1:07

Alexey

**This week, we'll talk about ways to put LLMs (large language models) into production. We have a special guest today, Meryam. Meryam is a “recovering” physicist (very interesting) and the co-founder of TitanML. Titan ML is an LLP development platform that focuses on the deployability of LLM and allows businesses to build smaller and cheaper deployments of language models. Welcome to the show!**

1:34

Meryam

Lovely to be here. Thanks for having me.

1:37

Alexey

**Yeah, our pleasure. The questions for today's interview were prepared by Johanna Bayer. Thanks, Johanna, as always, for your help.**

# Meryam's background

1:45

Alexey

**Before we go into our main topic of LLMs, let us start with your background. Can you tell us about your career journey so far?**

1:53

Meryam

Sure. As you said, I started my career, actually, in physics – specifically theoretical physics and philosophy. That's what I studied at Oxford. When I left, I became an investment banker, which I really, really enjoyed.

2:11

Alexey

**That's quite a change. [chuckles]**

2:13

Meryam

Well, mainly because I had a fantastic boss. But I left investment banking to join tech and tech startups. And the reason I did that is because I realized what I actually really, really love is building things – building products and systems that people (and users specifically) really, really like. That's why I've been in the tech ecosystem ever since. We started TitanML a couple years ago now – before ChatGPT, so we have the clout to be able to say that. We originally started TitanML as a research project (coming out of my cofounder's research at UCL) thinking about the deployability of computer vision models. Then we pivoted, over a year ago, to focus on the deployability of large language models and that's what we've been working on ever since.

3:07

Alexey

**That's quite an interesting career – working in physics and philosophy. When you mentioned that, I was like, “Hmm... How does that even work? What's the intersection between these two?”**

3:21

Meryam

Yeah. I mean, physics and philosophy are super interesting. The way that I say that they're related, is that they're actually trying to solve the same thing. They're trying to solve [questions] like, “How does the universe work and how should we understand it?” But they just go about it in really different ways, so there is actually a really nice intersection.

3:38

Alexey

**I never thought about this from that angle. Interesting. But then you worked in investment banking, which is like a completely different area. I don't know, maybe your background in theoretical physics did help. Did it, actually? Or was it like a completely new universe?**

3:57

Meryam

Other than the fact that I'm really good at maths, not really. [chuckles] So no, I learned everything completely from scratch when I became an investment banker. I think one of the streams that's followed me throughout my entire life is that I just like doing things that are pretty difficult and I learn pretty quickly. Moving from theoretical physics gave me a great thinking base, a great mathematical base, and that meant that when I became an investment banker, I was able to learn pretty quickly. And I had fantastic people that I was working with and a fantastic boss. I just liked the fact that investment banking was “difficult,” or, you know... that it was something I'd never done before, so it was really exciting. [inaudible]

# The constant evolution of startups

4:41

Alexey

**So what's the most difficult thing – theoretical physics, investment banking, or being a startup co-founder?**

4:49

Meryam

That's a great question. I think they're just different levels of sleep-deprived. [chuckles] I think being a co-founder is probably the most difficult thing I've done. Not necessarily because any individual thing you do is difficult, but you're just spinning so many different plates all the time, and you have to constantly be thinking about everything. So that's probably the most difficult thing that we've been doing so far. So I don't know what my next one will be. [chuckles]

5:21

Alexey

**And the main difficulty is this variety of things – you have to do pretty much everything. What's your official title? Are you a CEO, CTO, or...? Do you have a title?**

5:33

Meryam

We don't really... We don't really do titles within a role. We have three co-founders and we all look after different parts of the business: Fergus looks after product development, Jamie looks after product strategy and also the research angle, and then I look after operations and commercial and fundraising. So that's the kind of way that we dispurse ourselves.

5:57

Alexey

**But you don't officially have a title of Chief Fundraising Officer?**

6:02

Meryam

Well, you know... We always get asked this. It doesn't really make sense to have these really grandiose titles when you're still a startup. I think the thing that's important is that you just have a really clear division of labor. We'll get to the point where we'll have really official titles – at some point.

6:21

Alexey

**In half a year, you'll probably be doing a different thing, right?**

6:26

Meryam

Yeah, exactly. Because things always evolve and the challenges are always really different. For example, we're doing way more on the commercial side than we were doing a year and a half ago and we're doing less on the fundraising side now. It's ever-changing.

# How Meryam became interested in LLMs

6:42

Alexey

**I know we're kind of late to the party in terms of speaking about the LLM because, as I mentioned at the beginning, this is actually our first event ever about LLMs (where we explicitly talk about them). LLMs are large language models, in case you live under a rock and have not heard about them. [chuckles] Yeah, we're quite late to that, but finally, we're catching up and talking about them. You mentioned that your startup, your company, TitanML, pivoted from CV (computer vision) to NLP – to large language models. So apparently, you're quite interested in them. I'm wondering, how did it actually start for you? How did you become interested in LLMs, even before ChatGPT?**

7:30

Meryam

Yeah. It was actually driven by our customers. We were interested in deep learning and were interested in all of the cool things it could do from the computer vision point of view, from a large language model point of view, and from various others. And we were interested in it from the technology level. But we started getting interested in large language models from an application point of view and from what they could really, really do for society. When our customers kept on saying, “Okay, computer vision is kind of cool. But actually, the thing that's going to change our business is language modeling.” That's when we really started getting interested in that particular space.

