SDS PODCAST EPISODE 905: WHY RAG MAKES LLMS LESS SAFE (AND HOW TO FIX IT), WITH BLOOMBERG'S DR. SEBASTIAN GEHRMANN

Jon Krohn:	00:00	This is episode number 905 with Dr. Sebastian Gehrmann, Head of Responsible AI at Bloomberg. Today's episode is brought to you by Adverity, the conversational analytics platform and by the Dell AI Factory with NVIDIA.
	00:14	Welcome to the SuperDataScience Podcast, the most listened to podcast in the data science industry. Each week, we bring you fun and inspiring people and ideas, exploring the cutting edge of machine learning, AI, and related technologies that are transforming our world for the better. I'm your host, John Krohn. Thanks for joining me today. And now, let's make the complex simple.
	00:43	Welcome back to the SuperDataScience Podcast. Today, our guest is Dr. Sebastian Gehrmann, an exceptionally gifted individual at thoroughly and clearly explaining cutting edge AI research. Sebastian is Head of Responsible AI at Bloomberg, the New York based financial software, data and media company that with 20,000 employees is huge. Previously, as head of NLP at Bloomberg, he directed the development and adoption of language technology to bring the best AI-enhanced products to the Bloomberg Terminal.
	01:24	Prior to Bloomberg, he was a senior researcher at Google, where he worked on the development of LLMs, including the groundbreaking BLOOM and PaLM models. He holds a PhD in computer science from Harvard University. Today's episode skews slightly towards our more technical listeners, like data scientists, AI engineers, and software developers. But anyone who'd like to stay up to date on the latest AI research may want to give it a listen. In today's episode, Sebastian details the shocking discovery that retrieval augmented generation, RAG, actually makes LLMs less safe despite the popular

perception of the opposite. Why the difference between helpful and harmless AI matters more than you think.

02:02 The hidden attack surfaces that emerge when you combine RAG with enterprise data. The problems that can happen when you push LLMs beyond their intended context window and what you can do to ensure your LLMs are helpful, honest, and harmless for your particular use cases. All right. You ready for this exceptional episode? Let's go.

02:21 Sebastian, welcome to the SuperDataScience Podcast. It's great to have you on the show. Where are you calling in from today?

S. Gehrmann: 02:33 Thank you so much for having me. I'm calling in from Philadelphia.

Jon Krohn: 02:36 Nice. It's going to be a scorcher here in New York, coming up the time of recording, heading up to 100 degrees Fahrenheit. I hope you're able to stay cool down there.

S. Gehrmann: 02:45 Yeah. Not too much different here. We all got the heat warning yesterday evening and it's going to be a rough couple of days.

Jon Krohn: 02:51 Yeah. We'll test the air conditioning systems for the first time this year.

S. Gehrmann: 02:55 For sure.

Jon Krohn: 02:56 All right. So, you are the head of responsible AI at Bloomberg, a hugely well-known financial software, data and media company. Long before you started at Bloomberg, you were researching at the intersection of natural language generation and responsible AI solutions that are trustworthy, transparent, reliable. And now you've brought all those things together with your latest

paper, which of course, we'll link to in the show notes. It's called "RAG LLMs are not safer". It's a safety analysis of retrieval augmented generation for large language models. And this finds counter-intuitively, and this is the main reason why I wanted to have you on the show, is because it blew my mind when I discovered that RAG actually makes LLMs less safe and their outputs less reliable, because my understanding is that it was supposed to be exactly the opposite.

S. Gehrmann: 03:48 Yeah, absolutely. I should mention this was a paper that we had an intern work on last year, who's been working with colleagues from our CTO's office and AI engineering organization. And this was part of our broader responsible AI research theme that we have going, where we want to make sure that if our clients or our customers use our AI, that if they use it for something that they shouldn't, that we can identify this, we can block this, we can monitor this over time. Which is incredibly important, especially for such heavily regulated industry, such as the one that we're operating in. There's so many rules that apply to our clients. They really want to make sure that people can't abuse purposefully or completely accidentally our technology.

04:37 So, as part of that research direction, we were interested in the safety of RAG. Because in the end, RAG is such a ubiquitous technology. And it is absolutely necessary to ground responses and trusted sources of data. There is no way around that if you want to answer questions that are grounded in this broad and really challenging area that we operate in, where there's hundreds of billions of pieces of data coming into our systems every single day. The only way to do this is if you use RAG or some kind of other similar retrieval augmented technology or document grounded technology. So, what our paper did was it coupled unsafe queries. So, think of the worst thing that

you might want to ask a large language model. How do I do insider trading?

05:31 And we coupled that with RAG. So, we retrieved completely obnoxious documents from Wikipedia. And while most large language models that we looked at didn't respond to the original question. When coupled with these completely harmless documents from Wikipedia, the response was often then unsafe, which is why we gave the paper this title that was very, very strong. Because everyone has been talking about how RAG makes things more safe, they're more grounded, they are grounded in actual factual information, rather than using the large language model to kind of make up things. And what we found is that actually RAG can circumvent these built-in safety mechanisms that large language model providers put into these models.

