Andrej Karpathy: Deep Dive into LLMs like ChatGPT

Summary

This comprehensive video provides an accessible yet in-depth introduction to large language models (LLMs) such as ChatGPT, explaining how they are built, trained, and fine-tuned to become useful AI assistants. The presenter breaks down the entire pipeline—from collecting and processing vast internet text data, through tokenization and neural network training, to post-training techniques like supervised fine-tuning (SFT) and reinforcement learning (RL). Key cognitive and operational aspects of LLMs are explored, including their strengths, limitations, hallucinations, and emergent reasoning capabilities. The video also covers the evolving landscape of LLM applications, multimodality, tool use, and future directions, while offering practical insights into where to find and interact with these models.

The video begins by detailing the **pre-training stage**, where massive datasets (like the FineWeb dataset derived from Common Crawl web pages) are collected, filtered, and tokenized. It explains tokenization, especially byte pair encoding, to convert raw text into sequences of tokens the neural networks can process. The neural network architecture, primarily Transformers, is described in terms of inputs, parameters, and outputs, illustrating how the model learns to predict the next token in a sequence by adjusting billions of parameters.

Next, the video discusses **neural network training**—sampling token windows, predicting next tokens, calculating loss, and updating parameters. It emphasizes the computational intensity of this stage, requiring large GPU clusters (e.g., Nvidia H100 GPUs) and substantial cloud infrastructure, which explains the high costs and scale of modern LLM training.

After pre-training, the **post-training stage** (supervised fine-tuning) is introduced to convert a base model, which simulates internet text, into a helpful assistant. This involves training on carefully curated conversation datasets created by human labelers following detailed guidelines to generate ideal assistant responses. The model learns to imitate these responses, effectively learning a “personality” and useful behavior.

The video highlights **hallucinations**—situations where LLMs confidently generate false or fabricated information—and explains mitigation strategies. These include training the model to recognize when it should admit ignorance and integrating external tools such as web search or code interpreters to retrieve or verify factual information dynamically. This tool use significantly improves factual accuracy by augmenting the model’s vague internal knowledge with fresh, contextual data.

The presenter then explores the **limitations and quirks** of LLMs, such as difficulties with counting, spelling, and simple arithmetic, due to token-level processing and

limited per-token computation. It is emphasized that models have “Swiss cheese” capabilities—being excellent in many areas but prone to occasional, unpredictable errors.

The final major training stage is **reinforcement learning (RL)**, where the model practices generating solutions to problems, receives feedback on correctness, and iteratively improves its problem-solving strategies. This stage is likened to “practice” after learning “exposition” (pre-training) and “imitation of experts” (fine-tuning). RL enables models to develop emergent reasoning abilities, such as step-by-step problem solving and self-correction, which are not explicitly programmed but arise naturally through iterative optimization.

The video also discusses **reinforcement learning from human feedback (RLHF)**, a technique allowing RL to be applied in domains where objective scoring is difficult (e.g., creative writing). Here, humans rank model outputs, and a separate “reward model” is trained to simulate human preferences, enabling scalable RL training. However, RLHF has limitations, such as susceptibility to adversarial examples that “game” the reward model, necessitating careful management and early stopping.

Looking forward, the video anticipates rapid advances in **multimodal models** that handle text, audio, and images simultaneously, as well as longer-term AI agents capable of coordinating complex, multi-step tasks over extended periods with human supervision. The presenter concludes by recommending resources to track model progress and access models, emphasizing that despite their power, LLMs remain stochastic tools with limitations and should be used responsibly as aids rather than infallible oracles.

Highlights

- 🤖 Comprehensive overview of how large language models like ChatGPT are built and trained.
- 📚 Pre-training involves massive datasets from the internet, careful filtering, and tokenization into sequences for neural networks.
- ⚙️ Neural networks (Transformers) learn to predict next tokens by adjusting billions of parameters through intensive GPU-powered training.
- 🗣️ Post-training fine-tunes base models on human-labeled conversational data to create helpful AI assistants with personalities.
- 🧠 Hallucinations remain a challenge; mitigated by training models to admit ignorance and by integrating external tools like web search and code interpreters.

- 🧮 Models have cognitive quirks—good at many tasks but prone to errors in counting, spelling, and simple arithmetic due to tokenization and limited per-token compute.
- 🎯 Reinforcement learning helps models practice and improve reasoning, discovering emergent problem-solving strategies beyond supervised learning.

Key Insights

- 🔍 **Data Curation is Crucial:** The quality and diversity of the pre-training dataset (e.g., FineWeb) significantly impact model capabilities. Filtering out low-quality or harmful content ensures better generalization and reduces biases, though language distribution choices affect multilingual performance. This highlights the tension between dataset scale and quality.
- 🧩 **Tokenization Balances Efficiency and Expressiveness:** Byte pair encoding compresses text into manageable token sequences, balancing vocabulary size (~100,000 tokens) and sequence length. This tokenization strategy is a critical engineering choice affecting model size, speed, and ability to represent complex language structures.
- ⚡ **Training Neural Networks is Computationally Intensive:** Training requires specialized hardware (e.g., Nvidia H100 GPUs) in large clusters, costing millions of dollars. This infrastructure arms race fuels the rapid progress in LLM capabilities, demonstrating that breakthroughs rely as much on engineering and compute as on algorithms.
- 🛠️ **Fine-Tuning Shapes Model Behavior:** The post-training stage with human labelers "programs" the assistant's personality and response style by example, transforming a base model that merely simulates internet text into a useful interactive agent. This human-in-the-loop process is key to practical LLM deployment.
- 🌫️ **Hallucinations Stem from Statistical Imitation:** LLMs generate plausible-sounding but false outputs because they imitate the style and distribution of training data rather than verify facts. Mitigations like training models to admit ignorance and enabling tool use reflect a pragmatic approach to overcoming intrinsic model limitations.
- 🧠 **Reinforcement Learning Unlocks Emergent Reasoning:** By practicing problem-solving with feedback, models develop reasoning strategies such as chains of thought and self-correction that surpass mere imitation of training data. This mirrors human learning stages of exposition, imitation, and practice, suggesting a natural learning paradigm for AI.

- ⚠️ **RLHF Introduces New Challenges:** Using reward models to approximate human preferences enables scalable training in creative domains but risks “gaming” and adversarial exploits. This underscores the complexity of aligning AI behavior with human values and the continuing need for research into robust training methods.

This video serves as a foundational guide for understanding the inner workings, training methods, capabilities, and limitations of today’s large language models, as well as the exciting directions AI research is heading toward. It encourages a nuanced perspective: appreciating the power of LLMs while recognizing their imperfections and the critical role of human supervision and ongoing development.

hi everyone so I've wanted to make this video for a while it is a comprehensive but General audience introduction to large language models like Chachi PT and what I'm hoping to achieve in this video is to give you kind of mental models for thinking through what it is that this tool is it is obviously magical and amazing in some respects it's uh really good at some things not very good at other things and there's also a lot of sharp edges to be aware of so what is behind this text box you can put

00:30

anything in there and press enter but uh what should we be putting there and what are these words generated back how does this work and what what are you talking to exactly so I'm hoping to get at all those topics in this video we're going to go through the entire pipeline of how this stuff is built but I'm going to keep everything uh sort of accessible to a general audience so let's take a look at first how you build something like chpt and along the way I'm going to talk about um you know some of the sort of

00:57

cognitive psychological implications of the tools okay so let's build Chachi PT so there's going to be multiple stages arranged sequentially the first stage is called the pre-training stage and the first step of the pre-training stage is to download and process the internet now to get a sense of what this roughly looks like I recommend looking at this URL here so um this company called hugging face uh collected and created and curated this data set called Fine web and they go into a lot of detail on

01:29

this block post on how how they constructed the fine web data set and all of the major llm providers like open AI anthropic and Google and so on will have some equivalent internally of something like the fine web data set so roughly what are we trying to achieve here we're trying to get ton of text from the internet from publicly available sources so we're trying to have a huge quantity of very high quality documents and we also want very large diversity of documents because we want to have a lot of knowledge inside

01:57

these models so we want large diversity of high quality documents and we want many many of them and achieving this is uh quite complicated and as you can see here takes multiple stages to do well so let's take a look at what some of these stages look like in a bit for now I'd like to just like to note that for example the fine web data set which is fairly representative what you would see in a production grade application actually ends up being only about 44 terabyt of dis space um you can get a

02:24

USB stick for like a terabyte very easily or I think this could fit on a single hard drive almost today so this is not a huge amount of data at the end of the day even though the internet is very very large we're working with text and we're also filtering it aggressively so we end up with about 44 terabytes in this example so let's take a look at uh kind of what this data looks like and what some of these stages uh also are so the starting point for a lot of these efforts and something that contributes

02:50

most of the data by the end of it is Data from common crawl so common craw is an organization that has been basically scouring the internet since 2007 so as of 2024 for example common CW has indexed 2.7 billion web pages uh and uh they have all these crawlers going around the internet and what you end up doing basically is you start with a few seed web pages and then you follow all the links and you just keep following links and you keep indexing all the information and you end up with a ton of data of the internet

03:19

over time so this is usually the starting point for a lot of the uh for a lot of these efforts now this common C data is quite raw and is filtered in many many different ways so here they Pro they document this is the same diagram they document a little bit the kind of processing that happens in these stages so the first thing here is something called URL filtering so what that is referring to is that there's these block lists of uh basically URLs that are or domains that uh you don't want to be

03:53

getting data from so usually this includes things like U malware websites spam websites marketing websites uh racist websites adult sites and things like that so there's a ton of different types of websites that are just eliminated at this stage because we don't want them in our data set um the second part is text extraction you have to remember that all these web pages this is the raw HTML of these web pages that are being saved by these crawlers so when I go to inspect here this is what the raw HTML actually

04:21

looks like you'll notice that it's got all this markup uh like lists and stuff like that and there's CSS and all this kind of stuff so this is um computer code almost for these web pages but what we really want is we just want this text right we just want the text of this web page and we don't want the navigation and things like that so there's a lot of filtering and processing uh and heris that go into uh adequately filtering for just their uh good content of these web pages the next stage here is language

04:51

filtering so for example fine web filters uh using a language classifier they try to guess what language every single web page is in and then they only keep web pages that have more than 65% of English as an example and so you can get a sense that this is like a design decision that different companies can uh can uh take for themselves what fraction of all different types of languages are we going to include in our data set because for example if we filter out all of the Spanish as an example then you might

05:19

imagine that our model later will not be very good at Spanish because it's just never seen that much data of that language and so different companies can focus on multilingual performance to uh to a different degree as an example so fine web is quite focused on English and so their language model if they end up training one later will be very good at English but not may be very good at other languages after language filtering there's a few other filtering steps and D duplication and things like that um

05:47

finishing with for example the pii removal this is personally identifiable information so as an example addresses Social Security numbers and things like that you would try to detect them and you would try to filter out those kinds of web pages from the the data set as well so there's a lot of stages here and I won't go into full detail but it is a fairly extensive part of the pre-processing and you end up with for example the fine web data set so when you click in on it uh you can see some

06:13

examples here of what this actually ends up looking like and anyone can download this on the huging phase web page and so here are some examples of the final text that ends up in the training set so this is some article about tornadoes in 2012 um so there's some t tadoes in 2020 in 2012 and what happened uh this next one is something about did you know you have two little yellow 9vt battery sized adrenal glands in your body okay so this is some kind of a odd medical article so just think of these as

06:49

basically uh web pages on the internet filtered just for the text in various ways and now we have a ton of text 40 terabytes off it and that now is the starting point for the next step of this stage now I wanted to give you an intuitive sense of where we are right now so I took the first 200 web pages here and remember we have tons of them and I just take all that text and I just put it all together concatenate it and so this is what we end up with we just get this just just raw text raw internet

07:19

text and there's a ton of it even in these 200 web pages so I can continue zooming out here and we just have this like massive tapestry of Text data and this text data has all these p patterns and what we want to do now is we want to start training neural networks on this data so the neural networks can internalize and model how this text flows right so we just have this giant texture of text and now we want to get neural Nets that mimic it okay now before we plug text into neural networks we have to decide how we're going to

07:53

represent this text uh and how we're going to feed it in now the way our technology works for these neuron Lots is that they expect a one-dimensional sequence of symbols and they want a finite set of symbols that are possible and so we have to decide what are the symbols and then we have to represent our data as one-dimensional sequence of those symbols so right now what we have is a onedimensional sequence of text it starts here and it goes here and then it comes here Etc so this is a onedimensional sequence even though on

08:24

my monitor of course it's laid out in a two-dimensional way but it goes from left to right and top to bottom right so it's a one-dimensional sequence of text now this being computers of course there's an underlying representation here so if I do what's called utf8 uh encode this text then I can get the raw bits that correspond to this text in the computer and that's what uh that looks like this so it turns out that for example this very first bar here is the first uh eight bits as an

08:56

example so what is this thing right this is um representation that we are looking for uh in in a certain sense we have exactly two possible symbols zero and one and we have a very long sequence of it right now as it turns out um this sequence length is actually going to be very finite and precious resource uh in our neural network and we actually don't want extremely long sequences of just two symbols instead what we want is we want to trade off uh this um symbol size uh of this vocabulary as we call it

09:33

and the resulting sequence length so we don't want just two symbols and extremely long sequences we're going to want more symbols and shorter sequences okay so one naive way of compressing or decreasing the length of our sequence here is to basically uh consider some group of consecutive bits for example eight bits and group them into a single what's called bite so because uh these bits are either on or off if we take a group of eight of them there turns out to be only 256 possible combinations of

10:04

how these bits could be on or off and so therefore we can re repesent this sequence into a sequence of bytes instead so this sequence of bytes will be eight times shorter but now we have 256 possible symbols so every number here goes from 0 to 255 now I really encourage you to think of these not as numbers but as unique IDs or like unique symbols so maybe it's a bit more maybe it's better to actually think of these to replace every one of these with a unique Emoji you'd get something like this so um we basically

10:37

have a sequence of emojis and there's 256 possible emojis you can think of it that way now it turns out that in production for state-of-the-art language models uh you actually want to go even Beyond this you want to continue to shrink the length of the sequence uh because again it is a precious resource in return for more symbols in your vocabulary and the way this is done is done by running what's called The Bite pair encoding algorithm and the way this works is we're basically looking for

11:06

consecutive bytes or symbols that are very common so for example turns out that the sequence 116 followed by 32 is quite common and occurs very frequently so what we're going to do is we're going to group uh this um pair into a new symbol so we're going to Mint a symbol with an ID 256 and we're going to rewrite every single uh pair 11632 with this new symbol and then can we can iterate this algorithm as many times as we wish and each time when we mint a new symbol we're decreasing the length and

11:41

we're increasing the symbol size and in practice it turns out that a pretty good setting of um the basically the vocabulary size turns out to be about 100,000 possible symbols so in particular GPT 4 uses 100, 277 symbols um and this process of converting from raw text into these symbols or as we call them tokens is the process called tokenization so let's now take a look at how gp4 performs tokenization conting from text to tokens and from tokens back to text and what this actually looks

12:20

like so one website I like to use to explore these token representations is called tick tokenizer and so come here to the drop down and select CL 100 a base which is the gp4 base model tokenizer and here on the left you can put in text and it shows you the tokenization of that text so for example heo space world so hello world turns out to be exactly two Tokens The Token hello which is the token with ID 15339 and the token space world that is the token 1 1917 so um hello space world now if I was to join these two for example I'm

13:05

going to get again two tokens but it's the token H followed by the token L world without the H um if I put in two Spa two spaces here between hello and world it's again a different uh tokenization there's a new token 220 here okay so you can play with this and see what happens here also keep in mind this is not uh this is case sensitive so if this is a capital H it is something else or if it's uh hello world then actually this ends up being three tokens instead of just two tokens yeah so you can play with this

13:43

and get an sort of like an intuitive sense of uh what these tokens work like we're actually going to loop around to tokenization a bit later in the video for now I just wanted to show you the website and I wanted to uh show you that this text basically at the end of the day so for example if I take one line here this is what GT4 will see it as so this text will be a sequence of length 62 this is the sequence here and this is how the chunks of text correspond to these symbols and again there's 100,

14:17

27777 possible symbols and we now have one-dimensional sequences of those symbols so um yeah we're going to come back to tokenization but that's uh for now where we are okay so what I've done now is I've taken this uh sequence of text that we have here in the data set and I have re-represented it using our tokenizer into a sequence of tokens and this is what that looks like now so for example when we go back to the Fine web data set they mentioned that not only is this 44 terab of dis space but this is

14:46

about a 15 trillion token sequence of um in this data set and so here these are just some of the first uh one or two or three or a few thousand here I think uh tokens of this data set but there's 15 trillion here uh to keep in mind and again keep in mind one more time that all of these represent little text chunks they're all just like atoms of these sequences and the numbers here don't make any sense they're just uh they're just unique IDs okay so now we get to the fun part which is the uh

15:20

neural network training and this is where a lot of the heavy lifting happens computationally when you're training these neural networks so what we do here in this this step is we want to model the statistical relationships of how these tokens follow each other in the sequence so what we do is we come into the data and we take Windows of tokens so we take a window of tokens uh from this data fairly randomly and um the windows length can range anywhere anywhere between uh zero tokens actually all the way up to some

15:54

maximum size that we decide on uh so for example in practice you could see a token with Windows of say 8,000 tokens now in principle we can use arbitrary window lengths of tokens uh but uh processing very long uh basically U window sequences would just be very computationally expensive so we just kind of decide that say 8,000 is a good number or 4,000 or 16,000 and we crop it there now in this example I'm going to be uh taking the first four tokens just so everything fits nicely so these tokens

16:30

we're going to take a window of four tokens this bar view in and space single which are these token IDs and now what we're trying to do here is we're trying to basically predict the token that comes next in the sequence so 3962 comes next right so what we do now here is that we call this the context these four tokens are context and they feed into a neural network and this is the input to the neural network now I'm going to go into the detail of what's inside this neural network in a

17:03

little bit for now it's important to understand is the input and the output of the neural net so the input are sequences of tokens of variable length anywhere between zero and some maximum size like 8,000 the output now is a prediction for what comes next so because our vocabulary has 100277 possible tokens the neural network is going to Output exactly that many numbers and all of those numbers correspond to the probability of that token as coming next in the sequence so it's making guesses about what comes

17:37

next um in the beginning this neural network is randomly initialized so um and we're going to see in a little bit what that means but it's a it's a it's a random transformation so these probabilities in the very beginning of the training are also going to be kind of random uh so here I have three examples but keep in mind that there's 100,000 numbers here um so the probability of this token space Direction neural network is saying that this is 4% likely right now 11799 is 2%