Part of the reason they were telling us, “This is way more important,” is because there's way more text in the world than there are images – or at least from an enterprise setting and that's when we really switched our focus. Then personally, I got interested in them, I think, when I was playing with GPT-3 a couple years ago. It was super impressive to see the rate of improvement that we got from GPT-2 to GPT-3. From a personal point of view that really told us that something special is happening in the language modeling space.

8:53

Alexey

**To be honest, for me, these demos – when I saw GPT-3 appearing – they were kind of “Okay, cool,” but it wasn't like, “Wow, this is going to change everything.” For me, it was like, “Okay, whatever. It's cool that these things exist.” But when ChatGPT appeared, it was mind blowing for me. [Meryam agrees] The tech is still the same, right?**

9:17

Meryam

Yeah! And it was for us as well. We were already involved in this space, but there's something, you know... when we moved from GPT-3 to GPT-3.5 and you could have a conversation with it and it kind of felt like there was another really, really smart person behind the screen and it wouldn't say a lot of really, really stupid stuff. It felt like that free-flowing conversation of just a person that knows everything. It was huge. It was so, so, so amazing. Because before, with GPT-3, you had to do a lot of really clever prompting to actually get it to do the thing you wanted it to do.

So when we moved from “you have to do really clever prompting” to “you can just have a conversation with it,” that's a huge mindset shift in how accessible and relatable these systems are. Even us, in this space, when we saw ChatGPT and we were like, “This is going to be huge.” We saw it on the first day it came out, and we were like, “This is gonna be really huge.” And then it was. That was really exciting.

# What is an LLM (generative vs non-generative models)?

10:19

Alexey

**So what is an LLM? What is this thing, actually?**

10:24

Meryam

Yeah. LLMs are large language models. I would kind of distinguish large language models into two things – quite often we conflate these ideas. Large language models, as we typically talk about them, are generative models. What these are, are models that are able to generate human-sounding or convincing text. And then there are also language models, which are understanding models. So the way that I think about these, is that these are models that are really, really good at understanding text and understanding language. We have these two different kinds of language models.

The state of the art for both of them is held by transformer architecture, although I'm not going to get into the nitty gritty because you can go and read the transformer papers themselves. But that was really the breakthrough in creating the state of the art with both of these. So, when we think about language models, I tend to differentiate between generative and then non-generative models.

11:26

Alexey

**And ChatGPT is a generative model – and GPT-3.5? [Meryam agrees]**

11:31

Meryam

Yeah, they're both generative. Then you might have [also] heard of models like BERT or RoBERTa or ELECTRA – those are non-generative models that are very, very good at [things] like classification tasks and natural language understanding tasks.

11:44

Alexey

**So if I need to classify the intent of a search query – for example whether the customer wants to buy something or just conduct research – then I would go with BERT, right? I would not necessarily go with a generative model because it would be overkill. [Meryam agrees] But I would still be able to use an LLM, saying, “Hey, I have this query. Here are examples of other queries. Based on these examples, what do you think is the intent of this query?” Right?**

12:14

Meryam

Yeah. Those really massive models, like GPT-3 or 4 can also do those classification-based tasks. If you do a Few-Shot Learning, you can typically have very good results. The reason why for most deployments we recommend not doing that is because you have these massive, massive, massive models doing these tasks that can be done by models that are in the millions of parameters rather than the billions of parameters.

12:41

Alexey

**So, BERT, which is a non-generative model, is it a large language model? Is it even a language model?**

12:51

Meryam

It's definitely a language model. It used to be a large language model. We used to think of BERT and RoBERTa and ELECTRA, which are hundreds of millions of parameters, as being large language models. A lot of businesses still struggled to deploy them at good inference for a reasonable cost. But the GPT-3s of the world have kind of blown that out of the water and redefined what it means to be a large language model. So the scales have just completely changed. But BERTs are still relatively big when we compare them to typical machine learning models.

13:28

Alexey

**A usual language model can still be a generative model – maybe it will not have an output that is as good as GPT 3.5 or 4, but it can still generate some text, right?**

13:45

Meryam

Yeah. There's a whole range and ecosystem of language models and they're good at different things. For example, there's the Google FLAN-T5 range, which is able to generate text. But what that's particularly good at is translation and summarization. There are other models like LLaMA, which is an open source language model, which is much much smaller than what they suspect GPT-3 is. It is able to generate text and have a conversation [as well]. It's probably not as good as GPT-3 or 4, but it's still a language model that is able to both understand and generate and converse.

# Why LLMs are important

14:27

Alexey

**So the main advantage of LLMs is that they are better at what they do – they're better at generating text. Right? So why do we even care about them? Why do you focus on deploying these LLMs?**

14:42

Meryam

Why are these important?

14:44

Alexey

**Yeah.**

14:45

Meryam

Well... they are incredibly powerful when it comes to being able to work within our unstructured language paradigm. Most of the things that we work with as humans in everyday work is unstructured text, typically. We write emails, we write documents, we write code – all of this incredibly unstructured and actually very difficult for machines to understand. Now language models are that paradigm shift of being able to create a system that is actually able to understand this unstructured data to a level that appears human-like, or sometimes even better.

