's so much data. But for the specific custom model, that's not actually true. You can train those in the cloud. You can do the custom tuning afterward and all of that. But nevertheless – So we have own data center where we have our own GPUs, and actually NVIDIA is an investor in Deepgram. But like partially, or mostly, because like, “Hey, we’re like truly really utilizing GPUs in order to build this type of end-to-end deep learning system on the training side and the inference side.” So they definitely understand the technology side of like how Deepgram is building. But nevertheless, yeah. So we have her own data center, our own GPUs, our own data servers and all of that in order to make all these fast in the inter-process communication that happens there is all superfast.

But when you're serving it online, that's where I'll offer training, when you're serving it in the cloud, or on-prem, or any of that. Actually, those constraints are now lifted big time. It's just inference now. You use GPUs in order to host the model. And sort of the data access constraint, that's not a constraint anymore. The inter-process communication, that doesn't matter anymore. So you can actually host it there.

But the reason that we don't, like for our own like hosted service at Deepgram, the reason we don't host it in the cloud, that is because of cost, because it's expensive to – If people have tried to buy these GPUs or buy time on these GPUs from the cloud providers, it's very expensive. If you build it in-house, it's cheaper.

One of the problems there though is load balancing and making sure your resources are used well. The way that we do this internally at Deepgram is we actually have two types of loads. We have a training load for research and then we have an inference load. Our inference load is really high, which is like during the day, in the US at least. Then you take most of your resources and point it toward doing inference to actually doing the ASR for people. Then you back down your research load until the nighttime comes around and then you crank up your research load basically. That's how we built it. It's kind of spread all over, but this is a sign of the times that like," Hey, building these models is compute limited and you have to build a race car to do it, basically." So you're kind of in multiple areas in order to make it all work for you to train the model and for your customer to host it wherever they need to host it.

[00:23:32] JM: Is it a customer service heavy business? Because like I can imagine maintaining each of these individual models for different companies, like you're going to need to tweak the different models in different ways. The call center example necessarily the same as the – I don't know, mass podcast transcription and sentiment analysis example. Do you have, I guess, categories of model so that like over time you can improve the call center model and apply those updates to all the call center models, or does it have to be on this one-by-one ad hoc model basis?

[00:24:09] SS: Great question. It's a tree model inside Deepgram. And what I mean by that is it is a directed acyclic graph. You don't usually take like a model that has gone down that cascade

of general training, then necked down to a more specific category, and then again to a customer. You don't necessarily go backward with that or try to go backward with that. The reason is the general models are always learning in the background, and the category specific models are always learning in the background. Basically, you just want to keep like branching off over time.

The timescales for these are not like milliseconds or something. They're like months, or weeks, or something, right? Essentially, like every month, or couple weeks, or something, you're branching off these models and releasing them again. That's kind of our general model cadence on releasing new models and for updating custom models. It's a choice. A lot of customers don't necessarily want their models to be updated every week or month or something like that, because when you do that, your data signs team might need to verify that they're getting the results that they need or whatever, and they're happy with the results they were getting with the last model. So we'll just stick with it and pin to that model for a while and that sort of thing.

Anyway, we allow that flexibility inside Deepgram. So new models will come out. You're not forced to use it, but you can try it out and see if it's better accuracy or something, and that's generally the trend. That accuracy just keeps going up. And then you can switch over that model if you'd like.

To touch on like the, "Hey, this sounds complicated," part. It is in a lot of ways, but it was more complicated in the earlier years of Deepgram. We're four-years-old now. We got a pretty good handle on how to do all this stuff now. But the first few years of Deepgram, it was kind of a mess, but that's part of the progress of figuring out what you need in order to build these systems. But what we do now is allow our customers to manage which datasets should be labeled and fed to training and at what cadence and what percentage of audio should that happen to. It's generally a really small percent, like a 10% or less, or something like that. But nevertheless, maybe they want 10%, or maybe they want a way fewer percentage point there, lower percentage point.