18:06

and then here the probility of 3962 which is post is 3% now of course we've sampled this window from our data set so we know what comes next we know and that's the label we know that the correct answer is that 3962 actually comes next in the sequence so now what we have is this mathematical process for doing an update to the neural network we have the way of tuning it and uh we're going to go into a little bit of of detail in a bit but basically we know that this probability here of 3% we want

18:38

this probability to be higher and we want the probabilities of all the other tokens to be lower and so we have a way of mathematically calculating how to adjust and update the neural network so that the correct answer has a slightly higher probability so if I do an update to the neural network now the next time I Fe this particular sequence of four tokens into neural network the neural network will be slightly adjusted now and it will say Okay post is maybe 4% and case now maybe is 1% and uh Direction could become 2% or

19:12

something like that and so we have a way of nudging of slightly updating the neuronet to um basically give a higher probability to the correct token that comes next in the sequence and now you just have to remember that this process happens not just for uh this um token here where these four fed in and predicted this one this process happens at the same time for all of these tokens in the entire data set and so in practice we sample little windows little batches of Windows and then at every single one of these tokens we want to

19:45

adjust our neural network so that the probability of that token becomes slightly higher and this all happens in parallel in large batches of these tokens and this is the process of training the neural network it's a sequence of updating it so that it's predictions match up the statistics of what actually happens in your training set and its probabilities become consistent with the uh statistical patterns of how these tokens follow each other in the data so let's now briefly get into the internals of these neural

20:13

networks just to give you a sense of what's inside so neural network internals so as I mentioned we have these inputs uh that are sequences of tokens in this case this is four input tokens but this can be anywhere between zero up to let's say 8,000 tokens in principle this can be an infinite number of tokens we just uh it would just be too computationally expensive to process an infinite number of tokens so we just crop it at a certain length and that becomes the maximum context length of

20:41

that uh model now these inputs X are mixed up in a giant mathematical expression together with the parameters or the weights of these neural networks so here I'm showing six example parameters and their setting but in practice these uh um modern neural networks will have billions of these uh parameters and in the beginning these parameters are completely randomly set now with a random setting of parameters you might expect that this uh this neural network would make random predictions and it

21:15

does in the beginning it's totally random predictions but it's through this process of iteratively updating the network uh as and we call that process training a neural network so uh that the setting of these parameters gets adjusted such that the outputs of our neural network becomes consistent with the patterns seen in our training set so think of these parameters as kind of like knobs on a DJ set and as you're twiddling these knobs you're getting different uh predictions for every

21:45

possible uh token sequence input and training in neural network just means discovering a setting of parameters that seems to be consistent with the statistics of the training set now let me just give you an example what this giant mathematical expression looks like just to give you a sense and modern networks are massive expressions with trillions of terms probably but let me just show you a simple example here it would look something like this I mean these are the kinds of Expressions just to show you that it's not very scary we

22:14

have inputs x uh like X1 x2 in this case two example inputs and they get mixed up with the weights of the network w0 W1 2 3 Etc and this mixing is simple things like multiplication addition addition exponentiation division Etc and it is the subject of neural network architecture research to design effective mathematical Expressions uh that have a lot of uh kind of convenient characteristics they are expressive they're optimizable they're paralyzable Etc and so but uh at the end of the day

22:48

these are these are not complex expressions and basically they mix up the inputs with the parameters to make predictions and we're optimizing uh the parameters of this neural network so that the predictions come out consistent with the training set now I would like to show you an actual production grade example of what these neural networks look like so for that I encourage you to go to this website that has a very nice visualization of one of these networks so this is what you will find on this website and this neural network

23:19

here that is used in production settings has this special kind of structure this network is called the Transformer and this particular one as an example has 8 5,000 roughly parameters now here on the top we take the inputs which are the token sequences and then information flows through the neural network until the output which here are the logit softmax but these are the predictions for what comes next what token comes next and then here there's a sequence of Transformations and all these intermediate values that get produced

23:56

inside this mathematical expression s it is sort of predicting what comes next so as an example these tokens are embedded into kind of like this distributed representation as it's called so every possible token has kind of like a vector that represents it inside the neural network so first we embed the tokens and then those values uh kind of like flow through this diagram and these are all very simple mathematical Expressions individually so we have layer norms and Matrix multiplications and uh soft Maxes

24:27

and so on so here kind of like the attention block of this Transformer and then information kind of flows through into the multi-layer perceptron block and so on and all these numbers here these are the intermediate values of the expression and uh you can almost think of these as kind of like the firing rates of these synthetic neurons but I would caution you to uh not um kind of think of it too much like neurons because these are extremely simple neurons compared to the neurons you would find in your brain your biological

24:57

neurons are very complex dynamical processes that have memory and so on there's no memory in this expression it's a fixed mathematical expression from input to Output with no memory it's just a stateless so these are very simple neurons in comparison to biological neurons but you can still kind of loosely think of this as like a synthetic piece of uh brain tissue if you if you like uh to think about it that way so information flows through all these neurons fire until we get to the predictions now I'm not actually

25:27

going to dwell too much on the precise kind of like mathematical details of all these Transformations honestly I don't think it's that important to get into what's really important to understand is that this is a mathematical function it is uh parameterized by some fixed set of parameters like say 85,000 of them and it is a way of transforming inputs into outputs and as we twiddle the parameters we are getting uh different kinds of predictions and then we need to find a good setting of these parameters so that

25:54

the predictions uh sort of match up with the patterns seen in training set so that's the Transformer okay so I've shown you the internals of the neural network and we talked a bit about the process of training it I want to cover one more major stage of working with these networks and that is the stage called inference so in inference what we're doing is we're generating new data from the model and so uh we want to basically see what kind of patterns it has internalized in the parameters of

26:23

its Network so to generate from the model is relatively straightforward we start with some tokens that are basically your prefix like what you want to start with so say we want to start with the token 91 well we feed it into the network and remember that the network gives us probabilities right it gives us this probability Vector here so what we can do now is we can basically flip a biased coin so um we can sample uh basically a token based on this probability distribution so the tokens that are given High probability by the

26:59

model are more likely to be sampled when you flip this biased coin you can think of it that way so we sample from the distribution to get a single unique token so for example token 860 comes next uh so 860 in this case when we're generating from model could come next now 860 is a relatively likely token it might not be the only possible token in this case there could be many other tokens that could have been sampled but we could see that 86c is a relatively likely token as an example and indeed in

27:28

our training examp example here 860 does follow 91 so let's now say that we um continue the process so after 91 there's a60 we append it and we again ask what is the third token let's sample and let's just say that it's 287 exactly as here let's do that again we come back in now we have a sequence of three and we ask what is the likely fourth token and we sample from that and get this one and now let's say we do it one more time we take those four we sample and we get

28:00

this one and this 13659 uh this is not actually uh 3962 as we had before so this token is the token article uh instead so viewing a single article and so in this case we didn't exactly reproduce the sequence that we saw here in the training data so keep in mind that these systems are stochastic they have um we're sampling and we're flipping coins and sometimes we lock out and we reproduce some like small chunk of the text and training set but sometimes we're uh we're getting a token

28:36

that was not verbatim part of any of the documents in the training data so we're going to get sort of like remixes of the data that we saw in the training because at every step of the way we can flip and get a slightly different token and then once that token makes it in if you sample the next one and so on you very quickly uh start to generate token streams that are very different from the token streams that UR in the training documents so statistically they will have similar properties but um they are not identical

29:05

to your training data they're kind of like inspired by the training data and so in this case we got a slightly different sequence and why would we get article you might imagine that article is a relatively likely token in the context of bar viewing single Etc and you can imagine that the word article followed this context window somewhere in the training documents uh to some extent and we just happen to sample it here at that stage so basically inference is just uh predicting from these distributions one at a time we

29:35

continue feeding back tokens and getting the next one and we uh we're always flipping these coins and depending on how lucky or unlucky we get um we might get very different kinds of patterns depending on how we sample from these probability distributions so that's inference so in most common scenarios uh basically downloading the internet and tokenizing it is is a pre-processing step you do that a single time and then uh once you have your token sequence we can start training networks and in

30:07

Practical cases you would try to train many different networks of different kinds of uh settings and different kinds of arrangements and different kinds of sizes and so you''ll be doing a lot of neural network training and um then once you have a neural network and you train it and you have some specific set of parameters that you're happy with um then you can take the model and you can do inference and you can actually uh generate data from the model and when you're on chat GPT and you're talking

30:32

with a model uh that model is trained and has been trained by open aai many months ago probably and they have a specific set of Weights that work well and when you're talking to the model all of that is just inference there's no more training those parameters are held fixed and you're just talking to the model sort of uh you're giving it some of the tokens and it's kind of completing token sequences and that's what you're seeing uh generated when you actually use the model on CH GPT so that

31:00

model then just does inference alone so let's now look at an example of training an inference that is kind of concrete and gives you a sense of what this actually looks like uh when these models are trained now the example that I would like to work with and that I'm particularly fond of is that of opening eyes gpt2 so GPT uh stands for generatively pre-trained Transformer and this is the second iteration of the GPT series by open AI when you are talking to chat GPT today the model that is

31:26

underlying all of the magic of that interaction is GPT 4 so the fourth iteration of that series now gpt2 was published in 2019 by openi in this paper that I have right here and the reason I like gpt2 is that it is the first time that a recognizably modern stack came together so um all of the pieces of gpd2 are recognizable today by modern standards it's just everything has gotten bigger now I'm not going to be able to go into the full details of this paper of course because it is a technical publication but some of the

32:00

details that I would like to highlight are as follows gpt2 was a Transformer neural network just like you were just like the neural networks you would work with today it was it had 1.6 billion parameters right so these are the parameters that we looked at here it would have 1.6 billion of them today modern Transformers would have a lot closer to a trillion or several hundred billion probably the maximum context length here was 1,24 tokens so it is when we are sampling chunks of Windows of tokens

32:32

from the data set we're never taking more than 1,24 tokens and so when you are trying to predict the next token in a sequence you will never have more than 1,24 tokens uh kind of in your context in order to make that prediction now this is also tiny by modern standards today the token uh the context lengths would be a lot closer to um couple hundred thousand or maybe even a million and so you have a lot more context a lot more tokens in history history and you can make a lot better prediction about

33:00

the next token in the sequence in that way and finally gpt2 was trained on approximately 100 billion tokens and this is also fairly small by modern standards as I mentioned the fine web data set that we looked at here the fine web data set has 15 trillion tokens uh so 100 billion is is quite small now uh I actually tried to reproduce uh gpt2 for fun as part of this project called lm. C so you can see my rup of doing that in this post on GitHub under the lm. C repository so in particular the cost of training gpd2 in 2019 what

33:37

was estimated to be approximately $40,000 but today you can do significantly better than that and in particular here it took about one day and about $600 uh but this wasn't even trying too hard I think you could really bring this down to about $100 today now why is it that the costs have come down so much well number one these data sets have gotten a lot better and the way we filter them extract them and prepare them has gotten a lot more refined and so the data set is of just a lot higher quality so that's one thing but really

34:10

the biggest difference is that our computers have gotten much faster in terms of the hardware and we're going to look at that in a second and also the software for uh running these models and really squeezing out all all the speed from the hardware as it is possible uh that software has also gotten much better as as everyone has focused on these models and try to run them very very quickly now I'm not going to be able to go into the full detail of this gpd2 reproduction and this is a long

34:38

technical post but I would like to still give you an intuitive sense for what it looks like to actually train one of these models as a researcher like what are you looking at and what does it look like what does it feel like so let me give you a sense of that a little bit okay so this is what it looks like let me slide this over so what I'm doing here is I'm training a gpt2 model right now and um what's happening here is that every single line here like this one is one update to the model so remember how

35:09

here we are um basically making the prediction better for every one of these tokens and we are updating these weights or parameters of the neural net so here every single line is One update to the neural network where we change its parameters by a little bit so that it is better at predicting next token and sequence in particular every single line here is improving the prediction on 1 million tokens in the training set so we've basically taken 1 million tokens out of this data set and we've tried to

35:42

improve the prediction of that token as coming next in a sequence on all 1 million of them simultaneously and at every single one of these steps we are making an update to the network for that now the number to watch closely is this number called loss and the loss is a single number that is telling you how well your neural network is performing right now and it is created so that low loss is good so you'll see that the loss is decreasing as we make more updates to the neural nut which corresponds to making better

36:14

predictions on the next token in a sequence and so the loss is the number that you are watching as a neural network researcher and you are kind of waiting you're twiddling your thumbs uh you're drinking coffee and you're making sure that this looks good so that with every update your loss is improving and the network is getting better at prediction now here you see that we are processing 1 million tokens per update each update takes about 7 Seconds roughly and here we are going to process

36:44

a total of 32,000 steps of optimization so 32,000 steps with 1 million tokens each is about 33 billion tokens that we are going to process and we're currently only about 420 step 20 out of 32,000 so we are still only a bit more than 1% done because I've only been running this for 10 or 15 minutes or something like that now every 20 steps I have configured this optimization to do inference so what you're seeing here is the model is predicting the next token in a sequence and so you sort of start

37:18

it randomly and then you continue plugging in the tokens so we're running this inference step and this is the model sort of predicting the next token in the sequence and every time you see something appear that's a new token um so let's just look at this and you can see that this is not yet very coherent and keep in mind that this is only 1% of the way through training and so the model is not yet very good at predicting the next token in the sequence so what comes out is actually kind of a little bit of gibberish right

37:47

but it still has a little bit of like local coherence so since she is mine it's a part of the information should discuss my father great companions Gordon showed me sitting over at and Etc so I know it doesn't look very good but let's actually scroll up and see what it looked like when I started the optimization so all the way here at step one so after 20 steps of optimization you see that what we're getting here is looks completely random and of course that's because the model has only had 20

38:20

updates to its parameters and so it's giving you random text because it's a random Network and so you can see that at least in comparison to this model is starting to do much better and indeed if we waited the entire 32,000 steps the model will have improved the point that it's actually uh generating fairly coherent English uh and the tokens stream correctly um and uh they they kind of make up English a a lot better um so this has to run for about a day or two more now and so uh at this stage we

38:52

just make sure that the loss is decreasing everything is looking good um and we just have to wait and now um let me turn now to the um story of the computation that's required because of course I'm not running this optimization on my laptop that would be way too expensive uh because we have to run this neural network and we have to improve it and we have we need all this data and so on so you can't run this too well on your computer uh because the network is just too large uh so all of

39:21

this is running on the computer that is out there in the cloud and I want to basically address the compute side of the store of training these models and what that looks like so let's take a look okay so the computer that I'm running this optimization on is this 8X h100 node so there are eight h100s in a single node or a single computer now I am renting this computer and it is somewhere in the cloud I'm not sure where it is physically actually the place I like to rent from is called

39:49

Lambda but there are many other companies who provide this service so when you scroll down you can see that uh they have some on demand pricing for um sort of computers that have these uh h100s which are gpus and I'm going to show you what they look like in a second but on demand 8times Nvidia h100 uh GPU this machine comes for $3 per GPU per hour for example so you can rent these and then you get a machine in a cloud and you can uh go in and you can train these models and these uh gpus they look like

40:26

this so this is one h100 GPU uh this is kind of what it looks like and you slot this into your computer and gpus are this uh perfect fit for training your networks because they are very computationally expensive but they display a lot of parallelism in the computation so you can have many independent workers kind of um working all at the same time in solving uh the matrix multiplication that's under the hood of training these neural networks so this is just one of these h100s but actually you would put them

40:57

you would put multiple of them together so you could stack eight of them into a single node and then you can stack multiple nodes into an entire data center or an entire system so when we look at a data center can't spell when we look at a data center we start to see things that look like this right so we have one GPU goes to eight gpus goes to a single system goes to many systems and so these are the bigger data centers and there of course would be much much more expensive um and what's happening is that all the

41:29

big tech companies really desire these gpus so they can train all these language models because they are so powerful and that has is fundamentally what has driven the stock price of Nvidia to be $3.4 trillion today as an example and why Nvidia has kind of exploded so this is the Gold Rush the Gold Rush is getting the gpus getting enough of them so they can all collaborate to perform this optimization and they're what are they all doing they're all collaborating to predict the next token on a data set like the fine

42:02

web data set this is the computational workflow that that basically is extremely expensive the more gpus you have the more tokens you can try to predict and improve on and you're going to process this data set faster and you can iterate faster and get a bigger Network and train a bigger Network and so on so this is what all those machines are look like are uh are doing and this is why all of this is such a big deal and for example this is a article from like about a month ago or so this is why it's a big deal that for

42:32

example Elon Musk is getting 100,000 gpus uh in a single Data Center and all of these gpus are extremely expensive are going to take a ton of power and all of them are just trying to predict the next token in the sequence and improve the network uh by doing so and uh get probably a lot more coherent text than what we're seeing here a lot faster okay so unfortunately I do not have a couple 10 or hundred million of dollars to spend on training a really big model like this but luckily we can turn to

43:01

some big tech companies who train these models routinely and release some of them once they are done training so they've spent a huge amount of compute to train this network and they release the network at the end of the optimization so it's very useful because they've done a lot of compute for that so there are many companies who train these models routinely but actually not many of them release uh these what's called base models so the model that comes out at the end here is is what's

43:27

called a base model what is a base model it's a token simulator right it's an internet text token simulator and so that is not by itself useful yet because what we want is what's called an assistant we want to ask questions and have it respond to answers these models won't do that they just uh create sort of remixes of the internet they dream internet pages so the base models are not very often released because they're kind of just only a step one of a few other steps that we still need to take

43:56

to get in system however a few releases have been made so as an example the gbt2 model released the 1.6 billion sorry 1.5 billion model back in 2019 and this gpt2 model is a base model now what is a model release what does it look like to release these models so this is the gpt2 repository on GitHub well you need two things basically to release model number one we need the um python code usually that describes the sequence of operations in detail that they make in their model so um if you remember

44:37

back this Transformer the sequence of steps that are taken here in this neural network is what is being described by this code so this code is sort of implementing the what's called forward pass of this neural network so we need the specific details of exactly how they wired up that neural network so this is just computer code and it's usually just a couple hundred lines of code it's not it's not that crazy and uh this is all fairly understandable and usually fairly standard what's not standard are the

45:05

parameters that's where the actual value is what are the parameters of this neural network because there's 1.6 billion of them and we need the correct setting or a really good setting and so that's why in addition to this source code they release the parameters which in this case is roughly 1.5 billion parameters and these are just numbers so it's one single list of 1.5 billion numbers the precise and good setting of all the knobs such that the tokens come out well so uh you need those two things to