So that's why it's a huge paradigm shift because for the first time, it's really able to work with these documents in a way that humans generate these documents – in an unstructured format – and then also generate documents in a similar format. That's why I think it's a huge paradigm shift, because it's finally able to work within the paradigms of how we normally work. Because humans don't normally work in databases, unfortunately.

15:56

Alexey

**That would be convenient ,right?**

15:57

Meryam

I mean, it would be convenient. But you know – we don't need to anymore. [chuckles]

16:02

Alexey

**[chuckles] Right. You mentioned, LLaMA, which is an open source LLM. I know that with Open AI – they have all these models, but they are closed, even though the name “Open AI” kind of suggests that it will be open. But they are not – they're closed.**

16:17

Meryam

Don't get me started. [chuckles]

# Open source models vs API models

16:19

Alexey

**Yeah, I think the reason behind not opening them is that it can do harm in [the wrong people] hands, right? If they open it, then it will do more harm than if they keep it closed? Interesting. But still, there are models like LLaMA – the one you mentioned. Maybe there are others that are open? Can you tell us more about them? What is the main difference between these open models and the models from Open AI?**

16:48

Meryam

Sure. There are a whole bunch of open source language models, and they're getting better and better month by month. I think only two days ago, Meta released LLaMA 2, which is a massively improved version from LLaMA 1, trained on 40% more data. There are other examples like Falcon, which is released by Abu Dhabi. There are models like MPT, which is released by MosaicML. These open source models are very quickly approaching the benchmarks and levels and performance that we see set by the Open AI models, or at least the publicly released Open AI models. There is still a bit of a performance gap, but the beauty about these open source language models is that they're much easier to in-house fine-tune train on your data and for your particular use case. So with a bit of fine-tuning and with a bit of training, you're actually able to match or sometimes even beat the performance benchmarks from Open AI.

These open source language models are getting better and better, literally, by the day or by the week. They allow the user, or the business, to have control over these language models and really deploy them in the way that they want to, for the use cases they want to, rather than relying on an Open AI API – which also, as we realized this week, they've been changing the performance of slightly. There's just a lot more control with these open source models, which is why I am personally most excited in the entire field by the speed of improvement of these open source models. Because as soon as we get these open source models that are on par with what we're able to do with Open AI, then that's a huge paradigm shift for how we think about how we build and deploy these models.

18:36

Alexey

**What actually happened this week? I saw a couple of posts on social media about the drop in performance or GPT-4, but I don't really know any details. What happened?**

18:46

Meryam

Yeah. I had actually noticed this a while ago. I use GPT-3 and GPT-4 a lot and I'd noticed that it felt like it wasn't as smart as it used to be. I put it down to two things: either they'd distilled it a whole bunch and just tried to save more compute and it made the model worse, or two, which I thought was potentially more likely, as I was just getting used to what these models were able to do when I wasn't thinking they were as magical as I thought they were six months ago. But a paper came out recently by a group who had been secretly benchmarking these models' performance over time and benchmarking it for various hard tasks. One of them was asking whether a very, very big number was prime. Earlier versions of GPT-4 were able to answer this with like 97% accuracy and the latest version of GPT-4 was able to answer this with much, much much lower accuracy – I think less than 10%.

What that showed was that Open AI had been changing the models without the users' knowledge, which is super problematic. Firstly, because Open AI had been denying it for ages and ages and ages and secondly, businesses had built products off of these Open AI API's and these models were changing behind the scenes without their knowledge. So their products were changing – likely, deteriorating – over time without their knowledge. And that's hugely problematic. I think this goes to the idea of, if you're building AI systems that are important to your business, you need to be able to have control and visibility of what the systems are actually doing and that's why I think it's important that we use open source models when it's business-critical. Also, [it's important to remember] that when you use models from Open AI, you don't own the models – you're being leased them. They can change them or they can remove them at any point.

A couple of weeks ago, we saw Open AI stopped running a whole bunch of their legacy models, which meant that anyone who's running applications on top of these legacy models had to change their architectures and switch over – potentially change prompts. So it was kind of just this realization by the community that building models on top of Open AI – although it's really easy to get going, it does come with these limitations of “You don't actually know what they're doing with these models behind the scenes.” So you have to be careful there.

21:28

Alexey

**From what I understood – as a user of Open AI, I have an endpoint (a URL that I use for sending requests). And in the request, I say, “Okay, this is the prompt. This is the model I want to use.” And I send the request. But what happens under the hood is – the URL still stays the same, but behind the scenes, the model actually changes. I keep sending the same requests, but yesterday, it was one model, and today, it might be a different model that is serving this request – that is answering. And I have no control [over this]. I have no idea that this happened. At any point of time, they can just go ahead and change [things] – maybe take a GPT-4 model and use a distilled version of this model. This may be better for them because it's faster, but for me, the performance of my application drops – and I have no control whatsoever over these things.**

22:21

Meryam

And I think it's completely right that they are trying to distill these models and I think it's right that they're trying to make them cheaper and faster. They are also constantly fine-tuning them over time to make the performance better, once they get more data. But the user needs to be able to know that and be able to say, “Oh, I do want to switch or I don't want to switch.” Because when it happens behind the scenes, you just have these users who are really confused that “Oh, my application is not working like it was, two weeks ago. What happened?” So that's why I think it's kind of problematic.

