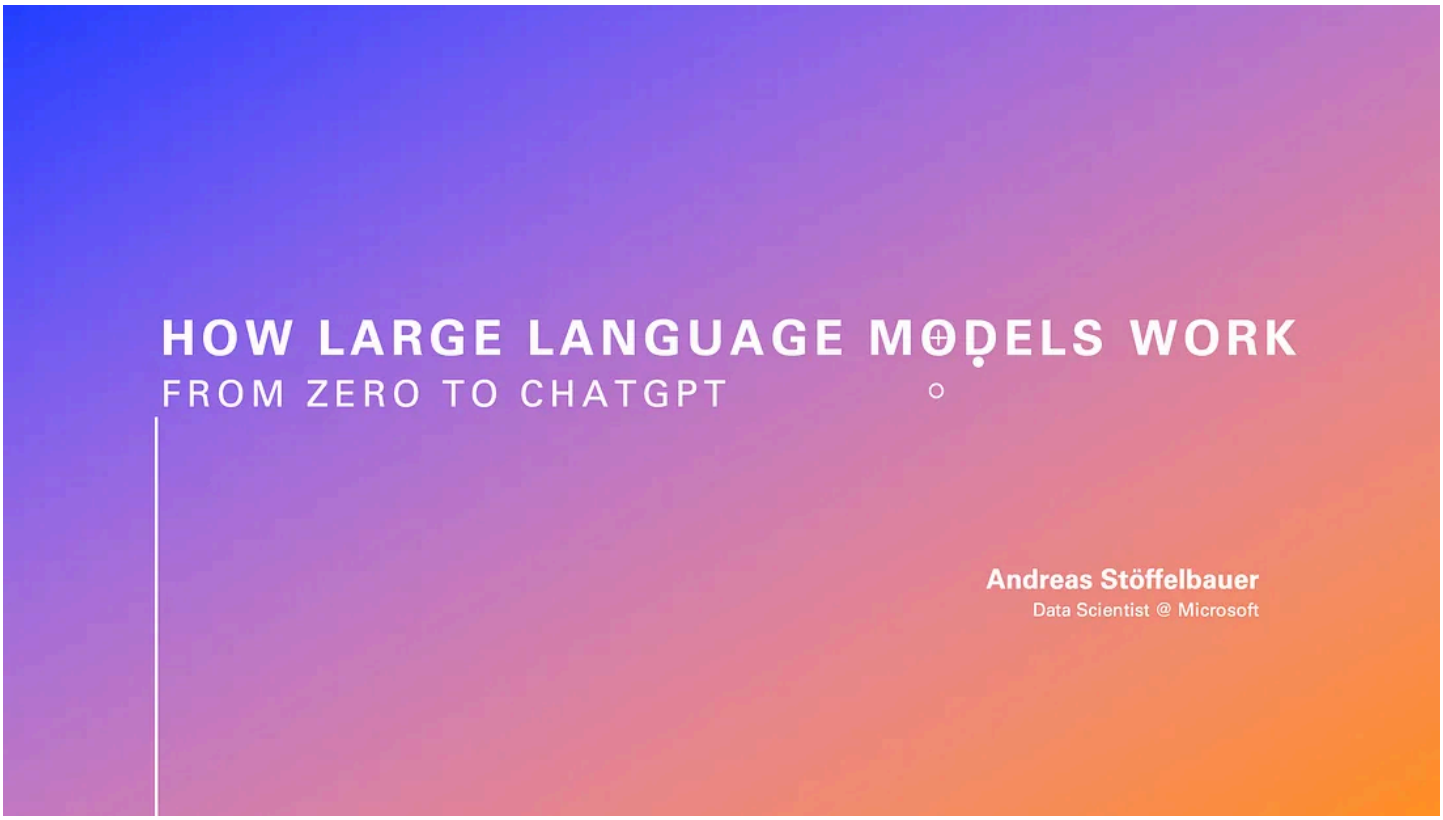


Title Image. How Large Language Models work. From zero to ChatGPT.

How Large Language Models work

From zero to ChatGPT

Andreas Stöffelbauer · Follow
Published in Data Science at Microsoft
25 min read · Oct 24, 2023

Listen Share More

Thanks to Large Language Models (or LLMs for short), Artificial Intelligence has now caught the attention of pretty much everyone. ChatGPT, possibly the most famous LLM, has immediately skyrocketed in popularity due to the fact that natural language is such a, well, natural interface that has made the recent breakthroughs in Artificial Intelligence accessible to everyone. Nevertheless, how

LLMs work is still less commonly understood, unless you are a Data Scientist or in another AI-related role. In this article, I will try to change that.

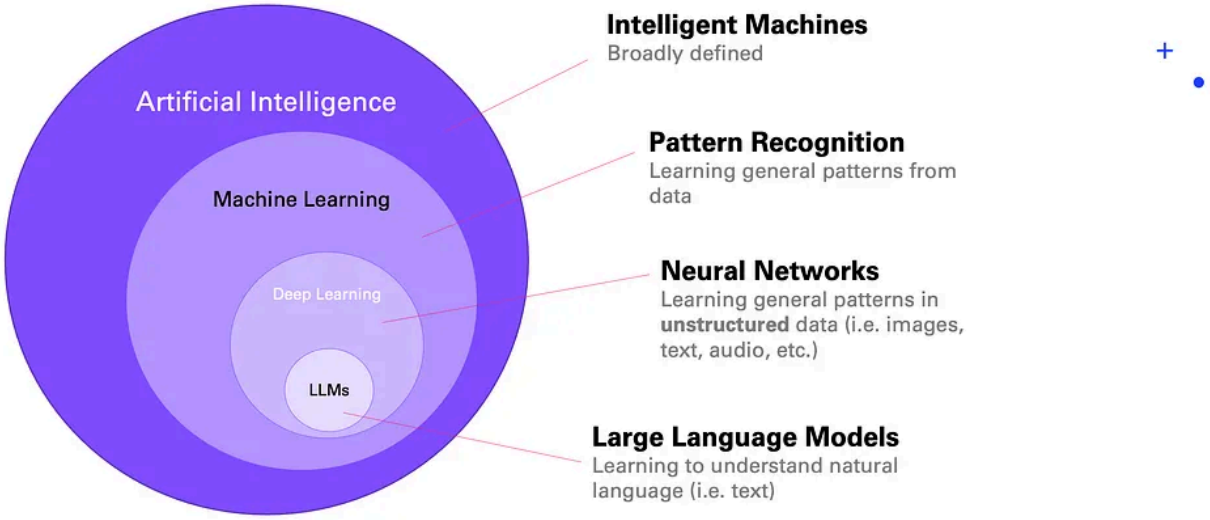
Admittedly, that's an ambitious goal. After all, the powerful LLMs we have today are a culmination of decades of research in AI. Unfortunately, most articles covering them are one of two kinds: They are either very technical and assume a lot of prior knowledge, or they are so trivial that you don't end up knowing more than before.

This article is meant to strike a balance between these two approaches. Or actually let me rephrase that, it's meant to take you from zero all the way through to how LLMs are trained and why they work so impressively well. We'll do this by picking up just all the relevant pieces along the way.

This is not going to be a deep dive into all the nitty-gritty details, so we'll rely on intuition here rather than on math, and on visuals as much as possible. But as you'll see, while certainly being a very complex topic in the details, the main mechanisms underlying LLMs are very intuitive, and that alone will get us very far here.

This article should also help you get more out of using LLMs like ChatGPT. In fact, we will learn some of the neat tricks that you can apply to increase the chances of a useful response. Or as Andrei Karparthy, a well-known AI researcher and engineer, recently and pointedly said: "English is the hottest new programming language."

But first, let's try to understand where LLMs fit in the world of Artificial Intelligence.

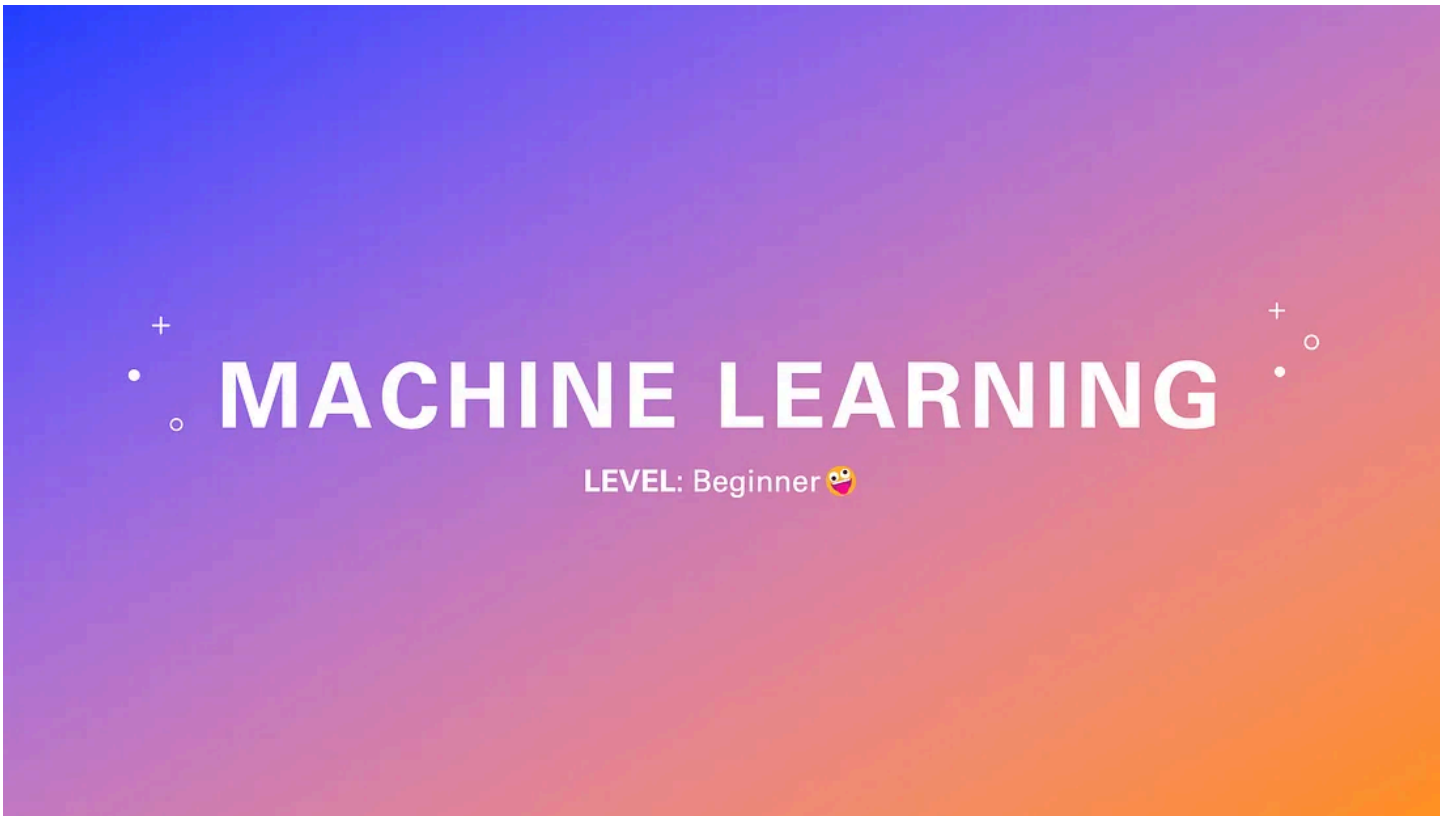


The field of Artificial Intelligence in layers.