But anyway, so yeah, there's a lot of flexibility there, but the way that we do it is through something that we're releasing very soon. I'm talking about it now, but we haven't necessarily announced it, but it's a way for a person to monitor via a console and control all of those

processes so that they can put them on autopilot or manually move them through their processes. So we've noticed that this is what our customers like to do. Generally, a very technical person will be monitoring these things and saying like, “Okay. I think it makes sense to release this new model soon. So we should start training on it,” or something like that.

There's a partial like offloading to our customers because they like it so much and they have access to it. They can check the accuracy. They can do all of that sort of thing. If that isn't your bag, we do it internally as well. Our researchers and model builders do that. But yeah, it’s a mostly automated process now, whereas before it was not. But yeah, now it is.

[00:27:08] JM: I see. So you give a customer a model, and there is some technical person at the company that is monitoring. That model closely, and may be occasionally they’ll take a sample, take a few customer service calls and make sure that that just scrutinize the data. Let's say, take few customer service calls and they’re like, “Okay. Is this being transcribed accurately?” “Yes.” “Is it producing the correct sentiment analysis?” “Yes.” “Is it doing the right topic modeling? I'm reading through this conversation. Are the right topics being created?” “Yes. It works well.” And then maybe they manually tested on the to be released Deepgram model version and they say, “Oh, actually, this one's even better.” And then they say, “Okay. Well, let's sexually upgrade the model, because that one's even better.” So it is kind of a manual sign off process.

[00:28:01] SS: Yeah. Generally, a manual sign up process. It doesn't have to be. You can make it automatic, so it's like totally hands-off. But usually if you're using it in that context and you’re building your own models and understanding models based on the input that Deepgram is giving you, then you want more control, and we allow that. Yeah. Yup.

[00:28:17] JM: Am I getting the main use cases right? Is it transcription, sentiment analysis, topic modeling? Is there anything else?

[00:28:26] SS: Yeah. Yeah. I mean, the big thing to talk about here is transcription. That's like what gets you from 0 to 1. If you're not doing transcription, then like start doing transcription first. Don't try to do any of the other things at first. Do the transcription and then build simple textual

models based on top of it in order to figure out like sentiment or other like category, topic modeling and things like that. Do it that way.

Under the hood, that happens at Deepgram, and the new products that are coming out. And the way that we think of ourselves as a company as well is we may go out and talk to people now and basically say, “Hey, we’re a speech recognition company.” That's actually underselling what happens on the research side at Deepgram a lot, where we’re really a speech understanding company and with speech recognition as the first product. We released other products now, like speaker diarization. So like you have a mono recording, like just a single microphone, but many people around that microphone speaking. But now you want to know like who said what, when, and that would be transcription plus speaker diarization. This is really important like for the meetings use case for collaboration platforms when everybody's in a meeting room.

We have things like language detection that are being developed now that aren’t released for general availability. But nevertheless, again, if you’re a meetings platform and you’re a global meetings platform, how do you know what language the people on the call are going to be speaking? You don't necessarily, and you might have a drop down that says like what language are you speaking? But like that's kind of stupid. You should have it be totally automatic. That kind of thing is what we’re building. But I like to think of it a lot like what is a person capable of when they jump into a conversation? A smart person that knows a lot about that conversation, but not necessarily the details or whatever, but what can they tell you about it? They could tell you who's speaking when. They could tell you what words they’re talking about. They could tell you a topic. They could tell you are they mad or not? That sort of thing. These are all things that we’re working on at Deepgram, but the thing that we sort of – We’re not going to release something unless it's like a big leap forward essentially.

The transcription is a huge leap forward. Language detection is a huge leap forward. Diarization is a huge leap forward. But some of these others, which like sentiment based on the audio rather than based on the text, that is a huge leap forward as well, but it’s not released yet. There are several reasons why basically, but mostly it's that the market is still young and people need to start with transcription. Start with transcription. Get all of your benefit from that. And then like six months a year, etc., later, you will be like, “Oh, okay. I understand how to test this. I understand how to value it,” etc. Then you can buy these more deeper products.