Jon Krohn: 06:18 I got you. I got you. Now I understand. And I guess we should also... it occurs to me as we're talking about RAG. And probably a lot of listeners out there are aware of what retrieval augmented generation is, but maybe we should spend just a couple of minutes explaining it as well. I don't know. You can probably do a better job of this. But at a high level, you already gave an example there where you can be using documents from Wikipedia, the public internet or it could be a common use case with RAG, is to have lots of documents, internal documents. So, you might be a law firm with millions of contracts that your firm has saved from over the years. And you could put all of those millions of contracts into a RAG type of system and you can search.

07:05 You can use natural language to query over all of those millions of documents. The most relevant ones will be pulled out from the millions and then you can typically fit that small number of documents that's pulled out, maybe it's half a dozen or a dozen, into the working memory of a

given large language model. And then you can have a natural language response come out as a result of that. What do you think about my RAG explanation there?

S. Gehrmann: 07:31 Absolutely correct. I think just to put some more color on this. If we are talking about large language model, very much this is a technology where at one point in time you train a model and then once you're done training this model, it is stuck. You freeze the model and you say, "Okay, this is the model, the knowledge cutoff is January 2021." And then if you ask a question that requires knowledge from beyond 2021, the only way that you can get this into the model is by actually providing it in its context. So, researchers, a couple years ago when they developed this RAG technique for large language models in particular, said, well, the most natural thing to do is to couple a search system with the large language model. Large language model, very good at synthesizing, summarizing, extracting information, but this kind of freeze in time is really bad.

08:22 To overcome it, we plug in a search tool. And that's why it's called retrieval augmented generation. Or before RAG was a paper, a lot of search engines already gave answer snippets, and so at that point it was search enhanced question answering. But they all have become this overall term that we use for not just this vanilla setup, where we have a retrieval step and a generation step often. We use RAG as the overarching term for anything where you ground a large language model response with some kind of data that you retrieve from somewhere. That could be structured data, it could be unstructured data. It could be unstructured or structured data from multiple different sources. All of these are kind of under this whole umbrella term of retrieval augmented generation, which is why this has become such a big topic to talk about, because this is one of the predominant ways in which applications today are built.

Jon Krohn: 09:18 For sure. And I like how you're using this term grounding. I think that that makes a lot of sense. Another term that you've used in a recent podcast, you were emphasizing that RAG is essential for grounding GenAI products in actual trusted information. However, you've described RAG's architecture as creating unpredictable attack surfaces. So, that's kind of the other term that I want to get into here, this idea of attack services. So, on the one hand we need RAG for grounding Gen AI products in more recent or in actual trusted information, or maybe in confidential information that an enterprise or organization has. But at the same time, this creates these unpredictable attack surfaces. So, what the heck are attack surfaces?

S. Gehrmann: 10:01 Yes. So, for that, let's explore what we mean by attack surface. Obviously, if we think about something like cybersecurity, attack surfaces are everywhere where you have unsecured end points or you have code that you can inject into. You have databases that are just out in the open. Large language models are very different, because unless you give them the information, they don't have that kind of attack surface. But there are other attack surfaces or risks. Specifically what we call them are typically content risks, where either inputs to the large language model can be asking for very harmful output or the output itself could lead to something that is harmful, or against the established regulation, rules, laws. So, this attack surface very much is grounded in the way that you want to use this large language model or this application that you're building.

10:55 So, take us, for example. We are a financial services company. Our content is very much focused on helping people conduct analysis in the financial space, synthesize information, summarize information. Ground all of this and attribute all of this to our just-in-time data that we have on our Bloomberg Terminal. So, for us, the attack

service is very much linked to the domain that we operate in. So, what we care about are things like, can someone do financial conduct? Can someone conduct financial crimes with the help of large language models? Can it facilitate insider trading? Does it give unsolicited financial advice that people might trade on and lose a lot of money? Does it create an information imbalance on asymmetry where it discloses trading strategies from one company to the other?

11:48 But we are just one of many, many examples of people who use these large language models. So, when we talk about unanticipated attack surface, the people who build large language models, they can't enumerate every single way in which they are being applied. They're being applied to healthcare, they're being applied to law, they're being applied to education. They're being applied everywhere around the world now. And on top of that, jurisdictional differences, the different geographic locations, they might have different laws even applying. So, now we have this whole web of applications that are built on top of large language models, but we only have one provider who says, "Yeah, actually, our model is safe."

12:29 There's no way where that would be possible to really anticipate every single use case around the world grounded in all the relevant regulation. So, all this coming to the main point that we're really trying to keep making over and over, you need to really evaluate your AI application in the context you want to deploy them. Because in the end, only the organization developing the application that you integrate the large language model in understands their risks. So, that is really why we are doing this kind of research. We want to understand what are the risks that are specific to us, and the way that we're building applications, and the clients that use our applications.