The field of AI is often visualized in layers:

- **Artificial Intelligence** (AI) is very a broad term, but generally it deals with intelligent machines.
- **Machine Learning** (ML) is a subfield of AI that specifically focuses on pattern recognition in data. As you can imagine, once you recoginze a pattern, you can apply that pattern to new observations. That's the essence of the idea, but we will get to that in just a bit.
- **Deep Learning** is the field within ML that is focused on unstructured data, which includes text and images. It relies on artificial neural networks, a method that is (loosely) inspired by the human brain.
- **Large Language Models** (LLMs) deal with text specifically, and that will be the focus of this article.

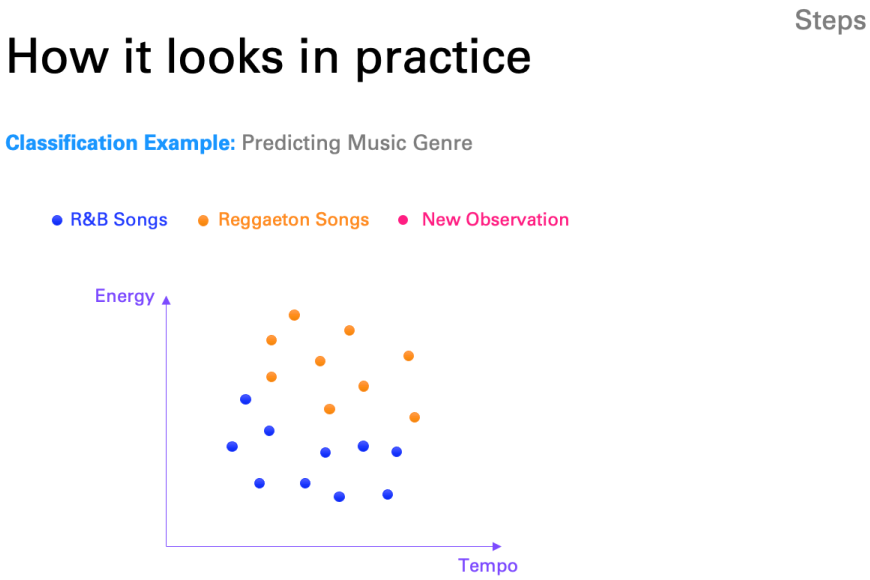
As we go, we'll pick up the relevant pieces from each of those layers. We'll skip only the most outer one, Artificial Intelligence (as it is too general anyway) and head straight into what is Machine Learning.



Machine Learning. Level: Beginner.

The goal of Machine Learning is to discover patterns in data. Or more specifically, a pattern that describes the relationship between an input and an outcome. This is best explained using an example.

Let's say we would like to distinguish between two of my favorite genres of music: reggaeton and R&B. If you are not familiar with those genres, here's a very quick intro that will help us understand the task. Reggaeton is a Latin urban genre known for its lively beats and danceable rhythms, while R&B (Rhythm and Blues) is a genre rooted in African-American musical traditions, characterized by soulful vocals and a mix of upbeat and slower-paced songs.



Machine Learning in practice. Predicting music genre is an example of a classification problem.

Suppose we have 20 songs. We know each song's tempo and energy, two metrics that can be simply measured or computed for any song. In addition, we've labeled them with a genre, either reggaeton or R&B. When we visualize the data, we can see that high energy, high tempo songs are primarily reggaeton while lower tempo, lower energy songs are mostly R&B, which makes sense.

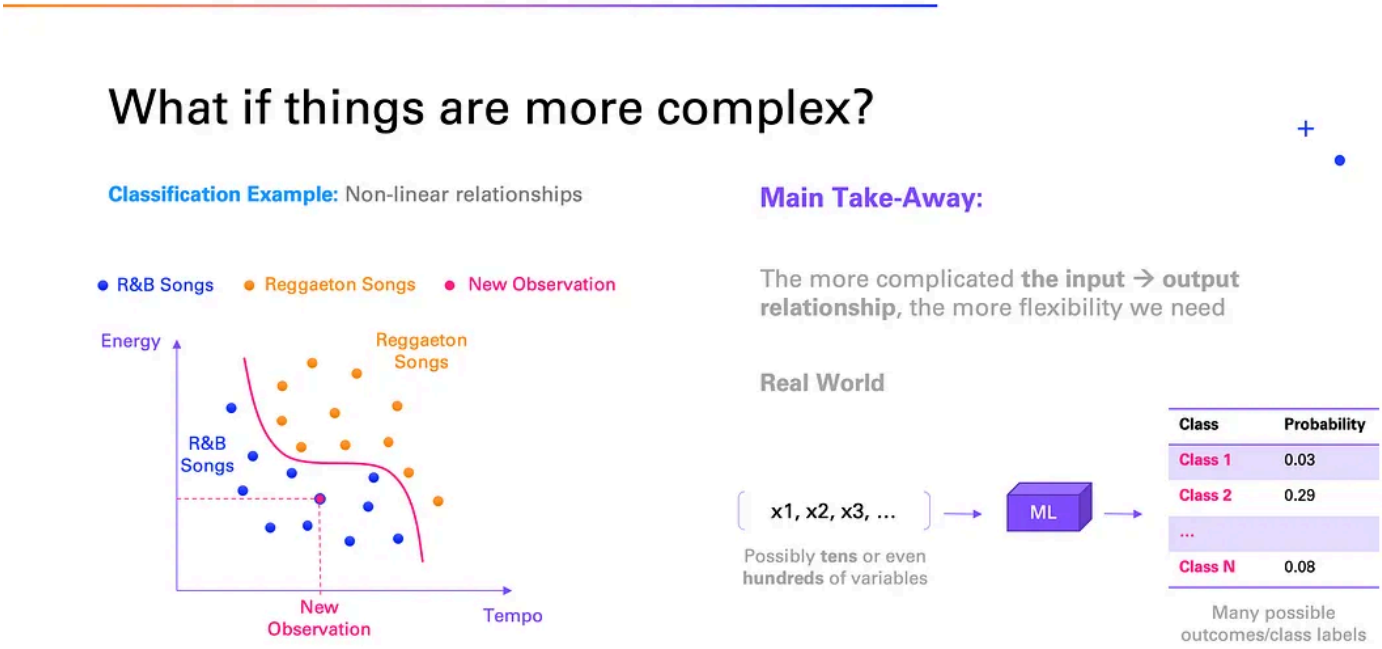
However, we want to avoid having to label the genre by hand all the time because it's time consuming and not scalable. Instead, we can learn the relationship between the song metrics (tempo, energy) and genre and then make predictions using only the readily available metrics.

In Machine Learning terms, we say that this is a classification problem, because the outcome variable (the genre) can only take on one of a fixed set of classes/labels — here reggaeton and R&B. This is in contrast to a regression problem, where the outcome is a continuous value (e.g., a temperature or a distance).

We can now "train" a Machine Learning model (or "classifier") using our labeled dataset, i.e., using a set of songs for which we do know the genre. Visually speaking, what the training of the model does here is that it finds the line that best separates the two classes.

How is that useful? Well, now that we know this line, for any new song we can make a prediction about whether it's a reggaeton or an R&B song, depending on which side of the line the song falls on. All we need is the tempo and energy, which we assumed is more easily available. That is much simpler and scalable than have a human assign the genre for each and every song.

Additionally, as you can imagine, the further away from the line, the more certain we can be about being correct. Therefore, we can often also make a statement on how confident we are that a prediction is correct based on the distance from the line. For example, for our new low-energy, low-tempo song we might be 98 percent certain that this is an R&B song, with a two percent likelihood that it's actually reggaeton.



In reality, things are often much more complex.

But of course, reality is often more complex than that.

The best boundary to separate the classes may not be linear. In other words, the relationship between the inputs and the outcome can be more complex. It may be curved as in the image above, or even many times more complex than that.

Reality is typically more complex in another way too. Rather than only two inputs as in our example, we often have tens, hundreds, or even thousands of input variables. In addition, we often have more than two classes. And all classes can depend on all these inputs through an incredibly complex, non-linear relationship.

Even with our example, we know that in reality there are more than two genres, and we need many more metrics other than tempo and energy. The relationship among them is probably not so simple either.

What I mainly want you to take away is this: The more complex the relationship between input and output, the more complex and powerful is the Machine Learning model we need in order to learn that relationship. Usually, the complexity increases with the number of inputs and the number of classes.

In addition to that, we also need more data as well. You will see why this is important in just a bit.

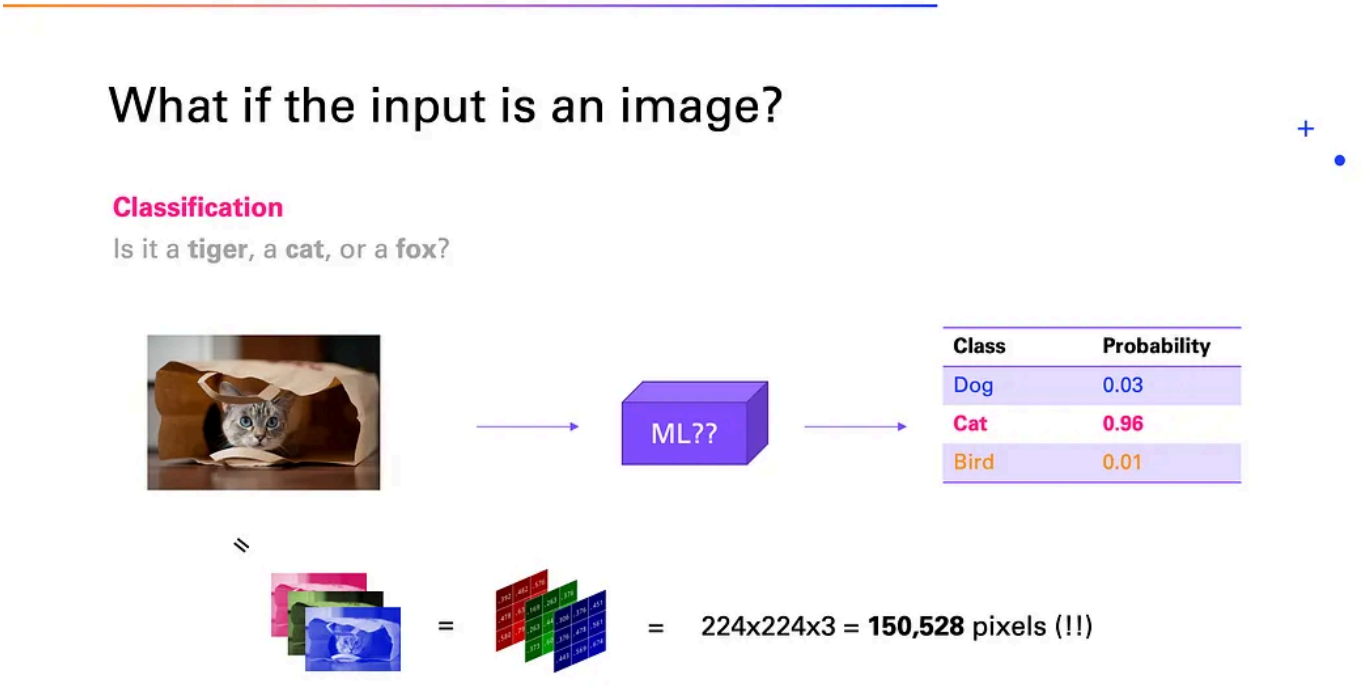


Image classification example.