[00:30:57] JM: When you say a speech understanding company, what would that actually mean? I mean, I can imagine that if I'm doing speech understanding, I can get topic modeling and sentiment analysis. Those are aspects of understanding what is actually being said. Is there something like condensing a transcript? Like you have a transcript and you want to condense it down to a paragraph that summarizes it? Is that the kind of thing you're talking about?

[00:31:26] SS: For sure. Yeah, topic modeling in there as well. Like, there're sort of general categories. There are many, many possibilities here. And essentially, it's what could a smart human do? Could they summarize? Yeah. Could they tell you who's speaking when and all of that? But there are limits to this, which is like can you tell me if somebody's lying? Like, "Okay. That one's a little –" Or some people don't have a very good sarcasm detector, like that kind of thing. There are kind of limits, but yeah, it's really thinking about this problem as more than transcription or more than just the basic data that you use in order to achieve that goal. It's like, "Hey, why do people buy transcription anyway?" It's really to understand what's inside the audio. Is transcription enough to understand what's inside the audio? You can understand a lot through transcription, but does it give you intonation? Does it tell you many, many other things? The answer is no.

So that's really the angle that we're taking there, is like, "Okay. Transcription is great for many things. It gets you 70%, 80% of the way there. I'm not talking about accuracy. I'm just talking about like to your goal. But that last like 20% is like maybe 10 times the value as the first 80%. That's how we really look at it. Like the transcription market now is it depends on how you count, but anywhere from like 15 to 25 billion for the automated transcription side. But if you look at understanding just machines being able to understand humans, like you're talking like 10 times that market size in 10 or 20 years. But the way that you get to that point is kind of through transcription, which funds the development of these other tools and products.

[00:32:59] JM: How do you build these models? Can you tell me some about the frameworks that you use?

[00:33:05] SS: Oh, yeah. Sorry. Yeah, you asked about this before, and I glossed over that a little bit. So we use – On the training side, it's typically Python that is being used for the data

handling, munching, like that sort of thing. If there's something fast that needs to be done on the data handling side, then it will be done in Rust.

Basically, we use Rust and Python, and that's on the frontend. For the data stuff, that's for the training, and that's for the inference as well. You just sort of use them wherever it makes sense. But on the training side, we're using PyTorch as the framework. This is especially for sequences, for time-based stuff, PyTorch is for sure better than TensorFlow currently. Then the other frameworks that are around like do all right, but PyTorch is what we use internally and like in building. But PyTorch is not what we do inference in. So we build our own stuff in Rust in order to do that. So that's part of it. How do you do a massively fast inference with a new deep learning architecture that Deepgram uses? You kind of have to build a lot of that yourself. So we're doing that with Python, Rust, and then the support of some libraries to help PyTorch for training and a few others. But it's mostly built by us. We don't use like an open source speech recognition engine that was a toolkit, but now we repurpose it for Deepgram or something. Like it's all just built from the ground-up using the bear elements.

[00:34:26] JM: The difference between the language used for training a model and the language for inference. If I train a model and Python, it's ingesting a bunch of audio. I'm training it through this neural network, and then the output is what some PyTorch file or something like that, and then I can actually execute that model or call that model using a Rust program?

[00:34:53] SS: Yeah, essentially. Yup. So we would output the model with all of its weights, and that would be imported via – The way that's actually used is like, "Hey, I'll put it into a Docker container." And so it's containerized and easy to move around and everything like that. But just compressed and with some security in there to verify that it hasn't been tampered with and all of that sort of thing and pass it over to you and then, yeah, it's ingested and hosted via a Rust API that loads that model.