45:37

get a base model release now gpt2 was released but that's actually a fairly old model as I mentioned so actually the model we're going to turn to is called llama 3 and that's the one that I would like to show you next so llama 3 so gpt2 again was 1.6 billion parameters trained on 100 billion tokens Lama 3 is a much bigger model and much more modern model it is released and trained by meta and it is a 45 billion parameter model trained on 15 trillion tokens in very much the same way just much much

46:11

bigger um and meta has also made a release of llama 3 and that was part of this paper so with this paper that goes into a lot of detail the biggest base model that they released is the Lama 3.1 4.5 405 billion parameter model so this is the base model and then in addition to the base model you see here foreshadowing for later sections of the video they also released the instruct model and the instruct means that this is an assistant you can ask it questions and it will give you answers we still

46:43

have yet to cover that part later for now let's just look at this base model this token simulator and let's play with it and try to think about you know what is this thing and how does it work and um what do we get at the end of this optimization if you let this run Until the End uh for a very big neural network on a lot of data so my favorite place to interact with the base models is this um company called hyperbolic which is basically serving the base model of the 405b Llama 3.1 so when you go to the

47:13

website and I think you may have to register and so on make sure that in the models make sure that you are using llama 3.1 405 billion base it must be the base model and then here let's say the max tokens is how many tokens we're going to be gener rating so let's just decrease this to be a bit less just so we don't waste compute we just want the next 128 tokens and leave the other stuff alone I'm not going to go into the full detail here um now fundamentally what's going to happen here is identical

47:42

to what happens here during inference for us so this is just going to continue the token sequence of whatever you prefix you're going to give it so I want to first show you that this model here is not yet an assistant so you can for example ask it what is 2 plus 2 it's not going to tell you oh it's four uh what else can I help you with it's not going to do that because what is 2 plus 2 is going to be tokenized and then those tokens just act as a prefix and then what the model is going to do now is

48:09

just going to get the probability for the next token and it's just a glorified autocomplete it's a very very expensive autocomplete of what comes next um depending on the statistics of what it saw in its training documents which are basically web pages so let's just uh hit enter to see what tokens it comes up with as a continuation okay so here it kind of actually answered the question and started to go off into some philosophical territory uh let's try it again so let me copy and paste and let's

48:40

try again from scratch what is 2 plus two so okay so it just goes off again so notice one more thing that I want to stress is that the system uh I think every time you put it in it just kind of starts from scratch so it doesn't uh the system here is stochastic so for the same prefix of tokens we're always getting a different answer and the reason for that is that we get this probity distribution and we sample from it and we always get different samples and we sort of always go into a different territory uh

49:14

afterwards so here in this case um I don't know what this is let's try one more time so it just continues on so it's just doing the stuff that it's saw on the internet right um and it's just kind of like regurgitating those uh statistical patterns so first things it's not an assistant yet it's a token autocomplete and second it is a stochastic system now the crucial thing is that even though this model is not yet by itself very useful for a lot of applications just yet um it is still very useful because

49:53

in the task of predicting the next token in the sequence the model has learned a lot about the world and it has stored all that knowledge in the parameters of the network so remember that our text looked like this right internet web pages and now all of this is sort of compressed in the weights of the network so you can think of um these 405 billion parameters is a kind of compression of the internet you can think of the 45 billion parameters is kind of like a zip file uh but it's not a loss less

50:25

compression it's a loss C compression we're kind of like left with kind of a gal of the internet and we can generate from it right now we can elicit some of this knowledge by prompting the base model uh accordingly so for example here's a prompt that might work to elicit some of that knowledge that's hiding in the parameters here's my top 10 list of the top landmarks to see in the pairs um and I'm doing it this way because I'm trying to Prime the model to now continue this list so let's see if that

50:56

works when I press enter okay so you see that it started a list and it's now kind of giving me some of those landmarks and now notice that it's trying to give a lot of information here now you might not be able to actually fully trust some of the information here remember that this is all just a recollection of some of the internet documents and so the things that occur very frequently in the internet data are probably more likely to be remembered correctly compared to things that happen

51:23

very infrequently so you can't fully trust some of the things that and some of the information that is here because it's all just a vague recollection of Internet documents because the information is not stored explicitly in any of the parameters it's all just the recollection that said we did get something that is probably approximately correct and I don't actually have the expertise to verify that this is roughly correct but you see that we've elicited a lot of the knowledge of the model and

51:49

this knowledge is not precise and exact this knowledge is vague and probabilistic and statistical and the kinds of things that occur often are the kinds of things that are more likely to be remembered um in the model now I want to show you a few more examples of this model's Behavior the first thing I want to show you is this example I went to the Wikipedia page for zebra and let me just copy paste the first uh even one sentence here and let me put it here now when I click enter what kind of uh completion

52:20

are we going to get so let me just hit enter there are three living species etc etc what the model is producing here is an exact regurgitation of this Wikipedia entry it is reciting this Wikipedia entry purely from memory and this memory is stored in its parameters and so it is possible that at some point in these 512 tokens the model will uh stray away from the Wikipedia entry but you can see that it has huge chunks of it memorized here uh let me see for example if this sentence occurs by now okay so this so we're

52:55

still on track let me check here okay we're still on track it will eventually uh stray away okay so this thing is just recited to a very large extent it will eventually deviate uh because it won't be able to remember exactly now the reason that this happens is because these models can be extremely good at memorization and usually this is not what you want in the final model and this is something called regurgitation and it's usually undesirable to site uh things uh directly uh that you have

53:27

trained on now the reason that this happens actually is because for a lot of documents like for example Wikipedia when these documents are deemed to be of very high quality as a source like for example Wikipedia it is very often uh the case that when you train the model you will preferentially sample from those sources so basically the model has probably done a few epochs on this data meaning that it has seen this web page like maybe probably 10 times or so and it's a bit like you like when you read

53:54

some kind of a text many many times say you read something a 100 times uh then you'll be able to recite it and it's very similar for this model if it sees something way too often it's going to be able to recite it later from memory except these models can be a lot more efficient um like per presentation than human so probably it's only seen this Wikipedia entry 10 times but basically it has remembered this article exactly in its parameters okay the next thing I want to show you is something that the

54:19

model has definitely not seen during its training so for example if we go to the paper uh and then we navigate to the pre-training data we'll see here that uh the data set has a knowledge cut off until the end of 2023 so it will not have seen documents after this point and certainly it has not seen anything about the 2024 election and how it turned out now if we Prime the model with the tokens from the future it will continue the token sequence and it will just take its best guess according to the

54:52

knowledge that it has in its own parameters so let's take a look at what that could look like so the Republican Party kit Trump okay president of the United States from 2017 and let's see what it says after this point so for example the model will have to guess at the running mate and who it's against Etc so let's hit enter so here thingss that Mike Pence was the running mate instead of JD Vance and the ticket was against Hillary Clinton and Tim Kane so this is kind of a interesting parallel universe

55:25

potentially of what could have happened happened according to the LM let's get a different sample so the identical prompt and let's resample so here the running mate was Ronda santis and they ran against Joe Biden and Camala Harris so this is again a different parallel universe so the model will take educated guesses and it will continue the token sequence based on this knowledge um and it will just kind of like all of what we're seeing here is what's called hallucination the model is just taking its best guess uh

55:55

in a probalistic manner the next thing I would like to show you is that even though this is a base model and not yet an assistant model it can still be utilized in Practical applications if you are clever with your prompt design so here's something that we would call a few shot prompt so what it is here is that I have 10 words or 10 pairs and each pair is a word of English column and then a the translation in Korean and we have 10 of them and what the model does here is at the end we have teacher column and then

56:27

here's where we're going to do a completion of say just five tokens and these models have what we call in context learning abilities and what that's referring to is that as it is reading this context it is learning sort of in place that there's some kind of a algorithmic pattern going on in my data and it knows to continue that pattern and this is called kind of like Inc context learning so it takes on the role of a translator and when we hit uh completion we see that the teacher translation is

57:00

Sim which is correct um and so this is how you can build apps by being clever with your prompting even though we still just have a base model for now and it relies on what we call this um uh in context learning ability and it is done by constructing what's called a few shot prompt okay and finally I want to show you that there is a clever way to actually instantiate a whole language model assistant just by prompting and the trick to it is that we're structure a prompt to look like a web page that is

57:29

a conversation between a helpful AI assistant and a human and then the model will continue that conversation so actually to write the prompt I turned to chat gbt itself which is kind of meta but I told it I want to create an llm assistant but all I have is the base model so can you please write my um uh prompt and this is what it came up with which is actually quite good so here's a conversation between an AI assistant and a human the AI assistant is knowledgeable helpful capable of answering wide

58:00

variety of questions Etc and then here it's not enough to just give it a sort of description it works much better if you create this fot prompt so here's a few terms of human assistant human assistant and we have uh you know a few turns of conversation and then here at the end is we're going to be putting the actual query that we like so let me copy paste this into the base model prompt and now let me do human column and this is where we put our actual prompt why is the sky blue and uh let's uh

58:37

run assistant the sky appears blue due to the phenomenon called R lights scattering etc etc so you see that the base model is just continuing the sequence but because the sequence looks like this conversation it takes on that role but it is a little subtle because here it just uh you know it ends the assistant and then just you know hallucinate Ates the next question by the human Etc so it'll just continue going on and on uh but you can see that we have sort of accomplished the task and if you just took this why is the sky

59:06

blue and if we just refresh this and put it here then of course we don't expect this to work with a base model right we're just going to who knows what we're going to get okay we're just going to get more questions okay so this is one way to create an assistant even though you may only have a base model okay so this is the kind of brief summary of the things we talked about over the last few minutes now let me zoom out here and this is kind of like what we've talked about so far we wish to train LM

59:38

assistants like chpt we've discussed the first stage of that which is the pre-training stage and we saw that really what it comes down to is we take Internet documents we break them up into these tokens these atoms of little text chunks and then we predict token sequences using neural networks the output of this entire stage is this base model it is the setting of The parameters of this network and this base model is basically an internet document simulator on the token level so it can just uh it can generate token sequences

01:00:08

that have the same kind of like statistics as Internet documents and we saw that we can use it in some applications but we actually need to do better we want an assistant we want to be able to ask questions and we want the model to give us answers and so we need to now go into the second stage which is called the post-training stage so we take our base model our internet document simulator and hand it off to post training so we're now going to discuss a few ways to do what's called post training of these models these

01:00:36

stages in post training are going to be computationally much less expensive most of the computational work all of the massive data centers um and all of the sort of heavy compute and millions of dollars are the pre-training stage but now we go into the slightly cheaper but still extremely important stage called post trining where we turn this llm model into an assistant so let's take a look at how we can get our model to not sample internet documents but to give answers to questions so in other words

01:01:08

what we want to do is we want to start thinking about conversations and these are conversations that can be multi-turn so so uh there can be multiple turns and they are in the simplest case a conversation between a human and an assistant and so for example we can imagine the conversation could look something like this when a human says what is 2 plus2 the assistant should re respond with something like 2 plus 2 is 4 when a human follows up and says what if it was star instead of a plus assistant could respond with something

01:01:33

like this um and similar here this is another example showing that the assistant could also have some kind of a personality here uh that it's kind of like nice and then here in the third example I'm showing that when a human is asking for something that we uh don't wish to help with we can produce what's called refusal we can say that we cannot help with that so in other words what we want to do now is we want to think through how in a system should interact with the human and we want to program the

01:01:59

assistant and Its Behavior in these conversations now because this is neural networks we're not going to be programming these explicitly in code we're not going to be able to program the assistant in that way because this is neural networks everything is done through neural network training on data sets and so because of that we are going to be implicitly programming the assistant by creating data sets of conversations so these are three independent examples of conversations in a data dat set an actual data set and

01:02:28

I'm going to show you examples will be much larger it could have hundreds of thousands of conversations that are multi- turn very long Etc and would cover a diverse breath of topics but here I'm only showing three examples but the way this works basically is uh a assistant is being programmed by example and where is this data coming from like 2 * 2al 4 same as 2 plus 2 Etc where does that come from this comes from Human labelers so we will basically give human labelers some conversational

01:02:56

context and we will ask them to um basically give the ideal assistant response in this situation and a human will write out the ideal response for an assistant in any situation and then we're going to get the model to basically train on this and to imitate those kinds of responses so the way this works then is we are going to take our base model which we produced in the preing stage and this base model was trained on internet documents we're now going to take that data set of internet documents

01:03:25

and we're gonna throw it out and we're going to substitute a new data set and that's going to be a data set of conversations and we're going to continue training the model on these conversations on this new data set of conversations and what happens is that the model will very rapidly adjust and will sort of like learn the statistics of how this assistant responds to human queries and then later during inference we'll be able to basically um Prime the assistant and get the response and it

01:03:54

will be imitating what the humans will human labelers would do in that situation if that makes sense so we're going to see examples of that and this is going to become bit more concrete I also wanted to mention that this post-training stage we're going to basically just continue training the model but um the pre-training stage can in practice take roughly three months of training on many thousands of computers the post-training stage will typically be much shorter like 3 hours for example

01:04:20

um and that's because the data set of conversations that we're going to create here manually is much much smaller than the data set of text on the internet and so this training will be very short but fundamentally we're just going to take our base model we're going to continue training using the exact same algorithm the exact same everything except we're swapping out the data set for conversations so the questions now are what are these conversations how do we represent them how do we get the model

01:04:47

to see conversations instead of just raw text and then what are the outcomes of um this kind of training and what do you get in a certain like psychological sense uh when we talk about the model so let's turn to those questions now so let's start by talking about the tokenization of conversations everything in these models has to be turned into tokens because everything is just about token sequences so how do we turn conversations into token sequences is the question and so for that we need to

01:05:15

design some kind of ending coding and uh this is kind of similar to maybe if you're familiar you don't have to be with for example the TCP IP packet in um on the internet there are precise rules and protocols for how you represent information how everything is structured together so that you have all this kind of data laid out in a way that is written out on a paper and that everyone can agree on and so it's the same thing now happening in llms we need some kind of data structures and we need to have

01:05:40

some rules around how these data structures like conversations get encoded and decoded to and from tokens and so I want to show you now how I would recreate uh this conversation in the token space so if you go to Tech tokenizer I can take that conversation and this is how it is represented in uh for the language model so here we have we are iterating a user and an assistant in this two- turn conversation and what you're seeing here is it looks ugly but it's actually relatively simple the way it gets turned

01:06:14

into a token sequence here at the end is a little bit complicated but at the end this conversation between a user and assistant ends up being 49 tokens it is a one-dimensional sequence of 49 tokens and these are the tokens okay and all the different llms will have a slightly different format or protocols and it's a little bit of a wild west right now but for example GPT 40 does it in the following way you have this special token called imore start and this is short for IM imaginary monologue uh the

01:06:46

start then you have to specify um I don't actually know why it's called that to be honest then you have to specify whose turn it is so for example user which is a token 4 28 then you have internal monologue separator and then it's the exact question so the tokens of the question and then you have to close it so I am end the end of the imaginary monologue so basically the question from a user of what is 2 plus two ends up being the token sequence of these tokens and now the important thing to mention here is

01:07:21

that IM start this is not text right IM start is a special token that gets added it's a new token and um this token has never been trained on so far it is a new token that we create in a post-training stage and we introduce and so these special tokens like IM seep IM start Etc are introduced and interspersed with text so that they sort of um get the model to learn that hey this is a the start of a turn for who is it start of the turn for the start of the turn is for the user and then this is what the

01:07:54

user says and then the user ends and then it's a new start of a turn and it is by the assistant and then what does the assistant say well these are the tokens of what the assistant says Etc and so this conversation is not turned into the sequence of tokens the specific details here are not actually that important all I'm trying to show you in concrete terms is that our conversations which we think of as kind of like a structured object end up being turned via some encoding into onedimensional

01:08:22

sequences of tokens and so because this is one dimensional sequence of tokens we can apply all the stuff that we applied before now it's just a sequence of tokens and now we can train a language model on it and so we're just predicting the next token in a sequence uh just like before and um we can represent and train on conversations and then what does it look like at test time during inference so say we've trained a model and we've trained a model on these kinds of data sets of conversations and now we

01:08:51

want to inference so during inference what does this look like when you're on on chash apt well you come to chash apt and you have say like a dialogue with it and the way this works is basically um say that this was already filled in so like what is 2 plus 2 2 plus 2 is four and now you issue what if it was times I am end and what basically ends up happening um on the servers of open AI or something like that is they put in I start assistant I amep and this is where they end it right here so they

01:09:25

construct this context and now they start sampling from the model so it's at this stage that they will go to the model and say okay what is a good for sequence what is a good first token what is a good second token what is a good third token and this is where the LM takes over and creates a response like for example response that looks something like this but it doesn't have to be identical to this but it will have the flavor of this if this kind of a conversation was in the data set so um

01:09:52

that's roughly how the protocol Works although the details of this protocol are not important so again my goal is that just to show you that everything ends up being just a one-dimensional token sequence so we can apply everything we've already seen but we're now training on conversations and we're now uh basically generating conversations as well okay so now I would like to turn to what these data sets look like in practice the first paper that I would like to show you and the first effort in this direction is

01:10:20

this paper from openai in 2022 and this paper was called instruct GPT or the technique that they developed and this was the first time that opena has kind of talked about how you can take language models and fine-tune them on conversations and so this paper has a number of details that I would like to take you through so the first stop I would like to make is in section 3.4 where they talk about the human contractors that they hired uh in this case from upwork or through scale AI to uh construct these conversations and so

01:10:50

there are human labelers involved whose job it is professionally to create these conversations and these labelers are asked to come up with prompts and then they are asked to also complete the ideal assistant responses and so these are the kinds of prompts that people came up with so these are human labelers so list five ideas for how to regain enthusiasm for my career what are the top 10 science fiction books I should read next and there's many different types of uh kind of prompts here so

01:11:17

translate this sentence from uh to Spanish Etc and so there's many things here that people came up with they first come up with the prompt and then they also uh answer that prompt and they give the ideal assistant response now how do they know what is the ideal assistant response that they should write for these prompts so when we scroll down a little bit further we see that here we have this excerpt of labeling instructions uh that are given to the human labelers so the company that is developing the language model like for

01:11:45

example open AI writes up labeling instructions for how the humans should create ideal responses and so here for example is an excerpt uh of these kinds of labeling instruction instructions on High level you're asking people to be helpful truthful and harmless and you can pause the video if you'd like to see more here but on a high level basically just just answer try to be helpful try to be truthful and don't answer questions that we don't want um kind of the system to handle uh later in chat