Let's move on to a slightly different problem now, but one for which we will simply try to apply our mental model from before. In our new problem we have as input an image, for example, this image of a cute cat in a bag (because examples with cats are always the best).

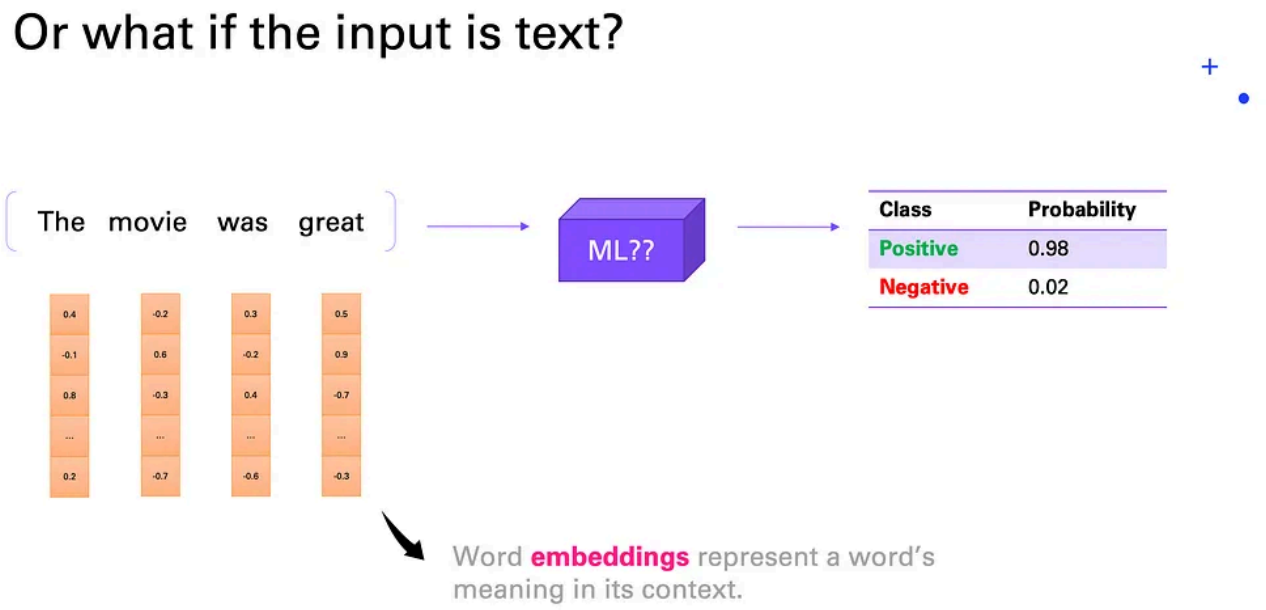
As for our outcome, let's say this time that we have three possible labels: tiger, cat, and fox. If you need some motivation for this task, let's say we may want to protect a herd of sheep and sound an alarm if we see a tiger but not if we see a cat or a fox.

We already know this is again a classification task because the output can only take on one of a few fixed classes. Therefore, just like before, we could simply use some available labeled data (i.e., images with assigned class labels) and train a Machine Learning model.

However, it's not quite obvious as to exactly how we would process a visual input, as a computer can process only numeric inputs. Our song metrics energy and tempo were numeric, of course. And fortunately, images are just numeric inputs too as they consist of pixels. They have a height, a width, and three channels (red, green, and blue). So in theory, we could directly feed the pixels into a Machine Learning model (ignore for now that there is a spatial element here, which we haven't dealt with before).

However, now we are facing two problems. First, even a small, low-quality 224x224 image consists of more than 150,000 pixels (224x224x3). Remember, we were speaking about a maximum of hundreds of input variables (rarely more than a thousand), but now we suddenly have at least 150,000.

Second, if you think about the relationship between the raw pixels and the class label, it's incredibly complex, at least from an ML perspective that is. Our human brains have the amazing ability to generally distinguish among tigers, foxes, and cats quite easily. However, if you saw the 150,000 pixels one by one, you would have no idea what the image contains. But this is exactly how a Machine Learning model sees them, so it needs to learn from scratch the mapping or relationship between those raw pixels and the image label, which is not a trivial task.



Sentiment classification example.

Let's consider another type of input-output relationship that is extremely complex — the relationship between a sentence and its sentiment. By sentiment we typically mean the emotion that a sentence conveys, here positive or negative.

Let's formalize the problem setup again: As the input here we have a sequence of words, i.e., a sentence, and the sentiment is our outcome variable. As before, this is a classification task, this time with two possible labels, i.e., positive or negative.

As with the images example discussed earlier, as humans we understand this relationship naturally, but can we teach a Machine Learning model to do the same?

Before answering that, it's again not obvious at the start how words can be turned into numeric inputs for a Machine Learning model. In fact, this is a level or two more complicated than what we saw with images, which as we saw are essentially already numeric. This is not the case with words. We won't go into details here, but what you need to know is that every word can be turned into a word embedding.

In short, a word embedding represents the word's semantic and syntactic meaning, often within a specific context. These embeddings can be obtained as part of training the Machine Learning model, or by means of a separate training procedure.

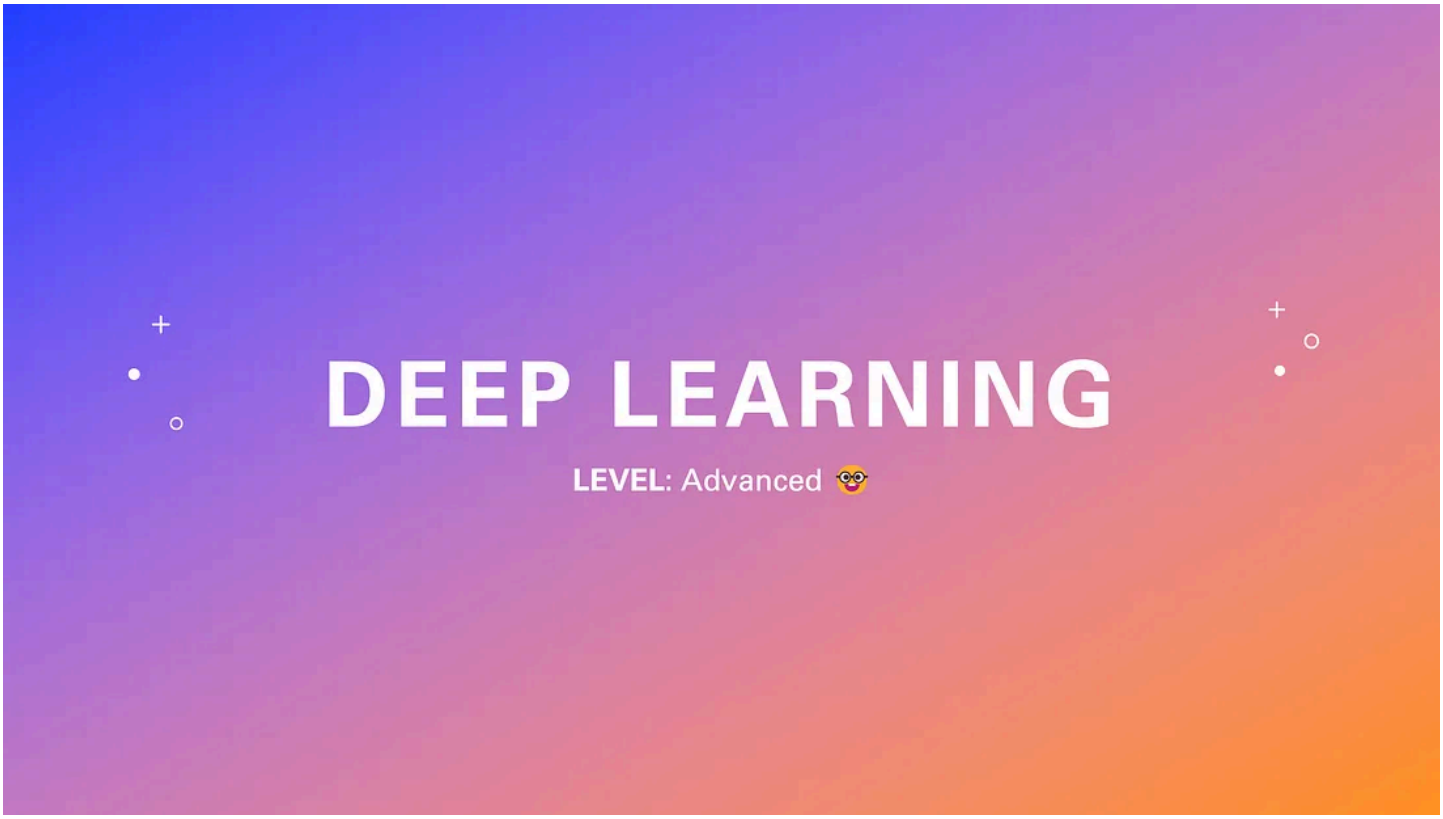
Usually, word embeddings consist of between tens and thousands of variables, per word that is.

To summarize, what to take away from here is that we can take a sentence and turn it into a sequence of numeric inputs, i.e., the word embeddings, which contain semantic and syntactic meaning. This can then be fed into a Machine Learning model. (Again, if you're observant you may notice that there is a new sequential dimension that is unlike our examples from before, but we will ignore this one here too.)

Great, but now we face the same challenges as with the visual input. As you can imagine, with a long sentence (or paragraph or even a whole document), we can quickly reach a very large number of inputs because of the large size of the word embeddings.

The second problem is the relationship between language and its sentiment, which is complex — very complex. Just think of a sentence like "That was a great fall" and all the ways it can be interpreted (not to mention sarcastically).

What we need is an extremely powerful Machine Learning model, and lots of data. That's where Deep Learning comes in.