[SPONSOR MESSAGE]

[00:35:31] JM: If you haven't tried Airtable, one way to think about it is like AWS with a visual component. And I say that because it's a very new way of creating software. Just like when

AWS came out, it was quite a new way of creating software. Airtable is a low code platform with the ability to become code-heavy if you want it to.

Airtable has gone through years of development as a modern approach to a spreadsheet. And beyond being a spreadsheet, it's a modern approach to a database backend. With that effort gone through, Airtable introduced blocks, which is a system for building rich application components that sit on top of your Airtable backend providing a user interface, or a map, or whatever you want out of your Airtable block.

Airtable is now releasing custom blocks so you can build your own Airtable components. And if this doesn't sound exciting, that's probably Because you haven't tried Airtable. You can make Airtable custom blocks out of JavaScript and React components and they work through the Airtable SDK to enable developers to make their own functionality on top of Airtable. You can make a modern authenticated real-time CRUD app for users, their admins or for yourself. And if you're looking for the right time to get started, that time is now. You can enter the Airtable custom blocks hackathon at airtable.devpost.com and get started. It's a great hackathon. It's $100,000 in cash prizes, and you can start by building a database, then move up the stack to build a fully-fledged custom block using the Airtable SDK.

You can really build anything with Airtable, and you can compete to win $100,000 in cash prizes. So it's quite a good time. So just go to airtable.devpost.com, try out a new way to build software with Airtable. Thank you to Airtable for being a sponsor of the show.

[INTERVIEW CONTINUED]

[00:37:38] JM: What about features? So we did a show recently about Techton, which is a platform that came out of Uber. And one of things they were very concerned with is how to manage features across an organization. I think this was more in the scope of an organization that has lots and lots of machine learning domains and you might want to share different features between different sectors of the company. Is feature management important to working with just language models?

[00:38:12] SS: There are many things that you might want to do with the language model. Bu typically, from a transcription side, you only care about words, timings and confidences. You might want some more things on top of that. But the basic product errors, like are the words right? When did they happen and what confidence does the model have been those words? Then you can do a ton with it, right? But yeah, there are more features on top of that, like punctuation. If you ask for it nicely, will it give you access to some of the underlying features that went into the output of that model? So like the phonemes and things like that? So like the alphabet of sound that went into it?

You as a data scientist might like that if you're a data scientist and say like, “Hey, I could verify, cross-check this and look at the phonemes,” and be like “Oh yeah, that really is what was said in other methods and whatnot.” Yeah, there are other features there. Like, “Hey, I have a mono channel and I’d like to know who is speaking even though multiple people were speaking, but it's only mono,” and that sort of thing.

But the number of features isn't like massive and it does really help that we only work in the audio domain. So we’re not working on text. We’re not working on images, not videos, none of that. All of them need their own kind of augmentation for data and a way that you handle the data and the way that you label the data and all of that. We focus and specialize on audio only. So that allows us to kind of have like a medium level. It's not like a super huge problem for us. It's a medium level problem for us to like make sure that everything is on the roadmap and communicated appropriately to our customers and how to use and tutorials and stuff like that. But it's not a massive problem at least.

[00:39:46] JM: Tell me more about the infrastructure side of things. I understand that if you're training a single model, you’re using a Python library. If you're deploying that model and serving it, you’re using a Rust library. But what's the actual infrastructure replicable systematic strategy for how you deploy these individual things and serve them and monitor them from your infrastructure?

[00:40:11] SS: Yeah. Are you talking about like the hardware level? Because we can go there.

[00:40:14] JM: Either way. I mean, I was thinking of just the software level, but we can go into what particular chips you're using or how you're aligning those chips with models.

[00:40:25] SS: Sure. Yes. Well, I mean, it's important to think about the hardware side just because it does limit like what you're able to do. It places some constraints on it. Generally, what you're doing is taking either a 4, 8 or a 16 GPU node, and these are generally like NVIDIA's V100, and very soon, like in the next week or two or three or whatever, A100s from NVIDIA, which is their newest chip. But nevertheless, you load up a server with either 4 V100s or A100s, or 8, or 16 of them, and each one of those would – It's typically more like 4 or 8, 16 is more exotic, lesson that you're generally not going to do, but you might. You might for inference, like in order to save space in your data center, like having only a two GPU machine or something, which is totally possible.