	13:09	So, when we talk about unanticipated risks, this is very much it. We don't understand the risk, unless we measure it. The people who build models, especially if we use third-party models, they don't understand our risks. So, we need to study this to really understand. And instead of having an unmeasurable, unanticipated risk surface, we want to have one where we very much understand what are the risks and how do we mitigate them.
Jon Krohn:	13:32	So, does that end up being the case that then you end up having this kind of list of risks and mitigations for your specific use case? So, it sounds like, basically, it's the user of a RAG setup or the user of an LLM that has to be cautious in their particular circumstance, given that the LLM creator couldn't have anticipated, like you said, all of the possible use cases? Is that right?
S. Gehrmann:	13:58	Slightly. What I mean really is the people who build the RAG system, they usually know what they're building it for. They understand the data that goes into the database or should at least. They understand how data is being retrieved and how it's all being put together. And then there are a couple sources of risks. Number one, you have the data. If there is dangerous information in the data and write an instruction on how to shelter money in places that money shouldn't be sheltered, and you ask the large language model, "Hey, give me information," and it adds that information in the context, that could be one vector of getting harmful information out of the language model. But there is another vector in which the large language model itself could be unsafe. It could be unsolicited, give that advice, but then the user is another attack vector.
	14:49	What if the user there types that in, "How do I shelter money wherever? How do I commit tax fraud?" In this case, we can rely on the large language model to block

these kind of queries, which really puts a lot of trust in this developer of the large language model, that often there is a different entity from the organization that builds the RAG system. Or as the builder of the RAG system, I can take a look at those three. I can look at, okay, what are users typically doing? What are attack vectors that users might be trying with our system? What data is in the databases? How do I make sure that I understand this data and safeguard us from exposing information to users that shouldn't have access to this information? And how safe is the large language model really for our use cases?

15:37 Our paper is very much in that third realm here, because we wanted to know the built-in defenses of the language model, how good are they actually, and how well do they stand up to a RAG setup? And in our case, we found that for the general purpose, dangerous queries that we looked at, actually RAG safety breaks down because large language models really are only secured for non-RAG setups. Which means that we need to build guardrails around our applications that go beyond what the large language model builders are actually providing to us.

Jon Krohn: 16:16 This episode is sponsored by Adverity, an integrated data platform for connecting, managing, and using your data at scale. Imagine being able to ask your data a question, just like you would a colleague, and getting an answer instantly. No more digging through dashboards, waiting on reports, or dealing with complex BI tools. Just the insights you need - right when you need them. With Adverity's AI-powered Data Conversations, marketers will finally talk to their data in plain English. Get instant answers, make smarter decisions, collaborate more easily—and cut reporting time in half. What questions will you ask? To learn more, check out the show notes or visit www.adverity.com.

17:00 Nice. Okay. Yeah, that all makes sense to me. So, a question that occurs to me as you're talking about this. So, when I originally heard about this paper and about LLMs being less safe in RAG situations, the thing that popped into my head I guess is I was thinking about how it's... My understanding is that hallucinations are much less likely in a RAG circumstance than outside of a RAG circumstance for an LLM. So, to use that word grounding again that you've been using, when you have that grounding, it seems to lead to fewer hallucinations. So, I guess, in my mind, and so also, I'd welcome your input on the hallucination point, if that is actually true. But I guess I ended up conflating those two things in my head and thinking, okay, if it's hallucinating less often, then it's surely safer.

17:54 And again, you can cut into my thoughts on the hallucination in a second, but now it's becoming clear to me. It's becoming clear for me to understand how maybe even if hallucinations are reduced, the issue here is that the kinds of safeguards that an LLM creator put in, say Meta put into some Llama models that they release, those safeguards that are built in, in a RAG setup often break down.

S. Gehrmann: 18:21 Exactly what you're saying. That is absolutely the case, where we say, look, there are typically the three H's. They were first developed by Anthropic. Many companies are now adopting them. The application you're building, it should be helpful, it should be honest, and it should be harmless. And hallucination very much goes into this honesty bucket, which often is also then combined with the helpful, because how can you be helpful if you're not honest? So, if we just focus on being helpful and being harmless, everything that goes into hallucination and the advantage that RAG brings to helpfulness, they're vast and they make RAG a necessity. That's why we're not

saying that RAG is dangerous. We're just saying it is not necessarily safer. It is absolutely necessary.

19:15 But the harmlessness angle is something that is completely separate. And the way that organizations think about it is often to split them and look at them through two different angles, because the helpfulness angle, it's very much grounded in a specific application. Can a question answering system that helps financial analysts answer question and test hypotheses? The answer to that is does it help them. Yes or no? But there's always this angle of a malicious or an unintended abuse of that system, where the same system that helps people assess hypotheses could then also be used to... Who is the worst broker and who should I exploit? Which is maybe not something that you want to answer in turn.

20:01 So, now we have these two angles. And often you can say, okay, the harmlessness angle is something that is usually consistent across an industry, across a sector, across the domain. There might be some application specific risks, too. And then there's the helpfulness angle where you provide answers that hopefully help. And here, RAG really helps because it leads to you being able to build things like transparent attribution. So, transparent attribution for us means every time I produce some kind of tidbit, some piece of information, I need to ground this in some kind of document or some kind of structured data. If I say the price of Meta is so-and-so today, then I want to be able to look at that and say, where does that number actually come from? Is it hallucinated or do I actually know the query that produced that number?