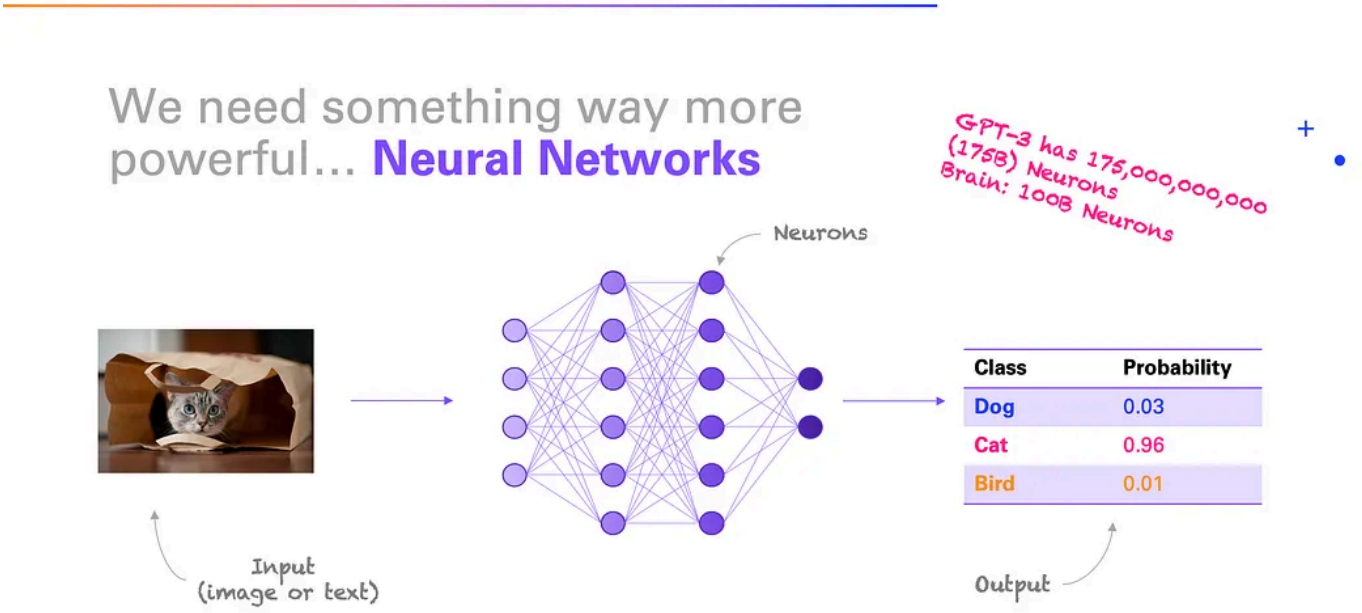


Deep Learning. Level: Advanced.

We already took a major step toward understanding LLMs by going through the basics of Machine Learning and the motivations behind the use of more powerful models, and now we'll take another big step by introducing Deep Learning.

We talked about the fact that if the relationship between an input and output is very complex, as well as if the number of input or output variables is large (and both are the case for our image and language examples from before), we need more flexible, powerful models. A linear model or anything close to that will simply fail to solve these kinds of visual or sentiment classification tasks.

This is where neural networks come in.



Neural Networks are the most powerful Machine Learning models we have today.

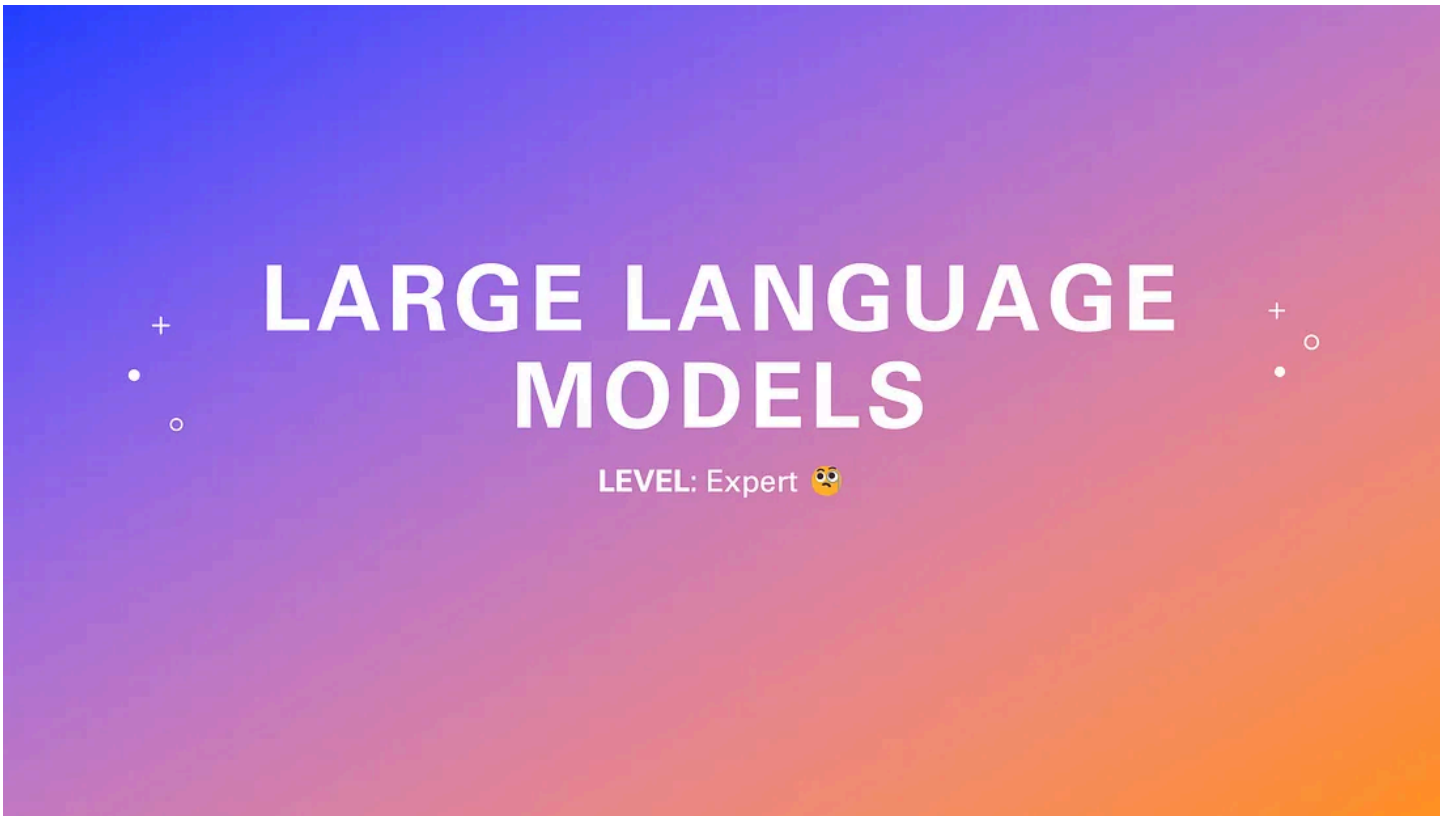
Neural networks are powerful Machine Learning models that allow arbitrarily complex relationships to be modeled. They are the engine that enables learning such complex relationships at massive scale.

In fact, neural networks are loosely inspired by the brain, although the actual similarities are debatable. Their basic architecture is relatively simple. They consist of a sequence of layers of connected "neurons" that an input signal passes through in order to predict the outcome variable. You can think of them as multiple layers of

linear regression stacked together, with the addition of non-linearities in between, which allows the neural network to model highly non-linear relationships.

Neural networks are often many layers deep (hence the name Deep Learning), which means they can be extremely large. ChatGPT, for example, is based on a neural network consisting of 176 billion neurons, which is more than the approximate 100 billion neurons in a human brain.

So, from here on we will assume a neural network as our Machine Learning model, and take into account that we have also learned how to process images and text.

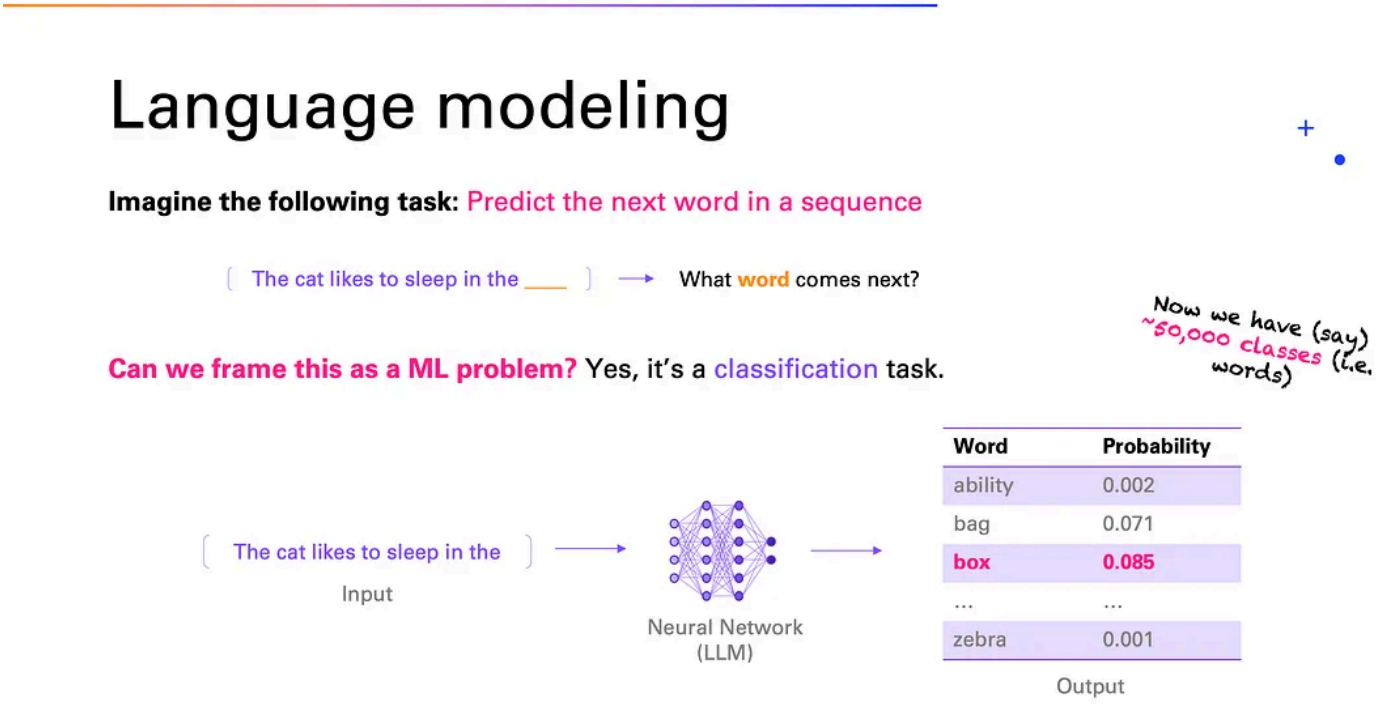


Large Language Models. Level: Expert.

Finally, we can start talking about Large Language Models, and this is where things get really interesting. If you have made it this far, you should have all the knowledge to also understand LLMs.

What's a good way to start? Probably by explaining what *Large Language Model* actually means. We already know what large means, in this case it simply refers to the number of neurons, also called parameters, in the neural network. There is no clear number for what constitutes a Large Language Model, but you may want to consider everything above 1 billion neurons as large.

With that established, what's a "language model"? Let's discuss this next — and just know that in a bit, we'll also get to learn what the GPT in ChatGPT stands for. But one step at a time.



Language modeling is learning to predict the next word.