But nevertheless, typically, if you're talking about training, you're doing like 4 or 8 and then you're cooking them together with InfiniBand or something similar that for this high-performance computing, this HPC type networking that has super low latency and very fast, very high throughput. So, gigabytes per second type stuff. That allows your models, like the GPUs, if they're connected in the right way and you have to set up like your number of CPUs and your PCE routes and everything appropriately and sort of order them and everything. It's a very like tuned process how this goes. But nevertheless, then you can have the GPUs all communicate to each other and work on the same model and shift those weights around and do their gradient passing and everything in as efficient way as possible with current technology. But basically, you're setting up.

It's rare that you're going above like 32 GPUs or something like that, because it's usually like some communication overhead or some other problem that you have. But you're setting up like maybe four machines working together via one of these networks and feeding it a ton of data and having it just chew through it. The frameworks and the way that that set up really is that you have a data server that has to be really fast, and that data server is – For deep learning, it's important to do like augmentation on your data. Because what you'll do is you'll show the model the same data over and over. You might show it to it five times, or a hundred times, or something like that. But if you're going to show it over and over, you might want to change it

slightly every time. You might want to stretch it, like change a frequency, add some noise in the background, like some stuff like that.

In the images, they do a similar thing of like flipping images or rotating them or whatever. But in audio, it's a bit different. You might add some reverb to it or something like that. But anyway, you're doing all this in a central data server and then serving that to all of the GPU machines that are crunching the numbers. And you're doing this like every hundred milliseconds or on that scale. It might be 500 milliseconds or something like that. And you're doing it to like minutes of audio at a time when you're doing it. The speed up factor is like huge. Meaning, like in real-time, you're crunching like hundreds of minutes every second or tens of minutes every second, that kind of thing, for just one model. So you have to do all that processing on the fly. So that's one key part of it.

The other part of it is the synchronization of the underlying training framework, where all of these GPUs are working together, and you sometime – Like they all are given different files. Those files may be of different links, etc. You may have like stragglers that are – Like you have 16 GPUs working on it or something, and one of them is taking longer than the other. Do you wait for it? Do you wait or do you say like, “Never mind. I don't care about your opinion anymore. We’re going to move on to the next one. You're going to abandon the job that you're working on and we’ll just switch and move on, press on forward.”

Basically, there's a way to calculate if you should do that or not. Of course, you don't sort of want to systematically do that over and over, because you'll probably screw up your model. But if you're doing and in a fairly random way and whatnot, then it works out. Anyway, but the way to make all of that work is all just like – There's no framework for that. You just have to build it all internally.

[00:44:21] JM: How big is your team at this point?

[00:44:22] SS: We’re over 50 people across the world. The biggest constituent being in the United States. But yeah – So, fairly large team. Now, that wasn't always the case actually. Deepgram was built with like a 10-person team, and only recently have we scaled up a lot as

sort of the market has come around and gained traction there and also our product is sort of finished and able to be rolled out. But yeah, it's still a fairly small company.

[00:44:47] JM: How do you divide up the teams?

[00:44:49] SS: Yeah, it's really important to have a – I think tech is super important. Deepgram is tech- first. Okay? So we're thinking what are the engineering constraints? What are the research constraints? What are the data constraints? Etc. We're always thinking about that first before releasing a product or anything like that. That might sound backwards to a venture capitalist or something, but there's no reason to build something in the speech world if it's going to like take up way too many resources or like you're not going to have the data to actually make the model good or something like that. We're always thinking about it from that perspective.

