

Aidan Sussman:

Hello and welcome to the Actuaries Institute podcasts. My name is Aidan and I'm a Fellow of the Actuaries Institute and am currently a manager at KPMG.

Today I'll be discussing LLMs applications and developments. I'm joined by Ean Chan, Minh Phan and William Zheng.

I'll just ask each of them to introduce themselves and say a little about themselves. Ean, you first.

Ean Chan:

Hi I'm Ean Chan. I'm a Senior Manager at EY in the Actuarial Services Team. I used to be in life insurance specifically but moved more broadly into a data analytics role and am now in the health and human services team.

Minh Phan:

Hi, my name is Minh I'm a data science actuary working at Finity predominantly in pricing and analytics. I started my career off heavily involved in pricing and valuations and have so a trended way to more obscure things that since relate to Cloud and interested in applications of LLMs.

Aidan Sussman:

Thanks and Will?

William Zheng:

Hello, my name is William I work in a data science position as a lead data scientist at George Western Food. I've come from an engineering background worked several years in medical and biomedical engineering then transitioning into the automotive as well as the finance industry in a data science capacity, so I'm interested in all things machine learning and AI.

Aidan Sussman:

Thanks, Will. So, what are we all doing on this call today? We've got actuaries and a traditional data scientist. What we're going to be doing is talking about large language models and Gen AI but focused on the applications and development. So that could mean how to code it yourself or like what kind of things you can do with it. So just for an easy question why don't we start of. How do we all use large language models in our jobs right now. So, I'll start again with you Ean.

Ean Chan:

Ah, for everything Aidan. I'll start basic. I think the answer which a lot of people will give is for coding - I think that's the number one use case of LLMs. Maybe we can have a discussion later or a debate about that. But I think a lot of people might agree.

But yeah, so for programming 100% gets used by myself to help develop things a lot. I do use it for I guess other stages of brainstorming, thinking.

Nowadays, I have started to use it as a first point before Google even sometimes. Once you get a sense of what it can tell the truth about and what it will lie about and know what kinds of questions work, I think a good starting point for a lot of things.

On the I guess a deeper development side of things yeah have been developing internal facing apps at EY for yeah people to use connected mainly through Microsoft Azure to GPT4.

Minh Phan:

Yeah, similar to Ean. It's like the number one practical use case of generative AI is just using ChatGPT for coding - that's the primary use case. I'd say that's 70% of my usage on ChatGPT.

And then, on top of that. It's also helpful for rewording, phrasing learning about everything. Like I use it also as my predominantly is my first point of contact for questions I have as well as perplexity when it needs to be kind of contemporary information. And it's also quite useful for brainstorming or just you know, understanding technical jargon for insurance terms, legal terms, contract terms, like reinsurance terms – hyperparametric and tuning in EGMs. It's pretty much you know, the primary application is just using it as a feed of knowledge and helping me code faster.

In terms of the programmatic side, I haven't really built applications but I've been using it more for augmenting modelling within pricing space and also to automate some otherwise tedious tasks.

Aidan Sussman:

So, you use it to augment pricing models. Can you tell me a bit more about that?

Minh Phan:

So there is a few applications I'm trying out either myself or with my colleagues. So one is, we're doing some work for liability pricing. And we know that worker-to-worker claims are a specific subset of claims that have quite a different development and a long-tail pricing pattern and so being able to kind of carve them out effectively is quite important and the data that we add is kind of not very well-coded.

One thing we've done is to try and reclassify it based on a very clear, so we provided a very detailed description of what it entails to be a worker-to-worker claim and so, I was able to classify things like you know, a simple string like injured back while jack-hammering or kind of third-party worker fell through scaffolding was able to recognise an injury of sustain because of a worker doing work onsite.

So, we kind of fed through a very multi-prompt like a very ah heavily scripted prompt around all the kinds of things that would classify as work-at-worker and yeah able to classify them and therefore you improve the pricing and the cost models for that particular peril.