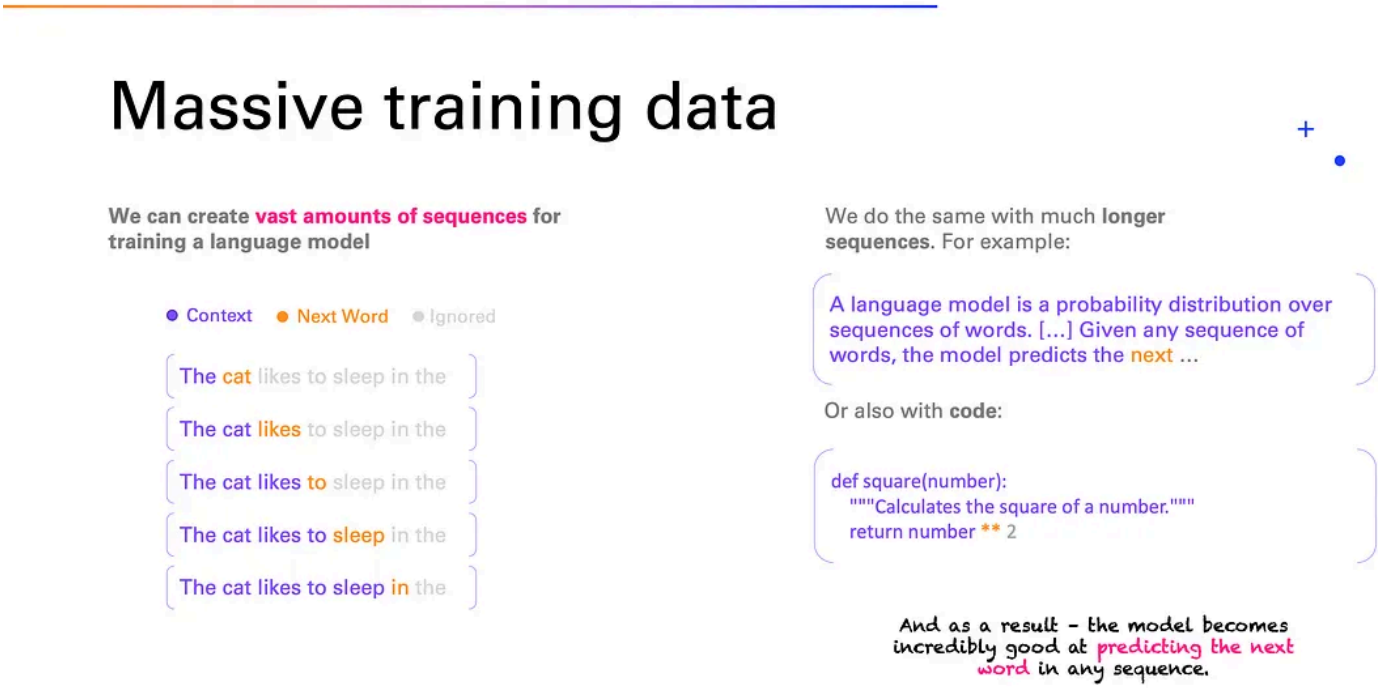
Let's take the following idea and frame it as a Machine Learning problem: What is the next word in a given sequence of words, i.e., in a sentence or paragraph? In other words, we simply want to learn how to predict the next word at any time. From earlier in this article we've learned everything we need to frame that as a Machine Learning problem. In fact, the task is not unlike the sentiment classification we saw earlier.

As in that example, the input to the neural network is a sequence of words, but now, the outcome is simply the next word. Again, this is just a classification task. The only difference is that instead of only two or a few classes, we now have as many classes as there are words — let's say around 50,000. This is what language modeling is about — learning to predict the next word.

Okay, so that's orders of magnitude more complex than the binary sentiment classification, as you can imagine. But now that we also know about neural

networks and their sheer power, the only response to that concern is really “why not?”

> ***Quick disclaimer**: Of course, we are simplifying many things here (as is done throughout the article). In reality things are a little more complex, but that shouldn’t hold us back from understanding the main mechanics, which is why we simplify and leave out some of the details.*



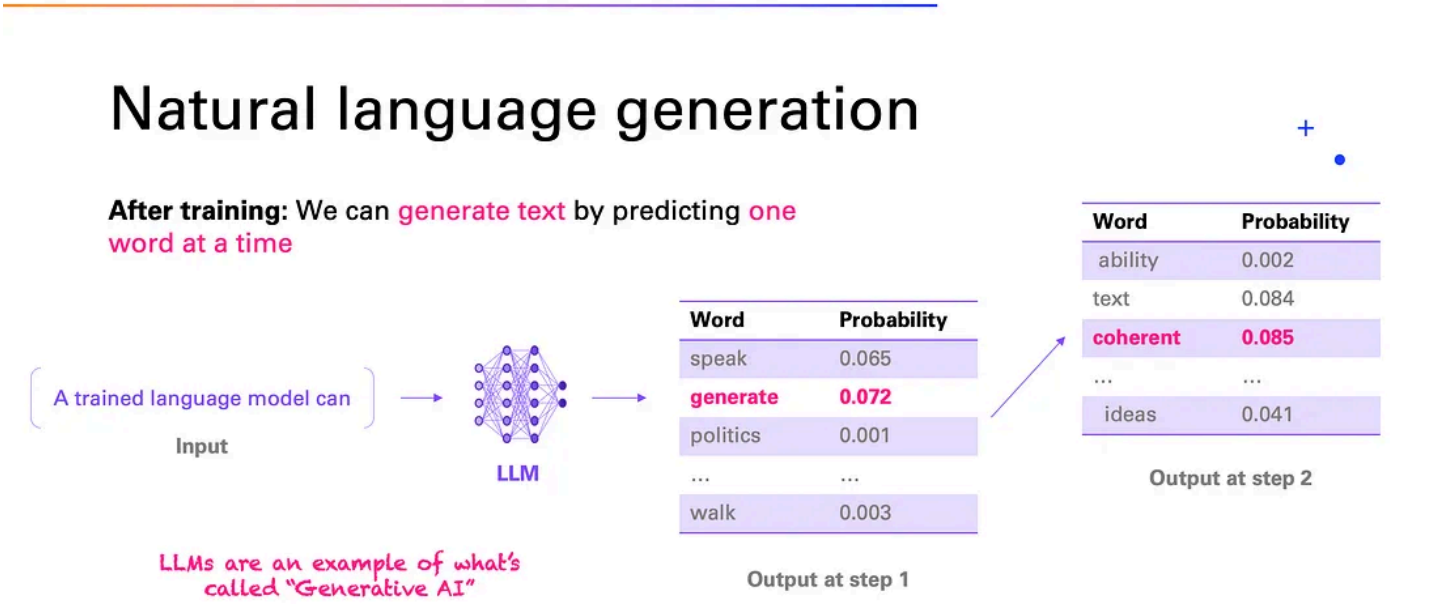
Massive amounts of traning data can be created relatively easily.

We know the task, and now we need data to train the neural network. It’s actually not difficult to create a lot of data for our “next word prediction” task. There’s an abundance of text on the internet, in books, in research papers, and more. And we can easily create a massive dataset from all of this. We don’t even need to label the data, because the next word itself is the label, that’s why this is also called *self-supervised learning*.

The image above shows how this is done. Just a single sequence can be turned into multiple sequences for training. And we have lots of such sequences. Importantly, we do this for many short and long sequences (some up to thousands of words) so that in every context we learn what the next word should be.

To summarize, all we are doing here is to train a neural network (the LLM) to predict the next word in a given sequence of words, no matter if that sequence is long or short, in German or in English or in any other language, whether it's a tweet or a mathematical formula, a poem or a snippet of code. All of those are sequences that we will find in the training data.

If we have a large enough neural network as well as enough data, the LLM becomes really good at predicting the next word. Will it be perfect? No, of course not, since there are often multiple words that can follow a sequence. But it will become good at selecting one of the appropriate words that are syntactically and semantically appropriate.



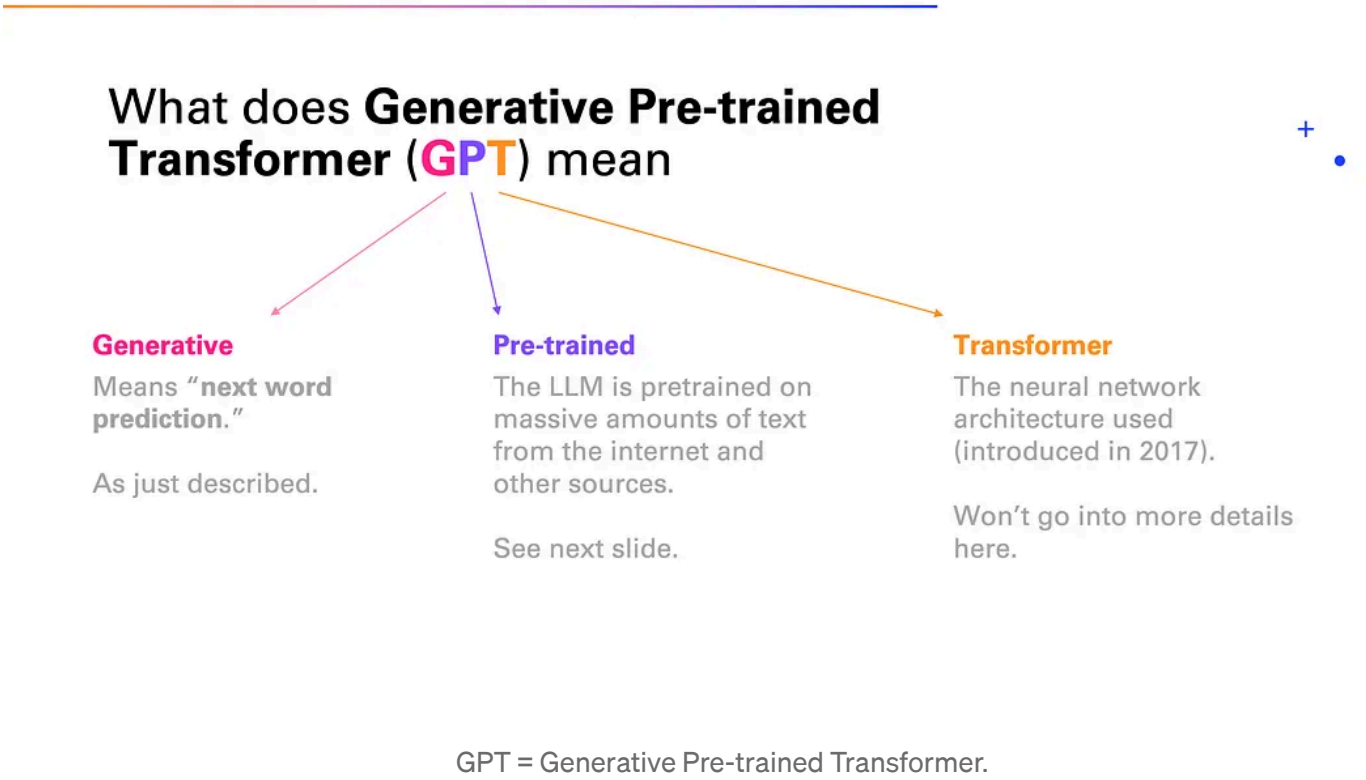
We can perform natural language generation by predicting one word at a time.

Now that we can predict one word, we can feed the extended sequence back into the LLM and predict another word, and so on. In other words, using our trained LLM, we can now generate text, not just a single word. This is why LLMs are an example of what we call Generative AI. We have just taught the LLM to speak, so to say, one word at a time.