22:56

Alexey

**And in case of open source LLMs – say we take LLaMA and we host LLaMA with Titan (or whatever tool). Then the model stays the same all the time until we want to change it ourselves. Right?**

23:12

Meryam

Yeah. Because you can go in and check the weights. You can go in and check that the model is the exact same. You can only upgrade your model when you want to and in ways that are appropriate for you and your business. You can say, “Oh, my application is not fast enough. Let's look at distilling it (or pruning it or whatever).” But that's something that you control and you can version-control that. Whereas you can't do that with the API's.

# What TitanML does

23:37

Alexey

**I kind of assumed that what you do is host open source models – I don't know if my assumption was wrong or right. Maybe you can tell us what you actually do at Titan?**

23:48

Meryam

Of course. What we do is help businesses build and deploy language models for their particular use cases and their data and their tasks. We do host open source language models that have been fine-tuned. But actually, a lot of our customers want to host these models on-premise or in their own cloud so they can have data security and privacy. Thus, we have three offerings: we have the Titan Train offering which fine-tunes large language models for particular use cases in particular domains. We have Titan Optimized, which is for natural language understanding models (the BERT-style models). With the Optimized module, we can compress those models very, very significantly – by 10-100x. And then, our latest offering, which we actually released on Monday is called the Titan Takeoff server. This is a massively optimized inference server that businesses can use to deploy large language models on-premise, even on CPU, or on their data center with really fast inference and load costs. So those are our three offerings and we help businesses train and then deploy those large language models. Or we can deploy them ourselves, which we've done before as well.

25:10

Alexey

**So if I understood correctly, previously, you were mostly focusing on training and fine-tuning, but now you're also moving in the serving space for LLMs. Correct?**

25:20

Meryam

Yeah, the serving part is really difficult. We...

25:23

Alexey

**Well, they are large, right? [chuckles]**

25:26

Meryam

Yeah, they're really large. That's why we got into the space originally – we were working on the compression of these deep learning models. They're really difficult to serve, so there's a huge amount of value that can be added by just making that infrastructure easier. Also, everything we do is about making infrastructure easier – fine-tuning is also really, really hard, because as you said, they're large. So the serving infrastructure is an extension of us making that infrastructure easier as well. What it means is that people need less GPUs, which is nice.

# How fine-tuning a model helps in LLM use cases

26:03

Alexey

**Okay. I have a question here about use cases. I'm also interested in learning more about fine-tuning. I was wondering, maybe you can give us a few use cases and how exactly fine-tuning helps there? What does fine-tuning do for the cases? Why can't we just take an off-the-shelf model like LLaMA and just use it for whatever we want to solve? Why do we even need to fine-tune?**

26:30

Meryam

Sure. When you take a model off the shelf, what it has and what it's very, very good at, is general language knowledge and understanding. Your model will speak English or speak whatever language it was trained in, and it'll have reasonably good grammar, and it'll have reasonably good knowledge about the world. But what it won't have is any domain-specific knowledge, or any company-specific knowledge, or it won't talk in the tone that you want it to. The process of fine-tuning – I like to think about it as the process of specialization. You have this really general, big language model – how can we take that and specialize it for your use case in your business so we end up with a language model that's bespoke and works well for you? So it doesn't just have, “okay” accuracy and performance on everything – it has really, really, really good performance for the thing that you care about, and probably “okay” on everything else.

So that's what I think about when we think about fine-tuning language models. I can give you a couple examples. If we think first with the natural language understanding tasks, here, fine-tuning is necessary. If we want a model that classifies intent (or some kind of classifier), you need to fine-tune it so it understands what your labels mean or what kind of text is it expected to see? You can fine-tune a model to classify for whatever it is you need to classify for. [cross-talk]

28:08

Alexey

**Here, fine-tuning means, “Here is a set of examples, input text, output label – please adjust the weights in whatever way you want, so that we get the best performance possible on this training set.”**

28:25

Meryam

Exactly. And the key there is getting good data. If you can get good examples of the kind of thing you want it to classify and what “correct” looks like, then you'll get a really good fine-tuning result – it'll adjust the labels in a really good way. The way that you get that wrong is if you have bad data or not enough data, then it makes fine-tuning much harder.

28:50

Alexey

**Here, usually we talk about classification, right? Intent classification, sentiment classification – basically, we have some text as input, and the output is a set of labels.**

29:05

Meryam

Exactly. Then we can think about... If we think about fine-tuning for more generative language models, one really use case (a good use case that we see for fine-tuning generative models) is – it's very hard, for example, to get a generative model to speak in a particular tone of voice or in a particular style. That's very hard to get working with prompting. So...

29:27

Alexey

**What do you mean by “particular voice or style”? You mean more colloquial or more formal?**

29:33

Meryam

Exactly. Maybe you're training a customer service agent and you, as an institution or a corporate, have particular brand guidelines and particular ways of speaking. You can fine-tune a large language model off of examples of conversations that your customer service agents have had in the past, and it'll start mimicking that style of what's in that, and that kind of tone. That's a great example of fine-tuning as well.