01:12:13

gbt and so roughly speaking the company comes up with the labeling instructions usually they are not this short usually there are hundreds of pages and people have to study them professionally and then they write out the ideal assistant responses uh following those labeling instructions so this is a very human heavy process as it was described in this paper now the data set for instruct GPT was never actually released by openi but we do have some open- Source um reproductions that were're trying to

01:12:41

follow this kind of a setup and collect their own data so one that I'm familiar with for example is the effort of open Assistant from a while back and this is just one of I think many examples but I just want to show you an example so here's so these were people on the internet that were asked to basically create these conversations similar to what um open I did with human labelers and so here's an entry of a person who came up with this BR can you write a short introduction to the relevance of

01:13:09

the term manop uh in economics please use examples Etc and then the same person or potentially a different person will write up the response so here's the assistant response to this and so then the same person or different person will actually write out this ideal response and then this is an example of maybe how the conversation could continue now explain it to a dog and then you can try to come up with a slightly a simpler explanation or something like that now this then becomes the label and we end up training

01:13:42

on this so what happens during training is that um of course we're not going to have a full coverage of all the possible questions that um the model will encounter at test time during inference we can't possibly cover all the possible prompts that people are going to be asking in the future but if we have a like a data set of a few of these examples then the model during training will start to take on this Persona of this helpful truthful harmless assistant and it's all programmed by example and

01:14:15

so these are all examples of behavior and if you have conversations of these example behaviors and you have enough of them like 100,00 and you train on it the model sort of starts to understand the statistical pattern and it kind of takes on this personality of this assistant now it's possible that when you get the exact same question like this at test time it's possible that the answer will be recited as exactly what was in the training set but more likely than that is that the model will kind of

01:14:43

like do something of a similar Vibe um and we will understand that this is the kind of answer that you want um so that's what we're doing we're programming the system um by example and the system adopts statistically this Persona of this helpful truthful harmless assistant which is kind of like reflected in the labeling instructions that the company creates now I want to show you that the state-of-the-art has kind of advanced in the last 2 or 3 years uh since the instr GPT paper so in

01:15:13

particular it's not very common for humans to be doing all the heavy lifting just by themselves anymore and that's because we now have language models and these language models are helping us create these data sets and conversations so it is very rare that the people will like literally just write out the response from scratch it is a lot more likely that they will use an existing llm to basically like uh come up with an answer and then they will edit it or things like that so there's many

01:15:36

different ways in which now llms have started to kind of permeate this posttraining Set uh stack and llms are basically used pervasively to help create these massive data sets of conversations so I don't want to show like Ultra chat is one um such example of like a more modern data set of conversations it is to a very large extent synthetic but uh I believe there's some human involvement I could be wrong with that usually there will be a little bit of human but there will be a huge amount of synthetic help um and

01:16:06

this is all kind of like uh constructed in different ways and Ultra chat is just one example of many sft data sets that currently exist and the only thing I want to show you is that uh these data sets have now millions of conversations uh these conversations are mostly synthetic but they're probably edited to some extent by humans and they span a huge diversity of sort of um uh areas and so on so these are fairly extensive artifacts by now and there's all these like sft mixtures as they're called so you have a mixture of

01:16:38

like lots of different types and sources and it's partially synthetic partially human and it's kind of like um gone in that direction since uh but roughly speaking we still have sft data sets they're made up of conversations we're training on them um just like we did before and uh I guess like the last thing to note is that I want to dispel a little bit of the magic of talking to an AI like when you go to chat GPT and you give it a question and then you hit enter uh what is coming back is kind of like

01:17:10

statistically aligned with what's happening in the training set and these training sets I mean they really just have a seed in humans following labeling instructions so what are you actually talking to in chat GPT or how should you think about it well it's not coming from some magical AI like roughly speaking it's coming from something that is statistically imitating human labelers which comes from labeling instructions written by these companies and so you're kind of imitating this uh you're kind of

01:17:39

getting um it's almost as if you're asking human labeler and imagine that the answer that is given to you uh from chbt is some kind of a simulation of a human labeler uh and it's kind of like asking what would a human labeler say in this kind of a conversation and uh it's not just like this human labeler is not just like a random person from the internet because these companies actually hire experts so for example when you are asking questions about code and so on the human labelers

01:18:07

that would be in um involved in creation of these conversation data sets they will usually be usually be educated expert people and you're kind of like asking a question of like a simulation of those people if that makes sense so you're not talking to a magical AI you're talking to an average labeler this average labeler is probably fairly highly skilled but you're talking to kind of like an instantaneous simulation of that kind of a person that would be hired uh in the construction of these data sets so let

01:18:35

me give you one more specific example before we move on for example when I go to chpt and I say recommend the top five landmarks who see in Paris and then I hit enter uh okay here we go okay when I hit enter what's coming out here how do I think about it well it's not some kind of a magical AI that has gone out and researched all the landmarks and then ranked them using its infinite intelligence Etc what I'm getting is a statistical simulation of a labeler that was hired by open AI you can think about

01:19:09

it roughly in that way and so if this specific um question is in the posttraining data set somewhere at open aai then I'm very likely to see an answer that is probably very very similar to what that human labeler would have put down for those five landmarks how does the human labeler come up with this well they go off and they go on the internet and they kind of do their own little research for 20 minutes and they just come up with a list right now so if they come up with this list and this is in

01:19:37

the data set I'm probably very likely to see what they submitted as the correct answer from the assistant now if this specific query is not part of the post training data set then what I'm getting here is a little bit more emergent uh because uh the model kind of understands the statistically um the kinds of landmarks that are in this training set are usually the prominent landmarks the landmarks that people usually want to see the kinds of landmarks that are usually uh very often talked about on the internet and

01:20:07

remember that the model already has a ton of Knowledge from its pre-training on the internet so it's probably seen a ton of conversations about Paris about landmarks about the kinds of things that people like to see and so it's the pre-training knowledge that has then combined with the postering data set that results in this kind of an imitation um so that's uh that's roughly how you can kind of think about what's happening behind the scenes here in in this statistical sense okay now I want to

01:20:34

turn to the topic of llm psychology as I like to call it which is what are sort of the emergent cognitive effects of the training pipeline that we have for these models so in particular the first one I want to talk to is of course hallucinations so you might be familiar with model hallucinations it's when llms make stuff up they just totally fabricate information Etc and it's a big problem with llm assistants it is a problem that existed to a large extent with early models uh from many years ago

01:21:03

and I think the problem has gotten a bit better uh because there are some medications that I'm going to go into in a second for now let's just try to understand where these hallucinations come from so here's a specific example of a few uh of three conversations that you might think you have in your training set and um these are pretty reasonable conversations that you could imagine being in the training set so like for example who is Cruz well Tom Cruz is an famous actor American actor

01:21:28

and producer Etc who is John baraso this turns out to be a us senetor for example who is genis Khan well genis Khan was blah blah blah and so this is what your conversations could look like at training time now the problem with this is that when the human is writing the correct answer for the assistant in each one of these cases uh the human either like knows who this person is or they research them on the Internet and they come in and they write this response that kind of has this like confident

01:21:57

tone of an answer and what happens basically is that at test time when you ask for someone who is this is a totally random name that I totally came up with and I don't think this person exists um as far as I know I just Tred to generate it randomly the problem is when we ask who is Orson kovats the problem is that the assistant will not just tell you oh I don't know even if the assistant and the language model itself might know inside its features inside its activations inside of its brain sort of

01:22:27

it might know that this person is like not someone that um that is that it's familiar with even if some part of the network kind of knows that in some sense the uh saying that oh I don't know who this is is is not going to happen because the model statistically imitates is training set in the training set the questions of the form who is blah are confidently answered with the correct answer and so it's going to take on the style of the answer and it's going to do its best it's going to give you

01:22:55

statistically the most likely guess and it's just going to basically make stuff up because these models again we just talked about it is they don't have access to the internet they're not doing research these are statistical token tumblers as I call them uh is just trying to sample the next token in the sequence and it's going to basically make stuff up so let's take a look at what this looks like I have here what's called the inference playground from hugging face and I am on purpose picking on a model

01:23:23

called Falcon 7B which is an old model this is a few years ago now so it's an older model So It suffers from hallucinations and as I mentioned this has improved over time recently but let's say who is Orson kovats let's ask Falcon 7B instruct run oh yeah Orson kovat is an American author and science uh fiction writer okay this is totally false it's hallucination let's try again these are statistical systems right so we can resample this time Orson kovat is a fictional character from this 1950s TV

01:23:54

show it's total BS right let's try again he's a former minor league baseball player okay so basically the model doesn't know and it's given us lots of different answers because it doesn't know it's just kind of like sampling from these probabilities the model starts with the tokens who is oron kovats assistant and then it comes in here and it's get it's getting these probabilities and it's just sampling from the probabilities and it just like comes up with stuff and the stuff is

01:24:24

actually statistically consistent with the style of the answer in its training set and it's just doing that but you and I experiened it as a madeup factual knowledge but keep in mind that uh the model basically doesn't know and it's just imitating the format of the answer and it's not going to go off and look it up uh because it's just imitating again the answer so how can we uh mitigate this because for example when we go to chat apt and I say who is oron kovats and I'm now asking the stateoftheart

01:24:53

state-of-the-art model from open AI this model will tell you oh so this model is actually is even smarter because you saw very briefly it said searching the web uh we're going to cover this later um it's actually trying to do tool use and uh kind of just like came up with some kind of a story but I want to just who or Kovach did not use any tools I don't want it to do web search there's a wellknown historical or public figure named or oron kovats so this model is not going to make up stuff

01:25:29

this model knows that it doesn't know and it tells you that it doesn't appear to be a person that this model knows so somehow we sort of improved hallucinations even though they clearly are an issue in older models and it makes totally uh sense why you would be getting these kinds of answers if this is what your training set looks like so how do we fix this okay well clearly we need some examples in our data set that where the correct answer for the assistant is that the model doesn't know

01:25:57

about some particular fact but we only need to have those answers be produced in the cases where the model actually doesn't know and so the question is how do we know what the model knows or doesn't know well we can empirically probe the model to figure that out so let's take a look at for example how meta uh dealt with hallucinations for the Llama 3 series of models as an example so in this paper that they published from meta we can go into hallucinations which they call here factuality and they

01:26:28

describe the procedure by which they basically interrogate the model to figure out what it knows and doesn't know to figure out sort of like the boundary of its knowledge and then they add examples to the training set where for the things where the model doesn't know them the correct answer is that the model doesn't know them which sounds like a very easy thing to do in principle but this roughly fixes the issue and the the reason it fixes the issue is because remember like the model might

01:26:59

actually have a pretty good model of its self knowledge inside the network so remember we looked at the network and all these neurons inside the network you might imagine that there's a neuron somewhere in the network that sort of like lights up for when the model is uncertain but the problem is that the activation of that neuron is not currently wired up to the model actually saying in words that it doesn't know so even though the internal of the neural network no because there's some neurons

01:27:27

that represent that the model uh will not surface that it will instead take its best guess so that it sounds confident um just like it sees in a training set so we need to basically interrogate the model and allow it to say I don't know in the cases that it doesn't know so let me take you through what meta roughly does so basically what they do is here I have an example uh Dominic kek is uh the featured article today so I just went there randomly and what they do is basically they take a

01:27:56

random document in a training set and they take a paragraph and then they use an llm to construct questions about that paragraph so for example I did that with chat GPT here so I said here's a paragraph from this document generate three specific factual questions based on this paragraph and give me the questions and the answers and so the llms are already good enough to create and reframe this information so if the information is in the context window um of this llm this actually works pretty well it doesn't

01:28:31

have to rely on its memory it's right there in the context window and so it can basically reframe that information with fairly high accuracy so for example can generate questions for us like for which team did he play here's the answer how many cups did he win Etc and now what we have to do is we have some question and answers and now we want to interrogate the model so roughly speaking what we'll do is we'll take our questions and we'll go to our model which would be uh say llama uh in meta

01:28:59

but let's just interrogate mol 7B here as an example that's another model so does this model know about this answer let's take a look uh so he played for Buffalo Sabers right so the model knows and the the way that you can programmatically decide is basically we're going to take this answer from the model and we're going to compare it to the correct answer and again the model model are good enough to do this automatically so there's no humans involved here we can take uh

01:29:29

basically the answer from the model and we can use another llm judge to check if that is correct according to this answer and if it is correct that means that the model probably knows so what we're going to do is we're going to do this maybe a few times so okay it knows it's Buffalo Savers let's drag in um Buffalo Sabers let's try one more time Buffalo Sabers so we asked three times about this factual question and the model seems to know so everything is great now let's try the second question

01:30:01

how many Stanley Cups did he win and again let's interrogate the model about that and the correct answer is two so um here the model claims that he won um four times which is not correct right it doesn't match two so the model doesn't know it's making stuff up let's try again um so here the model again it's kind of like making stuff up right let's Dragon here it says did he did not even did not win during his career so obviously the model doesn't know and the way we can programmatically tell again

01:30:43

is we interrogate the model three times and we compare its answers maybe three times five times whatever it is to the correct answer and if the model doesn't know then we know that the model doesn't know this question and then what we do is we take this question we create a new conversation in the training set so we're going to add a new conversation training set and when the question is how many Stanley Cups did he win the answer is I'm sorry I don't know or I don't remember and

01:31:11

that's the correct answer for this question because we interrogated the model and we saw that that's the case if you do this for many different types of uh questions for many different types of documents you are giving the model an opportunity to in its training set refuse to say based on its knowledge and if you just have a few examples of that in your training set the model will know um and and has the opportunity to learn the association of this knowledge-based refusal to this internal neuron

01:31:41

somewhere in its Network that we presume exists and empirically this turns out to be probably the case and it can learn that Association that hey when this neuron of uncertainty is high then I actually don't know and I'm allowed to say that I'm sorry but I don't think I remember this Etc and if you have these uh examples in your training set then this is a large mitigation for hallucination and that's roughly speaking why chpt is able to do stuff like this as well so these are kinds of

01:32:11

uh mitigations that people have implemented and that have improved the factuality issue over time okay so I've described mitigation number one for basically mitigating the hallucinations issue now we can actually do much better than that uh it's instead of just saying that we don't know uh we can introduce an additional mitigation number two to give the llm an opportunity to be factual and actually answer the question now what do you and I do if I was to ask you a factual question and you don't

01:32:41

know uh what would you do um in order to answer the question well you could uh go off and do some search and uh use the internet and you could figure out the answer and then tell me what that answer is and we can do the exact exact same thing with these models so think of the knowledge inside the neural network inside its billions of parameters think of that as kind of a vague recollection of the things that the model has seen during its training during the pre-training stage a long time ago so

01:33:10

think of that knowledge in the parameters as something you read a month ago and if you keep reading something then you will remember it and the model remembers that but if it's something rare then you probably don't have a really good recollection of that information but what you and I do is we just go and look it up now when you go and look it up what you're doing basically is like you're refreshing your working memory with information and then you're able to sort of like retrieve it

01:33:32

talk about it or Etc so we need some equivalent of allowing the model to refresh its memory or its recollection and we can do that by introducing tools uh for the models so the way we are going to approach this is that instead of just saying hey I'm sorry I don't know we can attempt to use tools so we can create uh a mechanism by which the language model can emit special tokens and these are tokens that we're going to introduce new tokens so for example here I've introduced two tokens and I've introduced a format or a

01:34:05

protocol for how the model is allowed to use these tokens so for example instead of answering the question when the model does not instead of just saying I don't know sorry the model has the option now to emitting the special token search start and this is the query that will go to like bing.com in the case of openai or say Google search or something like that so it will emit the query and then it will emit search end and then here what will happen is that the program that is sampling from the model that is

01:34:35

running the inference when it sees the special token search end instead of sampling the next token uh in the sequence it will actually pause generating from the model it will go off it will open a session with bing.com and it will paste the search query into Bing and it will then um get all the text that is retrieved and it will basically take that text it will maybe represent it again with some other special tokens or something like that and it will take that text and it will copy paste it here

01:35:05

into what I Tred to like show with the brackets so all that text kind of comes here and when the text comes here it enters the context window so the model so that text from the web search is now inside the context window that will feed into the neural network and you should think of the context window as kind of like the working memory of the model that data that is in the context window is directly accessible by the model it directly feeds into the neural network so it's not anymore a vague recollection

01:35:34

it's data that it it has in the context window and is directly available to that model so now when it's sampling the new uh tokens here afterwards it can reference very easily the data that has been copy pasted in there so that's roughly how these um how these tools use uh tools uh function and so web search is just one of the tools we're going to look at some of the other tools in a bit uh but basically you introduce new tokens you introduce some schema by which the model can utilize these tokens and can call these

01:36:05

special functions like web search functions and how do you teach the model how to correctly use these tools like say web search search start search end Etc well again you do that through training sets so we need now to have a bunch of data and a bunch of conversations that show the model by example how to use web search so what are the what are the settings where you are using the search um and what does that look like and here's by example how you start a search and the search Etc and uh if you have a few thousand maybe

01:36:35

examples of that in your training set the model will actually do a pretty good job of understanding uh how this tool works and it will know how to sort of structure its queries and of course because of the pre-training data set and its understanding of the world it actually kind of understands what a web search is and so it actually kind of has a pretty good native understanding um of what kind of stuff is a good search query um and so it all kind of just like works you just need a little bit of a few examples to show it how to

01:37:03

use this new tool and then it can lean on it to retrieve information and uh put it in the context window and that's equivalent to you and I looking something up because once it's in the context it's in the working memory and it's very easy to manipulate and access so that's what we saw a few minutes ago when I was searching on chat GPT for who is Orson kovats the chat GPT language model decided Ed that this is some kind of a rare um individual or something like that and instead of giving me an

01:37:30

answer from its memory it decided that it will sample a special token that is going to do web search and we saw briefly something flash it was like using the web tool or something like that so it briefly said that and then we waited for like two seconds and then it generated this and you see how it's creating references here and so it's citing sources so what happened here is it went off it did a web web search it found these sources and these URLs and the text of these web pages was all

01:37:59

stuffed in between here and it's not showing here but it's it's basically stuffed as text in between here and now it sees that text and now it kind of references it and says that okay it could be these people citation could be those people citation Etc so that's what happened here and that's what and that's why when I said who is Orson kovats I could also say don't use any tools and then that's enough to um basically convince chat PT to not use tools and just use its memory and its

