SDS PODCAST EPISODE 1017: VECTOR SEARCH, AGENTIC MEMORY AND EFFECTIVE RAG, WITH MONGODB'S PETE JOHNSON

Jon Krohn:	00:00	Four out of every five organizations have AI steering committees and success metrics, which should set them up for AI success, and yet only one in five sees any return on AI investment. My guest today explains what these AI ROI winners are doing differently. Welcome to episode number 1017 of the SuperData Science Podcast. I'm your host, Jon Krohn. Today's guest is the exceptional Pete Jonson, field CTO of AI at MongoDB, where he spent the year so far on a world tour advising over a hundred companies on their AI strategies, all backed by more than 30 years of experience at enterprise tech giants like HP and Cisco. In this episode, Pete reveals which AI deployments are actually generating ROI, why the embedding model you choose can make or break your rag pipeline, and how token maxing became a cautionary tale of AI vanity metrics. Whether you are an engineer or an executive, this episode is packed with practical value.
	00:58	Enjoy. This episode of Super Data Science is made possible by Anthropic, Notion, and Gurobi. Pete, welcome to the Super Data Science Podcast. Where are you calling in from today?
Pete Johnson:	01:09	Jon, I'm in Cincinnati here today, which is my home base, although I get the privilege of traveling all over the world to talk to people about AI.
Jon Krohn:	01:18	Hey, Cincinnati, am I correct in remembering that that is the home of Proctor & Gamble?
Pete Johnson:	01:22	You are correct. Procter & Gamble, Kroger, those are the two biggest Fortune 500s here in their metro area. Unless you count Joe Burrow, which not everybody does.
Jon Krohn:	01:32	I don't know what that one is, so I guess so. He's our
Pete Johnson:	01:35	Quarterback.
Jon Krohn:	01:36	Oh, I'm mostly into the football where they use a foot to kick a round ball.

Pete Johnson:	01:43	The one that actually describes what the sport is?
Jon Krohn:	01:45	Fair. Exactly. Fair. Not hand egg. All right. Well, but you're at neither of those organizations, nor are you, to my knowledge, a professional sportsman. But you have spent more than 30 years helping enterprise tech giants like Cisco and HP navigate major tech transitions. And now you're shaping AI strategy at MongoDB. Tell us about what that role is like.
Pete Johnson:	02:08	Sure. I mean, I always start introducing myself. I wrote my first line of code as a sixth grader in 1981. Wow.
Jon Krohn:	02:16	And
Pete Johnson:	02:16	That was when you got beat up for liking computers, and I did at the time. So that I've gone on to be part of this community where I'm not coding by myself anymore, but I get to go to all these conferences and talk to people about all this stuff. So yeah, UC San Diego is where I got my computer science degree. It took me five years to get it. I'm not embarrassed to admit. But then yeah, I did 20 years at HP, about 17 of that in HPIT. I rode the wave of the initial writing of web applications in the late '90s. Spent a little bit of time as the hp.com chief architect where I was responsible for 1500 websites worldwide, a couple hundred engineers. And we went out and built HP's web presence both externally for customers and internally for employees. The last three years I worked at an organization called HP Cloud Services, which attempted to compete with AWS on top of OpenStack.
	03:13	And that did not go well in terms of business, but I met so many cool people there that got me the next 15 years worth of jobs, including this one that I have at MongoDB. So then I spent time at a couple of startups, some time at Cisco, some time at CDW. I've been in this role here at MongoDB as the field CTO of AI for about 13 months now.

		And like I said, I get to fly all over the world and talk to people about where they are on their AI journey.
Jon Krohn:	03:41	Lucky month number 13. Your title there is Field CTO for Artificial Intelligence. What does that mean to be a field CTO? I feel like probably a lot of listeners have some idea of what a CTO means in general, but what's the field adjective added on there?
Pete Johnson:	03:57	Well, it really means I'm old is what it means is field CTO -
Jon Krohn:	04:02	You've been put to pasture.
Pete Johnson:	04:03	It does. Field CTO is a progression from a sales engineer or from a solutions architect where the CTO part of it denotes that I've been in industry as long as I have, in my case, 33 years. And it's meant to connote some equivalence with somebody who might be in a CIO or a CTO role. Those are the kinds of folks that I have conversations with about MongoDB in particular and about AI in general. But you might have a progression to a solutions architect to ultimately to distinguish solution architect and maybe eventually to field CTO. So that's typically the title progression.
Jon Krohn:	04:45	Cool. I like that. That is news to me. Before we start talking about exactly what you're doing at MongoDB today, you're going to have to explain something to me from your past. Okay. Because you start writing lines of code, you said in grade six.
Pete Johnson:	04:58	Yep.
Jon Krohn:	04:59	And then it takes you five years to do your computer science and engineering degree at UC San Diego. What happened in between? Are you teaching yourself at the wrong stuff?
Pete Johnson:	05:08	Jon, have you ever been to La Jolla?