Aidan Sussman:

So previously, before LLMs, someone might build a natural language processing machine learning classification model where they would provide a target of where they know there are those claims?

Aidan Sussman:

You're not even pre-training anything you're just providing it examples and it's making a judgment for you?

Minh Phan:

Yeah, that's right. It's quite amazing that it's able to do that because we've had a neural network that's trained to try to identify these exact type of claims with no training but just a series of like very comprehensive examples and a definition - it would say well to kind of classify it well for us and getting some good value out of that and the other. Obviously, the other thing that we tried historically was bag of words so using identify things like "third-party" or like "multiple parties" involved, and "onsite" and but you know you can only pick up like a handful of keywords.

Aidan Sussman:

Yeah, now that's really interesting to hear about a traditional a space using large language models. It's really cool.

Ean Chan:

So this is for the feature engineering side of it right? So that then gets fed into a model?

Minh Phan:

Yeah, well that this is this is more for classifying the peril so that's kind of one way to improve your models but from a feature engineering perspective, a different application, we've had was to try to for the in the context of commercial lines.

Again, describe occupations based on the things that drive risk so things like so if you've ever done modelling in commercial line space. You know that you get a series of ANZSIC which is 6000 occupations and you get a hierarchy and that's all you get.

But you know, in terms of trying to build predictive models. You want things that actually drive the risk, so you know does this occupation likely to involve heating or cooking or does it involve chemicals. Does it, you know and so being able to use LLMs to create these kind of binary indicators across the occupations, we're able to kind of generate new features that can yeah be fed into the models so that that's a feature engineering application that we've used.

Ean Chan:

Yeah, because I guess I am thinking it does sound like replacing the like embedding way of doing things sometimes like so instead of one of the features being the embedding of a longer description, we can feed it through an LLM first and that would give you the outputs into the next level.

Minh Phan:

Like the tricky thing but embeddings is that you don't really get an intuitive feel of what it represents.

So, you have this kind of mathematical representation that decomposes your texts into like a big, a big vector. But you ought to say oh like this like this occupation is risky because it involves heating and so therefore like I'm going to you know, apply a higher risk loading.

When you build your models like you're only as good as how you can explain it like no one cares about your Gini curves. No one cares about your AUC.

So, I'm always ready to sacrifice fit for explainability. But in this case, you get potentially better fit with explainability which is nice.

Aidan Sussman:

Now Will, as someone who's not an actuary, how do you use LLMs in your job?

William Zheng:

Yeah I think Ean and Minh have touched on many points that I agree with and indeed you can use it for coding assistance and then in addition to that learning about new topics and general topics especially. I treat it as like talking Wikipedia. So, you know, I think it a very powerful way to synthesise information that's out there in the world and there is a lot that I don't know and it's a great way to learn things in a time efficient way. So those two are the main productive boosters that I use myself.

In terms of the work that I use LLMs in a work setting, we've been using LLMs mainly for text summarisation as well as what Minh said, feature engineering. So, previously we had to rely on various NLPs techniques to extract or classify categories from the data that we consider freeform text and some of the text could be you know written in different- grammatical fashions and styles.

But what larger language models has been providing is a method that gave us a very flexible way to crawl through those data and ah quite effectively and generate the features we wanted, even doing kind of the traditional sentiment classification and categorisation of those texts.

So we found that LLM in general has been providing a super powerful tool for now for us to crawl through those data that we wouldn't have been able to do or without you know, otherwise we have to spend a lot of effort trying to create ah various kind of data transformation and NLPs -based scripts to crawl over so that's definitely a very surprising and very nice feature of a very powerful feature indeed.

Aidan Sussman:

So, just to summarise all around, it seems like whether you're working in the actual data science field or data science that LLMs are very good for both researching and coding but also for feature engineering, building models and analysing texts.

Next question I might shoot to Minh.

So, everybody knows ChatGPT. We've got new things like Sora coming out also by OpenAI where you've got some text to really awesome cinematic videos. What are some other tools out there that most people might not know about?