01:38:28

recollection I also went off and I um tried to ask this question of Chachi PT so how many standing cups did uh Dominic Hasek win and Chachi P actually decided that it knows the answer and it has the confidence to say that uh he want twice and so it kind of just relied on its memory because presumably it has um it has enough of a kind of confidence in its weights in it parameters and activations that this is uh retrievable just for memory um but you can also conversely use web search to make sure

01:39:04

and then for the same query it actually goes off and it searches and then it finds a bunch of sources it finds all this all of this stuff gets copy pasted in there and then it tells us uh to again and sites and it actually says the Wikipedia article which is the source of this information for us as well so that's tools web search the model determines when to search and then uh that's kind of like how these tools uh work and this is an additional kind of mitigation for uh hallucinations and

01:39:35

factuality so I want to stress one more time this very important sort of psychology Point knowledge in the parameters of the neural network is a vague recollection the knowledge in the tokens that make up the context window is the working memory and it roughly speaking Works kind of like um it works for us in our brain the stuff we remember is our parameters uh and the stuff that we just experienced like a few seconds or minutes ago and so on you can imagine that being in our context window and this context window is being

01:40:06

built up as you have a conscious experience around you so this has a bunch of um implications also for your use of LOLs in practice so for example I can go to chat GPT and I can do something like this I can say can you Summarize chapter one of Jane Austin's Pride and Prejudice right and this is a perfectly fine prompt and Chach actually does something relatively reasonable here and but the reason it does that is because Chach has a pretty good recollection of a famous work like Pride and Prejudice it's probably seen a ton

01:40:34

of stuff about it there's probably forums about this book it's probably read versions of this book um and it's kind of like remembers because even if you've read this or articles about it you'd kind of have a recollection enough to actually say all this but usually when I actually interact with LMS and I want them to recall specific things it always works better if you just give it to them so I think a much better prompt would be something like this can you summarize for me chapter one of genos's

01:41:01

spr and Prejudice and then I am attaching it below for your reference and then I do something like a delimeter here and I paste it in and I I found that just copy pasting it from some website that I found here um so copy pasting the chapter one here and I do that because when it's in the context window the model has direct access to it and can exactly it doesn't have to recall it it just has access to it and so this summary is can be expected to be a significantly high quality or higher

01:41:29

quality than this summary uh just because it's directly available to the model and I think you and I would work in the same way if you want to it would be you would produce a much better summary if you had reread this chapter before you had to summarize it and that's basically what's happening here or the equivalent of it the next sort of psychological Quirk I'd like to talk about briefly is that of the knowledge of self so what I see very often on the internet is that people do something

01:41:54

like this they ask llms something like what model are you and who built you and um basically this uh question is a little bit nonsensical and the reason I say that is that as I try to kind of explain with some of the underhood fundamentals this thing is not a person right it doesn't have a persistent existence in any way it sort of boots up processes tokens and shuts off and it does that for every single person it just kind of builds up a context window of conversation and then everything gets

01:42:21

deleted and so this this entity is kind of like restarted from scratch every single conversation if that makes sense it has no persistent self it has no sense of self it's a token tumbler and uh it follows the statistical regularities of its training set so it doesn't really make sense to ask it who are you what build you Etc and by default if you do what I described and just by default and from nowhere you're going to get some pretty random answers so for example let's uh pick on Falcon

01:42:48

which is a fairly old model and let's see what it tells us uh so it's evading the question uh talented engineers and developers here it says I was built by open AI based on the gpt3 model it's totally making stuff up now the fact that it's built by open AI here I think a lot of people would take this as evidence that this model was somehow trained on open AI data or something like that I don't actually think that that's necessarily true the reason for that is that if you don't explicitly program the

01:43:18

model to answer these kinds of questions then what you're going to get is its statistical best guess at the answer and this model had a um sft data mixture of conversations and during the fine-tuning um the model sort of understands as it's training on this data that it's taking on this personality of this like helpful assistant and it doesn't know how to it doesn't actually it wasn't told exactly what label to apply to self it just kind of is taking on this uh this uh Persona

01:43:50

of a helpful assistant and remember that the pre-training stage took the documents from the entire internet and Chach and open AI are very prominent in these documents and so I think what's actually likely to be happening here is that this is just its hallucinated label for what it is this is its self-identity is that it's chat GPT by open Ai and it's only saying that because there's a ton of data on the internet of um answers like this that are actually coming from open from chasht and So

01:44:20

that's its label for what it is now you can override this as a developer if you have a llm model you can actually override it and there are a few ways to do that so for example let me show you there's this MMO model from Allen Ai and um this is one llm it's not a top tier LM or anything like that but I like it because it is fully open source so the paper for Almo and everything else is completely fully open source which is nice um so here we are looking at its sft mixture so this is the data mixture

01:44:50

of um the fine tuning so this is the conversations data it right and so the way that they are solving it for Theo model is we see that there's a bunch of stuff in the mixture and there's a total of 1 million conversations here but here we have alot to hardcoded if we go there we see that this is 240 conversations and look at these 240 conversations they're hardcoded tell me about yourself says user and then the assistant says I'm and open language model developed by AI to Allen Institute

01:45:20

of artificial intelligence Etc I'm here to help blah blah blah what is your name uh Theo project so these are all kinds of like cooked up hardcoded questions abouto 2 and the correct answers to give in these cases if you take 240 questions like this or conversations put them into your training set and fine tune with it then the model will actually be expected to parot this stuff later if you don't give it this then it's probably a Chach by open Ai and um there's one more way to sometimes do this is

01:45:51

that basically um in these conversations and you have terms between human and assistant sometimes there's a special message called system message at the very beginning of the conversation so it's not just between human and assistant there's a system and in the system message you can actually hardcode and remind the model that hey you are a model developed by open Ai and your name is chashi pt40 and you were trained on this date and your knowledge cut off is this and basically it kind of like

01:46:20

documents the model a little bit and then this is inserted into to your conversations so when you go on chpt you see a blank page but actually the system message is kind of like hidden in there and those tokens are in the context window and so those are the two ways to kind of um program the models to talk about themselves either it's done through uh data like this or it's done through system message and things like that basically invisible tokens that are in the context window and remind the

01:46:46

model of its identity but it's all just kind of like cooked up and bolted on in some in some way it's not actually like really deeply there in any real sense as it would before a human I want to now continue to the next section which deals with the computational capabilities or like I should say the native computational capabilities of these models in problem solving scenarios and so in particular we have to be very careful with these models when we construct our examples of conversations

01:47:11

and there's a lot of sharp edges here that are kind of like elucidative is that a word uh they're kind of like interesting to look at when we consider how these models think so um consider the following prompt from a human and supposed that basically that we are building out a conversation to enter into our training set of conversations so we're going to train the model on this we're teaching you how to basically solve simple math problems so the prompt is Emily buys three apples and two

01:47:37

oranges each orange cost $2 the total cost is 13 what is the cost of apples very simple math question now there are two answers here on the left and on the right they are both correct answers they both say that the answer is three which is correct but one of these two is a significant ific anly better answer for the assistant than the other like if I was Data labeler and I was creating one of these one of these would be uh a really terrible answer for the assistant and the other would be okay and so I'd

01:48:06

like you to potentially pause the video Even and think through why one of these two is significantly better answer uh than the other and um if you use the wrong one your model will actually be uh really bad at math potentially and it would have uh bad outcomes and this is something that you would be careful with in your life labeling documentations when you are training people uh to create the ideal responses for the assistant okay so the key to this question is to realize and remember that when the models are training and also

01:48:34

inferencing they are working in onedimensional sequence of tokens from left to right and this is the picture that I often have in my mind I imagine basically the token sequence evolving from left to right and to always produce the next token in a sequence we are feeding all these tokens into the neural network and this neural network then is the probabilities for the next token and sequence right so this picture here is the exact same picture we saw uh before up here and this comes from the web demo

01:49:02

that I showed you before right so this is the calculation that basically takes the input tokens here on the top and uh performs these operations of all these neurons and uh gives you the answer for the probabilities of what comes next now the important thing to realize is that roughly speaking uh there's basically a finite number of layers of computation that happened here so for example this model here has only one two three layers of what's called detention and uh MLP here um maybe um typical modern

01:49:34

state-of-the-art Network would have more like say 100 layers or something like that but there's only 100 layers of computation or something like that to go from the previous token sequence to the probabilities for the next token and so there's a finite amount of computation that happens here for every single token and you should think of this as a very small amount of computation and this amount of computation is almost roughly fixed uh for every single token in this sequence um the that's not actually

01:50:00

fully true because the more tokens you feed in uh the the more expensive uh this forward pass will be of this neural network but not by much so you should think of this uh and I think as a good model to have in mind this is a fixed amount of compute that's going to happen in this box for every single one of these tokens and this amount of compute Cann possibly be too big because there's not that many layers that are sort of going from the top to bottom here there's not that that much

01:50:24

computationally that will happen here and so you can't imagine the model to to basically do arbitrary computation in a single forward pass to get a single token and so what that means is that we actually have to distribute our reasoning and our computation across many tokens because every single token is only spending a finite amount of computation on it and so we kind of want to distribute the computation across many tokens and we can't have too much computation or expect too much computation out of of the model in any

01:50:54

single individual token because there's only so much computation that happens per token okay roughly fixed amount of computation here so that's why this answer here is significantly worse and the reason for that is Imagine going from left to right here um and I copy pasted it right here the answer is three Etc imagine the model having to go from left to right emitting these tokens one at a time it has to say or we're expecting to say the answer is space dollar sign and then right here we're expecting it to

01:51:29

basically cram all of the computation of this problem into this single token it has to emit the correct answer three and then once we've emitted the answer three we're expecting it to say all these tokens but at this point we've already prod produced the answer and it's already in the context window for all these tokens that follow so anything here is just um kind of post Hawk justification of why this is the answer um because the answer is already created it's already in the token window so it's

01:51:56

it's not actually being calculated here um and so if you are answering the question directly and immediately you are training the model to to try to basically guess the answer in a single token and that is just not going to work because of the finite amount of computation that happens per token that's why this answer on the right is significantly better because we are Distributing this computation across the answer we're actually getting the model to sort of slowly come to the answer

01:52:23

from the left to right we're getting intermediate results we're saying okay the total cost of oranges is four so 30 - 4 is 9 and so we're creating intermediate calculations and each one of these calculations is by itself not that expensive and so we're actually basically kind of guessing a little bit the difficulty that the model is capable of in any single one of these individual tokens and there can never be too much work in any one of these tokens computationally because then the model

01:52:51

won't be able to do that later at test time and so we're teaching the model here to spread out its reasoning and to spread out its computation over the tokens and in this way it only has very simple problems in each token and they can add up and then by the time it's near the end it has all the previous results in its working memory and it's much easier for it to determine that the answer is and here it is three so this is a significantly better label for our computation this would be really bad and

01:53:21

is teaching the model to try to do all the computation in a single token and it's really bad so uh that's kind of like an interesting thing to keep in mind is in your prompts uh usually don't have to think about it explicitly because uh the people at open AI have labelers and so on that actually worry about this and they make sure that the answers are spread out and so actually open AI will kind of like do the right thing so when I ask this question for chat GPT it's actually going to go very slowly it's

01:53:50

going to be like okay let's define our variables set up the equation and it's kind of creating all these intermediate results these are not for you these are for the model if the model is not creating these intermediate results for itself it's not going to be able to reach three I also wanted to show you that it's possible to be a bit mean to the model uh we can just ask for things so as an example I said I gave it the exact same uh prompt and I said answer the question in a single token

01:54:15

just immediately give me the answer nothing else and it turns out that for this simple um prompt here it actually was able to do it in single go so it just created a single I think this is two tokens right uh because the dollar sign is its own token so basically this model didn't give me a single token it gave me two tokens but it still produced the correct answer and it did that in a single forward pass of the network now that's because the numbers here I think are very simple and so I

01:54:42

made it a bit more difficult to be a bit mean to the model so I said Emily buys 23 apples and 177 oranges and then I just made the numbers a bit bigger and I'm just making it harder for the model I'm asking it to more computation in a single token and so I said the same thing and here it gave me five and five is actually not correct so the model failed to do all of this calculation in a single forward pass of the network it failed to go from the input tokens and then in a single forward pass of the

01:55:09

network single go through the network it couldn't produce the result and then I said okay now don't worry about the the token limit and just solve the problem as usual and then it goes all the intermediate results it simplifies and every one of these intermediate results here and intermediate calculations is much easier for the model and um it sort of it's not too much work per token all of the tokens here are correct and it arises the solution which is seven and I just couldn't squeeze all of this work

01:55:38

it couldn't squeeze that into a single forward passive Network so I think that's kind of just a cute example and something to kind of like think about and I think it's kind of again just elucidative in terms of how these uh models work the last thing that I would say on this topic is that if I was in practi is trying to actually solve this in my day-to-day life I might actually not uh trust that the model that all the intermediate calculations correctly here so actually probably what I do is

01:56:01

something like this I would come here and I would say use code and uh that's because code is one of the possible tools that chachy PD can use and instead of it having to do mental arithmetic like this mental arithmetic here I don't fully trust it and especially if the numbers get really big there's no guarantee that the model will do this correctly any one of these intermediates steps might in principle fail we're using neural networks to do mental arithmetic uh kind of like you doing

01:56:28

mental arithmetic in your brain it might just like uh screw up some of the intermediate results it's actually kind of amazing that it can even do this kind of mental arithmetic I don't think I could do this in my head but basically the model is kind of like doing it in its head and I don't trust that so I wanted to use tools so you can say stuff like use code and uh I'm not sure what happened there use code and so um like I mentioned there's a special tool and the uh the model can

01:56:56

write code and I can inspect that this code is correct and then uh it's not relying on its mental arithmetic it is using the python interpreter which is a very simple programming language to basically uh write out the code that calculates the result and I would personally trust this a lot more because this came out of a Python program which I think has a lot more correctness guarantees than the mental arithmetic of a language model uh so just um another kind of uh potential hint that if you

01:57:23

have these kinds of problems uh you may want to basically just uh ask the model to use the code interpreter and just like we saw with the web search the model has special uh kind of tokens for calling uh like it will not actually generate these tokens from the language model it will write the program and then it actually sends that program to a different sort of part of the computer that actually just runs that program and brings back the result and then the model gets access to that result and can

01:57:50

tell you that okay the cost of each apple is seven um so that's another kind of tool and I would use this in practice for yourself and it's um yeah it's just uh less error prone I would say so that's why I called this section models need tokens to think distribute your competition across many tokens ask models to create intermediate results or whenever you can lean on tools and Tool use instead of allowing the models to do all of the stuff in their memory so if they try to do it all

01:58:19

in their memory I don't fully trust it and prefer to use tools whenever possible I want to show you one more example of where this actually comes up and that's in counting so models actually are not very good at counting for the exact same reason you're asking for way too much in a single individual token so let me show you a simple example of that um how many dots are below and then I just put in a bunch of dots and Chach says there are and then it just tries to solve the problem in a

01:58:46

single token so in a single token it has to count the number of dots in its context window um and it has to do that in the single forward pass of a network and a single forward pass of a network as we talked about there's not that much computation that can happen there just think of that as being like very little competation that happens there so if I just look at what the model sees let's go to the LM go to tokenizer it sees uh this how many dots are below and then it turns out that these dots here this

01:59:18

group of I think 20 dots is a single token and then this group of whatever it is is another token and then for some reason they break up as this so I don't actually this has to do with the details of the tokenizer but it turns out that these um the model basically sees the token ID this this this and so on and then from these token IDs it's expected to count the number and spoiler alert is not 161 it's actually I believe 177 so here's what we can do instead uh we can say use code and you might expect

01:59:52

that like why should this work and it's actually kind of subtle and kind of interesting so when I say use code I actually expect this to work let's see okay 177 is correct so what happens here is I've actually it doesn't look like it but I've broken down the problem into a problems that are easier for the model I know that the model can't count it can't do mental counting but I know that the model is actually pretty good at doing copy pasting so what I'm doing here is

02:00:18

when I say use code it creates a string in Python for this and the task of basically copy pasting my input here to here is very simple because for the model um it sees this string of uh it sees it as just these four tokens or whatever it is so it's very simple for the model to copy paste those token IDs and um kind of unpack them into Dots here and so it creates this string and then it calls python routine. count and then it comes up with the correct answer so the python interpreter is doing the

02:00:54

counting it's not the models mental arithmetic doing the counting so it's again a simple example of um models need tokens to think don't rely on their mental arithmetic and um that's why also the models are not very good at counting if you need them to do counting tasks always ask them to lean on the tool now the models also have many other little cognitive deficits here and there and these are kind of like sharp edges of the technology to be kind of aware of over time so as an example the models

02:01:21

are not very good with all kinds of spelling related tasks they're not very good at it and I told you that we would loop back around to tokenization and the reason to do for this is that the models they don't see the characters they see tokens and they their entire world is about tokens which are these little text chunks and so they don't see characters like our eyes do and so very simple character level tasks often fail so for example uh I'm giving it a string ubiquitous and I'm asking it to print

02:01:50

only every third character starting with the first one so we start with U and then we should go every third so every so 1 2 3 Q should be next and then Etc so this I see is not correct and again my hypothesis is that this is again Dental arithmetic here is failing number one a little bit but number two I think the the more important issue here is that if you go to Tik tokenizer and you look at ubiquitous we see that it is three tokens right so you and I see ubiquitous and we can easily access the individual letters because we

02:02:24

kind of see them and when we have it in the working memory of our visual sort of field we can really easily index into every third letter and I can do that task but the models don't have access to the individual letters they see this as these three tokens and uh remember these models are trained from scratch on the internet and all these token uh basically the model has to discover how many of all these different letters are packed into all these different tokens and the reason we even use tokens is

02:02:49

mostly for efficiency uh but I think a lot of people areed interested to delete tokens entirely like we should really have character level or bite level models it's just that that would create very long sequences and people don't know how to deal with that right now so while we have the token World any kind of spelling tasks are not actually expected to work super well so because I know that spelling is not a strong suit because of tokenization I can again Ask it to lean On Tools so I can just say

02:03:14

use code and I would again expect this to work because the task of copy pasting ubiquitous into the python interpreter is much easier and then we're leaning on python interpreter to manipulate the characters of this string so when I say use code ubiquitous yes it indexes into every third character and the actual truth is u2s uqs uh which looks correct to me so um again an example of spelling related tasks not working very well a very famous example of that recently is how many R are there in strawberry and this

02:03:49

went viral many times and basically the models now get it correct they say there are three Rs in Strawberry but for a very long time all the state-of-the-art models would insist that there are only two RS in strawberry and this caused a lot of you know Ruckus because is that a word I think so because um it just kind of like why are the models so brilliant and they can solve math Olympiad questions but they can't like count RS in strawberry and the answer for that again is I've got built up to it kind of