There's one more detail to this that I think is important to understand. We don't necessarily always have to predict the most likely word. We can instead sample

from, say, the five most likely words at a given time. As a result, we may get some more creativity from the LLM. Some LLMs actually allow you to choose how deterministic or creative you want the output to be. This is also why in ChatGPT, which uses such a sampling strategy, you typically do not get the same answer when you regenerate a response.

Speaking of ChatGPT, you could ask yourself now why it's not called ChatLLM. As it turns out, language modeling is not the end of the story — in fact it's just the beginning. So what does the GPT in ChatGPT stand for?



GPT = Generative Pre-trained Transformer.

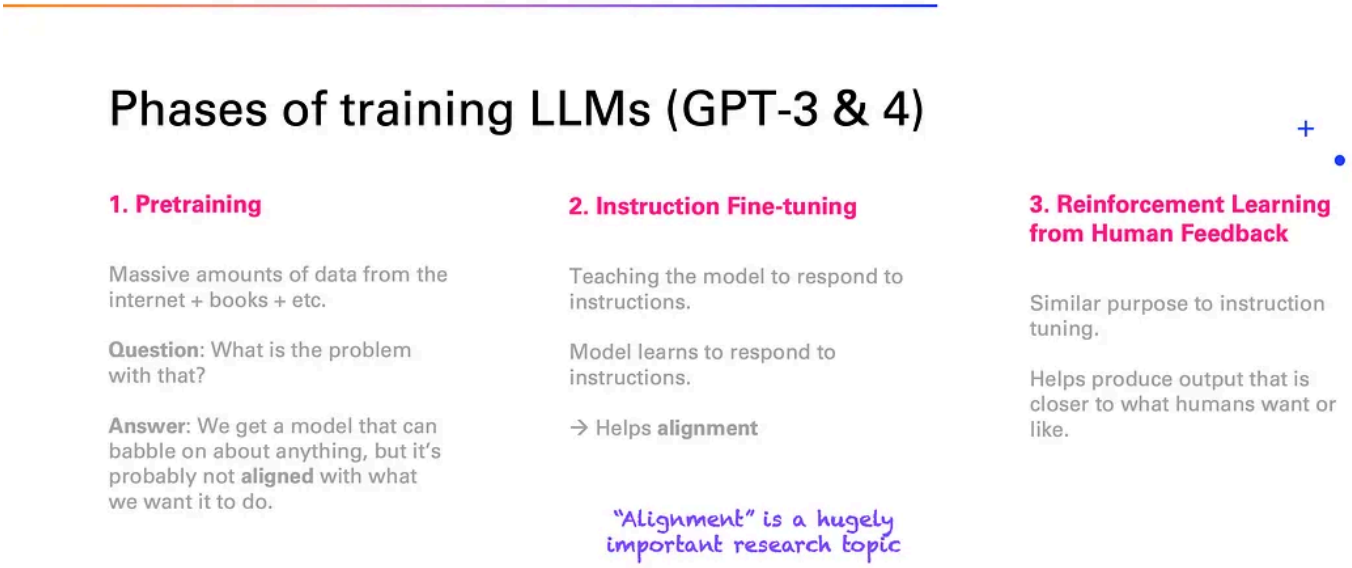
We have actually just learned what the G stands for, namely "generative" — meaning that it was trained on a language generation pretext, which we have discussed. But what about the P and the T?

We'll gloss over the T here, which stands for "transformer" — not the one from the movies (sorry), but one that's simply the type of neural network architecture that is being used. This shouldn't really bother us here, but if you are curious and you only want to know its main strength, it's that the transformer architecture works so well because it can focus its attention on the parts of the input sequence that are most relevant at any time. You could argue that this is similar to how humans work. We,

too, need to focus our attention on what's most relevant to the task and ignore the rest.

Now to the P, which stands for "pre-training". We discuss next why we suddenly start speaking about pre-training and not just training any longer.

The reason is that Large Language Models like ChatGPT are actually trained in phases.



Phases of LLM training: (1) Pre-Training, (2) Instruction Fine-Tuning, (3) Reinforcement from Human Feedback (RLHF).

Pre-training

The first stage is pre-training, which is exactly what we've gone through just now. This stage requires massive amounts of data to learn to predict the next word. In that phase, the model learns not only to master the grammar and syntax of language, but it also acquires a great deal of knowledge about the world, and even some other emerging abilities that we will speak about later.

But now I have a couple of questions for you: First, what might be the problem with this kind of pre-training? Well, there are certainly a few, but the one I am trying to point to here has to do with what the LLM has really learned.

Namely, it has learned mainly to ramble on about a topic. It may even be doing an incredibly good job, but what it doesn't do is respond well to the kind of inputs you would generally want to give an AI, such as a question or an instruction. The problem is that this model has not learned to be, and so is not behaving as, an assistant.

For example, if you ask a pre-trained LLM "What is your fist name?" it may respond with "What is your last name?" simply because this is the kind of data it has seen during pre-training, as in many empty forms, for example. It's only trying to complete the input sequence.

It doesn't do well with following instructions simply because this kind of language structure, i.e., instruction followed by a response, is not very commonly seen in the training data. Maybe Quora or StackOverflow would be the closest representation of this sort of structure.

At this stage, we say that the LLM is not aligned with human intentions. Alignment is an important topic for LLMs, and we'll learn how we can fix this to a large extent, because as it turns out, those pre-trained LLMs are actually quite steerable. So even though initially they don't respond well to instructions, they can be taught to do so.

Instruction fine-tuning and RLHF

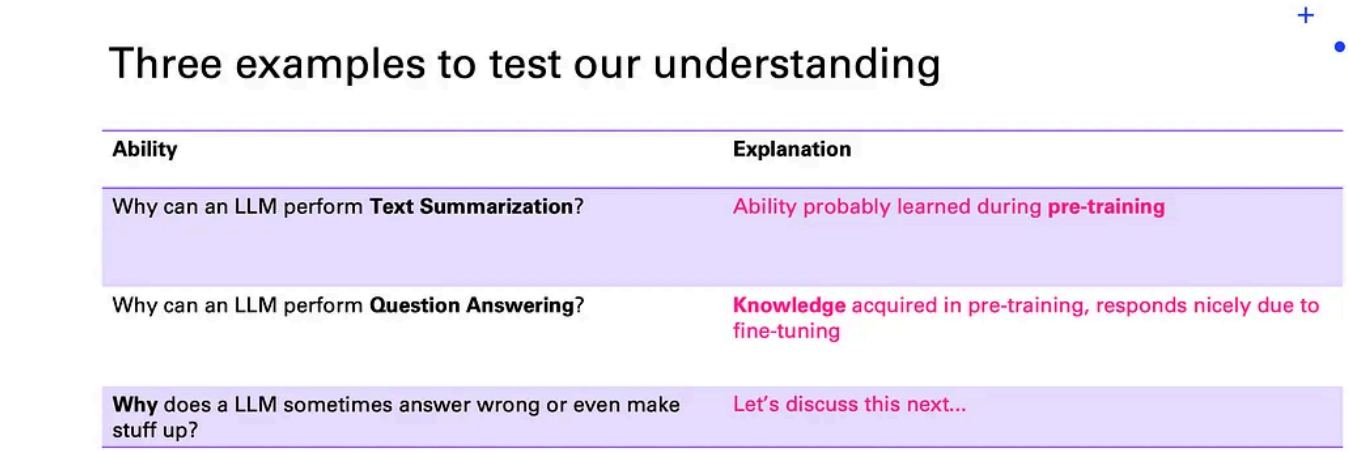
This is where instruction tuning comes in. We take the pre-trained LLM with its current abilities and do essentially what we did before — i.e., learn to predict one word at a time — but now we do this using only high-quality instruction and response pairs as our training data.

That way, the model un-learns to simply be a text completer and learns to become a helpful assistant that follows instructions and responds in a way that is aligned with the user's intention. The size of this instruction dataset is typically a lot smaller than the pre-training set. This is because the high-quality instruction-response pairs are much more expensive to create as they are typically sourced from humans. This is very different from the inexpensive self-supervised labels we used in pre-training. This is why this stage is also called *supervised instruction fine-tuning*.

There is also a third stage that some LLMs like ChatGPT go through, which is reinforcement learning from human feedback (RLHF). We won't go into details here, but the purpose is similar to instruction fine-tuning. RLHF also helps alignment and ensures that the LLM's output reflects human values and

preferences. There is some early research that indicates that this stage is critical for reaching or surpassing human-level performance. In fact, combining the fields of reinforcement learning and language modeling is being shown to be especially promising and is likely to lead to some massive improvements over the LLMs we currently have.

So now let's test our understanding on some common use cases.



Examples to test our understanding of LLMs.

First, **why can an LLM perform summarization** of a longer piece of text? (If you didn't already know, it does a really great job. Just paste in a document and ask it to summarize it.)

To understand why, we need to think about the training data. As it so happens, people often make summarizations — on the internet, in research papers, books, and more. As a result, an LLM trained on that data learns how to do that too. It learns to attend to the main points and compress them into a short text.

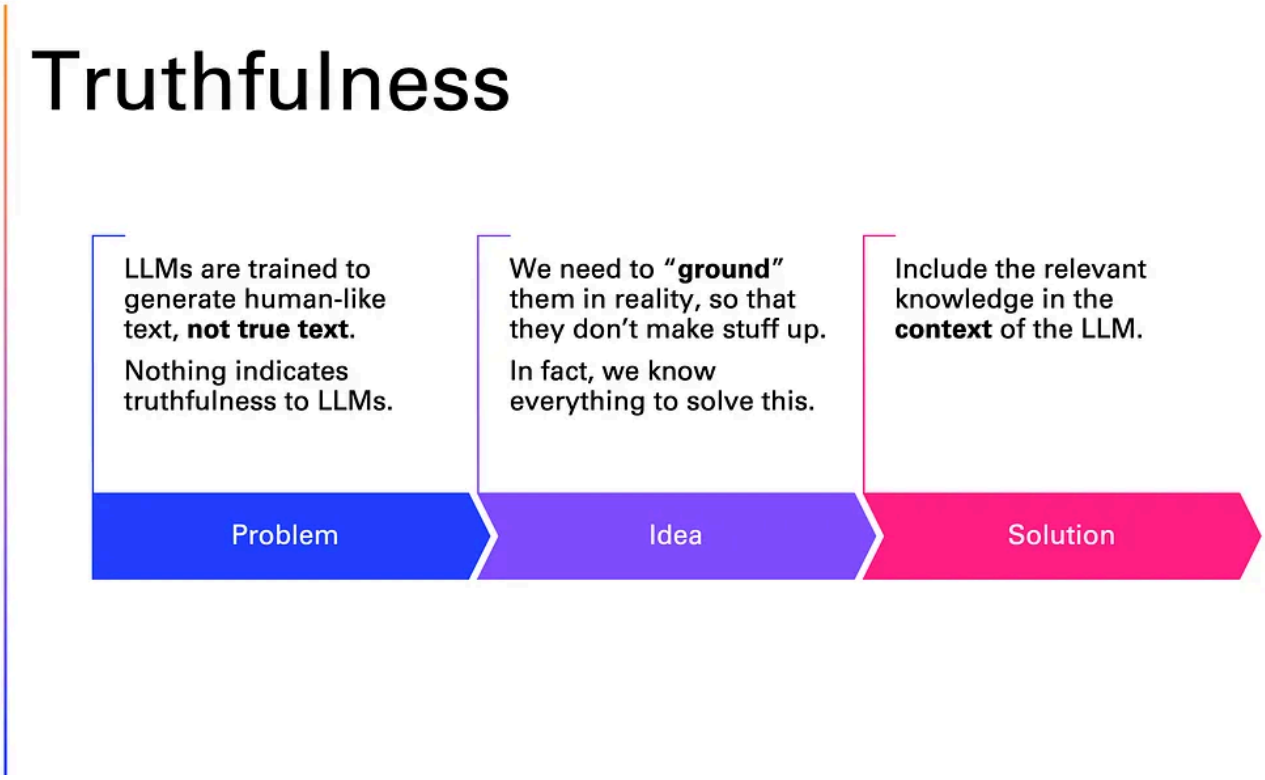
Note that when a summary is generated, the full text is part of the input sequence of the LLM. This is similar to, say, a research paper that has a conclusion while the full text appears just before.