Minh Phan:

Yeah, so there's a few other ones that I tried using and sometimes do use so creating assistance like Copilot and aider.chat, some good ones I've use. Obviously I can't use some of this stuff at work because, potential leakage into the Cloud universe and who knows but these tools help you write code directly into your console so rather than having to copy and paste it into ChatGPT and then copy it back, you can basically type in it all so Copilot lets you type code and it will try and prefill it for you whereas aider.chat you can kind of set up your GitHub repository and then write comments inside your code and say like write this kind of function and write this and then hit go and it'll actually write it for you within your script. So, it can edit ah scripts for you which is quite nice.

Another the tool that I've like to use I mentioned before is Perplexity which is basically like retrieval augmented generation on the internet. So, you think about ChatGPT, it's trained on a corpus of data but it only knows information up to a certain point. So, if you can effectively see what it does is it does a Google search or effectively Google search it opens those pages and reads information and then it will then take your query and then read the information synthesize a response for you. So, really nice way to efficiently learn stuff that efficiently research things that are a little bit more contemporary.

So, things that are recent I use kind of Perplexity to kind of summarise so it might be like summarise for me the EU AI act and synthesize your point in dot points. So, ChatGPT won't know that because it's only been released recently.

Other thing that I like to do sometimes as well is Chat PDF. So, what you can do is upload your PDF and ask questions about it. But the benefit of using something like ChatPDF is that it provides you the citations and you can kind of click and find within that PDF what is answering the question.

Adobe has just released its own version of this. So, if you subscribe to Adobe Cloud you can do this well and there's various other ones on the internet - ChatGPT has its own version. But I find the interface of Chat PDF much nicer to use.

So, those are kind of 3 tools that I like to also use in my workflow.

Aidan Sussman:

So, Chat PDF - you're able to, instead of looking through perhaps a very long 200-page report, you can ask Chat PDF certain questions to where to find things or even what the report's trying to tell you?

Minh Phan:

Yeah, a good retro effective use case is like pricing promises.

I don't know if anyone's involved in pricing promise reviews but you can kind of say tell me any instance in this document where a pricing promise is made and that you will save your premium if you do X or if you change your excess you will save or you may save. Like I'm drawing through documents it can quickly isolate for you - the sections that you can kind of interrogate.

Aidan Sussman:

Have you created any applications on your own that use in your work or personal life that leverage large language models?

William Zheng:

Yeah, I sort of have created one. I wouldn't say the applications is published in public but somewhat embarrassingly, I'm using LLMs to cheer myself up. So, it is pretty much a LLM with a prompt add-on.

From the prompt I type in, it's setup to spin it in a positive way, so no matter what kind of sad messages I put in, is always going to give me a cheerful advice. Um, so you know that that's the kind of the LLM application that I made for find for myself. But I think it's ah it's useful in some ways.

Ean Chan:

Maybe, I'll follow with a similarly, not productivity focused one necessarily.

I'm in the midst of developing... Ah I guess you can call it like a LLM game type thing where you know those kinds of games where you're trying to um, what are they called like articulate or taboo where you're trying to describe something to someone else without using the word. So, the next level of that to me is I'm trying to describe something to you Minh, and I'm trying to describe it without a Chatbot being able to know what I'm saying.

So, I've used this for celebrities before and ChatGPT is very good... like we tried very hard to use just in jokes to describe who this celebrity was like just internet slang and just trying to avoid true facts about this celebrity until you know someone would guess it. But then when you click submit on the ChatGPT prompt it gets it like 9 out 10 times.

William Zheng

That's really impressive.

William Zheng:

I Look forward to playing with it. Is it going to be on the app store?

Ean Chan:

I think because I'm using GPT 4 at the moment, I think there's no way this is going to be a mass market thing because it's on my personal API key.

Minh Phan:

I don't know if you have time for one last application Aidan, but I've I'm using LLMs to summarise content about LLMs to learn it faster.

So, I wrote an application that scrapes the transcripts of these YouTube channels and media articles and then pre-summarises them and then renders them in HTML format to email myself every day and a little app that called knomad. So, that's how I say in the know with knomad.