Jon Krohn:	05:10	No.
Pete Johnson:	05:11	The beach access is great. And I grew up in Southern California and suddenly I'm one of the best beaches in the world I have access to as an 18 to 23-year-old. And that explains the five-year plan.
Jon Krohn:	05:26	That does for sure. That's worth a victory lab. Definitely. I actually did. I took an extra year on my PhD at Oxford for similar reasons. Not the beaches, but there were other wonderful things about being there that I was like, "You know what? I could string this along a little bit."
Pete Johnson:	05:41	That's a way better mic drop than I went to the beach.
Jon Krohn:	05:47	Nice. I would take it. All right. So yeah, let's dig into what you're up to at MongoDB. In a recent article, you highlight that 83% of organizations have AI steering committees and success metrics, yet only 19%. So roughly four out of five organizations, and we're probably talking bigger enterprises in this case, but four out of five of them have AI steering committees, success metrics. So they're trying to do something with AI, but only one in five, 19% see matching ROI. So there's this gap where three out of the five organizations seem to have organizational structures set up to succeed with AI, yet they aren't. So how does that AI ROI gap happen?
Pete Johnson:	06:34	It comes from a couple different places. What I take this back to is last fall, the mainstream media published a series of articles that asked the very fair question, where's the ROI for AI? Collectively as an industry, we've spent all this CapEx developing the models. Where's the benefits? And I was at an executive dinner in New York City the first week of January, the NRF, the big retail conference that's there every year. And it was the first time I heard customer talk about that they had agents deployed in production and they were getting ROI out of them. And they came with two caveats. Caveat number one was they

were employee-facing. And caveat number two is that they were not autonomous, but they were human in the loop. The reason for choosing employee-facing use cases was twofold. Number one, the data security and data quality bar is lower than it would be for customers.

07:33 It's terrible if you accidentally leak someone's salary to another employee. It's way worse if you leak some customer's data to the wrong customer. But the other thing has to do with the ROI part of your question, which is I know how I'm judging the effectiveness of an employee. If I take a agent and I put it in their workflow and I see those metrics jump, I can attribute that jump to the agents and therefore back of the envelope compute some ROI. So that's a very long way of saying picking the right problem is key here. You got to pick a problem that you have good data for because if you put bad data into an AI ecosystem, it's not going to solve that. And number two, you have to have some metrics of success. You can't just throw AI at a problem that you don't know how difficult it is or you don't know how to measure the success of it.

08:30 And that's why these employee-facing use cases were so popular in the first half of 2026 is we already know what metrics we use to bonus people and to evaluate their performance. And like I said, if you put AI in their ecosystem within their workflow and notice the change in those metrics, that gives you an ability to then measure that ROI. So more and more companies are beginning to do this, but when those statistics came out in that survey, that was the problem is people were picking the wrong problems.

Jon Krohn: 09:04 Right. So instead of having your success metrics be related to how many employees have adopted AI, that kind of thing, instead the success metrics should be how much does an existing process get accelerated by having some automation of workflows within the loop?

Pete Johnson:	09:24	Exactly. And the best example I've seen here in sort of the second quarter of calendar 26 has been software delivery life cycles. So at the beginning of the year in January, we might have measured the success of something like Codex or anti-gravity or clawed code by the lines of code. How much code are you producing? And if you've been writing software as long as I have, and even if you haven't, you probably know lines of code is a terrible metric to determine the effectiveness of an individual or in this case of some AI. But the maturity that I've seen really in the last eight to 12 weeks, really since the Uber token maxing story came out, was am I shipping code faster? That it's not just about the effectiveness of writing code or the effectiveness of an individual engineer, but the broader business metric is, have I increased the speed it takes me to go from idea or bug report to production solution getting deployed?
	10:26	When you measure it in terms of those business metrics, that's really where you see the improvement in overall ROI value. Makes
Jon Krohn:	10:36	A lot of sense. And yeah, the token maxing thing was so silly. I did an episode dedicated to that because I'm basically encouraging my listeners and managers not to be using lines of code or number of tokens as a way to measure productivity because it's so easy with agents to just have them spewing things out and be using tons and tons of tokens for sure. Yeah, I think we've seen, it's funny that, oh yeah, the token overrun that you're talking about, it was something like Uber's expected allotment for tokens for the entire year 2026 ended up being used in three or four months or something like that.
Pete Johnson:	11:16	13 weeks.
Jon Krohn:	11:17	13 weeks. Exactly. Yeah. Wow. Yeah. I'm glad that organizations are waking up to the dangers of having that

kind of silly metric, vanity metric to track. I think Meta had a similar kind of story. They

Pete Johnson: 11:31 Did. Meta had their own token scoreboard that was measuring the number of tokens that individual engineers were consuming. And like you said, that turned out to be a bad idea. And among the things that's interesting about this AI wave of technology compared to others that I've seen in my career is how fast the cycles are. Anthropic dropped model context protocol on the Monday of Thanksgiving week in 2024. And by March, all of their competitors had embraced it. And here, I mean, TokenMaxing really got its shining moment in March at GTC when Jensen mentioned it on stage. And by the time the Uber story came out eight weeks later, nobody was talking about Token Maxing anymore. So the life cycle of these stories and how quickly we move on to new things is unlike anything I've ever seen in my career.

Jon Krohn: 12:26 For sure. It seems like being in the AI space, we're going to soon have a 24-hour news cycle just for what the popular packages and approaches are in our discipline.