30:00

Alexey

**ChatGPT sometimes speaks too formally and then I say “Hey, it's too formal. Can you make it less formal?” And then it speaks like a teenager from Reddit.**

30:08

Meryam

Exactly! It's so hard to get it to speak kind of... Normally? [chuckles] I find that as well. But fine-tuning is a really good way of getting around that. There are also other use cases you can do for fine-tuning. Let's say you work in a domain that has a lot of domain-specific knowledge – finance is an example of that. You can fine-tune an open source language model or, a language model, on information from your particular domain. What that will allow the model to do is start understanding phrases from your domain that might not have been in its training set. For example, let's say you work in the finance industry – you want your model to think that a “bear market” means something to do with the financial markets and not a market where you're selling bears. It's those kinds of things that you can do with fine-tuning that really allows you to get this domain adaptation.

31:10

Alexey

**Funny example – a market selling bears – because that's exactly what I would think. [laughs]**

31:14

Meryam

[chuckles] Exactly.

# Fine-tuning generative models

31:17

Alexey

**How does this process of fine-tuning look for generative models? Because for these BERT-style models, as we discussed, it's more like you have an input set of data with labels, “Here you go. Adjust your weights based on that.” But for generative models, how should the data look for us to fine-tune?**

31:38

Meryam

Yeah. This kind of changes depending on the end task that you want it to get it to do. But in cases that we've done, you can literally just have strings of documents, you can just have raw text that you can fine-tune on. So you don't need to have the super structured format of the before and after that you need for the understanding models – unless that is what you want it to do in the end.

32:05

Alexey

**You said it depends on the end task. I was wondering what kind of end tasks there actually are.**

32:12

Meryam

Summarization is a good one – where you have a long passage of text and you want to say, “Okay, just give me the gist of this five-page document.” Summarization is one. All of the natural language understanding tasks, you can also mimic with generative tasks. Another might be a tone change. I can ask my generative model, “Can you give me a before and after with the right tone and without the right tone?” And then there's all of the other ChatGPT-kind of tasks, like knowledge retrieval and just generating text, or all the kinds of things that you could do with these language models. I like to think that anything that you can kind of do with textual stock as a human, you can probably get working with a large language model.

# How generative models change the landscape of human work

33:04

Alexey

**A few days ago, I read an article about a copywriter who lost her job because of ChatGPT. At the beginning, instead of getting 10 articles per week (or whatever the actual number was), she would get eight, then six, then four, then two... And then nothing. Eventually, she lost her job. Then one of the jobs she found was training another language model to replace copywriters. So I think this is like what you said – end tasks can be generative, “I want to generate text in a particular way, so I need to hire a human who would produce text in the way I want. Then we can fire this human and just use the model.”**

33:58

Meryam

Wow. It's a really strange time. One of the industries that is expected to be most impacted by this is actually engineering, which typically has been very, very safe. Someone was saying to me the other week that engineers are actually replacing their own jobs because they're creating these large language model systems and integrating large language models into their work, which actually, in 10 years, will largely replace what they do. This is kind of interesting. There's huge ethical implications for what we're doing.

34:33

Alexey

**I tried using GPT-4 for creating a website in Django. I would say, “Okay, this is the website I want to create. These are the tasks that I want to perform with this website. Generate the code for me.” Then, step by step it would actually generate code for me that I would copy/paste to the terminal (to VS Code). Then sometimes it doesn't work – I try to run it, and it doesn't work. In this case I just copy the StackTrace, put it to ChatGPT and say, “Hey, this is the StackTrace I got.” and it replies with, “Oh, sorry, I forgot to mention that you should have done this thing too. So do it now.” Then I do this, and then it works. And I'm like, “Wow!” [chuckles]**

35:14

Meryam

It's probably... My guess is that took you way less time than it would have taken you to just do it yourself.

35:20

Alexey

**Yeah! I used Django like 10 years ago. For me, I would need to look up many, many things. But it will just tell me what I need to do. At the beginning it generated a requirements.txt file and I told it, “Hey, I want to use pipenv.” And then it replied, “Oh, here you go. This is the command that you need to run with pipenv.” It's like a person who is doing it for me, and I'm just telling this person what to do and then it's doing it.**

35:52

Meryam

Yeah, it's crazy. We use it and it definitely massively increases the productivity of our engineers. There are, obviously, risks. For example, there was a malware attack that I read about a couple months ago, where ChatGPT kept on making up packages that didn't exist. What some bad actors did is – they then created those packages and put malware in it.

36:23

Alexey

**Ingenious. [chuckles]**

36:24

Meryam

Exactly! They would maybe slightly misspell a commonly used package and then that became an attack. So there are obviously risks associated with doing this, but my guess is that a lot of these will be ironed out over the coming months and years.

36:39

Alexey

**Today, you still need an engineer to see if whatever the model outputs makes sense. Right? That's why, for now, engineering jobs are kind of safe? [Meryam agrees] Somebody still needs to do that. But what you're saying is that in 10 years, it might be a completely different situation.**

36:57

Meryam

Yeah. I mean, people have different approaches to this. Some people just think more code is gonna get written. They think, in a similar way, that we saw a productivity increase when we moved to factory production lines – more cars were built, and the same number of people were employed, but we just had more efficient processes. So some people think we're just gonna have more code written. I have a slightly more pessimistic view and I think that, if not in 10 years then in the longer term, we are just going to have less software engineers, which is sad.

