SPEAKER: Welcome back. So now that we've taken a look at the history of artificial intelligence, some of the major stepping stones that have led us to where we are today, let's look at generative AI and really get into understanding how generative AI tools work. So the thing that got us from AI that had sort of been building and building for 40-some years to where we are today, to generative AI, to generalized machine learning was something called a transformer. And no, I'm not talking about the Hasbro toy here.

But this was a massive leap forward in terms of what artificial intelligence algorithms could do. In a seminal 2017 paper called "Attention is All You Need," researchers at Google realized that they could make simple, powerful transformers that would take language, in particular, and, with these transformers that were self contained, that didn't require a lot of other software systems that things like text-based translation had long relied on, they could take these self-contained transformer boxes and perform the kind of tasks that used to take minutes or hours to complete in a matter of seconds.

So without getting too technical, a transformer sort of takes input and outputs something that is very similar. It takes text in, and it puts text out it. Takes audio in. It puts audio out.

But most of the time, and in relation to ChatGPT and other generative AI, it uses text here. It's a black box that takes an input and gives us an output. In this case, the transformer on the screen is focusing on language translation, and it takes the phrase, je suis estudiant, and turns it into the English version, I am a student. And what the Google team realized is they could do this very, very rapidly in a much less complex way using something called a self-attention decoder, which is a specific kind of transformer.

And I promise we're going to get back to transformers a little bit later on in this lecture. And because of this work around transformers and specifically the self-attention decoder, a special kind of transformer, ChatGPT-1 was created in 2018. So what is a GPT anyway? What does it stand for since we talk about ChatGPT all the time?

Well, GPT stands for generative pre-trained transformer. It's generative in that the model itself is capable of generating continuations to the provided input. That means, given some text, the model tries to guess which words come next and generate text-based output on that. It's also pre-trained, so the model is trained on a very large corpus of general text, and the model is also meant to be trained once and used for a lot of different things without needing to be retrained from scratch all the time. That's why it is pre-trained.

And there are a lot of pieces of data in this pre-trained model, this generative pre-trained transformer. So in the original GPT model, GPT-1, which was released back in June of 2018, it was comprised of 117-million parameters, or pieces of text. So it looked at 117-million different combinations of text to learn how to generate more text.

GPT-2 was released in-- well, not released in February 2019. Now, it was able to produce coherent multiparagraph texts, but it didn't have any guardrails in place to avoid hate speech and things like that. And so it was never really released to the public because they're like, wow, we forgot this really important thing. And we're going to talk more about the importance of guardrails and preventing hate speech later on in this class.

Now, GPT-2 went from 117-million to 1.5-billion pieces of text. GPT-3 was released in June 2020. That's largely what ChatGPT is based upon. ChatGPT, the public, free version, uses GPT-3.5, which is based on the GPT-3 model, that was then trained on 175-billion parameters, or pieces of text. And GPT-2 could make full paragraphs, but GPT-3 really generated pretty advanced text, and it was able to answer factual questions and translate between languages, pretty cool.

And then GPT-4, which was released in March 2023, had 1.8-trillion pieces of information and text as training data. Now, this did not all actually go into a single model. They called GPT-4 kind of a single model, but it's not.

It uses something called mixture of expert models. It has 16 of those around specific topic domains. And then it uses lots of software to route specific words to the appropriate expert model. But you can see that's a lot of parameters being passed into these models and then being processed by GPUs during training. It's an almost unfathomable amount of data.

But it still doesn't tell us how large language models work. So without a master's in computer science, can we explain how they work? Well, of course, we can because large language models are actually based on a very simple statistical model.

So you can think of a GPT or ChatGPT kind of like how meteorologists predict hurricane paths. Millions of pieces of data go into a system which predicts the path of a hurricane in real time. Now, this system isn't going to be 100% accurate all the time. But it's going to make its best guess as to where the hurricane is headed in the next 24 hours. It uses data to generate probabilities for the path of the hurricane.

And that is exactly what large language models do. The classic definition in computer science of a language model is a probability distribution over a sequence of tokens or, if you want to use the word token, of words, of words. So let's say we have a vocabulary of words represented by the v here. And then we also have a sequence of tokens of individual words, word number one, word number two, word number three, word number four, word number five, and so on.

And a language model represented here by p assigns each different sequence of tokens, each different sequence of words, a probability, a number between 0 and 1 that tells us, the users of this model, how, quote, good or accurate a sequence of tokens actually is. So let me give you an example.

Let's say we have a vocabulary containing six words-- ate, ball, cheese, mouse, the, green. Then the language model assigns a specific probability to a specific sequence of these words, or tokens, in the vocabulary. So the language model would say that the sequence "mouse, the, the, cheese, ate" has a very low probability score. And the language model would say that the sequence "the, cheese, ate, the, mouse" also has a pretty low probability score, although not as low as the other one.