Aidan Sussman:

I'm emailing you about this app.

Aidan Sussman:

So, that's interesting that you've all developed your own unique applications.

Maybe a question to that. Before open AI announced the use of agents in a marketplace and to have agents, a lot of different developers were working on similar LLM-based apps.

A lot of augmentation and a lot of different type of use cases got wiped out a lot of those startups or ideas or projects got made a bit redundant. OpenAI overnight brought out some new features, and I guess my question is if these big players in this the LLM industry, having such huge budgets and rapid development, is there much point of us trying to create apps off their products if they could just come out with a new product in the next second which makes your idea irrelevant?

Ean Chan:

So, I have opinions on this and I think first off it, it does depend on what your use case is and maybe I'll split it between like internal use vs. external use and trying to sell a product and then also split whether you know you're doing it for a company or for yourself.

So, I guess looking at the external facing side. I wouldn't try and compete with OpenAI and Microsoft on their products right.

So, like if all you're doing is adding one little layer on top to make it specifically do one type of thing, but the core sell is the model itself, I think that's a losing battle, right? I think they're always a model - half a model - ahead of you right? And similar to I guess early days of the app store with Apple where they took all of the best apps and it became a core part of the next iPhone by the next product.

So, I think that if all you're doing is just adding a little bit on to the top then yeah, probably don't compete against Microsoft, Google, OpenAI. If instead like you're developing a product whose draw isn't the fact that it's an LLM, but you know there's one LLM part of it that sprinkles the magic in that I think is valuable. You need to I guess make sure you have some kind of unfair advantage there. So, you're not actually directly competing against them.

In terms of internal use and so creating tools that you're going to use yourself that your company is going to use itself for its specific use case, I think go nuts.

Yeah, like I think that's the perfect use of developing LLM applications. Like no one knows your problems the way you do, no one knows exactly the way you're going to use your data. Use the LLM models to help do your own tasks and your company's tasks. And then add the extra capability in when the new models get released and I think that's the best way to go about it.

Aidan Sussman:

So maybe before anyone else jumps in, from what I'm hearing Ean, if you're trying to build a more general purpose external applications leveraging LLMs you could face a lot of trouble, but the more purpose built practical, maybe internal applications, go for your life...

Minh you look very keen to talk.

Minh Phan:

Ah I was just, furiously agreeing with Ean.

Like if you phrase it very well like if you if you try and build an application which is like just a thin layer of value add, then you're going to get wiped out so you see a lot of these startups that you know, "you can just summarise text or rewrite this in a different character form" it's such a thin layer of value add it's not worth pursuing.

When you say applications, even ah, something as simple as like iterating the API through like your claim lists like I don't know if that you call that application but it's technically by definition, it is. Like that's you get high value because you take their take OpenAI LLMs products and then use it and you get an output and which then plugs into like ah like the next part of your modelling process or the next part of your value chain pipeline.

Finding like the small use cases, finding ways to leverage what the guys have built in in terms of its like strong foundational structure for your particular use case is useful. I think the only thing that'd add on top of that as well, if you were to build an application that is heavily wrapped around like OpenAI but if you have you know a large corpus of domain data that basically no one else has access to then that's where you potentially can have the ability to compete.

So, I'm thinking like someone like Instagram has this trove of images that um, potentially. Based on the, I don't know what the terms conditions are for users, but like they have like behemoth of images that they can use and so the kind of model they can build on top of that is going to outweigh what OpenAI can source you know from an open source perspective and so.

But that's a very extreme example. But if you have a good corpus of data that these companies don't have then and go nuts as well.

Aidan Sussman:

So, if you if you have your own data or your own judgment that you can integrate into your application then you've got a value add.

Minh Phan:

And a not easily replicated value add.

Ean Chan:

Yeah, you need to build your moat and protect it fiercely.

Aidan Sussman:

Do you have similar thoughts or different opinions?