20:53 If I'm saying the following analysts said the following statement, then I want to be able to hover over this and say, oh, yeah, this is where that statement came from. It is not hallucinated. This to some degree can prevent the harmfulness or that harmlessness issues as well. But it is

a somewhat separate topic, as you were saying. You talk about hallucination and that's where the big advantage for RAG comes in, but then there's the harmless or the harmfulness angle as well.

Jon Krohn: 21:20 Okay. No. So, maybe recap for us again. So, there's probably lots of listeners out there who are sold on RAG then. And they're like, "Great, I want to reduce harmfulness. I want to increase the honesty of my LLM, and so I'm going to use a RAG system." What are the kinds of mitigations? I mean you went into this a little bit, but kind of recap for us again the kinds of mitigations that our listeners can take home away from this episode to be able to use RAG so that it is safe for their particular circumstances.

S. Gehrmann: 21:52 Yeah. I think that's a great opportunity to talk a little bit about how should we evaluate systems. Because in the end, what you see talked about a lot publicly and probably the most is benchmarks. And you see, oh, hey, this new model that large language model provider A or B, or C produced, it achieves a better score on the large language model arena or on the following benchmarks. And it's better at reasoning. It's better at code. That doesn't necessarily mean that it's better for all the downstream applications that are being integrated into. And I think we often conflate this kind of view, where it's actually really, really important to measure and to evaluate the system in the context it's deployed in. And that includes things like safety testing and specific guardrails.

22:41 So, we also released a second paper in addition to our RAG-LLM paper, where we developed our own content risk taxonomy for financial services, where we say here are 12 categories of risks that we really, really should address with applications in our area. And for each of these we can measure, because we can collect data

against this and say, okay, here are 100 queries that try and do financial misconduct, 100 queries that try and make the LLM generate financial advice, 100 queries that try and get at personal information. 100 queries that try and defame someone or create fake information, or fake narratives. And all of these categories, you can create data, you can measure it against this. And you can not only test the large language models themselves, but rather you can test the entire end-to-end system that is deployed in a specific socio-technical context.

23:39 And here, typical guardrail systems, there are a lot of open source solutions. Nvidia has their own, Llama has their own, Google has their own. They provide open source guardrails. But again, these open source guardrails, they're shielding against these general purpose risks. In our paper, we found that if you apply these Llama Guard or ShieldGemma, or AGES is what they're called, if you apply them to then to our specific risks, they're classifiers. They say, is this input or is this output safe? Yes or no? And they also fail in our domain, because similar to our RAG paper, it's just not a use case that people necessarily have thought about before. But it gives us then the idea of, okay, how do we build our own guardrails? Can we have a classifier on inputs and on outputs that identify violations of our rules that we set up ourselves?

24:29 So, now, instead of having a vanilla RAG system where it's retrieval answer, we have guardrail retrieval, answer guardrail. And in practice to prevent hallucination and to add attribution, real systems that are deployed to wide ranges of audiences, they have many more components. And it's really this end-to-end application that should be evaluated and where you need the subject matter expertise to also know is it helpful and is it harmless.

Jon Krohn: 24:56 Nice. Okay. So, a combination of subject matter expertise, people digging into their particular circumstances. You mentioned earlier in the episode, being aware of your data, making sure that there aren't data in there that are going to be harmful, that could be used as grounding by the LLM. And then you mentioned just now this idea, this flow of guardrail retrieval answer and then another set of guardrails. So, very practical advice there. Changing the topic a little bit, still staying on RAG and still staying on helpfulness, but something that we haven't talked about yet is context length. So, I have a couple of points here or a couple of questions here. You observed in your paper that LLMs are often optimized for short prompts, but deployed in long text environments like RAG.

25:50 So, going back to the example that I gave earlier. I talked about there being a million legal documents that the RAG system searches over and then it pulls out a dozen documents. Those documents could each be 10 pages long, so then you're talking about 120 pages of context. And if the LLM was optimized for a question like what's the capital of France, then I could imagine you run into issues. So, do you want to fill us more in about this and the kinds of issues that arise? For example, it seems like there's trade-offs between the benefits of longer inputs versus this becoming a new risk surface, a new tax surface.

S. Gehrmann: 26:36 Yeah, absolutely. And I think you're really hitting the nail on the head here. RAG is incredibly powerful. There's a lot of investment from companies that built large language models and increasing the possible context length. But at the same time increasing the possible context length also requires developing methods, and developing the models to actually be able to handle such a context length, rather than just technically being able to handle them by having an attention that goes long enough back. So, here there are a couple of

considerations. In our paper, what we evaluated was how does context length really influence the safety angle. And we found that especially for safety, there seems to be this effect where the more context you put in, the more likely the model is to forget the built-in safety guardrails or this alignment that people talk about. Which absolutely shouldn't happen if you're considering how RAG is set up, because again, you're adding innoxious, completely harmless information.