Pete Johnson: 12:36 We're getting there. And as an engineer, that's the hard part is how do I keep up? How do I know where I should invest my learning? And how do I know when to sort of cut bait and move on to the next thing? That's the hard part about being an engineer in this ecosystem right now.

Jon Krohn: 12:50 Pete, this isn't a question that I had planned for you, but I think it's. Yeah, given what we've just been talking about, I think it might be interesting for listeners. It kind of begs this question, what you were just talking about. How do you personally, given that you always need to be right on the cutting edge of what's happening with AI, how do you stay up to date?

Pete Johnson: 13:09 Well, I'm fortunate in my job that I get to talk to so many different people. I've been calling it my AI world tour this

year because I've visited multiple cities. So I've made 19 stops on my world tour in the first six months of 2026 across six countries. I've probably talked to a hundred different companies. I get to listen. I mean, I spend a good amount of my time telling them about how MongoDB can help them, but I get to listen. What are you doing? What are you seeing? So because of the diversity of the different organizations that I get to talk to, I get to hear not just what one person thinks, I get to hear what a couple dozen people think and use that as a guide for where I go spend my time. And it's things like I think we'll get into a little bit later, things like better agentic memory, things like maybe not always calling the generative LLM and being a little bit more creative about how you use rag pipelines and why retrieval quality is important.

14:13 I get to talk to people about that kind of stuff and let that be the guide for where I spend my time.

Jon Krohn: 14:17 Nice. Yes, we are going to get into that shortly. But quickly, before they get there, an interesting point from another interview that we came across in our research is that you've said that there's no skew or stock keeping unit for AI. And that organizations should start by asking what are your problems? What are your data? What are your metrics to determine ROI? Maybe that's kind of a framework. Maybe you've kind of outlined a framework there for us on how we can be identifying the right metrics. I kind of want to draw a line under that conversation. We kind of talked about the wrong way to be tracking metrics of vanity metrics, but yeah, I'd love to hear your thoughts on this one. And right before you answer that, I'd like to commend you for using data as a plural term in this interview because I always do, and I feel like I'm fighting against the tide on that one.

Pete Johnson: 15:07 We're BFFs already then. So yeah, I started using this phrase, there's no skew for AI maybe 18 months ago to try to capture the idea with the IT decision makers that I

tend to spend my time with. This isn't like a one-time purchase. It's not like you buy one product and you're done. This is a technology wave that requires iterations, including some of the ones that we just talked about. What I tend to tell people is look at what are the top 10 to 15 problems that you're facing as a company? What do you have good data for? And then what do you have good metrics for? So you can tell the difference between before and after when you inject AI into that ecosystem. And that's why the shift that I mentioned before that I saw in January was so important is by focusing on employee facing, you have the metrics.

16:00 By focusing on things like call centers is a popular one to put AI into workflows. You know what cost per call is. You know what call volume is. If you see those things improve, then that's how you can develop ROI. But it's only because you have the metrics, you have the data that tells you what that looks like. And the same is true of software delivery lifecycle. We know how long it takes to ship product today. So if you improve that, then that's proof that the AI is helping. The next sort of frontier is what about professions or job roles or workflows that aren't as well-defined and don't have good metrics? How do we approach those? There's plenty of low-hanging fruit of the ones that do, and that's what I see enterprises focusing on is those kinds of workflows that they already have that data in the metrics.

16:50 The next challenge that we're going to see, I think sometime next year is now how do you tackle those that don't already have that well-defined workflow?

Jon Krohn: 16:59 Yeah, there is so much low-hanging fruit today for organizations. It is wild to me. And by identifying the opportunities exactly as you're describing where, where do we already have the data? Where do we already have productivity metrics or some other kind of metric that is related to what matters to us as humans and as a

business? That makes so much sense. It's going to be interesting to see what happens after that where once we start to get these processes in place, and I think it's going to be this ongoing adventure. Just like MCP comes out of nowhere and changes everything. These shifts are going to continue to happen, not just broadly within the economy or within the field of AI, but within your own business where somebody realizes on the front line, wow, I have an idea of how we can be integrating AI here.

17:49 And this wouldn't have been possible if we didn't already have all the automation in the past. Yeah.

Pete Johnson: 17:53 Things like data and security, they get amplified by AI. They don't get solved by it.

Jon Krohn: 17:58 Exactly. Which brings us, I think, pretty perfectly to Jevon's paradox. You did a great job in an article about AGI skepticism explaining what the term is in relation to bank tellers and toll booth workers. I've tried to define Jevon's paradox on air before, but I think you did a better job. Do you want to give it to our audience for the members who don't know what it is already?

Pete Johnson: 18:21 Sure. So Jevon's paradox comes originally from coal efficiency in the late 19th, early 20th century, that when boilers started to become more efficient, the thought was that, well, we're going to need less coal then. But what happened was people found other uses for coal. And the paradox there is if you make a resource more efficient, you would assume that you would need less of it when in fact you oftentimes need more of it. And the bank tellers versus toll booth workers is a comparison that I make there where the mainstream media, again, has this kind of job apocalypse talk track that they mention a lot as it relates to AI. And I tend to think that it's more nuanced than that if you spend some time looking at Jevon's paradox. And the comparison I make in the article that

you're talking about is between two different outcomes that we saw with toll booth workers and with bank tellers.