William Zheng:

Ah yeah I can agree with most of it I think that um you know is ah the big question is it worthwhile building your own or like do you see a cell close to sound you burn yourself right?

There is a big meme about OpenAI every time that OpenAI does. Like when they hold a dev workshop a thousand startup dies.

I think in general the mental kind of model I have is thinking of this you know previously people would think you know LLMs is like a mode of capability that that sort of gets unleashed, but it is a tool that most people have access to and the barrier to entry to it is actually quite low - so you can think of that as almost like when the mobile platform got introduced - and just being mobile, that doesn't determine business success but some businesses are enabled by mobile whereas if it's not mobile then the business may not exist so you know, it may be worthwhile thinking about in this framework where LLMs is not a make or break kind of thing but it might be a critical component of any business for a potential business idea or undertaking that you know people want to consider.

So, I think, you know if you have an idea that works already and it's based on a real-world use case and real-world demand and demand is there, and LLM adding LLM can you know, accelerate or boost 1000% of what you're doing, then that is a potentially a real advantage there.

And I will say also through this acceleration if you can gather more of your more of the data so through this if you can acquire more users acquire more adoption of whatever you're building through LLMs and then gather custom data that you can keep. This generates what we call a data flow/flywheel, and it's the real moat.

Aidan Sussman:

So gotcha. So, it really puts the onus on us as data scientists or actuaries to think creatively about the data we have or potential applications for LLMs because it does seem like they're definitely out there. So, I guess we all got to start thinking a bit harder?

Ean Chan:

Yeah, your million dollar startup idea can't be "use an LLM" right? You got to have some other core purpose and then also LLM.

Aidan Sussman:

Yeah, it's funny because even when you think about the Chat PDF like when you think about it at the core. That's a very simple, simple use case but like how many users would that have.

Minh Phan:

I feel like their lifespan is running thin. I still use them just because it's the interfaces right now better than chat GPTs but give it like chat GPT haven't so OpenAI's haven't invested time in.

I'm not I'm not wedded to any application. It's just right now. It's the best one and it's free up to a certain point.

Aidan Sussman:

So, one question I'll ask maybe I'll go to Ean for this one so we're talking a lot about developing our own applications.

Using LLMs. So, say if I was a data analyst or an actuary, a data scientist who haven't had a whole lot of experience, doing a little bit of coding, where could I get started if I want to start developing my own applications?

Ean Chan:

I think if you know some Python that's perfect that's the language everyone seems to be using for this kind of development. Like if you're so starting out at the very beginning I would I would say get used to using ChatGPT like a lot itself but then also try to use they call it Playground View still, where you're looking at the backend you can edit the system prompts and like kind of understanding what some of those parameters are and what it means.

In terms of actually developing your own applications, I think go download an LLM, if you want or connect to your ChatGPT, OpenAI API key and then just start with as basic a tutorial as you can right? Like find something which lets you put in a prompt and get a response and then just start going from there.

I use OpenAI resources a lot – because EY is also partnered with Microsoft across a lot of things. So, I use the Microsoft Azure platform a lot for work but in terms of personal, you can just load up your favourite IDE for using Python and yeah, connecting directly with packages. Langchain is very good, if you want a nice front end you can use streamlit. Just work on getting that very first response, and then you might get addicted along the way to making that response better and better for your use case.

Aidan Sussman:

Ah, cool, really interesting. Does anyone else have any other thoughts?

William Zheng:

Yeah, we just call out HuggingFace also has ah it has a wealth of resources, and you know they have a lot of documentation as well on that if you want to explore ah models from different providers such as Mistral.

Google just add a bunch of models today as well, or yesterday, and you know there are other Open Source ones like Llama and all of its derivatives.

Minh Phan:

I'd say another thing to add is if you want to learn in a structured manner deeplearning.ai has some really good courses that are all free.

The other way I like to I'm a video person so, I watch people build applications on YouTube.