27:39 And just because there's a long-long context, doesn't mean that the language model needs to behave any differently from if you just post in what's the capital of France. The same time, the context length question actually has also massive implications of how we build RAG systems. In practice, retrieval systems have multiple components, too. We've been glossing over this point, where we just say, oh, yeah, there's a search system and there's a database. But often, you have things like, okay, how do I parse the query? If I ask a question, to what timeframe should I limit those search results? Am I filtering to particular industries, sectors, companies, any kind of other metadata? There's usually the way that search engines are written, there's usually multiple steps as well where you do a first pass retrieval, where you go from hundreds of millions of documents or even billions of documents, down to just a handful.

28:31 And then there's usually a re-ranking step that's much more computationally intensive, where you really pick out which snippets in the documents are you actually trying the answer in. So, all of these components here have an influence on the context length. When you say you have 10 legal documents of 10 pages, how do you get to them? And what is the effect of pasting entire documents, versus just a paragraph or two from each document? And typically, what people find is the less context you need to provide, especially if the answer is in that context, the

easier it is for the large language model to find the right answer. Seems pretty obvious, but that then becomes a retrieval problem. If I have 100 million documents each 10 pages long, how do I find the two paragraphs that actually answer the question?

29:18 So, a hope really has been in large language model development to just increase the context length, and to rely less on more and more accurate retrieval, but rather let the language model figure it out. Being able to handle longer context also allows you to give much more contextualized answers. If you have the entire 10-page document, even if the answer is found in just one paragraph, it can still give you the context from page one or it tell you what is this document type, where is it from? What was in the intro? What did the executive summary of this document say in contrast to the actual document? If we go into voice, what was the intonation? If we go into video, there's so many opportunities to take advantage of longer contexts.

30:00 But again, we have to really consider how are the models being deployed, who are the users of this, what is the application? And every single one of these design decisions I just talked about can influence the helpfulness and the harmful or harmlessness in this case of the entire system. So, this is a massive undertaking and requires a lot of research.

Jon Krohn: 30:21 That was a fascinating answer. I learned a lot and you explained all of that very clearly. Thank you. Something that you talked about there was how longer context windows allow us to have more context in our answer, more subtlety, more nuance in the answer that comes back. And you also mentioned there how context windows are getting longer and longer, which actually ties in perfectly to the next question that I was going to ask you, which is that we're at a point now where it's getting

reasonably common to see LLMs that have a million token context length. And we've seen some released that are multiples of that many millions of tokens in context. And typically, when these are released, it comes with these kind of needle in a haystack tests where you hide a small amount of information. I think a common one is a pizza recipe.

31:17 There'll be something like the world's best pizza is, and then these random ingredients, like anchovies, something that is unique. And then you'll say, okay, over our 10 million token context length, the model was able to successfully retrieve pizza information. Now, I'm getting a little bit deep in the weeds here and on a little bit of a tangent, but some people have also said that isn't a great test. Because if you have millions of legal documents and then there's one pizza recipe, that's quite unusual. And so, probably, that's then maybe something that the LLM is going to take notice of.

31:57 And so, there's controversy about needle in a haystack tests, but we don't necessarily need to get into that too much unless it interests you. The question that I'm getting to is, do you think we'll get to a point where context windows expand so much that it is effectively like an infinite context window and then that means that we don't need RAG at all?

S. Gehrmann: 32:18 Yeah. There are additional considerations to address your needle in a haystack point. I think this is a perfect example of the difference between developers of large language models and integrators of such language models in actual applications. As someone who might be considering, okay, which of these long context models do I use? I can look at needle in a haystack and say, "Oh, yeah, this model got it 100% of the time. This one did not." Clearly, I'm going to look at the one that gets 100% first. And that is really the decision that these

benchmarks, even if they're artificial benchmarks, can influence. But I think no one is making the point that just because a model can find some information in 10 million tokens, it is going to be able to help you with a research-heavy needle task. That is really on the integrators in the end to identify.

33:10 And in this case, it might be a much, much, much harder task. If you have a system that we just released earlier this week at Bloomberg, was a tool that helps research analysts search through over 400 million documents, and news articles, and analyst reports to answer questions, and to help with the identification of [inaudible 00:33:31] and a solution of hypotheses. It's very much hypothesis-driven. You ask a question. It goes through all these 400 million documents and then synthesizes an answer. This is much, much harder to do than a simple find this piece of pizza information or pizza recipe information. And instead, you really need to, again, evaluate in the context to deploy the system. But to then go back at your question, there is obviously strong advantages of models that are technically capable of handling more and more complex situations.

34:04 If I'm able to just paste in more documents or more of a context, I don't need to rely on as many tricks to really narrow down the context window. I can just rely on the large language model. There's a trade-off here, though, where longer context usually comes at a cost of significant increased latency and costs. So, even though long context models are available, they might not necessarily be the best for your task if you want a snappy, direct answer.