The model would assign a much higher probability to the sequence of words "the, mouse, ate, the, cheese." so why would it assign that particular sequence a much higher probability? They're just words, right? Well, the model itself has code, has lines of software, algorithms that evaluate both the syntactic probability of a sequence of words as well as a world knowledge about how those words are used in the real world.

And it gets both of these things from its training data from the millions, or billions, or trillions of pieces of text that it gets when the model is being trained. And let's not forget-- it also gets some foundational knowledge about how the English language works, in this case, from the decades of research and commercial deployment of dictation software, like Siri or Dragon NaturallySpeaking. So the sequence "mouse, the, the, cheese, ate" gets a very low syntactic probability score because the algorithms inside the model say, this is not how these words go together in the real world.

The sequence "the, cheese, ate, the, mouse" also gets a fairly low probability because the world knowledge that the model has gained through its training also says, uh uh, cheese doesn't eat mice. Mice eat cheese.

But the sequence "the, mouse, ate, the, cheese" gets a much higher probability because, during its training, it has learned that this sequence has both good syntactic probability and world knowledge probability. So if someone were to type in the words "the, mouse, ate," it would probably guess "the,

cheese" as being the next couple of words because that particular sequence of words has a much higher probability than other sequences of the same words.

So how does the large language model know what's good syntactic probability and what's good world knowledge? Well, it does this through a training process. So in the training process, you have a data set. In this case, it's the web. It's all the public textual information you can find on the web translated it into 300-billion individual tokens, or words, or sequences of words.

And then the language model is given an objective. In this case, the objective is to predict the next appropriate word in the sequence. So the system has a data set. It has its objective. And it's given an example.

The mouse ate the, what? Predict the next word. Or more likely, because is this is generative AI and it's only picking the next word in the sequence, the example begins with something like just, the mouse, and nothing else, no verb to help it along. What does it do next when it's presented with the words, the mouse?

So the language model looks at the words, the mouse, and then whatever data it has in its training set-- and this might be a very, very limited amount of data to start with, but it's going to build, obviously, over time. It's going to make a guess. It's going to say, OK, based on the probability values that I have for what the next word might be, I'm going to make a guess.

And it guesses the word, ate, as in, the mouse ate. And it's going to get labeled by the algorithm as correct because, in the algorithm, the algorithm has 1,473 different examples where the words "the mouse ate" go together. So it says, yes, this is correct. This is an appropriate next word after the words "the mouse."

And then the language model picks the next word, the word "the". And the model says, based on the information I have already in my data set, my training data set, this sounds good. This works. I see lots of examples for this, so we're going to mark this as correct.

The next word it picks is "cheese". The system says, this is great. The model says, yep, this is right because I have lots and lots of examples of the phrase "the mouse ate the cheese." So this is good. We're going to mark this as correct.

And then the language model says, this looks like a complete sentence to me, because it understands the basic of sentences. And it says this is great. This is a valid, real answer. We're going to say stop here. "The mouse ate the cheese" is a perfectly good response to filling out this objective, to completing our objective and making a whole sentence.

This has good syntactic probability and good world knowledge probability. So we're good with this. We're set. Let's move on to the next task.

So the model saves that data and then moves on to trying a different set of words. OK, we're going to do this task again because it does these tasks over and over, millions of times with millions upon billions and, ultimately, trillions of different combinations of words. So it sees the words "the mouse" again, and it comes up with maybe a different response as the next word. And it comes up with a word "cooked". But then the language model goes, whoa, whoa. In my 1,794 different examples I have of this, none of them follow the words "the mouse" with "cooked". That doesn't sound right.

Yeah, it might be syntactically correct, but world knowledge wise, this is not correct at all. This is going to be marked as incorrect, and I'm going to save this information, put it in my database, put it back in my model itself so that the next time the model sees the words "the mouse", it won't return "cooked". But it's much more likely to return the word "ate" because that has better syntactic and real-world probability. And it's going to do this billions upon billions of times with all sorts of different combinations of words in its vocabulary.

And because it stores the results of each of these training loops as simple mathematical vectors, graphical processing units are able to do this work really, really quickly because, again, graphical processing units excel at processing vectors of information. So we've seen how a language model can assign a probability to a sequence of words. But how does it really understand? How does it really understand what those words mean and what the context is? If I put the word "it" in a sentence, how does it know what the word "it" is?

So this is where that transformer, that specific kind of transformer that I mentioned earlier, a self-attention decoder, comes in. So the self-attention decoder takes a word out of a sentence, and looks at the entire sequence of words in a sentence, and then figures out what that word means. So in this example, the self-attention decoder could look at the word "her" and then be able to look at the whole sentence and say, ah, that refers to mother or look at the word "child" and know that it refers to the words "it".