As a channel - Mervin Praisson, he does these 5 minute videos on tiny applications gives the code like just watch people write this stuff and then copy their code and then run it. Like that's a really good starting point for building the application is like run someone else's application and then off the back of that you can once you see that how it's done, you can gain confidence like that you can do something similar or like you know fork or fork the code um and build something at yourself.

Aidan Sussman:

That's really interesting. Watching other people, do a bit of learning playing around with GPT, and having a look at other models and documentation.

So, getting a bit technically here. So especially when ChatGPT first came out, a lot of people thought the best use case for models was "we need a train one specifically on all the data I have" but now this term 'rag' gets thrown out around a lot. So, what is 'rag' and what's fine tuning when it comes to LLMs.

William Zheng:

Yeah, sure I'll have a crack at it.

You know feel free to join Ean and adding additional perspective on it.

So I think that the most intuitive understanding of this is when you use ChatGPT right? And it kind of absorbing all of the world's knowledge and it's just talking about you know when you talk to when you're type in your chats with it?

Now the big idea of using LLMs for your own is that can you make ChatGPT in such a way that you know when it spits out text at you. It's more relevant to your use case you know make it more relevant and make it more compatible with your business use case.

For example, if you work with legal you want to have ChatGPT that talks about law instead of just general. You know Donald Trump stuff you know? So, this concept is that you have your own your business or your organisation or your team have your own documentation or your own text, own knowledge bank essentially and can you then tune.

The large language model to such extent will produce text that is more relevant to what your knowledge bank is about. Now there's two kind of main ways of doing that.

One is 'RAG' - stands for Retrieval Augmented Generation which I'll briefly describe what it is and then the other one is fine tuning. I'll cover fine tuning first which comes from like a data science/machine learning background that can be more directed explained.

So essentially (one fine tuning), it involves a big neural network. It has weights so weights are numbers in a neural network - just mathematical numbers. You retrain the large language model by updating the numbers the weights in the large language model itself. So, the large language model gets changed and altered so that it produces outputs closer to what you wanted – closer to your own knowledge bank. However, this process has a certain cost to it.

First of all, there's a computation cost of running the fine tuning, and there need some additional knowledge/expertise, from people who are familiar with working with a large language model itself in terms of code - you may need people with machine learning background to do the fine tuning. In addition to that once you have a fine tune model. And you may want to serve that model yourself.

So, there are some cases where you may want to deploy that larger language like large language models on your own infrastructure and that could that could involve additional resources.

There are other ways to deploy them, for example where OpenAI would allow you do that, as well as companies like MosaicML, DataBricks and many other platforms now.

But traditionally, serving your own model has involved a significant cost.

Now what is RAG? RAG is alternative way (to fine tuning). Large language models are really good at synthesizing and producing text. So alternative way is - instead of doing the numerical update of the neural network model, what you can do is you don't change a large language model - you just use it directly. In this way you can do search for the relevant knowledge in the knowledge bank. Then you want the large language model to look at it, with the relevant snippets of text retrieved. Here you are using a large language model and you led the LLM to reason with it and to take it in into the context you know into the kind of prompt context and then having the LLM spits out the result.

So, the advantage is you can relatively quickly develop (the application), it's also a relatively new newer field and the disadvantage would be the latency. So sometimes, if your application requires a very short latency then every round-tripping of the back and forth of the RAG pipeline could involve longer latency depending on your application - you know use case – so, I think that's kind of the main things I think for the pros and cons.

Ean Chan:

So I do have some opinions on RAG. Um, yeah, but some of them I'm qualified to have.

So, I think I think fine-tuning like it - it exists with LLMs because it was such a good concept with the rest of machine learning right? So, when you're when you're training other models fine tuning by you know, taking a neural net, redoing the last layer or retraining all the weights or whatever you want to be doing with it.

I think made a lot of sense with a lot of the you know categorisation models or predictive models or anything like that. But I think for LLMs unless you're an AI researcher or you're you know a professional software developer, I think RAG is king. I think RAG is just easier and better in a lot of cases.