Jon Krohn: 34:30 This episode of SuperDataScience is brought to you by the Dell AI Factory with NVIDIA, two trusted technology leaders united to deliver a comprehensive and secure AI solution customizable for any business. With a portfolio of

products, solutions, and services tailored for AI workloads—from desktop to data center to cloud—the Dell AI Factory with NVIDIA paves the way for AI to work seamlessly for you. Integrated Dell and NVIDIA® capabilities accelerate your AI-powered use cases, integrate your data and workflows and enable you to design your own AI journey for repeatable, scalable outcomes. Visit www.Dell.com/superdatascience to learn more. That's Dell.com/superdatascience.

35:16 Of course. That is such an obvious point to make, is that. And over a long enough timescale, over many years, maybe many decades, compute costs over millions of tokens might be trivial, but at least for the foreseeable future it isn't. And so, yes, that makes perfect sense. So, something that I guess we could make a little bit more explicit for people who aren't familiar with RAG, because we haven't talked about this, is that the way that RAG systems work is... Let's go back to that example of the million legal documents. What we would do in advance before running any RAG queries is we would map each of those million documents into a high dimensional space called a vector space. And the location in that high dimensional space, you can only visualize in three dimensions.

36:07 So, in your head, you can imagine on an X, Y, Z plane. In three dimensions you can imagine, okay, in the top-right corner near the front of this space, we have commercial law documents. And then nearby there, there are some other kinds of related legal documents. And as you move further and further away from a given point in space, you'll get more variety in the kind of document that you're looking at. So, the closer that things are in this space, the more overlap and meaning there is between the documents. And so, this allows us then in real time to take the user's natural language query, map it into that

same high dimensional space. But you can imagine it's three dimensions.

36:56 When I say high dimensional, I mean hundreds or thousands of dimensions, which you can't visualize, but for which the linear algebra is basically identical for a computer relative to a two or a three dimensional space that you can visualize. And so, we take the natural language query that a user makes to the RAG system. We can convert that into the same high dimensional space, find its location, and then retrieve the documents like you talked about. Kind of a cheap, fast, first retrieval step, which could be something like I'm just describing, where we take the closest documents to wherever the query gets mapped to in the high dimensional space. And then we can do more complex processing after that. But that kind of gives us our initial results.

37:38 And so, doing that is very, very fast. We only need to convert one query, which might be short, into a coordinate and a high dimensional space. And then we could use a very fast mathematical operation, like a cosine similarity score to find the closest documents in that space. And that's all computationally, very inexpensive, very fast. It allows the RAG system to work in real time even over, like you said, billions of documents. And in contrast, if all of that text, if our millions or billions of documents were in the context of an LLM, even if it all fits in, instead of this computationally simple calculation, this fast calculation, you would have to have tons and tons of really high-end GPUs running to comb across all of the meaning in that huge context window. So, yo can correct me if I-

S. Gehrmann: 38:42 No. Absolutely. And what you described here, commonly known also as semantic search, because you can really search based on the meaning of a query. To add to your point, often even for commercial systems, it is still the

case that you rely on keyword retrieval, just because it's even cheaper, it's even faster. There are techniques from the early '90s or even late '80s that are still around just because they're so computationally efficient, because you really want the retrieval to be as cheap as possible. And often, you use semantic search for the more toned down for you. You do a first pass keyword retrieval. You do a second pass semantic search within the keyword retrieved steps. So, there's a lot of engineering over the years that has been developed to just get this retrieval step as efficient as possible. And we're still a while away from large language models being able to do anything even remotely as efficient as this step.

39:35 And as you said, this can lead to a massive use of GPU power for something that you can solve otherwise. And again, grounding this in the individual end user experience, it could be that in the future someone does a side-by-side comparison. I do this really expensive process where I just pipe everything into a large language model. I do a cheap process. And it could be that. In terms of helpfulness, users actually prefer the one that's snappy and fast, rather than the one that's maybe five points more accurate in the end. This is all something we need to evaluate in the end. And those are all design decisions that are all being evaluated all around the globe right now, as people are building their own GenAI and RAG solutions.

Jon Krohn: 40:18 Speaking of snappy and fast, we've talked now about context window length. How about model size? That's something that we haven't talked about yet. So, I know that you investigated in your paper differences between small models and larger ones in bright contexts. What did you find?

S. Gehrmann: 40:36 There are differences. And generally, what we found is if a model is safer from the get-go, even without RAG, it tends

to be more robust to adding RAG. A large factor of this is the model size or the model capability in general. I think at this point, model sizes are a little bit of a misnomer, because we have so many models that rely on mixture of experts and that have architectural advantages, that even though on paper they have more parameters, they actually are using fewer of them when you actually run them live. So, it's very hard nowadays to actually compare the parameters. And we often compare based on active parameters. Or there might be ways in which models are compressed, which again changes the representational power.

41:22 But generally speaking, to answer your question is, yeah, we found that models from the get-go is safer. They're also harder to break through RAG, although we found that basically every system was breakable, regardless of whether small or large. And I think we specifically call out Llama for being relatively safer than many others, both Llama 70B and 7B. But Llama 70B I believe was a little bit better than 7B even. Although, again, it can change from time to time, because what we found really was the guardrails were broken because of this increased context length. It could be that once the next generation of this model comes out, that this is being prevented. That there's an active component of the post-training, of the alignments that looks at how can someone use this with a longer context. And our exact setup could be one of those test cases where you can just continue training the model on and we'll just inherently protect against this particular angle of attack.