So a good self-attention decoder understands that words are related to other words by function, by referring to the same thing, or by informing the meanings of different words. And what the Google researchers did in 2017 was figure out that you didn't need a lot of other external resources to determine sentence structure. All you need is an attention decoder paying attention to itself to figure out how to understand the structure and meaning of sentences.

This made using these kinds of tools both simpler and orders of magnitude faster. So it's from the self-attention decoder that the system can say, this word relates to this word, the verb is related to the noun, and all the other functions of grammar that we understand intuitively as humans. This is then, again, translated into a mathematical vector.

So it can be computed and worked upon extremely quickly by a graphical processing unit. So in addition to the probability scores that we saw earlier, the transformer, the self-attention transformer, also takes these semantic embeddings, the fact that "it" referred to the "child" or "the mother" refers to "she" in the sentence we just saw and turns this into a series of mathematical representations, these vector-based embeddings. And it can be difficult for us as humans to visualize what these embeddings look like and how they might work.

But in this particular example, this particular visualization, although not very friendly to people with blue- or red-based color blindness, each bar represents a set of embeddings and how that word that precedes each bar might be correct or incorrect based on another word in the list. So in this visualization, a dark blue bar means that the probability of that word being correlated to the other word that it has in its vector embeddings is very low whereas a dark red bar indicates that that word is very highly correlated in its training to whatever the other word was.

So for example, "sea" and the word "orange" probably don't have high correlation in this visualization. They would be represented by a dark blue bar whereas the word "sea" and "fish" would have a very high correlation in the training data through the self-attention decoder and have this red bar here. But the point of this visualization shows you how those numeric values really get translated into low- or high-correlation values in both syntax and world knowledge when it comes to training via the self-attention decoder.

Now, ChatGPT goes one extra layer here in a way that traditionally, until ChatGPT came around, most large language models didn't. And it put all of these contextual embeddings, all of this information about how words relate to one another in a sentence, into another decoder layer called a feed-forward neural network. Now, this is a large neural network, or sort of artificial brain, that can usually tell what words come after other words. It's much more advanced. It's much faster and much more accurate than previous decoders that did this kind of work.

So you take this huge pre-trained model. You mix in this feed-forward neural network as people type in questions. And you use the self-attention decoder to analyze the text, the individual words in the questions and the prompts people are making, and that is what transforms that input into an output for you in ChatGPT, Claude, and other current generative AI tools. Now, again, I am grossly simplifying here, and there are many, many more steps in the process, including normalizing a lot of these mathematical vectors.

But it all still boils down to this simple probability. That's how LLMs work. And remember, these are probabilities, not certainties. These are predictions, not facts.

And this is going to be really important to remember as we progress through these lectures. Some people call ChatGPT, Claude, Bing Chat, and pretty much all the currently available large language models stochastic parrots. Now, this phrase was coined by Emily Bender in her 2021 paper "On the Dangers of Stochastic Parrots-- Can Language Models be Too Big?" And the phrase stochastic parrots has been used to deride the actual usefulness of LLMs.

Stochastic parrot or, in less graceful language, BS machines as a label, though, raises a valid point. These tools are good at generating convincing text that they really do not understand. Now, this is an oversimplification, especially given what we know about how ChatGPT combines a self-attention decoder with a feed-forward neural network to produce pretty startlingly good results.

But it is a truth about current large language models. They generate probabilities, not certainties. And I want to point out that it's not just text-based generative AI that's trained this way. Image-generation tools, like Midjourney and Microsoft Image Creator, work very much in the same way.

So these tools were also trained on vast data sets that used long-standing object-classification artificial intelligence methods to say that an image contained a person, or a dog, or a mountain. But the genius of Midjourney and other image-generative tools was combining this neural network, this artificial brain of images and their associated labels, with a pre-trained language model like GPT, GPT-3 to be more specific.

So these systems use the contextual language understandings of a self-attention decoder to match the prompt, the thing you're asking, say, Midjourney to create with the classification tags from the neural network. So then Midjourney or another image-generative tool uses those tags to create a low resolution, very low, rough resolution image based on those tags. At that point, another kind of algorithm takes over, a diffusion model, or, more specifically, a diffusion probabilistic model.

So in short, diffusion models take an image, that rough image that Midjourney already created, and then gradually add in noise on top of that image until it is not recognizable. From that point, the generative AI inside of Midjourney tries to reconstruct that image to its original form. And in doing so, the AI learns how to generate pictures or other data.

So when you ask Midjourney or Microsoft Designer to create an image, it essentially adds random noise to the rough shapes from earlier in the process and then refines and refines that noise until the shapes look like what was originally requested based off its own internal model of how a person, or a dog, or a mountain, or, in this case, the COVID-19 virus is supposed to look. And again, Midjourney is, at its heart, a prediction machine, just like ChatGPT. Midjourney, and Stable Diffusion, and other image generators can guess what you're after, but as systems, they cannot produce deterministic output because they are always guessing.