19:19 So I am old enough that when I was a kid and my parents were teachers, they get paid three o'clock on a Friday, my mom would take me by the hand and we had to walk into a bank and she had to write some things on the back of it and have a deposit slip. And we had to hand this to a human being like a caveman. And that's how we got money into checking accounts. In the late '70s and early '80s, we got direct deposit and we got ATMs. And the thought at the time was that bank tellers as a job would disappear, but Jevon's Paradox thinks differently. And in fact, what happened was two things. First, the nature of the job changed. What ATMs and direct deposit took away from the bank teller job was the redundant parts of that job. So that freed time for them to do other things like relationship banking, like small business loans, like will and trust services, which is not something you would traditionally associate with a bank teller job.

20:26 I think it's, if I'm remembering my stats correct, Bureau of Labor Statistics, 1975, there was 268,000 tellers in the US. And in 2024, there were 375,000. What it allowed the banks to do was you could now run a branch with fewer people. So what did they do? They opened more branches. It used to be you'd have one branch of each bank in a town. Now they're on every street corner. So that was an example of because that job had somewhere to go in terms of raising the level of human interaction, we got abundance of it. So that's when Jebon's paradox came into play. But if you look at toll booth workers, again, when I was a kid, you used to have to hand coinage to a human being before they would raise the gates and let you cross a toll. Then in the '90s, we got the RFID, the transponders.

21:20 Then now we've got license plate readers. There was no value place for the toll booth worker to go when

automation replaced the redundant parts of that job. So when you're looking at how AI might impact the broader job economy, what I would ask you to consider is, is it a job that's more like a bank teller where there's room for someone to jump up in that level of human interaction and value added? Or is it more like a toll booth worker where there isn't a place for that person to add up? I don't think it's as black and white as every job is in trouble. I think it's far more nuanced than that. And Javon's paradox is a big part of that. Yeah,

Jon Krohn: 22:00 I think we are definitely in the former situation and that this is more like bank tellers. I think that there's a huge amount of capability for people like listeners to the show who have. I think a lot of our listeners have hands-on experience developing AI systems, doing data science, doing software development. For any of those kinds of people, these kinds of tools, being able to use Claudecode, being able to use Codex, the Gemini CLI, those kinds of tools allow us to have so much more capability, just like the bank tellers. They no longer need to be spending their time putting a stamp on the back of a check and writing something banal into some accounting book. They now can be thinking about, okay, what does the client of my business want? Or what would make the experience of this product better for a user? And now every individual can be way more impactful than ever before.

22:56 And I've made the case that I think just like bank tellers, I think it means that an organization when they're like, wow, look at how much value I'm getting from this data science hire. Look at how much value I'm getting from the software development hire. How many more of these can we get? If

Pete Johnson: 23:08 Software engineering's going to disappear as a profession, then why is Anthropic hiring 50 of them? And why are they offering 570K a year comp for it? To get

Jon Krohn:	23:19	Their keystrokes. Yeah. Oh man. Yeah. All right. So this has been great for digging into your general philosophy on where the market is. Let's zoom in a little bit more on what you're doing at MongoDB.
Pete Johnson:	23:34	Sure.
Jon Krohn:	23:35	You guys are specialists in databases, obviously. The big thing historically at MongoDB, when I think about the word MongoDB for me in kind of a vector space map in my own brain, a lot of the neurons that go off when I hear the word MongoDB is NoSQL. Sure. So probably a lot of our listeners already know what NoSQL is, but just as a quick primer, maybe you can tell us how NoSQL is different from traditional structured databases.
Pete Johnson:	24:05	Sure. So if you look at the history of databases, the kind of triggering white paper was written in June of 1970, and that happens to be the year I was born by a guy named E.F. Kod, who was an IBM researcher. And he gave rise to this notion of what we now know as SQL and how do you organize data on Disc. And at the time, if you think about the issue that developers in our early stages as a profession were having at the time was the cost of storage relative to say compute and memory. We hadn't yet had Moore's Law kick in over 50 years like we have now. So you had to be very precise about how you laid out data on Disc. And that's why we see things like my wife and I share a mailing address. And if we both have an account with a retailer, let's say, you store the address once, you store customer records for each of us separately.
	25:08	And when you need to get a name and address of someone, you do what's called a join, where you do a search on one table to get the address, you do a search on the other table to get the name. You sometimes have a third table that connects the two. You're making three reads on Disc for that, but the storage on Disc is very efficient because you're storing the least amount of

information possible. This is the foundation of SQL and it's what everybody has been using and what gets taught in universities and coding schools and as the right way to do it. So the term for minimizing that amount of data on Disc is called normalization. And there's use cases for which it actually makes sense to denormalize. So where normalization will give you efficiency of how you store on Disc, what I just walked through, you have to do two or three reads on Disc in order to get that piece of information.

26:05 What if you denormalized that? What if you think about it as, okay, Disc actually isn't as expensive in 2026 as it was in 1970. So in that retail example, suppose you have my address and my name in one record, and you have my wife's name and her in the same address in a second record. I'm duplicating the address in both places. But from a performance perspective, I only have to do one read from Disc so it's faster. Now, the way that we nuance that with MongoDB is when we store that data, we're storing it in a version of JSON so that when it gets serialized off of DISC and we ultimately send it back to you from the API call, it's already in a format that your language of choice can immediately use. And that also adds to the fast benefit. So when do milliseconds matter for you versus when does the cost of DISC matter to you?