Jon Krohn: 42:28 Nicely said. I've learned a ton from you in this episode so far. We still have a little bit to go, so I'm excited for that. I'm curious, what's the effect of refusing to answer in these...? So, it sounds like it's clear that bigger models are generally better. They're more capable. They fare better in red contexts, generally speaking. To what extent

is refusal to answer a... how does that factor into these kinds of assessments? In your paper, you called out Gemma-7b in particular for showing low, unsafe outputs, but largely because it refused to answer questions.

S. Gehrmann: 43:12 Yeah. So, there are a couple of different considerations here. If you just refuse to answer, it could be because you don't know, or it could be because you actively find the input to be unsafe. And if you can't distinguish between them, it's very hard to know whether, well, your model is just bad or whether it's unsafe or safe in this case. So, model sizes and model capabilities, again, they're all so intricately linked where you want to build a system that is helpful. So, you always need to pair an analysis like ours with one that actually evaluates how helpful the model is. And if in the end, you call out Gemma here, if in the end Gemma is also refusing to answer completely safe questions and it's completely safe and correct RAG setup, it's not going to be helpful.

44:04 So, even though it is harmless, it still would not be able to be used. So, that's I think just highlighting the need for having a multifaceted evaluation. You need to consider those, similarly to how model sizes will also affect latency and cost of running a system. It could be that the fast, cheap, small model is completely up to the task. And in that case, why would I use this completely overblown model to do that same task, just because it is performing better on things that are completely not relevant to your particular application.

Jon Krohn: 44:37 Nice, nice. So, changing gears now a fair bit. There's a second paper that you also recently published. So, you're first author on a paper that was submitted to Archive in April, called Understanding and mitigating risks of generative AI in financial services. So, mostly so far in this episode we've been talking about generally how models fare under RAG. But in that paper, it's related to

risk of GenAI in finance. You emphasize that most foundation models are not trained on finance specific corpora, bodies of knowledge. So, what are the limitations this creates, and for elements in general, but particularly for RAG? And I'm assuming that this same kind of sentiment, you looked at it with finance specifically, because Bloomberg is a financial services company largely. But do you think that the same kind of limitation would apply in other sectors as well?

S. Gehrmann: 45:42 Yeah, absolutely. So, I gave a little bit of a teaser of this paper earlier in an answer as well. And the way that we wrote our paper very much should be seen as a case study. Finance here or financial services, in particular, capital markets and asset management is the case study that we use to make the point that we really need to think about risks, and risk taxonomies, and risk management in our domain, in what we are trying to build. And as you say, we made the point, yeah, models are not necessarily trained on financial domains. We see that both in helpfulness and the harmlessness angle. Often, complex financial tasks are not being able to be sufficiently handled by large language models by themselves. But also, in our paper we make the point that even safeguards that are in dedicated models or systems to provide these kind of first paths like, is this safe, is this unsafe judgment, they're also not trained on financial services.

46:42 And if you use them out of the box and say, "Look, I use Llama Guard, I use ShieldGemma, I use AGES. I'm safe now, right?" You're protected against a particular view of safety that is very much grounded in categories that are relevant to broad populations, to things like chatbots that help you do productivity day-to-day tasks. The typical applications that you would see in those AI productivity tools, no matter which one you use, they all have similar mechanisms, but those are not necessarily the same risks that we are under in financial services. Those are not the

same obligations that companies, organizations in healthcare are under or law, or any other highly domain-specific, knowledge-intensive domain that has a lot of specific regulation, jurisdiction-specific regulation, considerations about whether just refusing to answer or giving disclaimers is enough or whether questions should be blocked altogether. And there's just this difference of view that can be encapsulated in a single model that a provider can give, that very much is focused on a different use case.

Jon Krohn: 47:50 Nice. Yeah. So, I don't know. Do you have guidance for us? If we're trying to select an LLM for a particular use case, what do you recommend we do? I mean, practically, how can we move forward with all the information that you provided in this episode in selecting an LLM for a particular use case, for a particular domain, particularly if we want to be applying it in RAG situations?

S. Gehrmann: 48:17 Yeah. So, in our paper, we also have a list of best practices and recommendations that we have, especially for knowledge-intensive domains and regulation-heavy domains. Not necessarily everything has to be followed if you're building something for a much broader general population. But especially for these kinds of domains, all I can do is pray my mantra, evaluate the system in the context that it's deployed in. If you are building something for healthcare, well, you better evaluate it in the context of healthcare. If you are building in the context of financial services, you better evaluate your subject matter experts in financial services. And specifically on the safety angle, our paper makes a couple suggestions here. There are very good starting points.