37:38

Alexey

**What do you feel about this? Because you're practically building a startup to makу it happen. [chuckles]**

37:45

Meryam

Yeah, I... It's a great question. We don't actually create any code generation stuff, typically. We don't replace anyone's jobs directly. But we're working in an ecosystem that's going to be really, really disruptive for our society as a whole. Unfortunately, I don't think there's any way of stopping it. The cat's out of the bag, kind of. I think, as a society, we need to look at ourselves really closely and think, “How are we, as a society, going to organize ourselves when we have transitioned to a post-work society, where the majority of people don't have jobs and don't have productive mechanisms in society?” We need to reorganize ourselves, and what will that look like? That's what I kind of think. I know a lot of people have the contrary view, where we'll just have, “Everyone will be way more productive and it'll be great!” But I actually think this AI will get very, very, very good very, very quickly.

38:50

Alexey

**I think a few episodes (a few interviews) ago – I don't remember what we talked about, but then, at some point, we talked about a TV show called the Mandalorian. In this TV show, there was an episode where a bunch of droids who went rouge – they started behaving differently, strangely. And the whole society actually relied on the droids to do the work. People would just sit back and enjoy life, while droids do all the work. I think this is related to this post-work society that you mentioned. And then when something happens with these droids, they have to hire somebody to go and fix this problem because nobody knows how to work – they rely on droids.**

39:40

Meryam

But I also think there's a view that “Oh, it will be great when we don't have to work. That'd be wonderful and all of these things.” I think people will really struggle to find meaning in their lives and stay attached to some kind of, “I'm adding that value,” which I think is really important for the human psyche. We need to figure out a way that we as humans can attribute value to ourselves and generate value to society without having to have an official workplace. Because I don't know – I don't know that if everyone didn't have jobs tomorrow, whether people would be happier in the long term. I think actually...

40:26

Alexey

**Sounds like communism.**

40:28

Meryam

Exactly. That was like Karl Marx's ideal that we would have gotten up in the morning and painted in the afternoon. But people don't know how to work like that. As a society, we don't know how to generate meaning without having a purpose in society. And that will be a really interesting transition.

# How to adjust models over time

40:46

Alexey

**So we're getting a bit too philosophical and we actually have a few questions. One of them is related to the topic of fine-tuning. Here, the question is about creating a chatbot with LLMs. Let's say you have 1 million conversations and then you take these conversations, feed it to an LLM to fine-tune it – for it to learn, I guess the style and the tone of how chatbots should behave and answer. But then, the answers change with time. Maybe something that is the correct answer today will not necessarily be a correct answer tomorrow. How do we deal with this situation? We cannot just go and hire all our customer support agents, right?**

41:36

Meryam

For sure. Yeah, that's a really great question. There's a couple of things we can do there. The one that I actually prefer is using information retrieval. When you're doing customer service, typically, you'll have huge streams of documentation – of what the product looks like, what kind of responses are acceptable, blah, blah, blah.

42:01

Alexey

**A huge knowledge base, right?**

42:02

Meryam

Yeah, exactly, a huge knowledge base. And I think most companies have those kinds of knowledge bases, whether in Confluence, or Notion, etc. What you can do is embed all of that documentation and reinvent it every single time it changes in any substantial way and essentially look up the right part of the documentation as part of your answer and base your LLM's knowledge in the truth of what's in the documentation. That way, you can always stay grounded to whatever the base truth is in the docs. Or you could obviously refine-tune, but that's a much more expensive process than just re-embedding this big knowledge base and this documentation.

42:48

Alexey

**So as I understood, there are two ways. Let's say we have a knowledge base and we want our model to use this knowledge base in replies. In the same way, as let's say, I use ChatGPT to create a website in Django. For example, there is a less famous library – something else that is not widely known – but we have some internal documentation about this, so what we can do is just feed all this documentation to an LLM and fine-tune it. Then we can ask it, “Hey, I want to do this task. How do I do this?” Then if we just took all our knowledge base and put it into an LLM by fine-tuning it, it will reply and say, “This is what you need to do.” But the problem with this approach is that our knowledge base changes with time.**

**People go to Confluence, people edit the files there, and we cannot constantly retrain the model. We cannot constantly keep it updated because I guess it's also not very cheap. It's an expensive thing to do. The alternative to that would be, instead of retraining the whole model from scratch every time, we have a knowledge base, and we index this knowledge base, and then when there is a question like, “How do I do X with this library?” Instead of just using the weights of the model, we look the answer up somewhere in this knowledge base.**

44:17

Meryam

We'll look the answer up , typically, using our LLMs – probably using some kind of natural language understanding LLM. Another key reason why I prefer this particular method over the fine-tuning method, is that you get much less hallucination. Because your answers are grounded in the truth of a particular section in your documentation, you know that that's true rather than “It sounds like your documentation.” So fine-tuning is much better for things like style rather than substance of “it needs to say this particular thing.” And then you can have the model rephrase it in a way that sounds conversational.