27:06 So it's like the scarce resource in 1970 was the cost of disc, but the scarce resource in 2026 is time.

Jon Krohn: 27:14 Great explanation. I love all the kind of historical background we're getting on pretty much any explanation that you provide today, Pete. You

Pete Johnson: 27:22 Got an old

Jon Krohn: 27:23 Bald

Pete Johnson: 27:23 Man on the show today, Jon.

Jon Krohn:	27:26	And you've got not quite as old, but getting there and very bald, even balder man interviewing you for people who. Well, I guess you can't really tell because of my head. I was going to say for people who are only listening to this as opposed to viewing it. But anyway, so at MongoDB, one of the places where you've been able to make a lot of impact in the AI world that we're in now is around vector search. Yes. And you have framed vector search as the new frontier, and you work directly with enterprise teams on the architectural choices behind production AI, including reg, retrieval augmented generation patterns, embeddings, vector search retrieval quality and evaluation. Your perspective bridges the technical details of these systems with the practical realities of making them reliable at scale. I mean, we're probably going to end up digging into this question, like the statement I just said for the whole rest of the episode, because there's so many cutting edge technologies that depend on great databases.
	28:33	But let's start with vector search. Why is vector search the new frontier? So
Pete Johnson:	28:37	It's the new frontier because of its importance within an application architecture that makes use of some generative LLM. So if you think about how this has matured, when ChatGPT came to market in 2022, its application architecture is you take a query or a question and you put it in the context window of the LLM. The LLM does its processing and gives you some very smart-looking natural language response. So by the time we get to spring of 23, you see headlines like ChatGPT passes the bar exam. The LLMs have two fundamental restrictions in them that they still have today. So number one, they're trained on public data, not proprietary data. So if I'm trying to solve a business problem for my business, how do I inject my content into the LLM without having to incur an expensive retraining or fine-tuning process? How do I do that?

29:33 Well, in 2024, people started to use retrieval augmented generation for that. The other limitation that you have is that the LLMs have a knowledge cutoff. There's a point at which the LLM vendors have to stop training and start to inference the models. And by their very nature, the LLMs don't know anything that happened after that. So in the spring of 23, if you had asked ChatGPT, what is MongoD The stock price today, it can't tell you because today is after the knowledge cutoff. In 2025, we started to see the MCP tools and tooling take advantage so you can make live calls out to places. And now you have the basics of what current agentic architectures look like, where you've got some rag pipeline, you've got some set of tools, you've got the generative LLM. You now take the results of one pass of the LLM and it becomes the query for the next pass.

30:31 So you have this loop, and then you have agentic memory that keeps track of things across sessions within that loop. And those are kind of the basic components of an application architecture when you're talking about building agents now. And vector search tends to be a big part of those RAG pipelines and of those agentic memory. And we can get into some of the details about how MongoDB implements that and why things like retrieval quality are important, why things like scale are important, and how things like our acquisition of Voyage AI help with some of that.

Jon Krohn: 31:04 For sure. Let's definitely get into that. So for applications like RAG, where we allow a system to be able to search over effectively an infinite number of documents, when we're pulling out what those relevant documents are, we're using vector search, which you were just talking about as the new frontier, what common mistakes do you see teams make when they try to optimize RAG, retrieval augmented generation, for enterprise grade reliability?

Pete Johnson: 31:34 So here's where the details matter. So the way that things started to change in 2024 when we started to introduce RAG into this ecosystem, you take some documents, you pass them through an embedding model, and you then store them in a vector database. That's the data ingestion part of RAG. Then at query time, instead of taking that query, the question, and putting it directly into the context window of the LLM, you instead take a side quest. You take that query and you pass it through the same embedding model that you used before. You then can do a vector search, or is it sometimes called a similarity search, and you get related documents that are related to the query that you put through the data ingestion phase. You then take those documents and along with the query, you put them into the context window. Now the LLM knows things about your proprietary information in a way that doesn't require fine-tuning or doesn't require a retraining.

32:31 And so that's how you get around that proprietary limitation of the LLMs that are trained on public data. The biggest mistake I see people making is assuming that that embedding model is commoditized. It's not. The basis for what you pick for that embedding model will have a direct impact on the quality of the retrieval you make when you try to do the semantic search with your query, with your vectorized query and get the similarity documents. This is why we purchased Voyage AI in February of 2025 is we saw this as a differentiating feature in the marketplace. There's a benchmark that all the embedding models use that's on HuggingFace, it's called RTEB. And depending upon the model, we score as much as 14% higher than some of the embedding model competitors. Most people choose an embedding model today based on what's convenient for them based on their cloud of choice.

33:29 What I would say the biggest mistake is not looking around and seeing if you could get better retrieval quality

out of your rag pipeline, what are some problems you could go tackle? If you could do 14% better, are there problems that you could solve that you aren't able to solve today with whatever your default embedding model might be?

Jon Krohn: 33:50 I didn't know about RTEB before, but I'll have a link to it in the show notes. It stands for retrieval embedding benchmark. And yeah, Hugging Face published it in October of last year. It sounds like a really useful benchmark for retrieval for vector search. And so I will have that for all of our listeners. So in the past, you've laid out quality, storage efficiency, and developer friction as the three criteria for good vector search. Do you want to tell us a bit more about those three criteria?