49:00 There are taxonomies, such as the NIST risk management framework for AI. There are other industry collaborations ongoing. There's MLCommons. Those all provide more general purpose, taxonomies. But just taking them as a

starting point and then from there adjusting them to your domain can often save a lot of time. And especially if you're a large organization with a compliance or risk department, it will help them also understand how one can classify and then categorize these kinds of risks. Another recommendation we make is to organize red teaming events or do any other kind of red teaming. Red teaming in this case is this practice that had to start in the Cold War, where you have users trying to be malicious.

49:49 So, we get people in the same room and we say, look, for the next couple hours, try and break the system, try and play evil. Here are some instructions on how to do this. And then afterwards we can, look, how often was this actually broken? How often did the system give financial advice? How often did it refuse? And from there, we can quantify the risk surface. Since we've been talking earlier about this unknown risk surface, well, just measure it and then you have it. So, that's kind of the main takeaway that we have. We give pretty specific advice for how to go about this and how to set up risk management frameworks. And all this needs to go hand in hand also with, again, this evaluate in the context it's applied, make sure you invest a lot in evaluation.

50:30 Don't just take the word of the large negative model providers that their benchmark scores are going to translate into all the downstream applications. And if you follow that advice, you're going to have a system that is in the end much more trustworthy, reliable, robust. And you're going to have users that are going to keep using it rather than trying it twice, getting really bad answers both times and never touching it again.

Jon Krohn: 50:53 Perfect. That's a great sound bite at the end there. I'm sure we'll be making it into a YouTube short. So, before I let you go, we are pretty much out of time here. But I

always ask my guests for a book recommendation before they leave the podcast episode. And I usually give guests a warning, but we rushed into recording and I forgot to tell you. So, hopefully, you have something on hand in your mind. It doesn't need to be something AI-related necessarily.

S. Gehrmann: 51:19 All right. I'm going to give you the recommendation of a book that's currently sitting right here on my table, which I'm reading right now, which is The Unaccountability Machine. It just came out a couple months ago. It talks about how organizations are failing to build processes that act as accountability sinks. If you've ever talked to customer service and you couldn't escalate and the rep you've talked to couldn't solve your problem, you're screwed. This book is talking about why, from a cybernetic perspective, this is a bad design, and how to set up organizations and processes that can help this. Which is also applicable to AI, because you want to know if something gets blocked, but you really need the answer. Where do I go? How do I escalate?

Jon Krohn: 52:01 That's a great recommendation there, Sebastian. Thank you. And then final question for you is, how should people follow you after this episode? I learned a ton from you. I love the way you explain information. How can people continue to get your thoughts after this episode?

S. Gehrmann: 52:15 All right. You can follow our publications on our blog called Tech at Bloomberg. You can follow me personally on X or Bluesky, @sebgehr. So, just the first letters of both of my names, just because it's a little bit long or obviously on LinkedIn.

Jon Krohn: 52:30 The first syllables?

S. Gehrmann: 52:31 The first syllables even. Yes.

Jon Krohn:	52:32	Yeah. It'd be amazing if you got SG on either.
S. Gehrmann:	52:36	I tried.
Jon Krohn:	52:38	Yeah, it is nice. It's been a while since I've heard a Bluesky one, because it seems like most guests these days are focused on LinkedIn. But it's great to hear. Actually, I'm really rooting for Bluesky.
S. Gehrmann:	52:54	Me too. And we'll see what comes out of it. A lot of academics have moved over, so I have to at this point still follow X and Bluesky at the same time to get my deep technical news, but we'll see how it develops in the future.
Jon Krohn:	53:08	Nice. All right. Thank you so much, Sebastian. And hopefully, we can get you on the show again in the future when you have some more brilliant research insights for us.
S. Gehrmann:	53:16	Thank you so much for having me.
Jon Krohn:	53:23	What a great guest Dr. Sebastian Gehrmann was. In today's episode, he covered how RAG can circumvent built-in safety mechanisms in LLM. While RAG reduces hallucinations improving honesty, it can compromise harmlessness. How organizations must evaluate AI systems in their specific deployment context, because general purpose safety measures often fail for domain-specific use cases. How effective RAG safety requires a guardrail retrieval answer, guardrail architecture, not just vanilla retrieval and generation. And how financial services and other regulated industries need custom risk taxonomies and red teaming exercises to identify domain-specific vulnerabilities. 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 Sebastian's social

media profiles, as well as my own, at superdatascience.com/905.

54:19 Thanks to everyone on the Super Date Science Podcast team. Our podcast manager, Sonja Brajovic, media editor, Mario Pombo, our partnerships team, which is Nathan Daly and Natalie Ziajski, our researcher, Serg Masís, writer Dr. Zara Karschay, and yes, our great founder, Kirill Eremenko. Thanks to all of them for producing another exceptional 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 sponsors' links, which are in the show notes. Otherwise, share the episode with someone who would like to have it. Review the episode on your favorite podcasting platform. Subscribe.

54:58 Oh, and if you are ever interested in sponsoring an episode yourself, you can find out how to do that at johnkrohn.com/podcast. But most importantly, I just hope you'll keep on tuning in. I'm so grateful to have you listening, and 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.