You're right it'll be slower. I don't know that any of us are producing apps where it'll matter the latency or the cost. So that's just from I guess a more actuarial perspective in cases where we're going to use it which is like I don't know reading PDSs and then trying to take the information learned from the PDS that that you've fed into it. Trying to Finetune for something like that. Like it's because it's language right? It's like so subjective. It's as if it's a better or worse response as well. So, you need like a lot of data with someone giving a subjective thumbs up, thumbs down approach to it.

You need, you just need a lot of manpower, data, effort. Which I think RAG just because of how magical LLMs are like RAG just works. That's my opinion.

Minh Phan:

I was just going to jump in with like another way that I like to think about RAG versus fine-tuning is kind of short-term versus long-term learning and in the context of actuarial exams.

So, you've got your open book exam basically you can if you take. Those notes and study them and commit them to memory and like learn the rich concepts. That's the equivalent of fine tuning. So, you're updating the weights inside your brain whereas when you go into these open book exams - you can kind of skim through the notes but not really learn them properly but know that you can refer back to them when you need to.

So, as long as you've indexed those notes very carefully then you go into the exam, you get the question you find the relevant chunk you read it really carefully and then you synthesize your response. So, like that's the process of RAG so being able to refer to your notes and produce an answer.

You don't have to learn as hard but you're kind of reliant on the fact that you have to be able to find the relevant chunks of your notes and produce a good answer just from those things. So it's less computationally intensive And there's less effort in doing that.

But then you potentially don't get as rich in the domain and interrelated concepts. So, like I think fine tuning is like is worthwhile if you need to kind of like truly to understand the subject matter and not have to refer to like other notes.

William Zheng:

I want to add one more good benefit for the RAG. So you know when you have knowledge bank that changes with time but let's say for example, you have deleted some text or your knowledge bank actually changes over time. Then, every time it changes you sort of have to fine tune again which is additional cost whereas with RAG, the unwanted content is automatically removed from the knowledge bank. So, it doesn't get retrieved anyway in the RAG process. So that's one you know big benefit I think with RAG.

Futuristic sound

Ean Chan:

Hello, this is Ean from the future. I'm just adding a little note here because it's been about three weeks since the recording. And with the insane rate of progress in the space, there's now become an alternative to RAG and fine-tuning which might be practically viable very soon. So just backing up a bit, if we're trying to solve the problem of getting a customized LLM which is attuned to your specific knowledge, grounded in information that you give it, there's fine-tuning we talked about which was actually changing the weights and biases by training it on your textbook. Hopefully, it's structured data but I guess it doesn't need to be.

There's RAG which is getting specific pages and paragraphs from your textbook and inserting them into your prompt every time you ask a question. So it's an architectural approach which allows it to specialize knowledge per query. The third option I want to talk about now is why not just grab your entire textbook and insert the whole thing into the prompt? And so, the limitation here would be the context window. So context window could be anywhere from 4,000 to 32,000 amongst the actual commercially viable options nowadays.

And 32,000 such as GPT-4-32k, as the name suggests, it's the 32,000 token context window size. That translates to about 24,000 words, 40 to 50 pages. A lot... Enough for RAG, not enough for your entire textbook. So some of the technology which has been announced/released just recently is Claude 3 which has 200,000 token context window which is a lot. But then, there's also Google's Gemini which has a 1 million token context window. But that one, I don't think, is available to the public yet.

So these would let you drop your entire textbook in pretty comfortably with room to spare for a whole bunch of prompts and responses that you're getting. The other issue that would usually exist or that does exist is

the lost in the middle problem. And it's a tendency for LLMs to focus more on the start and end of your prompt but its retrieval rate for information in the middle is sometimes lower. It seems like Anthropic with their Opus model has really solved this. So a bunch of experiments they did were called needle in the haystack experiments where they'd get 150,000 words of text and drop one sentence into the middle. So for example, all of the 150,000 words would be talking about software development. And then, there's one sentence talking about pizza toppings.