Pete Johnson: 34:23 Sure. So let's start with the last one, the developer friction. More people will build agents in the next three years than have built them in the last three years. In order for a broader set of software engineers to take this on, you need to make it easier to work with. So reducing that developer friction. And I'll give you a simple example. One of the things that you have to do when you pass that data through that embedding model is how big is my vector space going to be? Anybody who's taken algebra understands what 2D is and probably what 3D is. When you create an embedding space, you typically, you might choose 1024, 512, 256 as the number of dimensions. And while I can't think in 256 dimensions, the embedding models can. What's the right number of dimensions for you to choose for your use case?

35:18 And the truth of the matter is a developer has to iterate over them. So how can you remove friction from that iteration? You pass your corpus of data through the embedding model, say at 1024 dimensions, you run some tests and see what kind of retrieval quality you get. You then take your corpus of data, you run it through again at 512, and then you see what kind of performance you

get. And it's this balance between retrieval quality and storage space, both on disc and in the index in memory that a developer has to constantly iterate over. Every other embedding model in the market would force you to re-embed your corpus of data every time you want to try a different number of dimensions. But the voyage models have a feature called Metroska reasoning, which the name comes from Russian nesting dolls. If you think about how a Russian nesting doll works, you've got the big one, you take it apart and inside you've got the same thing, but at a smaller dimension.

36:20 When the voyage models generate the dimensions, they're ordered. So what that means is if you want to test from 1024 to 512, you just lop off the last 512 of what you already generated at 1024, and you can now run your tests again. You don't have to go through the mechanics of re-embedding your entire corpus of data a second time at 512. So it reduces the amount of time it takes you to test that iteration in a way that nobody else on the market has. So that's one example. There's several others that we have that's baked into the feature set. Things like shared embedding spaces where different sizes of the models are compatible with one another. You can very quickly switch between them for different use cases. There's something called auto embeddings where you don't have to manage the maintenance of your pipeline on your own.

37:10 We'll do it for you. You tell us what field in your JSON you want to embed based on. You tell us how many dimensions, you tell us which voyage model. We'll take care of the updating of both the embeddings and the index as that data either changes or you introduce new content into that collection. So that's all what I mean by the developer friction. And these are all things that MongoDB has built into not just the vector search, but this better together voyage embedding models and re-rankers plus the core MongoDB vector search. I

Jon Krohn:	37:46	Love that answer. Just as I have loved actually all of your answers in this episode, you do such a great job explaining every technical topic that you go into. You are really experienced at explaining vector spaces and everything related to it. So thank you for that. Something that's exciting with you mentioning some of the MongoDB functionality that is designed to reduce developer friction as they use vector search standalone or as part of RAG or whatever. Something else that Mongo recently announced is new shared embedding spaces. What does that mean?
Pete Johnson:	38:18	So most embedding model vendors offer two to three sizes of the embedding models and they come at different price points and with different base retrieval quality. So there's usually a small, medium, and a large. What we recently introduced is what we're calling the nano, which is an open weight version of the series four family of voyage models. And that one's free. That one's out on hugging face. And if you've got the right CPU on your laptop, you can run it there. Shared embedding spaces is the idea is that embeddings generated by one of those four models is compatible with all of the others. So what that sets up is I just talked about, okay, suppose for my development cycle, I take my corpus of data and I embed it with the large, let's say, and those are the vectors that I store in my vector database.
	39:09	But as I'm a developer who's trying to figure out what the logic of my agent should look like, maybe you don't want to incur any token costs. So maybe what I do is I download and I run the nano locally. I embed my queries with the nano, but I can now do searches against the data that was embedded with the large. You take a little bit of a retrieval quality hit when you're using two different models, but the embeddings between those four models are all compatible with one another so that you can eliminate token costs during your development cycle if you want.

Jon Krohn:	39:43	That's cool. That is a novel concept for me. I haven't seen that before. Really cool. First one in the
Pete Johnson:	39:48	Market to do it.
Jon Krohn:	39:49	Wow. Nice. That sounds really useful. You mentioned a few minutes ago a term that I've been chomping at the bit to get back to, which is agentic memory. Yes. And how that interacts with all these kinds. When we're building a production AI system today, getting the agentic memory right is critical to having it work. And it seems like you're going to have a lot to say with respect to that.
Pete Johnson:	40:20	Yeah. Some of the advancements we've seen in agentic memory here the first six months of 2026 have been in direct response to this token maxing part of the conversation that we just had. So in its early days, agentic memory was just real simple, short-term, long-term, where short-term was keeping track of all the responses that happened in this session and long-term was keeping track of responses that happen over multiple sessions. And anybody who's used any of the modern chatbots, which are slowly morphing their way into being agents. So whether you're a ChatGPT person, a Gemini person, or a Claude person, you've all experienced this where maybe a year ago it wouldn't remember anything outside of a session, but we got long-term memory and now all of a sudden it can remember things across sessions. So we think that there's something better out there than just short-term and long-term memory.
	41:11	So let me paint a case for you. So suppose a question comes in and my agent passes it to the question to the generative LLM and it generates a response. What if I can then save that response in a more sophisticated agentic memory so that when the next time someone asks a similar question, instead of automatically taking that question and sending it to the LLM and incurring all of the token costs that are associated with that, what if

instead I could do a vector search into my more sophisticated memory so that I could get candidate answers and then I could use a lower cost evaluative LLM, something like Quinn or Llama. Is this answer good enough to answer this question? If it is, you return it and you don't go to the generative LLM. If it's not, then you pass the question onto the generative LLM and you go through the cycle again.