44:57

Alexey

**When I think about documentation that I, as a software engineer or as a data scientist, create, I usually have something like a big Confluence page with all the things there. But oftentimes, when I have a question, the answer is in a specific paragraph of this document. It's not the entire document, but just one part of this document. Does this mean that I need to be careful about how exactly I index my knowledge base? I guess I still need to have some training data to say “For this question, this is where the answer is”. Correct?**

45:32

Meryam

Yeah. It's a good question. The open source models aren't as good at this reasoning as those that are like GPT-4. It's not as good at pulling from five different sources and then stitching them all together. However, there are cool things you can do when you do that information retrieval. Instead of it just returning the top one answer, you can get it to return the top five answers and get the LLM to search for the answer based on five different sections, rather than one different section. What that means is that you might get more variety in the answer, and it might be able to pull... Let's say, if your answer can be found at three different places – it can pull those three different sources together.

46:22

Alexey

**And how does this work in practice? I guess we need to put the entire document in the prompt, right? We need to somehow find a way that, “For this question, these are the relevant documents. Let's put them all in the prompt and let the model figure out the answer.” Right? How exactly does it look in practice?**

46:42

Meryam

That's exactly – that's one way that you could do it. You can just, essentially, inject relevant parts into a prompt. You can say something like, “Answer this query. You may use the information from these documents or these sections.” And then put the sections below. That's definitely one way that you can do it. Another way that you can do it, if you really want to ground it in truth for a very, very sensitive task – you can say, “I just want you to pull up the relevant section and then I just want you to summarize it (or change the tone or do something like that).” And then you have one model that does the information retrieval and then you have another model that just does a summarization, and is really grounded in truth. So that's another thing that you can do. You can ground it in truth through information retrieval, and then put it through a summarizer.

# Vector databases and LLMs

47:47

Alexey

**The next question we have is about a vector database. Before we talk about this, maybe you can tell us what these vector databases are and how they are relevant to LLMs.**

48:01

Meryam

Yeah, so these vector databases are very, very similar to the information retrieval systems that I was talking about. It's essentially a way of indexing this unstructured data and being able to search it through natural language. You could use the vector database as the first step in order to find the relevant parts of a document that you need to answer your query. That's, essentially, the first part of that system that I was just talking about – the information retrieval part.

48:33

Alexey

**Let's say in practice, we have a Confluence with all the documentation. What we do is get each document from this Confluence and then somehow index it with a vector database. Then for each of the documents, we have a vector, which we put into the database. Then, when there is a query from the user, we somehow turn this query into a vector again, and then try to look up the most similar vectors from the database. Is this how it works?**

49:11

Meryam

Yeah. You're very good at explaining it – much better than I am explaining it.

49:15

Alexey

**LLMs can also “vectorize” a document, right? They can take a document and create embeddings from this document so we can put them in a vector database. Correct?**

49:30

Meryam

Yeah, exactly. Vector databases are hugely, hugely popular to do those kinds of information retrieval systems that I was talking about. People really love them, which is why they've exploded in the startup scene.

# How to choose an open source LLM or an API

49:44

Alexey

**For this task, do you know if we should go with an open source LLM or go with GPT-3.5 or 4? Are there any pros and cons?**

49:57

Meryam

Yeah. The way that I tend to think of whether you should use GPT 3.5 or 4, or an open source model, is – if you are still in the prototyping stage, if you're still at the stage where you want to figure out if a model or an LLM is right for your use case, then you should definitely use one of the API-based models like GPT-3.5 or 4. The reason for that is that you're just able to get to results really, really quickly. You're able to get to demos within a day or two, which is very, very impressive. But in the long term, businesses tend to want to use open source models to build their applications for a bunch of reasons. One is that you actually know that the model is not changing under the hood. Another reason is data privacy. Another is that it's much cheaper and faster – all of these really, really good things. So in the longer term, you can move over from using that Open AI-based model, once you've proven out the business case to build something a bit more scalable with open source models. Given that an application works in Open AI, you can almost certainly get it working to a very similar standard with a fine-tuned large language model.

51:14

Alexey

**So the benefits of going with an open source model eventually (after the prototyping stage) is that the models do not change, data privacy, it's cheaper, and you also mentioned faster. I wanted to ask – faster? Because for me, these Open AI models seem to be pretty fast. Can you be even faster when you host your own model?**

51:35

Meryam

I mean, they are really fast. They're really, really fast, because they're hosted on very expensive hardware. If you were to host your model on the same hardware, using good techniques – using something like the Titan Takeoff server or other fast inference servers, you can get very, very, very fast models on very powerful hardware. But what is also really nice is that most businesses don't deploy their models on the state of the art, very expensive hardware – they're deploying on smaller GPUs or even CPUs, because that's typically what they have available. These models will be much, much faster, than what you would be able to do if you're hosting an Open AI model on that kind of server. You can get it even faster if you use really powerful hardware, or you could just get comparable speeds for much, much lower prices.

52:31

Alexey

**But you have to put in some effort. For example, in the case of Open AI, you just take an off-the-shelf API and start using it. Of course, you pay, but you can move very fast. But then, at some point, you start thinking about costs, data privacy, and other things. This is the time when you invest time into adopting and adjusting to an open source model.**

52:57

Meryam

The way that I describe it is – if you've read Peter Thiel's Zero to One – once you're trying to get to product market fit, or MVP or whatever the “one” is in your case, definitely use whatever makes you move quickest. But in the long term, when you start thinking about scalability and data privacy and performance in the long term, then you can start transitioning over. It is a bit more effort than just using API's (because they're trivial), however, there are a lot of tools that make it much easier nowadays. Fine-tuning used to be a really laborious process but it's now much easier to do that.