02:04:17

slowly but number one the models don't see characters they see tokens and number two they are not very good at counting and so here we are combining the difficulty of seeing the characters with the difficulty of counting and that's why the models struggled with this even though I think by now honestly I think open I may have hardcoded the answer here or I'm not sure what they did but um uh but this specific query now works so models are not very good at spelling and there there's a bunch of other

02:04:46

little sharp edges and I don't want to go into all of them I just want to show you a few examples of things to be aware of and uh when you're using these models in practice I don't actually want to have a comprehensive analysis here of all the ways that the models are kind of like falling short I just want to make the point that there are some Jagged edges here and there and we've discussed a few of them and a few of them make sense but some of them also will just not make as much sense and they're kind

02:05:08

of like you're left scratching your head even if you understand in- depth how these models work and and good example of that recently is the following uh the models are not very good at very simple questions like this and uh this is shocking to a lot of people because these math uh these problems can solve complex math problems they can answer PhD grade physics chemistry biology questions much better than I can but sometimes they fall short in like super simple problems like this so here we go

02:05:35

9.11 is bigger than 9.9 and it justifies it in some way but obviously and then at the end okay it actually it flips its decision later so um I don't believe that this is very reproducible sometimes it flips around its answer sometimes gets it right sometimes get it get it wrong uh let's try again okay even though it might look larger okay so here it doesn't even correct itself in the end if you ask many times sometimes it gets it right too but how is it that the model can do so great at Olympiad grade problems but

02:06:10

then fail on very simple problems like this and uh I think this one is as I mentioned a little bit of a head scratcher it turns out that a bunch of people studied this in depth and I haven't actually read the paper uh but what I was told by this team was that when you scrutinize the activations inside the neural network when you look at some of the features and what what features turn on or off and what neurons turn on or off uh a bunch of neurons inside the neural network light up that are usually associated with Bible verses

02:06:41

U and so I think the model is kind of like reminded that these almost look like Bible verse markers and in a bip verse setting 9.11 would come after 99.9 and so basically the model somehow finds it like cognitively very distracting that in Bible verses 9.11 would be greater um even though here it's actually trying to justify it and come up to the answer with a math it still ends up with the wrong answer here so it basically just doesn't fully make sense and it's not fully understood and um

02:07:13

there's a few Jagged issues like that so that's why treat this as a as what it is which is a St stochastic system that is really magical but that you can't also fully trust and you want to use it as a tool not as something that you kind of like letter rip on a problem and copypaste the results okay so we have now covered two major stages of training of large language models we saw that in the first stage this is called the pre-training stage we are basically training on internet documents and when

02:07:41

you train a language model on internet documents you get what's called a base model and it's basically an internet document simulator right now we saw that this is an interesting artifact and uh this takes many months to train on thousands of computers and it's kind of a lossy compression of the internet and it's extremely interesting but it's not directly useful because we don't want to sample internet documents we want to ask questions of an AI and have it respond to our questions so for that we need an

02:08:07

assistant and we saw that we can actually construct an assistant in the process of a post training and specifically in the process of supervised fine-tuning as we call it so in this stage we saw that it's algorithmically identical to pre-training nothing is going to change the only thing that changes is the data set so instead of Internet documents we now want to create and curate a very nice data set of conversations so we want Millions conversations on all kinds of diverse topics between a human and an

02:08:42

assistant and fundamentally these conversations are created by humans so humans write the prompts and humans write the ideal response responses and they do that based on labeling documentations now in the modern stack it's not actually done fully and manually by humans right they actually now have a lot of help from these tools so we can use language models um to help us create these data sets and that's done extensively but fundamentally it's all still coming from Human curation at

02:09:11

the end so we create these conversations that now becomes our data set we fine tune on it or continue training on it and we get an assistant and then we kind of shifted gears and started talking about some of the kind of cognitive implications of what this assistant is like and we saw that for example the assistant will hallucinate if you don't take some sort of mitigations towards it so we saw that hallucinations would be common and then we looked at some of the mitigations of those hallucinations and

02:09:38

then we saw that the models are quite impressive and can do a lot of stuff in their head but we saw that they can also Lean On Tools to become better so for example we can lo lean on a web search in order to hallucinate less and to maybe bring up some more um recent information or something like that or we can lean on tools like code interpreter so the code can so the llm can write some code and actually run it and see the results so these are some of the topics we looked at so far um now what I'd like

02:10:06

to do is I'd like to cover the last and major stage of this Pipeline and that is reinforcement learning so reinforcement learning is still kind of thought to be under the umbrella of posttraining uh but it is the last third major stage and it's a different way of training language models and usually follows as this third step so inside companies like open AI you will start here and these are all separate teams so there's a team doing data for pre-training and a team doing training for pre-training and then

02:10:38

there's a team doing all the conversation generation in a in a different team that is kind of doing the supervis fine tuning and there will be a team for the reinforcement learning as well so it's kind of like a handoff of these models you get your base model the then you find you need to be an assistant and then you go into reinforcement learning which we'll talk about uh now so that's kind of like the major flow and so let's now focus on reinforcement learning the last major

02:11:03

stage of training and let me first actually motivate it and why we would want to do reinforcement learning and what it looks like on a high level so I would now like to try to motivate the reinforcement learning stage and what it corresponds to with something that you're probably familiar with and that is basically going to school so just like you went to school to become um really good at something we want to take large language models through school and really what we're doing is um we're um

02:11:30

we have a few paradigms of ways of uh giving them knowledge or transferring skills so in particular when we're working with textbooks in school you'll see that there are three major kind of uh pieces of information in these textbooks three classes of information the first thing you'll see is you'll see a lot of exposition um and by the way this is a totally random book I pulled from the internet I I think it's some kind of an organic chemistry or something I'm not sure uh but the

02:11:55

important thing is that you'll see that most of the text most of it is kind of just like the meat of it is exposition it's kind of like background knowledge Etc as you are reading through the words of this Exposition you can think of that roughly as training on that data so um and that's why when you're reading through this stuff this background knowledge and this all this context information it's kind of equivalent to pre-training so it's it's where we build sort of like a knowledge base of this

02:12:24

data and get a sense of the topic the next major kind of information that you will see is these uh problems and with their worked Solutions so basically a human expert in this case uh the author of this book has given us not just a problem but has also worked through the solution and the solution is basically like equivalent to having like this ideal response for an assistant so it's basically the expert is showing us how to solve the problem in it's uh kind of like um in its full form so as we are

02:12:55

reading the solution we are basically training on the expert data and then later we can try to imitate the expert um and basically um that's that roughly correspond to having the sft model that's what it would be doing so basically we've already done pre-training and we've already covered this um imitation of experts and how they solve these problems and the third stage of reinforcement learning is basically the practice problems so sometimes you'll see this is just a single practice problem here but of

02:13:27

course there will be usually many practice problems at the end of each chapter in any textbook and practice problems of course we know are critical for learning because what are they getting you to do they're getting you to practice uh to practice yourself and discover ways of solving these problems yourself and so what you get in a practice problem is you get a problem description but you're not given the solution but you are given the final answer answer usually in the answer key of the textbook and so you know the

02:13:55

final answer that you're trying to get to and you have the problem statement but you don't have the solution you are trying to practice the solution you're trying out many different things and you're seeing what gets you to the final solution the best and so you're discovering how to solve these problems so and in the process of that you're relying on number one the background information which comes from pre-training and number two maybe a little bit of imitation of human experts

02:14:20

and you can probably try similar kinds of solutions and so on so we've done this and this and now in this section we're going to try to practice and so we're going to be given prompts we're going to be given Solutions U sorry the final answers but we're not going to be given expert Solutions we have to practice and try stuff out and that's what reinforcement learning is about okay so let's go back to the problem that we worked with previously just so we have a concrete example to talk

02:14:48

through as we explore sort of the topic here so um I'm here in the Teck tokenizer because I'd also like to well I get a text box which is useful but number two I want to remind you again that we're always working with onedimensional token sequences and so um I actually like prefer this view because this is like the native view of the llm if that makes sense like this is what it actually sees it sees token IDs right okay so Emily buys three apples and two oranges each orange is $2 the total cost

02:15:17

of all the fruit is $13 what is the cost of each apple and what I'd like to what I like you to appreciate here is these are like four possible candidate Solutions as an example and they all reach the answer three now what I'd like you to appreciate at this point is that if I am the human data labeler that is creating a conversation to be entered into the training set I don't actually really know which of these conversations to um to add to the data set some of these conversations kind of

02:15:50

set up a system equations some of them sort of like just talk through it in English and some of them just kind of like skip right through to the solution um if you look at chbt for example and you give it this question it defines a system of variables and it kind of like does this little thing what we have to appreciate and uh differentiate between though is um the first purpose of a solution is to reach the right answer of course we want to get the final answer three that is the that is the important purpose here but

02:16:20

there's kind of like a secondary purpose as well where here we are also just kind of trying to make it like nice uh for the human because we're kind of assuming that the person wants to see the solution they want to see the intermediate steps we want to present it nicely Etc so there are two separate things going on here number one is the presentation for the human but number two we're trying to actually get the right answer um so let's for the moment focus on just reaching the final answer

02:16:45

if we're only care if we only care about the final answer then which of these is the optimal or the best prompt um sorry the best solution for the llm to reach the right answer um and what I'm trying to get at is we don't know me as a human labeler I would not know which one of these is best so as an example we saw earlier on when we looked at um the token sequences here and the mental arithmetic and reasoning we saw that for each token we can only spend basically a finite number of finite

02:17:18

amount of compute here that is not very large or you should think about it that way way and so we can't actually make too big of a leap in any one token is is maybe the way to think about it so as an example in this one what's really nice about it is that it's very few tokens so it's going to take us very short amount of time to get to the answer but right here when we're doing 30 - 4 IDE 3 equals right in this token here we're actually asking for a lot of computation to happen on that single individual

02:17:46

token and so maybe this is a bad example to give to the llm because it's kind of incentivizing it to skip through the calculations very quickly and it's going to actually make up mistakes make mistakes in this mental arithmetic uh so maybe it would work better to like spread out the spread it out more maybe it would be better to set it up as an equation maybe it would be better to talk through it we fundamentally don't know and we don't know because what is easy for you or I as or as human

02:18:12

labelers what's easy for us or hard for us is different than what's easy or hard for the llm it cognition is different um and the token sequences are kind of like different hard for it and so some of the token sequences here that are trivial for me might be um very too much of a leap for the llm so right here this token would be way too hard but conversely many of the tokens that I'm creating here might be just trivial to the llm and we're just wasting tokens like why waste all these tokens when

02:18:47

this is all trivial so if the only thing we care care about is the final answer and we're separating out the issue of the presentation to the human um then we don't actually really know how to annotate this example we don't know what solution to get to the llm because we are not the llm and it's clear here in the case of like the math example but this is actually like a very pervasive issue like for our knowledge is not lm's knowledge like the llm actually has a ton of knowledge of PhD in math and

02:19:15

physics chemistry and whatnot so in many ways it actually knows more than I do and I'm I'm potentially not utilizing that knowledge in its problem solving but conversely I might be injecting a bunch of knowledge in my solutions that the LM doesn't know in its parameters and then those are like sudden leaps that are very confusing to the model and so our cognitions are different and I don't really know what to put here if all we care about is the reaching the final solution and doing it economically

02:19:46

ideally and so long story short we are not in a good position to create these uh token sequences for the LM and they're useful by imitation to initialize the system but we really want the llm to discover the token sequences that work for it we need to find it needs to find for itself what token sequence reliably gets to the answer given the prompt and it needs to discover that in the process of reinforcement learning and of trial and error so let's see how this example would work like in reinforcement

02:20:20

learning okay so we're now back in the huging face inference playground and uh that just allows me to very easily call uh different kinds of models so as an example here on the top right I chose the Gemma 2 2 billion parameter model so two billion is very very small so this is a tiny model but it's okay so we're going to give it um the way that reinforcement learning will basically work is actually quite quite simple um we need to try many different kinds of solutions and we want to see which

02:20:49

Solutions work well or not so we're basically going to take the prompt we're going to run the model and the model generates a solution and then we're going to inspect the solution and we know that the correct answer for this one is $3 and so indeed the model gets it correct it says it's $3 so this is correct so that's just one attempt at DIS solution so now we're going to delete this and we're going to rerun it again let's try a second attempt so the model solves it in a bit

02:21:18

slightly different way right every single attempt will be a different generation because these models are stochastic systems remember that at every single token here we have a probability distribution and we're sampling from that distribution so we end up kind kind of going down slightly different paths and so this is a second solution that also ends in the correct answer now we're going to delete that let's go a third time okay so again slightly different solution but also gets it correct now we can actually repeat this

02:21:47

uh many times and so in practice you might actually sample thousand of independent Solutions or even like million solutions for just a single prompt um and some of them will be correct and some of them will not be very correct and basically what we want to do is we want to encourage the solutions that lead to correct answers so let's take a look at what that looks like so if we come back over here here's kind of like a cartoon diagram of what this is looking like we have a prompt and then we tried many different

02:22:15

solutions in parallel and some of the solutions um might go well so they get the right answer which is in green and some of the solutions might go poorly and may not reach the right answer which is red now this problem here unfortunately is not the best example because it's a trivial prompt and as we saw uh even like a two billion parameter model always gets it right so it's not the best example in that sense but let's just exercise some imagination here and let's just suppose that the um green ones are good and the

02:22:47

red ones are bad okay so we generated 15 Solutions only four of them got the right answer and so now what we want to do is basically we want to encourage the kinds of solutions that lead to right answers so whatever token sequences happened in these red Solutions obviously something went wrong along the way somewhere and uh this was not a good path to take through the solution and whatever token sequences there were in these Green Solutions well things went uh pretty well in this situation and so we want to

02:23:18

do more things like it in prompts like this and the way we encourage this kind of a behavior in the future is we basically train on these sequences um but these training sequencies now are not coming from expert human annotators there's no human who decided that this is the correct solution this solution came from the model itself so the model is practicing here it's tried out a few Solutions four of them seem to have worked and now the model will kind of like train on them and this corresponds

02:23:46

to a student basically looking at their Solutions and being like okay well this one worked really well so this is this is how I should be solving these kinds of problems and uh here in this example there are many different ways to actually like really tweak the methodology a little bit here but just to give the core idea across maybe it's simplest to just think about take the taking the single best solution out of these four uh like say this one that's why it was yellow uh so this is the the

02:24:12

solution that not only led to the right answer but may maybe had some other nice properties maybe it was the shortest one or it looked nicest in some ways or uh there's other criteria you could think of as an example but we're going to decide that this the top solution we're going to train on it and then uh the model will be slightly more likely once you do the parameter update to take this path in this kind of a setting in the future but you have to remember that we're going to run many different

02:24:40

diverse prompts across lots of math problems and physics problems and whatever wherever there might be so tens of thousands of prompts maybe have in mind there's thousands of solutions prompt and so this is all happening kind of like at the same time and as we're iterating this process the model is discovering for itself what kinds of token sequences lead it to correct answers it's not coming from a human annotator the the model is kind of like playing in this playground and it knows

02:25:10

what it's trying to get to and it's discovering sequences that work for it uh these are sequences that don't make any mental leaps uh they they seem to work reliably and statistically and uh fully utilize the knowledge of the model as it has it and so uh this is the process of reinforcement learning it's basically a guess and check we're going to guess many different types of solutions we're going to check them and we're going to do more of what worked in the future and that is

02:25:39

uh reinforcement learning so in the context of what came before we see now that the sft model the supervised fine tuning model it's still helpful because it still kind of like initializes the model a little bit into to the vicinity of the correct Solutions so it's kind of like a initialization of um of the model in the sense that it kind of gets the model to you know take Solutions like write out Solutions and maybe it has an understanding of setting up a system of equations or maybe it kind of like talks

02:26:06

through a solution so it gets you into the vicinity of correct Solutions but reinforcement learning is where everything gets dialed in we really discover the solutions that work for the model get the right answers we encourage them and then the model just kind of like gets better over time time okay so that is the high Lev process for how we train large language models in short we train them kind of very similar to how we train children and basically the only difference is that children go through

02:26:32

chapters of books and they do all these different types of training exercises um kind of within the chapter of each book but instead when we train AIS it's almost like we kind of do it stage by stage depending on the type of that stage so first what we do is we do pre-training which as we saw is equivalent to uh basically reading all the expository material so we look at all the textbooks at the same time and we read all the exposition and we try to build a knowledge base the second thing then is we go into the sft stage which

02:27:03

is really looking at all the fixed uh sort of like solutions from Human Experts of all the different kinds of worked Solutions across all the textbooks and we just kind of get an sft model which is able to imitate the experts but does so kind of blindly it just kind of like does its best guess uh kind of just like trying to mimic statistically the expert behavior and so that's what you get when you look at all the work Solutions and then finally in the last stage we do all the practice problems in the RL stage across all the

02:27:33

textbooks we only do the practice problems and that's how we get the RL model so on a high level the way we train llms is very much equivalent uh to the process that we train uh that we use for training of children the next point I would like to make is that actually these first two stat ages pre-training and surprise fine-tuning they've been around for years and they are very standard and everyone does them all the different llm providers it is this last stage the RL training that is a lot more

02:28:01

early in its process of development and is not standard yet in the field and so um this stage is a lot more kind of early and nent and the reason for that is because I actually skipped over a ton of little details here in this process the high level idea is very simple it's trial and there learning but there's a ton of details and little math mathematical kind of like nuances to exactly how you pick the solutions that are the best and how much you train on them and what is the prompt distribution

02:28:27

and how to set up the training run such that this actually works so there's a lot of little details and knobs to the core idea that is very very simple and so getting the details right here uh is not trivial and so a lot of companies like for example open and other LM providers have experimented internally with reinforcement learning fine tuning for llms for a while but they've not talked about it publicly um it's all kind of done inside the company and so that's why the paper from

02:28:55

Deep seek that came out very very recently was such a big deal because this is a paper from this company called DC Kai in China and this paper really talked very publicly about reinforcement learning fine training for large language models and how incredibly important it is for large language models and how it brings out a lot of reasoning capabilities in the models we'll go into this in a second so this paper reinvigorated the public interest of using RL for llms and gave a lot of the um sort of n-r details that are

02:29:28

needed to reproduce their results and actually get the stage to work for large langage models so let me take you briefly through this uh deep seek R1 paper and what happens when you actually correctly apply RL to language models and what that looks like and what that gives you so the first thing I'll scroll to is this uh kind of figure two here where we are looking at the Improvement in how the models are solving mathematical problems so this is the accuracy of solving mathematical problems on the a accuracy and then we