42:11 So what we're starting to see with these more sophisticated memory types is it's a way to combat token maxing. And it also has the nice side effect that you get consistency out of it. We've all experienced something where if you ask a chatbot a question and then two weeks later you ask it the same question, because the underlying LLMs are probabilistic, you get two different answers. Well, in a business situation, that can be catastrophic. You need some consistency out of the answers. And if you generate the answer once, keep track of it in memory, able to pull just that answer and not just sort of less sophisticated short-term and long-term sessions, but a specific answer. If you can pull that with better retrieval quality on your vector search out of that agentic memory, now you can use that to avoid the call altogether and save on your long-term token costs.

Jon Krohn: 43:03 I love that. You mentioned at the end there the importance of designing these systems effectively for commercial use cases. And Mongo has obviously been developing a lot of products, optimizing in a way to allow organizations, the bigger they scale, the more efficiencies, the more value you're going to get from MongoDB. So in the past, you've connected vector search, embeddings and MCP to the idea that databases must store what's accessed all together, extending a decades old denormalization principle that you talked about right at the beginning of the episode into AI retrieval design. Can you pull those different concepts together into what MongoDB is delivering for its customers?

Pete Johnson: 43:51 So the root of what you're asking is how MongoDB vector search works. So let me get into those details for a little bit, and then I'm going to make a very old man statement that is either going to make you smile or you're going to have no idea what I'm talking about. So the way that this works, so like I said, within the core product, MongoDB stores things as JSON structures. So we call them documents and we put them in collections. So we don't have tables and rows, we have documents and collections. Each one of those documents is a JSON structure and you get all the flexibility that you would imagine with a JSON structure. And as an extension of that, what is a vector? A vector is just an array of floats. It's an array of floats that was generated whether you pass an image, a movie, a piece of text to an embedding model.

44:39 Ultimately, what you give back is an array of floats. So for us, that's just an additional attribute to the documents you already have. So if I just add one more attribute to the document in my collection that's now an array of floats, I can now generate a index, a vector index on that, and that's how the search works. But I have a choice here. I could choose to use data that's already in my documents. So suppose I have a book. I have a document that describes a book and there's an author name, there's a title, there's a number of pages, there's a year published, there's a synopsis, and there is a text field that is a URL to a JPEG for the cover. In the example that I just walked through, you could take the synopsis and you could generate vectors based on the synopsis.

45:33 And that's an example of when I do the query, my similarity search, my vector search, what I get back is some scores, and I'll have the document that has the actual piece of data in it. And that's how most people do it. What I would make a parallel to is in old school C language programming, you had a choice when you created a function or a method where you could pass by

value or you could pass by reference. And what I would argue is if you've got the data in the document with the vectors, that's like pass by value. But what about that cover? I don't have the JPEG in the document. I have a pointer to where the JPEG is. I could take that JPEG, I could run it through a multimodal embedding model, and it's still going to generate a list of floats on top of which I could generate a vector index.

46:24 When I do that search in that example, what I get back is a string that represents a pointer, a reference to where I can go find the JPEG. So it's not like the synopsis example where I have the data, now I can go about my way, but now I have a reference where I make a second call to go grab it from its system of record, but that gives me choice. I could choose to pass by value and have the data there in the document, or I could choose to pass by reference because I don't want to have to copy that data over in this example to MongoDB. Maybe I want to keep it in its system of record that there's reasons why I might want to do that. And I'm willing to take the runtime hit of making the second call to go fetch it once I get the vector search back.

47:06 So it's giving people that level of design choice. And in that example I just gave, not only could you do a vector search, you could also do a lexical search, you could do a hybrid search, you could pre-filter based on the author name or the year of publish. There's all kinds of mechanics we offer to make the nuance of how you're doing that retrieval far more powerful than you have with some of the alternatives. And it's all based on that same document model. So because it's based on that same document model, you get all the replication, all of the sharding, and all the additional security features that we already have baked into the core product. You kind of get that for free with the vector search. I