# Measuring input data quality

53:34

Alexey

**We have a few interesting questions from Tara. The first question he's asking is, “How can you measure if the data you feed into an LLM is good enough?” Do you even think about these things or are you just saying, “This is the data I have. Let's just throw it all in and then it will figure everything out?”**

53:58

Meryam

It's a great question. How do you know if it's good? There's no hard measure of “This is exactly the quality of data that you need.” Obviously, the cleaner the better. I tend to measure these kinds of things based on the quality of the output. So if the quality of the output is good and exactly what you want, then clearly, what you put in was probably fine. When it comes to what you should put in the model, what we tend to find is that it's better to use a slightly smaller sample of what we would deem “gold standard”, really, really high quality data, preferably that someone has had labeled or at least checked the labels for. Then once you have that, you can use interesting methods, like dataset expansion or data augmentation, to get a large LLM to generate more examples from that gold standard data. That typically tends, in my experience anyway, to yield better results than just throwing in a bunch of rubbish data into there just for volume – have a smaller amount of quality and then expand it with an LLM.

55:12

Alexey

**Date expansion is a strategy, I guess, that is similar to how we use data augmentation for computer vision. We take a picture and rotate it slightly or crop it, ow whatever – we basically use existing data to generate more data. Dataset expansion is something similar, but for NLP, right?**

55:32

Meryam

Yeah, it's super similar. A very basic example is – if I have a dataset where one example is “the pig is pink,” I might get my LLM to say “the cat is black”. It just kind of switches words out, but it's semantically similar. Another way that you can do it, which is just much easier, is if you just put in a bunch of examples into another LLM and just say, “Generate more data points that look like this.” It actually tends to do pretty well. We recently did a benchmarking of this – it was called GPTeacher – where we compared the performance of a fine-tuned BERT model off of gold standard data that was all hand-labeled, versus an LLM-augmented version. We took like 10 examples from the gold standard one and just said, “create more like this,” and we got very, very good performance for that augmented data. It was obviously much quicker and cheaper than hand-labeling all of them.

56:39

Alexey

**You mentioned good performance. This is another question from Taras, “How do you actually benchmark in LLMs? How can you tell that performance is actually good?” You need to have an objective way of saying that. Of course, you can subjectively say, “Okay, the answer to this prompt is good,” or “the answer to this prompt is not good.” But it doesn't scale, right? When you retrain the model, you want to somehow automatically say that, “Okay, this is good – it's better, or it's worse than the previous version.”**

57:10

Meryam

It's a really good question and I don't think it's a solved question. For natural language understanding tasks, it's much easier because when it's classification, you can say, “it's this percent right,” or “it's this percent wrong.” That's much easier. For generation tasks, there are a lot of different ways you can be right and a lot of different ways that you can be wrong. That is hard to systematically measure. When we're doing projects, we actually measure it by hand. We will just play with a model for a given amount of time and say, “Does this kind of look sensible?” And that's the most reliable way that we found to really get a feel for quality.

There are interesting things you might have to do with automating it if you're refine-tuning – maybe coming up with example benchmark test datasets that you can then go in and hand-evaluate. I would always recommend hand-evaluating at least some of them. You might also be able to get an LLM to mark it for you, which could be an interesting way of doing it as well. But unfortunately, with the generative models, there's not an easy way to check performance other than to go by a human mind.

58:26

Alexey

**So right now, currently, with the existing methods, we still need to keep the human in the loop to assess the performance.**

58:35

Meryam

Yeah. Also partly because this technology is new and businesses aren't used to it. So anything that you can do to make it feel safer or make it feel like it's going to do sensible things, the better. Very often that just takes people to check, “Does this work? What about this edge case? What about this edge case?” All of that.

58:56

Alexey

**Maybe the job of copywriters might be not safe as of yet, but at least they can be these humans in the loop.**

59:07

Meryam

Maybe.

# Meryam's resource recommendations

59:08

Alexey

**To help eliminate them in the future. It's kind of an interesting dilemma. Anyway, we should be wrapping up. Maybe before we finish – is there any good resource, like a book or a course or a blog, that you can recommend for those who want to learn more about LLM and fine-tuning LLMs, and all the things we talked about today?**

59:29

Meryam

Yeah, there's a whole bunch of resources. I follow a lot of LinkedIn influencers in this space. If you follow me on LinkedIn, I typically retweet a lot of them or repost a lot of them, so you can see a lot of that really great content. There's also cool things like, Cohere has a great LLM University thing. The Hugging Face course is also very, very good. There's also a bunch of really good resources on YouTube. And that's where I would kind of get started. It's one of those fields that just has a lot of scattered very very good influences that you can just follow over time.

**60:10  
Alexey  
And this is so rapidly developing. Maybe by the time we release this interview (this episode) there will be a new course.**

60:17

Meryam

Exactly. Everything that I say will be completely out of date within a week, probably. [chuckles]

60:22

Alexey **Well let's hope that for at least half a year it will still be relevant. Okay. Thanks a lot for joining us today, and thanks, everyone, also for joining us today. With that, I think we'll finish.**

60:36

Meryam  
Thank you so much Alexey.

60:38

Alexey

**Yeah. Have a great rest of your week. Bye, everyone!**