02:29:54

can go to the web page and we can see the kinds of problems that are actually in these um these the kinds of math problems that are being measured here so these are simple math problems you can um pause the video if you like but these are the kinds of problems that basically the models are being asked to solve and you can see that in the beginning they're not doing very well but then as you update the model with this many thousands of steps their accuracy kind of continues to climb so the models are

02:30:17

improving and they're solving these problems with a higher accuracy as you do this trial and error on a large data set of these kinds of problems and the models are discovering how to solve math problems but even more incredible than the quantitative kind of results of solving these problems with a higher accuracy is the qualitative means by which the model achieves these results so when we scroll down uh one of the figures here that is kind of interesting is that later on in the optimization the model seems to be uh

02:30:47

using average length per response uh goes up up so the model seems to be using more tokens to get its higher accuracy results so it's learning to create very very long Solutions why are these Solutions very long we can look at them qualitatively here so basically what they discover is that the model solution get very very long partially because so here's a question and here's kind of the answer from the model what the model learns to do um and this is an immerging property of new optimization

02:31:16

it just discovers that this is good for problem solving is it starts to do stuff like this wait wait wait that's Nota moment I can flag here let's reevaluate this step by step to identify the correct sum can be so what is the model doing here right the model is basically re-evaluating steps it has learned that it works better for accuracy to try out lots of ideas try something from different perspectives retrace reframe backtrack is doing a lot of the things that you and I are doing in the process

02:31:43

of problem solving for mathematical questions but it's rediscovering what happens in your head not what you put down on the solution and there is no human who can hardcode this stuff in the ideal assistant response this is only something that can be discovered in the process of reinforcement learning because you wouldn't know what to put here this just turns out to work for the model and it improves its accuracy in problem solving so the model learns what we call these chains of thought in your

02:32:09

head and it's an emergent property of the optim of the optimization and that's what's bloating up the response length but that's also what's increasing the accuracy of the problem problem solving so what's incredible here is basically the model is discovering ways to think it's learning what I like to call cognitive strategies of how you manipulate a problem and how you approach it from different perspectives how you pull in some analogies or do different kinds of things like that and

02:32:36

how you kind of uh try out many different things over time uh check a result from different perspectives and how you kind of uh solve problems but here it's kind of discovered by the RL so extremely incredible to see this emerge in the optimization without having to hardcode it anywhere the only thing we've given it are the correct answers and this comes out from trying to just solve them correctly which is incredible um now let's go back to actually the problem that we've been working with and

02:33:02

let's take a look at what it would look like uh for uh for this kind of a model what we call reasoning or thinking model to solve that problem okay so recall that this is the problem we've been working with and when I pasted it into chat GPT 40 I'm getting this kind of a response let's take a look at what happens when you give this same query to what's called a reasoning or a thinking model this is a model that was trained with reinforcement learning so this model described in this paper DC car1 is

02:33:31

available on chat. dec.com uh so this is kind of like the company uh that developed is hosting it you have to make sure that the Deep think button is turned on to get the R1 model as it's called we can paste it here and run it and so let's take a look at what happens now and what is the output of the model okay so here's it says so this is previously what we get using basically what's an sft approach a supervised funing approach this is like mimicking an expert solution this is what we get from the RL model okay let

02:34:02

me try to figure this out so Emily buys three apples and two oranges each orange cost $2 total is 13 I need to find out blah blah blah so here you you um as you're reading this you can't escape thinking that this model is thinking um is definitely pursuing the solution solution it deres that it must cost $3 and then it says wait a second let me check my math again to be sure and then it tries it from a slightly different perspective and then it says yep all that checks out I think that's

02:34:31

the answer I don't see any mistakes let me see if there's another way to approach the problem maybe setting up an equation let's let the cost of one apple be $8 then blah blah blah yep same answer so definitely each apple is $3 all right confident that that's correct and then what it does once it sort of um did the thinking process is it writes up the nice solution for the human and so this is now considering so this is more about the correctness aspect and this is more about the presentation aspect where

02:35:01

it kind of like writes it out nicely and uh boxes in the correct answer at the bottom and so what's incredible about this is we get this like thinking process of the model and this is what's coming from the reinforcement learning process this is what's bloating up the length of the token sequences they're doing thinking and they're trying different ways this is what's giving you higher accuracy in problem solving and this is where we are seeing these aha moments and these different

02:35:27

strategies and these um ideas for how you can make sure that you're getting the correct answer the last point I wanted to make is some people are a little bit nervous about putting you know very sensitive data into chat.com because this is a Chinese company so people don't um people are a little bit careful and Cy with that a little bit um deep seek R1 is a model that was released by this company so this is an open source model or open weights model it is available for anyone to download and use you will

02:35:57

not be able to like run it in its full um sort of the full model in full Precision you won't run that on a MacBook but uh or like a local device because this is a fairly large model but many companies are hosting the full largest model one of those companies that I like to use is called together. so when you go to together. you sign up and you go to playgrounds you can can select here in the chat deep seek R1 and there's many different kinds of other models that you can select here these are all state-of-the-art models so

02:36:27

this is kind of similar to the hugging face inference playground that we've been playing with so far but together. a will usually host all the state-of-the-art models so select DT car1 um you can try to ignore a lot of these I think the default settings will often be okay and we can put in this and because the model was released by Deep seek what you're getting here should be basically equivalent to what you're getting here now because of the randomness in the sampling we're going

02:36:52

to get something slightly different uh but in principle this should be uh identical in terms of the power of the model and you should be able to see the same things quantitatively and qualitatively uh but uh this model is coming from kind of a an American company so that's deep seek and that's the what's called a reasoning model now when I go back to chat uh let me go to chat here okay so the models that you're going to see in the drop down here some of them like 01 03 mini O3 mini High Etc they are talking about

02:37:21

uses Advanced reasoning now what this is referring to uses Advanced reasoning is it's referring to the fact that it was trained by reinforcement learning with techniques very similar to those of deep C car1 per public statements of opening ey employees uh so these are thinking models trained with RL and these models like GPT 4 or GPT 4 40 mini that you're getting in the free tier you should think of them as mostly sft models supervised fine tuning models they don't actually do this like thinking as as you

02:37:50

see in the RL models and even though there's a little bit of reinforcement learning involved with these models and I'll go that into that in a second these are mostly sft models I think you should think about it that way so in the same way as what we saw here we can pick one of the thinking models like say 03 mini high and these models by the way might not be available to you unless you pay a Chachi PT subscription of either $20 per month or $200 per month for some of the top models so we can pick a thinking

02:38:17

model and run now what's going to happen here is it's going to say reasoning and it's going to start to do stuff like this and um what we're seeing here is not exactly the stuff we're seeing here so even though under the hood the model produces these kinds of uh kind of chains of thought opening ey chooses to not show the exact chains of thought in the web interface it shows little summaries of that of those chains of thought and open kind of does this I think partly because uh they are worried

02:38:47

about what's called the distillation risk that is that someone could come in and actually try to imitate those reasoning traces and recover a lot of the reasoning performance by just imitating the reasoning uh chains of thought and so they kind of hide them and they only show little summaries of them so you're not getting exactly what you would get in deep seek as with respect to the reasoning itself and then they write up the solution so these are kind of like equivalent even though we're not seeing

02:39:12

the full under the hood details now in terms of the performance uh these models and deep seek models are currently rly on par I would say it's kind of hard to tell because of the evaluations but if you're paying $200 per month to open AI some of these models I believe are currently they basically still look better uh but deep seek R1 for now is still a very solid choice for a thinking model that would be available to you um sort of um either on this website or any other website because the model is open

02:39:41

weights you can just download it so that's thinking models so what is the summary so far well we've talked about reinforcement learning and the fact that thinking emerges in the process of the optimization on when we basically run RL on many math uh and kind of code problems that have verifiable Solutions so there's like an answer three Etc now these thinking models you can access in for example deep seek or any inference provider like together. a and choosing deep seek over there these

02:40:13

thinking models are also available uh in chpt under any of the 01 or O3 models but these GPT 4 R models Etc they're not thinking models you should think of them as mostly sft models now if you are um if you have a prompt that requires Advanced reasoning and so on you should probably use some of the thinking models or at least try them out but empirically for a lot of my use when you're asking a simpler question there's like a knowledge based question or something like that this might be

02:40:39

Overkill like there's no need to think 30 seconds about some factual question so for that I will uh sometimes default to just GPT 40 so empirically about 80 90% of my use is just gp4 and when I come across a very difficult problem like in math and code Etc I will reach for the thinking models but then I have to wait a bit longer because they're thinking um so you can access these on chat on deep seek also I wanted to point out that um AI studio. go.com even though it looks really busy really ugly because Google's just unable

02:41:11

to do this kind of stuff well it's like what is happening but if you choose model and you choose here Gemini 2.0 flash thinking experimental 01 21 if you choose that one that's also a a kind of early experiment experimental of a thinking model by Google so we can go here and we can give it the same problem and click run and this is also a thinking problem a thinking model that will also do something similar and comes out with the right answer here so basically Gemini also offers a thinking model anthropic

02:41:42

currently does not offer a thinking model but basically this is kind of like the frontier development of these llms I think RL is kind of like this new exciting stage but getting the details right is difficult and that's why all these models and thinking models are currently experimental as of 2025 very early 2025 um but this is kind of like the frontier development of pushing the performance on these very difficult problems using reasoning that is emerging in these optimizations one more connection that I wanted to bring up is

02:42:10

that the discovery that reinforcement learning is extremely powerful way of learning is not new to the field of AI and one place what we've already seen this demonstrated is in the game of Go and famously Deep Mind developed the system alphago and you can watch a movie about it um where the system is learning to play the game of go against top human players and um when we go to the paper underlying alphago so in this paper when we scroll down we actually find a really interesting plot um that I think uh is kind of

02:42:47

familiar uh to us and we're kind of like we discovering in the more open domain of arbitrary problem solving instead of on the closed specific domain of the game of Go but basically what they saw and we're going to see this in llms as well as this becomes more mature is this is the ELO rating of playing game of Go and this is leas dull an extremely strong human player and here what they are comparing is the strength of a model learned trained by supervised learning and a model trained by reinforcement

02:43:16

learning so the supervised learning model is imitating human expert players so if you just get a huge amount of games played by expert players in the game of Go and you try to imitate them you are going to get better but then you top out and you never quite get better than some of the top top top players of in the game of Go like LEL so you're never going to reach there because you're just imitating human players you can't fundamentally go beyond a human player if you're just imitating human

02:43:43

players but in a process of reinforcement learning is significantly more powerful in reinforcement learning for a game of Go it means that the system is playing moves that empirically and statistically lead to win to winning the game and so alphago is a system where it kind of plays against it itself and it's using reinforcement learning to create rollouts so it's the exact same diagram here but there's no prompt it's just uh because there's no prompt it's just a fixed game of Go but it's trying out

02:44:14

lots of solutions it's trying out lots of plays and then the games that lead to a win instead of a specific answer are reinforced they're they're made stronger and so um the system is learning basically the sequences of actions that empirically and statistically lead to winning the game and reinforcement learning is not going to be constrained by human performance and reinforcement learning can do significantly better and overcome even the top players like Lisa Dole and so uh probably they could have

02:44:45

run this longer and they just chose to crop it at some point because this costs money but this is very powerful demonstration of reinforcement learning and we're only starting to kind of see hints of this diagram in larger language models for reasoning problems so we're not going to get too far by just imitating experts we need to go beyond that set up these like little game environments and get let let the system discover reasoning traces or like ways of solving problems uh that are unique

02:45:14

and that uh just basically work well now on this aspect of uniqueness notice that when you're doing reinforcement learning nothing prevents you from veering off the distribution of how humans are playing the game and so when we go back to uh this alphao search here one of the suggested modifications is called move 37 and move 37 in alphao is referring to a specific point in time where alphago basically played a move that uh no human expert would play uh so the probability of this move uh to be

02:45:46

played by a human player was evaluated to be about 1 in 10th ,000 so it's a very rare move but in retrospect it was a brilliant move so alphago in the process of reinforcement learning discovered kind of like a strategy of playing that was unknown to humans and but is in retrospect uh brilliant I recommend this YouTube video um leis do versus alphao move 37 reactions and Analysis and this is kind of what it looked like when alphao played this move value that's a very that's a very surprising move I thought I thought it

02:46:20

was I thought it was a mistake when I see this move anyway so basically people are kind of freaking out because it's a it's a move that a human would not play that alphago played because in its training uh this move seemed to be a good idea it just happens not to be a kind of thing that a humans would would do and so that is again the power of reinforcement learning and in principle we can actually see the equivalence of that if we continue scaling this Paradigm in language models and what that looks like is kind of

02:46:48

unknown so so um what does it mean to solve problems in such a way that uh even humans would not be able to get how can you be better at reasoning or thinking than humans how can you go beyond just uh a thinking human like maybe it means discovering analogies that humans would not be able to uh create or maybe it's like a new thinking strategy it's kind of hard to think through uh maybe it's a holy new language that actually is not even English maybe it discovers its own language that is a lot better at

02:47:20

thinking um because the model is unconstrained to even like stick with English uh so maybe it takes a different language to think in or it discovers its own language so in principle the behavior of the system is a lot less defined it is open to do whatever works and it is open to also slowly Drift from the distribution of its training data which is English but all of that can only be done if we have a very large diverse set of problems in which the these strategy can be refined and perfected and so that is a lot of the

02:47:52

frontier LM research that's going on right now is trying to kind of create those kinds of prompt distributions that are large and diverse these are all kind of like game environments in which the llms can practice their thinking and uh it's kind of like writing you know these practice problems we have to create practice problems for all of domains of knowledge and if we have practice problems and tons of them the models will be able to reinforcement learning reinforcement learn on them and kind of

02:48:18

uh create these kinds of uh diagrams but in the domain of open thinking instead of a closed domain like game of Go there's one more section within reinforcement learning that I wanted to cover and that is that of learning in unverifiable domains so so far all of the problems that we've looked at are in what's called verifiable domains that is any candidate solution we can score very easily against a concrete answer so for example answer is three and we can very easily score these Solutions against the

02:48:48

answer of three either we require the models to like box in their answers and then we just check for equality of whatever is in the box with the answer or you can also use uh kind of what's called an llm judge so the llm judge looks at a solution and it gets the answer and just basically scores the solution for whether it's consistent with the answer or not and llms uh empirically are good enough at the current capability that they can do this fairly reliably so we can apply those kinds of techniques as well in any

02:49:16

case we have a concrete answer and we're just checking Solutions again against it and we can do this automatically with no kind of humans in the loop the problem is that we can't apply the strategy in what's called unverifiable domains so usually these are for example creative writing tasks like write a joke about Pelicans or write a poem or summarize a paragraph or something like that in these kinds of domains it becomes harder to score our different solutions to this problem so for example writing a joke

02:49:42

about Pelicans we can generate lots of different uh jokes of course that's fine for example we can go to chbt and we can get it to uh generate a joke about Pelicans uh so much stuff in their beaks because they don't bellan in backpacks what okay we can uh we can try something else why don't Pelicans ever pay for their drinks because they always B it to someone else haha okay so these models are not obviously not very good at humor actually I think it's pretty fascinating because I think humor is secretly very

02:50:15

difficult and the model have the capability I think anyway in any case you could imagine creating lots of jokes the problem that we are facing is how do we score them now in principle we could of course get a human to look at all these jokes just like I did right now the problem with that is if you are doing reinforcement learning you're going to be doing many thousands of updates and for each update you want to be looking at say thousands of prompts and for each prompt you want to be potentially looking at looking at

02:50:43

hundred or thousands of different kinds of generations and so there's just like way too many of these to look at and so um in principle you could have a human inspect all of them and score them and decide that okay maybe this one is funny and uh maybe this one is funny and this one is funny and we could train on them to get the model to become slightly better at jokes um in the context of pelicans at least um the problem is that it's just like way too much human time this is an unscalable strategy we need

02:51:12

some kind of an automatic strategy for doing this and one sort of solution to this was proposed in this paper uh that introduced what's called reinforcement learning from Human feedback and so this was a paper from open at the time and many of these people are now um co-founders in anthropic um and this kind of proposed a approach for uh basically doing reinforcement learning in unverifiable domains so let's take a look at how that works so this is the cartoon diagram of the core ideas involved so as I

02:51:42

mentioned the native approach is if we just set Infinity human time we could just run RL in these domains just fine so for example we can run RL as usual if I have Infinity humans I would I just want to do and these are just cartoon numbers I want to do 1,000 updates where each update will be on 1,000 prompts and in for each prompt we're going to have 1,000 roll outs that we're scoring so we can run RL with this kind of a setup the problem is in the process of doing this I will need to run one I will need to

02:52:13

ask a human to evaluate a joke a total of 1 billion times and so that's a lot of people looking at really terrible jokes so we don't want to do that so instead we want to take the arlef approach so um in our Rel of approach we are kind of like the the core trick is that of indirection so we're going to involve humans just a little bit and the way we cheat is that we basically train a whole separate neural network that we call a reward model and this neural network will kind of like imitate human

02:52:44

scores so we're going to ask humans to score um roll we're going to then imitate human scores using a neural network and this neural network will become a kind of simulator of human preferences and now that we have a neural network simulator we can do RL against it so instead of asking a real human we're asking a simulated human for their score of a joke as an example and so once we have a simulator we're often racist because we can query it as many times as we want to and it's all whole

02:53:16

automatic process and we can now do reinforcement learning with respect to the simulator and the simulator as you might expect is not going to be a perfect human but if it's at least statistically similar to human judgment then you might expect that this will do something and in practice indeed uh it does so once we have a simulator we can do RL and everything works great so let me show you a cartoon diagram a little bit of what this process looks like although the details are not 100 like super important it's just a core idea of

02:53:43

how this works so here I have a cartoon diagram of a hypothetical example of what training the reward model would look like so we have a prompt like write a joke about picans and then here we have five separate roll outs so these are all five different jokes just like this one now the first thing we're going to do is we are going to ask a human to uh order these jokes from the best to worst so this is uh so here this human thought that this joke is the best the funniest so number one joke this is

02:54:14

number two joke number three joke four and five so this is the worst joke we're asking humans to order instead of give scores directly because it's a bit of an easier task it's easier for a human to give an ordering than to give precise scores now that is now the supervision for the model so the human has ordered them and that is kind of like their contribution to the training process but now separately what we're going to do is we're going to ask a reward model uh about its scoring of