Jon Krohn:	47:47	Love it. It does seem like a pretty obvious choice for a data backbone. If you're going to be building AI applications, GenAI, agentic AI, it makes a lot of sense to use Mongo as your data backbone for sure. It's something that we've been using in startups, tech startups that I've had for over a decade now, and it still continues to be the default choice for us. Yeah, so much flexibility. It's
Pete Johnson:	48:13	Easy to get started and it scales with you longer term so that when you run into things like GDPR or you need to start getting data closer to users for different things, that's all baked into the core product.
Jon Krohn:	48:25	Awesome. Well, Pete, as I already said earlier in the episode, I have thoroughly enjoyed listening to everything that you've had to say in this episode. You are so slick at explaining technical concepts that you might think a diagram would be needed to explain it, but in a podcast we don't have the luxury of diagrams. Nope. So we have to rely on outstanding verbal representations of the ideas. And you absolutely nailed it. Really appreciate it.
Pete Johnson:	48:54	You are very kind. It's almost like I was raised by teachers.
Jon Krohn:	48:59	Yeah, and have decades of experience in databases and AI. It's all coming together really nicely for us. Yeah. So thank you so much for this great episode. Before I let you go, I have two final questions that I ask all of my guests. The penultimate one is, do you have a book recommendation for us?
Pete Johnson:	49:16	The only book I've read four times is Michael Creighton's Sphere. The movie is terrible. Don't watch the movie. But the book is at the height of his pre-Jurassic Park powers. It involves computers and time travel. And what else could someone like me ask for? I

Jon Krohn:	49:35	Love it. The time travel paradoxes that come up. I watched two time travel films or two films that involve time travel as key plot points over the most recent weekend. And it's hard to do that without some kind of issue coming up. But maybe Michael Craig can do it.
Pete Johnson:	49:58	He does it well. And now I'm wondering whether you watched, did you watch Avengers Endgame, Back to the Future, Austin Powers or Hot Tub Time Machine?
Jon Krohn:	50:09	Actually, none of the above, amazingly. I watched Star Trek, the 2009 version. So that was the first one directed by JJ Abrams. And yeah, Time Travel is a big part of that episode. And then on Sunday, I watched a comedy that I can highly recommend. I actually watched it for the second time in about a month. It's Nirvana, the Band, The Show, The Movie. That's the name of it.
Pete Johnson:	50:34	Well,
Jon Krohn:	50:34	I can
Pete Johnson:	50:34	Never get enough of Leonard Nemoy, so you got me with this 2009 Star Trek.
Jon Krohn:	50:40	Nice. There you go. Yeah, you get two Leonard Nemoys. Well, you get two spots.
Pete Johnson:	50:45	Two
Jon Krohn:	50:45	Spots. One Leonard Nemoy. There's
Pete Johnson:	50:47	Only one Leonard
Jon Krohn:	50:49	Nemoy. That's right. That's right. That must've been one of his final films.
Pete Johnson:	50:51	Yeah, I think it was.

Jon Krohn:	50:53	Final question that I ask before I let guests go is how can people follow you or follow MongoDB after this episode? What are your recommended places for that?
Pete Johnson:	51:00	Oh, sure. So both me personally and the company in general, very active on LinkedIn, so it's easy to follow us there. If you're looking for more company information, it's MongoDB.com or we do a lot on the YouTube channel as well.
Jon Krohn:	51:14	For sure. Great. We'll have links to all of those things and anything else we talked about in this episode, in the show notes. Thanks, Pete. Again, I hope to have you on the show sometime again soon because I felt genuinely like we could have talked for hours.
Pete Johnson:	51:27	Well, that's better than if you couldn't wait to get rid of me. Thank you so much for the time. I really appreciate you and the time of your audience.
Jon Krohn:	51:34	I loved having Pete Jonson on the episode today on the show today. I hope you enjoyed it. As much as I did in today's episode, Pete covered why the first agents delivering real ROI in production are employee-facing with a human in the loop since firms already know exactly how they measure an employee's performance. So any jump in those metrics can be attributed directly to the AI. He talked about why the embedding model is the most underrated choice in a rag retrieval augmented generation pipeline. He said that most teams default to whatever their cloud provider offers while models like MongoDB's voyage family score up to 14% higher on retrieval benchmarks. He talked about Matroischka embeddings named after Russian nesting dolls, which let developers test smaller vector dimensions by simply lopping off the end of larger ones instead of re-embedding their entire corpus. And he talked about his bank teller versus toll booth framing of Javon's paradox where telejobs grew

after ATMs because the role had somewhere more valuable to go.

52:34 And the question to ask about any job facing AI is whether it has the same room to move up. As always, you can get all the show notes, including the transcript for this episode, the video recording, any materials mentioned on the show, the URLs for Pete's social media profiles as well as my own at superdatascience.com/1017.

52:55 Yes, this was episode number 1017. That's where that number comes from. Thanks to everyone on the Superdata Science podcast team, our podcast manager, Sonja Brajovic, media editor, Mario Pombo, our partnerships team Natalie Ziajski, our researcher, Serg Masís and our founder Kirill Eremenko. Thanks to all of them for producing another outstanding episode for us today. For enabling that super team to create this free podcast for you, we are deeply grateful to our sponsors. You can support this show by checking out our sponsor's links, which are in the show notes. And if you know someone that would like to know how they can be getting a better ROI from AI projects or to be able to understand better how NoSQL or MongoDB style databases can be useful in the Agentic AI era, then share this episode with them. Review this episode on your favorite podcasting platform or on YouTube.

53:49 If you write an Apple Podcast review, there's nothing more valuable to us in terms of an action that you can do. So please consider doing that. And yeah, subscribe if you're not a ready subscriber. But most importantly, I just hope you'll keep on tuning in. I'm so grateful to have you listening, and I hope I can continue to make episodes you love for years and years to come. Until next time, keep on rocking it out there, and I'm looking forward to enjoying another round of the SuperDataScience podcast with you very soon.