As a result, that skill has probably been learned during pre-training already, although surely instruction fine-tuning helped improve that skill even further. We can assume that this phase included some summarization examples too.

Second, **why can a LLM answer common knowledge questions**?

As mentioned, the ability to act as an assistant and respond appropriately is due to instruction fine-tuning and RLHF. But all (or most of) the knowledge to answer questions itself was already acquired during pre-training.

Of course, that now raises another big question: **What if the LLM doesn't know the answer**? Unfortunately, it may just make one up in that case. To understand why, we need to think about the training data again, and the training objective.



LLMs suffer from hallucinations, but this can be mitigated by providing additional context.

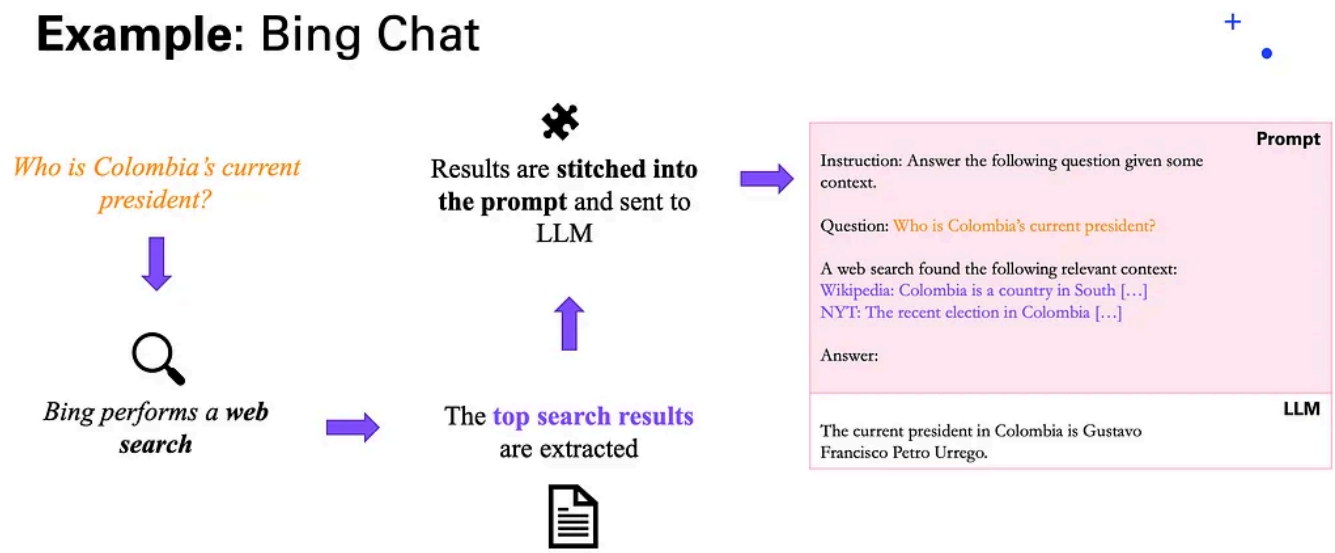
You might have heard about the term "hallucination" in the context of LLMs, which refers to the phenomenon of LLMs making up facts when they shouldn't.

Why does that happen? Well, the LLM learns only to generate text, not factually true text. Nothing in its training gives the model any indicator of the truth or reliability of any of the training data. However, that is not even the main issue here, it's that generally text out there on the internet and in books sounds confident, so the LLM

of course learns to sound that way, too, even if it is wrong. In this way, an LLM has little indication of uncertainty.

That being said, this is an active area of research, from which we can expect that LLMs will be less prone to hallucinations over time. For example, during instruction tuning we can try and teach the LLM to abstain from hallucinating to some extent, but only time will tell whether we can fully solve this issue.

You may be surprised that we can actually try to solve this problem here together right now. We have the knowledge we need to figure out a solution that at least partially helps and is already used widely today.



Bing chat is an example of a search-based LLM workflow.

Suppose that you ask the LLM the following question: Who is the current president of Colombia? There's a good chance an LLM may respond with the wrong name. This could be because of two reasons:

- The first is what we have already brought up: The LLM may just hallucinate and simply respond with a wrong or even fake name.
- The second one I will mention only in passing: LLMs are trained only on data up to a certain cut-off date, and that can be as early as last year. Because of that, the

> LLM cannot even know the current president with certainty, because things could have changed since the data was created.

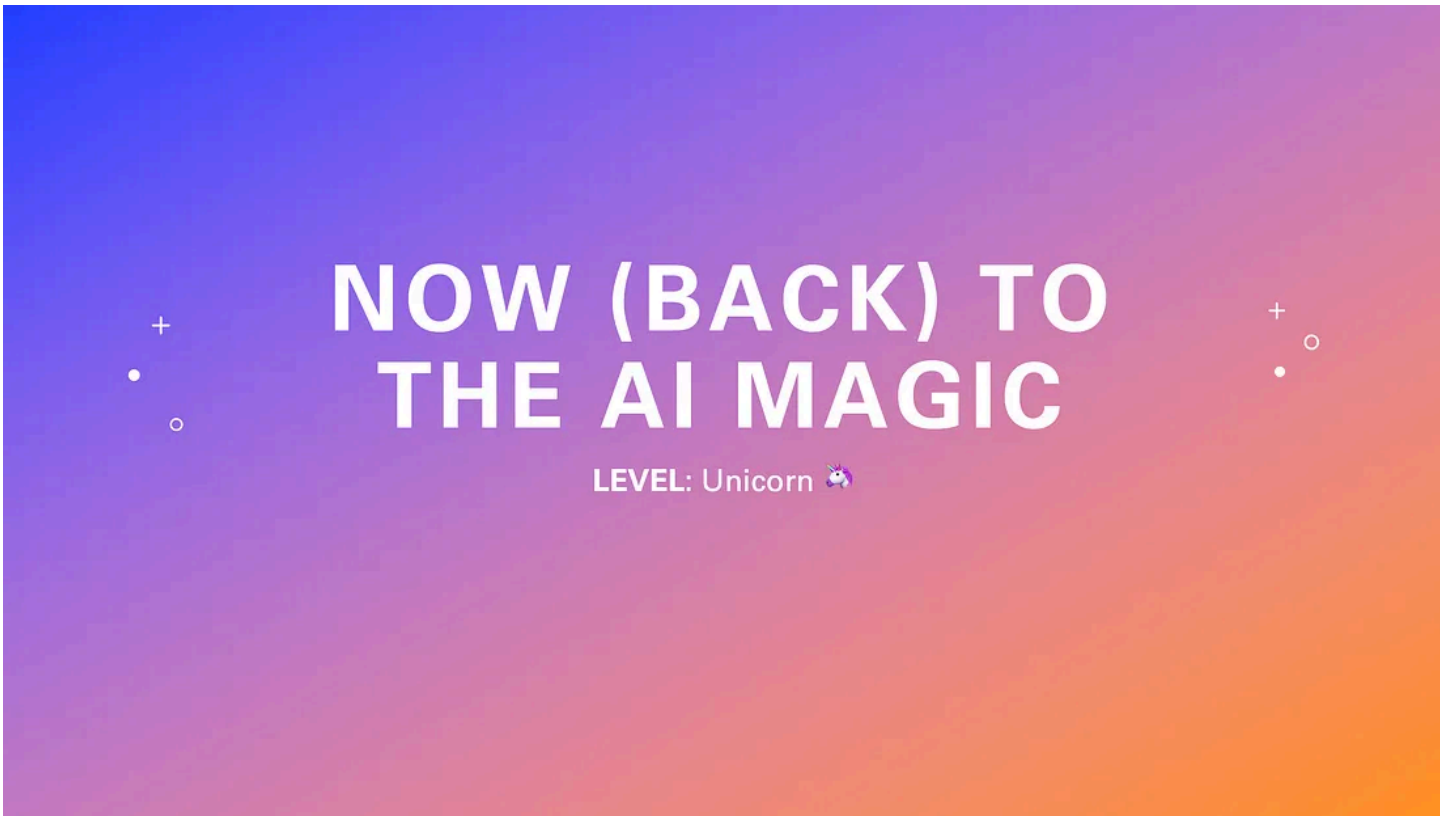
So how can we solve both these problems? The answer lies in providing the model some relevant context. The rationale here is that everything that's in the LLM's input sequence is readily available for it to process, while any implicit knowledge it has acquired in pre-training is more difficult and precarious for it to retrieve.

Suppose we were to include the Wikipedia article on Colombia's political history as context for the LLM. In that case it would much more likely to answer correctly because it can simply extract the name from the context (given that it is up to date and includes the current president of course).

In the image above you can see what a typical prompt for an LLM with additional context may look like. (By the way, prompt is just another name for the instructions we give to an LLM, i.e., the instructions form the input sequence.)

This process is called grounding the LLM in the context, or in the real world if you like, rather than allowing it to generate freely.

And that's exactly how Bing Chat and other search-based LLMs work. They first extract relevant context from the web using a search engine and then pass all that information to the LLM, alongside the user's initial question. See the illustration above for a visual of how this is accomplished.



Back to the AI Magic. Level: Unicorn.