Of course, like the research avenues that we go down are led by our customers. But nevertheless, it's mostly that. So that's reflected in our organization of the company, where we have more researchers than we have engineers. We have more engineers than we have product people. We have more product people than we have marketers. We're very focused on the technical side and making it all work first. And we have even more people working on the data side than researchers actually. It's done that way. The go-to-market side, Deepgram is sales-heavy. We have a pretty big sales team. This is because there are so many companies that need it and it said difficult product to test out, basically. It's very multifaceted. Another way to say this is like if you're going to try to test out the prowess and accuracy of a speech recognition model, like how would you do it? You have to gather your own data on your side. You have to – And this data is precious to you in a lot of ways, like you don't want to necessarily just scattered around to make it public on the Internet, right? These are like internal recordings. You're not going to do that. So you have to go from a security perspective, that company. You have to then either do the test in their cloud or do it on-premise. That takes effort and everything on that side.

But also, like which data are you going to use? Do you go download a YouTube video and upload it or do you use like real data? If you want to get real data, how do you get representative data? That kind of thing. Anyway, there's a big education process right now in the

world, because it's a very new thing to do this the right way. But yeah, so we have a team that helps with that and an awesome customer success team that helps on that side of getting people up to speed on how they can use it and then being successful along the way. It's built out many areas, but it's definitely tech-first with like data being huge, research being huge and then sort of cascade down from there.

[00:47:08] JM: If you fast-forward really far into the future, what do you envision natural language understanding or the other domains that Deepgram might get into looking like?

[00:47:21] SS: The way I look at it is that you serve the enterprise market first, and they need transcription right now and they're going to need it for the next five years, basically. And that's a big business already. You can build billion-dollar companies there. No problem. But the next step after that is the understanding. And now you're talking like $10 billion company type stuff. Okay, fine. But that's still just enterprise, and the enterprise angle is awesome and I think it's where you need to start and that's where like you're going to get all of your product development done properly and everything. But then you can flip it back to the consumer side and say – Like you can actually go compete with Google and Amazon on the consumer side.

There are reasons why I think in the next like 5 or 10 years, Google and Amazon are not going to get this right. Basically, Google and Amazon care about ads and consumer-packaged goods or selling their like AWS or GCP infrastructure. Those are their number one goals in life. Their number one goals in life are not to build like an all-encompassing, awesome understanding system. Right now, they do perception, understanding and interaction, like these three crucial steps to like a conversational agent. They do perception using like a hybrid approach of some deep learning along with the traditional techniques. The interaction approach is a similar thing, but mostly deep learning actually on the interaction. What I mean by that is like their text-to-speech. So they come up with some words that they want to say and then they can say them and they use deep learning techniques for that, at least for their good models. But nevertheless, that in between, that understanding, is they're not making progress on that in a way that I don't think it's going to turn into the awesome way to have a conversation with a machine. It's going to be the awesome way to buy stuff or the awesome way for them to gather information about you so they can serve better ads to you. That's how they're going to build it, whereas Deepgram is like centrally speech understanding, focused. So we'll be able to come back 5, 10 years from

now and then say like partner with a phone company, partner with Samsung or others, or partner with Asus or a Wi-Fi company where you have Wi-Fi in your house, but now all those devices also have acoustic microphone arrays in them. And now you actually have a real home assistant. But Deepgram has taken care of the understanding in a real way.

We see that evolution happening in the home, in the car, on the phone, etc. But that's not what we're tackling first. That's a good way to kill your company to focus on that. You need to go secure a really rock-solid market that other people are not focusing on, that they're neglecting, and then you can go attack that much larger front later.

[00:49:49] JM: Well, thanks for coming on the show, Scott. It's been a real pleasure talking to you, and that's a bright vision for the future.

[00:49:55] SS: Thanks. I really appreciate. It was fun. I love diving into the tech side of things. I really appreciate you having me on the show.

[00:50:01] JM: Absolutely.

[END OF INTERVIEW]

[00:50:11] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[END]