So it was able to correctly identify when queried what the sentence was with perfect accuracy. But it would also comment on how out of place that sentence looked and even suggest that maybe this is a practical joke or a test that... Yeah. So the LLM seemed quite aware, which I shouldn't use that word, but conversation for another time.

In summary, I guess this technique of just being able to drop your entire textbook in might... Well, by the time you're listening to this, in the future future, might already be viable from a commercial point of view. It is more expensive than RAG because you get charged per token. So 200,000 tokens of cost is going to be more than the maybe 1,000 you have for a RAG prompt. But yeah, could be cheap enough soon that it's a viable way to approach this issue. Anyway, I will now pass you back to past Aidan.

Futuristic sound

Aidan Sussman:

Okay so Sora was released recently. You provide text and it can give you a really beautiful cinematic video that looks like it's taken a big budget to film. It's pretty huge. Before we're just talking to Chat GPT and it gave us responses and now we've got cinematic videos. What's next?

Ean Chan:

So, I have a big prediction for this by the end of year. I reckon there'll be a feature length movie. So, 2024 like 90-minute movie text-to-video. No actors. No camera crew required, maybe editing over the top – we will see about that one but that's where ChatGPT is going next.

It actually reminds me when people talk about LLMs like their next word predictors and they say ah an LLM is just predicting the next word. It doesn't know anything. It doesn't understand anything.

And not to get philosophical about what “know” and “understand” actually mean... but I think that's a very reductive view where you know sometimes to predict the next word someone saying you need to know a lot about a topic.

If an astrophysicist was giving a lecture and then he stopped mid-sentence and you tried to predict his next word... the best way to predict it would just be to go and learn astrophysics and you know and 100 billion parameters, who knows what's going on there.

So, I think that transition to video... Potentially in these models, there's a lot being understood behind the scenes to be able to produce a video of waves crashing.

William Zheng:

I would just say that I want to like, and I hope people can keep open mind about these things. AI (like LLM) is just one aspect of such technology in this space. There's diffusion models as well and we know the transformer architecture which is what the LLM is based on is working quite well - but we should keep an open mind about future architectures that could radically improve the capability of these AI systems - LLM is just a subset of the AI field.

There are different model architectures that the researchers are currently pursuing, such as using joint embedding architectures that for example, Prof Yann LeCun (from the Meta AI research group) is exploring

and which I think has a lot of legs. And also with reinforcement learning architectures that could be fused with various LLMs architectures and transformer models. So AI as an industry is going to be a big deal and will continue to be a big deal in the future, extending beyond the LLM architecture.

Minh Phan:

My closing thought on this is that I think development has been just insanely rapid over the last year and a half and while I think it's currently so in terms of capability versus the hype I think it's overhyped around what it can do right now but give it another you know one and a half years maybe it actually lives up to the hype to that around saying and that it will be able to displace portions of the labour workforce - which is not that scary because those people they get displaced will just go to another occupation. We're humans we'll will adapt and use our skills for a different purpose or produce more output.

So, I'm just kind of excited to see how this technology develops and I think as actuaries, data science actuaries we should make sure that we stay abreast and learn how to use the technology so that we can kind of continue to kind of add value to our companies and within the industry so that and we won't be the first to be displaced.

William Zheng:

I have actually have like a contrarian bet on this I don't think that any job much of the job will be displaced. You know I observe that the LLM technology or AI is being democratised today. And the low barrier entry is low.

What this means that everyone or businesses will get these technology, so the differentiating profit moat will likely need to be in something else right? And the job will still be there to work on that something else.

Aidan Sussman:

Cool I think that's a good note to end there.

Listeners, we hope you enjoyed this session with Ean, Minh and William. If you're hungry for more conversation and insights we've got you covered, check out the Actuaries Institute Podcast hub via the podcast section of actuaries digital or via all mainstream podcast apps. If you'd like to continue conversation with us on social media join us on Facebook LinkedIn at Actuaries Institute or you can reach also via Instagram on @ActuariesInst. Please write in your questions and comments on the show, we'd love to hear from you. I'm Aidan, bye for now.