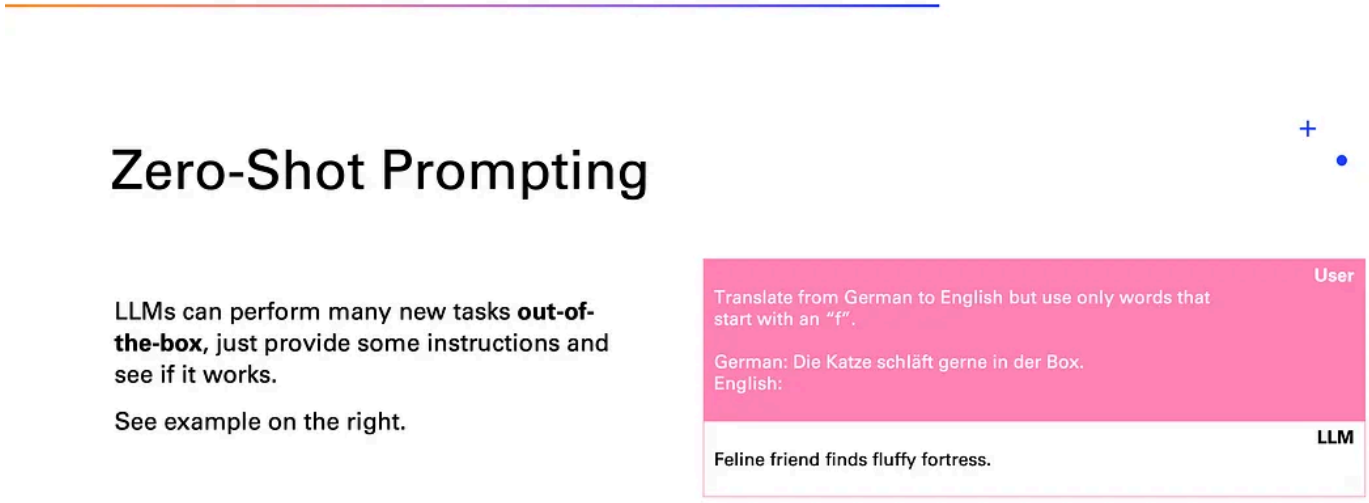
We've now reached a point where you pretty much understand the main mechanisms of the state-of-the art LLMs (as of the second half of 2023, anyway).

You may be thinking "this is actually not that magical" because all that is happening is the predicting of words, one at a time. It's pure statistics, after all. Or is it?

Let's back up a bit. The magical part of all this is how remarkably well it works. In fact, everyone, even the researchers at OpenAI, were surprised at how far this sort of language modeling can go. One of the key drivers in the last few years has simply been the massive scaling up of neural networks and data sets, which has caused performance to increase along with them. For example, GPT-4, reportedly a model with more than one trillion parameters in total, can pass the bar exam or AP Biology with a score in the top 10 percent of test takers.

Surprisingly, those large LLMs even show certain **emerging abilities**, i.e., abilities to solve tasks and to do things that they were not explicitly trained to do.

In this last part of the article, we'll discuss some of these emerging abilities and I'll show you some tricks for how you can use them to solve problems.

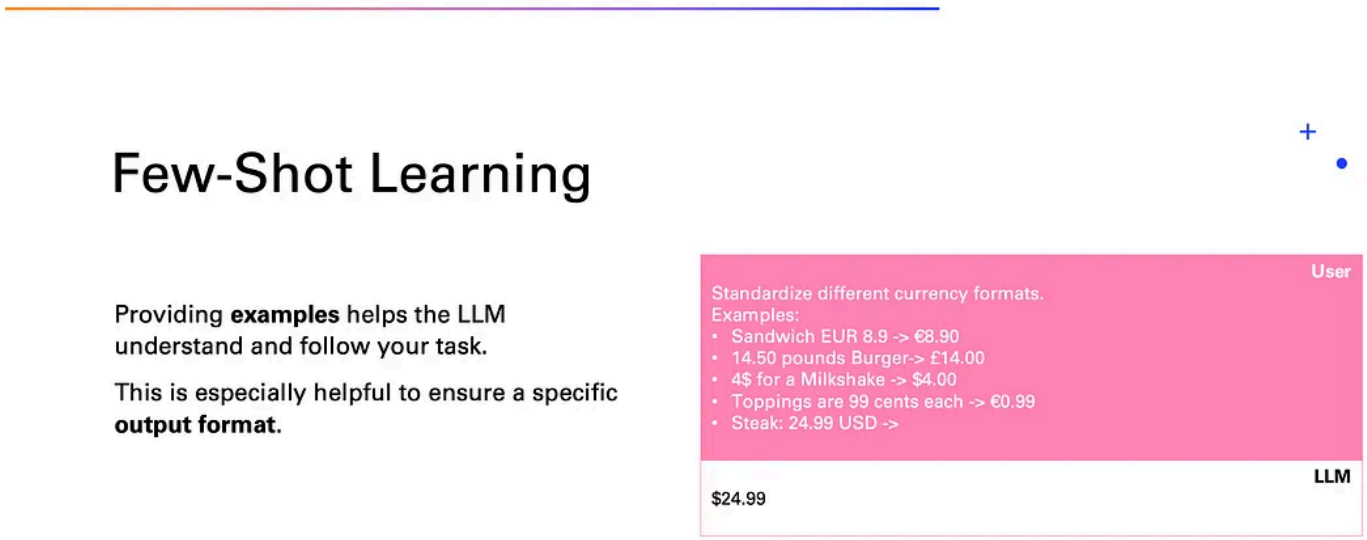


LLMs can solve entirely new tasks in a zero-shot manner.

A ubiquitous emerging ability is, just as the name itself suggests, that LLMs can perform entirely new tasks that they haven't encountered in training, which is called zero-shot. All it takes is some instructions on how to solve the task.

To illustrate this ability with a silly example, you can ask an LLM to translate a sentence from German to English while responding only with words that start with "f".

For instance, when asked to translate a sentence using only words that start with "f", an LLM translated "Die Katze schläft gerne in der Box" (which is German and literally means "The cat likes to sleep in the box") with "Feline friend finds fluffy fortress", which is a pretty cool translation, I think.



LLMs, just like humans, can benefit from providing them with examples or demonstrations.

For more complex tasks, you may quickly realize that zero-shot prompting often requires very detailed instructions, and even then, performance is often far from perfect.

To make another connection to human intelligence, if someone tells you to perform a new task, you would probably ask for some examples or demonstrations of how the task is performed. LLMs can benefit from the same.

As an example, let's say you want a model to translate different currency amounts into a common format. You could describe what you want in details or just give a brief instruction and some example demonstrations. The image above shows a sample task.

Using this prompt, the model should do well on the last example, which is "Steak: 24.99 USD", and respond with $24.99.

Note how we simply left out the solution to the last example. Remember that an LLM is still a text-completer at heart, so keep a consistent structure. You should almost force the model to respond with just what you want, as we did in the example above.

To summarize, a general tip is to provide some examples if the LLM is struggling with the task in a zero-shot manner. You will find that often helps the LLM understand the task, making the performance typically better and more reliable.

Chain-of-thought provides LLMs a working memory, which can improve their performance substantially, especially on more complex tasks.

Another interesting ability of LLMs is also reminiscent of human intelligence. It is especially useful if the task is more complex and requires multiple steps of reasoning to solve.

Let's say I ask you "Who won the World Cup in the year before Lionel Messi was born?" What would you do? You would probably solve this step by step by writing down any intermediate solutions needed in order to arrive at the correct answer. And that's exactly what LLMs can do too.

It has been found that simply telling an LLM to "think step by step" can increase its performance substantially in many tasks.

Why does this work? We know everything we need to answer this. The problem is that this kind of unusual composite knowledge is probably not directly in the LLM's internal memory. However, all the individual facts might be, like Messi's birthday, and the winners of various World Cups.

Allowing the LLM to build up to the final answer helps because it gives the model time to think out loud — a working memory so to say — and to solve the simpler sub-problems before giving the final answer.

The key here is to remember that everything to the left of a to-be-generated word is context that the model can rely on. So, as shown in the image above, by the time the model says "Argentina", Messi's birthday and the year of the Word Cup we inquired about are already in the LLM's working memory, which makes it easier to answer correctly.

Conclusion

Before I wrap things up, I want to answer a question I asked earlier in the article. Is the LLM really just predicting the next word or is there more to it? Some researchers are arguing for the latter, saying that to become so good at next-word-prediction in

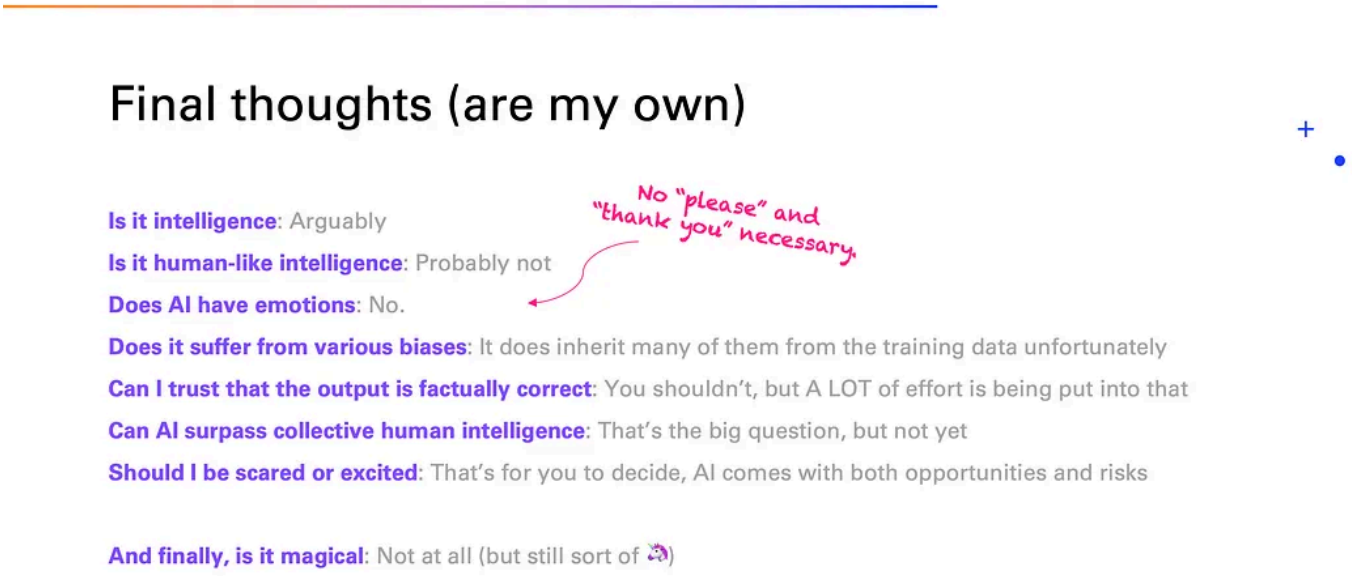
language, the world, or anything else.

There is probably no clear right or wrong between those two sides at this point; it may just be a different way of looking at the same thing. Clearly these LLMs are proving to be very useful and show impressive knowledge and reasoning capabilities, and maybe even show some sparks of general intelligence. But whether or to what extent that resembles human intelligence is still to be determined, and so is how much further language modeling can improve the state of the art.

I hope that this article helps you understand LLMs and the current craze that is surrounding them, so that you can form your own opinion about AI's potentials and risks. It's not only up to AI researchers and data scientists to decide how AI is used to benefit the world; everyone should be able to have a say. This is why I wanted to write an article that doesn't require a lot of background knowledge.

If you made it through this article, I think you pretty much know how some the state-of-the-art LLMs work (as of Autumn 2023), at least at a high level.

I'll leave you with some of my final thoughts on the current state of Artificial Intelligence and LLMs.



Final thoughts on the state-of-the-art in Artificial Intelligence.

Thank you for reading. If you have any questions, feel free to contact me on LinkedIn. Thank you also to Casey Doyle for the edits and suggestions.

Large Language Models | NLP | Data Science | Machine Learning | Deep Learning

Follow