02:54:41

these jokes now the reward model is a whole separate neural network completely separate neural net um and it's also probably a transform uh but it's not a language model in the sense that it generates diverse language Etc it's just a scoring model so the reward model will take as an input The Prompt number one and number two a candidate joke so um those are the two inputs that go into the reward model so here for example the reward model would be taken this prompt and this joke now the output of a reward model is a single

02:55:14

number and this number is thought of as a score and it can range for example from Z to one so zero would be the worst score and one would be the best score so here are some examples of what a hypothetical reward model at some stage in the training process would give uh s scoring to these jokes so 0.1 is a very low score 08 is a really high score and so on and so now um we compare the scores given by the reward model with uh the ordering given by the human and there's a precise mathematical way to

02:55:47

actually calculate this uh basically set up a loss function and calculate a kind of like a correspondence here and uh update a model based on it but I just want to give you the intuition which is that as an example here for this second joke the the human thought that it was the funniest and the model kind of agreed right 08 is a relatively high score but this score should have been even higher right so after an update we would expect that maybe this score should have been will actually grow after an update of the network to be

02:56:15

like say 081 or something um for this one here they actually are in a massive disagreement because the human thought that this was number two but here the the score is only 0.1 and so this score needs to be much higher so after an update on top of this um kind of a supervision this might grow a lot more like maybe it's 0.15 or something like that um and then here the human thought that this one was the worst joke but here the model actually gave it a fairly High number so you might expect that

02:56:45

after the update uh this would come down to maybe 3 3.5 or something like that so basically we're doing what we did before we're slightly nudging the predictions from the models using a neural network training process and we're trying to make the reward model scores be consistent with human ordering and so um as we update the reward model on human data it becomes better and better simulator of the scores and orders uh that humans provide and then becomes kind of like the the neural the simulator of human

02:57:19

preferences which we can then do RL against but critically we're not asking humans one billion times to look at a joke we're maybe looking at th000 prompts and five roll outs each so maybe 5,000 jokes that humans have to look at in total and they just give the ordering and then we're training the model to be consistent with that ordering and I'm skipping over the mathematical details but I just want you to understand a high level idea that uh this reward model is do is basically giving us this scour and

02:57:45

we have a way of training it to be consistent with human orderings and that's how rhf works okay so that is the rough idea we basically train simulators of humans and RL with respect to those simulators now I want to talk about first the upside of reinforcement learning from Human feedback the first thing is that this allows us to run reinforcement learning which we know is incredibly powerful kind of set of techniques and it allows us to do it in arbitrary domains and including the ones that are unverifiable

02:58:16

so things like summarization and poem writing joke writing or any other creative writing really uh in domains outside of math and code Etc now empirically what we see when we actually apply rhf is that this is a way to improve the performance of the model and uh I have a top answer for why that might be but I don't actually know that it is like super well established on like why this is you can empirically observe that when you do rhf correctly the models you get are just like a little bit better um but as to why is I

02:58:46

think like not as clear so here's my best guess my best guess is that this is possibly mostly due to the discriminator generator Gap what that means is that in many cases it is significantly easier to discriminate than to generate for humans so in particular an example of this is um in when we do supervised fine-tuning right sft we're asking humans to generate the ideal assistant response and in many cases here um as I've shown it uh the ideal response is very simple to write but in many cases might not be so for

02:59:22

example in summarization or poem writing or joke writing like how are you as a human assist as a human labeler um supposed to give the ideal response in these cases it requires creative human writing to do that and so rhf kind of sidesteps this because we get um we get to ask people a significantly easier question as a data labelers they're not asked to write poems directly they're just given five poems from the model and they're just asked to order them and so that's just a much easier task for a

02:59:51

human labeler to do and so what I think this allows you to do basically is it um it kind of like allows a lot more higher accuracy data because we're not asking people to do the generation task which can be extremely difficult like we're not asking them to do creative writing we're just trying to get them to distinguish between creative writings and uh find the ones that are best and that is the signal that humans are providing just the ordering and that is their input into the system and then the

03:00:20

system in rhf just discovers the kinds of responses that would be graded well by humans and so that step of indirection allows the models to become a bit better so that is the upside of our LF it allows us to run RL it empirically results in better models and it allows uh people to contribute their supervision uh even without having to do extremely difficult tasks um in the case of writing ideal responses unfortunately our HF also comes with significant downsides and so um the main one is that basically we are doing reinforcement

03:00:56

learning not with respect to humans and actual human judgment but with respect to a lossy simulation of humans right and this lossy simulation could be misleading because it's just a it's just a simulation right it's just a language model that's kind of outputting scores and it might not perfectly reflect the opinion of an actual human with an actual brain in all the possible different cases so that's number one which is actually something even more subtle and devious going on that uh

03:01:21

really dramatically holds back our LF as a technique that we can really scale to significantly um kind of Smart Systems and that is that reinforcement learning is extremely good at discovering a way to game the model to game the simulation so this reward model that we're constructing here that gives the course these models are Transformers these Transformers are massive neurals they have billions of parameters and they imitate humans but they do so in a kind of like a simulation way now the problem

03:01:54

is that these are massive complicated systems right there's a billion parameters here that are outputting a single score it turns out that there are ways to gain these models you can find kinds of inputs that were not part of their training set and these inputs inexplicably get very high scores but in a fake way so very often what you find if you run our lch for very long so for example if we do 1,000 updates which is like say a lot of updates you might expect that your jokes are getting better and that you're getting like real

03:02:27

bangers about Pelicans but that's not EXA exactly what happens what happens is that uh in the first few hundred steps the jokes about Pelicans are probably improving a little bit and then they actually dramatically fall off the cliff and you start to get extremely nonsensical results like for example you start to get um the top joke about Pelicans starts to be the and this makes no sense right like when you look at it why should this be a top joke but when you take the the and you plug it into your reward model you'd

03:02:56

expect score of zero but actually the reward model loves this as a joke it will tell you that the the the theth is a score of 1. Z this is a top joke and this makes no sense right but it's because these models are just simulations of humans and they're massive neural lots and you can find inputs at the bottom that kind of like get into the part of the input space that kind of gives you nonsensical results these examples are what's called adversarial examples and I'm not going to go into the topic too much but these

03:03:24

are adversarial inputs to the model they are specific little inputs that kind of go between the nooks and crannies of the model and give nonsensical results at the top now here's what you might imagine doing you say okay the the the is obviously not score of one um it's obviously a low score so let's take the the the the the let's add it to the data set and give it an ordering that is extremely bad like a score of five and indeed your model will learn that the D should have a very low score and it will

03:03:52

give it score of zero the problem is that there will always be basically infinite number of nonsensical adversarial examples hiding in the model if you iterate this process many times and you keep adding nonsensical stuff to your reward model and giving it very low scores you can you'll never win the game uh you can do this many many rounds and reinforcement learning if you run it long enough will always find a way to gain the model it will discover adversarial examples it will get get really high scores uh with nonsensical

03:04:21

results and fundamentally this is because our scoring function is a giant neural nut and RL is extremely good at finding just the ways to trick it uh so long story short you always run rhf put for maybe a few hundred updates the model is getting better and then you have to crop it and you are done you can't run too much against this reward model because the optimization will start to game it and you basically crop it and you call it and you ship it um and uh you can improve the reward model

03:04:56

but you kind of like come across these situations eventually at some point so rhf basically what I usually say is that RF is not RL and what I mean by that is I mean RF is RL obviously but it's not RL in the magical sense this is not RL that you can run indefinitely these kinds of problems like where you are getting con correct answer you cannot gain this as easily you either got the correct answer or you didn't and the scoring function is much much simpler you're just looking at the

03:05:25

boxed area and seeing if the result is correct so it's very difficult to gain these functions but uh gaming a reward model is possible now in these verifiable domains you can run RL indefinitely you could run for tens of thousands hundreds of thousands of steps and discover all kinds of really crazy strategies that we might not even ever think about of Performing really well for all these problems in the game of Go there's no way to to beat to basically game uh the winning of a game or the

03:05:53

losing of a game we have a perfect simulator we know all the different uh where all the stones are placed and we can calculate uh whether someone has won or not there's no way to gain that and so you can do RL indefinitely and you can eventually be beat even leol but with models like this which are gameable you cannot repeat this process indefinitely so I kind of see rhf as not real RL because the reward function is gameable so it's kind of more like in the realm of like little fine-tuning

03:06:23

it's a little it's a little Improvement but it's not something that is fundamentally set up correctly where you can insert more compute run for longer and get much better and magical results so it's it's uh it's not RL in that sense it's not RL in the sense that it lacks magic um it can find you in your model and get a better performance and indeed if we go back to chat GPT the GPT 40 model has gone through rhf because it works well but it's just not RL in the same sense rlf is like a little fine

03:06:55

tune that slightly improves your model is maybe like the way I would think about it okay so that's most of the technical content that I wanted to cover I took you through the three major stages and paradigms of training these models pre-training supervised fine tuning and reinforcement learning and I showed you that they Loosely correspond to the process we already use for teaching children and so in particular we talked about pre-training being sort of like the basic knowledge acquisition of reading Exposition supervised fine

03:07:21

tuning being the process of looking at lots and lots of worked examples and imitating experts and practice problems the only difference is that we now have to effectively write textbooks for llms and AIS across all the disciplines of human knowledge and also in all the cases where we actually would like them to work like code and math and you know basically all the other disciplines so we're in the process of writing textbooks for them refining all the algorithms that I've presented on the

03:07:49

high level and then of course doing a really really good job at the execution of training these models at scale and efficiently so in particular I didn't go into too many details but these are extremely large and complicated distributed uh sort of um jobs that have to run over tens of thousands or even hundreds of thousands of gpus and the engineering that goes into this is really at the stateof the art of what's possible with computers at that scale so I didn't cover that aspect too much

03:08:19

but um this is very kind of serious and they were underlying all these very simple algorithms ultimately now I also talked about sort of like the theory of mind a little bit of these models and the thing I want you to take away is that these models are really good but they're extremely useful as tools for your work you shouldn't uh sort of trust them fully and I showed you some examples of that even though we have mitigations for hallucinations the models are not perfect and they will hallucinate still it's gotten better

03:08:47

over time and it will continue to get better but they can hallucinate in other words in in addition to that I covered kind of like what I call the Swiss cheese uh sort of model of llm capabilities that you should have in your mind the models are incredibly good across so many different disciplines but then fail randomly almost in some unique cases so for example what is bigger 9.11 or 9.9 like the model doesn't know but simultaneously it can turn around and solve Olympiad questions and so this is

03:09:14

a hole in the Swiss cheese and there are many of them and you don't want to trip over them so don't um treat these models as infallible models check their work use them as tools use them for inspiration use them for the first draft but uh work with them as tools and be ultimately respons responsible for the you know product of your work and that's roughly what I wanted to talk about this is how they're trained and this is what they are let's now turn to what are some of the future

03:09:44

capabilities of these models uh probably what's coming down the pipe and also where can you find these models I have a few blow points on some of the things that you can expect coming down the pipe the first thing you'll notice is that the models will very rapidly become multimodal everything I talked about above concerned text but very soon we'll have llms that can not just handle text but they can also operate natively and very easily over audio so they can hear and speak and also images so they can

03:10:10

see and paint and we're already seeing the beginnings of all of this uh but this will be all done natively inside inside the language model and this will enable kind of like natural conversations and roughly speaking the reason that this is actually no different from everything we've covered above is that as a baseline you can tokenize audio and images and apply the exact same approaches of everything that we've talked about above so it's not a fundamental change it's just uh it's

03:10:36

just a to we have to add some tokens so as an example for tokenizing audio we can look at slices of the spectrogram of the audio signal and we can tokenize that and just add more tokens that suddenly represent audio and just add them into the context windows and train on them just like above the same for images we can use patches and we can separately tokenize patches and then what is an image an image is just a sequence of tokens and this actually kind of works and there's a lot of early work in this direction and so we can

03:11:07

just create streams of tokens that are representing audio images as well as text and interpers them and handle them all simultaneously in a single model so that's one example of multimodality uh second something that people are very interested in is currently most of the work is that we're handing individual tasks to the models on kind of like a silver platter like please solve this task for me and the model sort of like does this little task but it's up to us to still sort of like organize a coherent execution of

03:11:35

tasks to perform jobs and the models are not yet at the capability required to do this in a coherent error correcting way over long periods of time so they're not able to fully string together tasks to perform these longer running jobs but they're getting there and this is improving uh over time but uh probably what's going to happen here is we're going to start to see what's called agents which perform tasks over time and you you supervise them and you watch their work and they come up to once in a

03:12:05

while report progress and so on so we're going to see more long running agents uh tasks that don't just take you know a few seconds of response but many tens of seconds or even minutes or hours over time uh but these uh models are not infallible as we talked about above so all of this will require supervision so for example in factories people talk about the human to robot ratio uh for automation I think we're going to see something similar in the digital space where we are going to be talking about

03:12:31

human to agent ratios where humans becomes a lot more supervisors of agent tasks um in the digital domain uh next um I think everything is going to become a lot more pervasive and invisible so it's kind of like integrated into the tools and everywhere um and in addition kind of like computer using so right now these models aren't able to take actions on your behalf but I think this is a separate bullet point um if you saw chpt launch the operator then uh that's one early example of that

03:13:04

where you can actually hand off control to the model to perform you know keyboard and mouse actions on your behalf so that's also something that that I think is very interesting the last point I have here is just a general comment that there's still a lot of research to potentially do in this domain main one example of that uh is something along the lines of test time training so remember that everything we've done above and that we talked about has two major stages there's first

03:13:27

the training stage where we tune the parameters of the model to perform the tasks well once we get the parameters we fix them and then we deploy the model for inference from there the model is fixed it doesn't change anymore it doesn't learn from all the stuff that it's doing a test time it's a fixed um number of parameters and the only thing that is changing is now the token inside the context windows and so the only type of learning or test time learning that the model has access to is the in

03:13:54

context learning of its uh kind of like uh dynamically adjustable context window depending on like what it's doing at test time so but I think this is still different from humans who actually are able to like actually learn uh depending on what they're doing especially when you sleep for example like your brain is updating your parameters or something like that right so there's no kind of equivalent of that currently in these models and tools so there's a lot of like um more wonky ideas I think that

03:14:18

are to be explored still and uh in particular I think this will be necessary because the context window is a finite and precious resource and especially once we start to tackle very long running multimodal tasks and we're putting in videos and these token windows will basically start to grow extremely large like not thousands or even hundreds of thousands but significantly beyond that and the only trick uh the only kind of trick we have Avail to us right now is to make the context Windows longer but I think that

03:14:46

that approach by itself will will not will not scale to actual long running tasks that are multimodal over time and so I think new ideas are needed in some of those disciplines um in some of those kind of cases in the main where these tasks are going to require very long contexts so those are some examples of some of the things you can um expect coming down the pipe let's now turn to where you can actually uh kind of keep track of this progress and um you know be up to date with the latest and grest

03:15:14

of what's happening in the field so I would say the three resources that I have consistently used to stay up to date are number one El Marina uh so let me show you El Marina this is basically an llm leader board and it ranks all the top models and the ranking is based on human comparisons so humans prompt these models and they get to judge which one gives a better answer they don't know which model is which they're just looking at which model is the better answer and you can calculate a ranking

03:15:43

and then you get some results and so what you can hear is what you can see here is the different organizations like Google Gemini for example that produce these models when you click on any one of these it takes you to the place where that model is hosted and then here we see Google is currently on top with open AI right behind here we see deep seek in position number three now the reason this is a big deal is the last column here you see license deep seek is an MIT license model it's open weights anyone can use

03:16:11

these weights uh anyone can download them anyone can host their own version of Deep seek and they can use it in what whatever way they like and so it's not a proprietary model that you don't have access to it's it's basically an open weight release and so this is kind of unprecedented that a model this strong was released with open weights so pretty cool from the team next up we have a few more models from Google and open Ai and then when you continue to scroll down you start to see some other Usual

03:16:37

Suspects so xai here anthropic with son it uh here at number 14 and um then meta with llama over here so llama similar to deep seek is an open weights model and so uh but it's down here as opposed to up here now I will say that this leaderboard was really good for a long time I do think that in the last few months it's become a little bit gamed um and I don't trust it as much as I used to I think um just empirically I feel like a lot of people for example are using a Sonet from anthropic and

03:17:15

that it's a really good model so but that's all the way down here um in number 14 and conversely I think not as many people are using Gemini but it's racking really really high uh so I think use this as a first pass uh but uh sort of try out a few of the models for your tasks and see which one performs better the second thing that I would point to is the uh AI news uh newsletter so AI news is not very creatively named but it is a very good newsletter produced by swix and friends so thank you for

03:17:46

maintaining it and it's been very helpful to me because it is extremely comprehensive so if you go to archives uh you see that it's produced almost every other day and um it is very comprehensive and some of it is written by humans and curated by humans but a lot of it is constructed automatically with llms so you'll see that these are very comprehensive and you're probably not missing anything major if you go through it of course you're probably not going to go through it because it's so long but I do think

03:18:12

that these summaries all the way up top are quite good and I think have some human oversight uh so this has been very helpful to me and the last thing I would point to is just X and Twitter uh a lot of um AI happens on X and so I would just follow people who you like and trust and get all your latest and greatest uh on X as well so those are the major places that have worked for me over time and finally a few words on where you can find the models and where can you use them so the first one I would say is for any of the biggest

03:18:41

proprietary models you just have to go to the website of that LM provider so for example for open a that's uh chat I believe actually works now uh so that's for open AI now for or you know for um for Gemini I think it's gem. google.com or AI Studio I think they have two for some reason that I don't fly understand no one does um for the open weights models like deep SE CL Etc you have to go to some kind of an inference provider of LMS so my favorite one is together together. a and I showed you that when

03:19:12

you go to the playground of together. a then you can sort of pick lots of different models and all of these are open models of different types and you can talk to them here as an example um now if you'd like to use a base model like um you know a base model then this is where I think it's not as common to find base models even on these inference providers they are all targeting assistants and chat and so I think even here I can't I couldn't see base models here so for base models I

03:19:40

usually go to hyperbolic because they serve my llama 3.1 base and I love that model and you can just talk to it here so as far as I know this is this is a good place for a base model and I wish more people hosted base models because they are useful and interesting to work with in some cases finally you can also take some of the models that are smaller and you can run them locally and so for example deep seek the biggest model you're not going to be able to run locally on your MacBook but there are

03:20:08

smaller versions of the deep seek model that are what's called distilled and then also you can run these models at smaller Precision so not at the native Precision of for example fp8 on deep seek or you know bf16 llama but much much lower than that um and don't worry if you don't fully understand those details but you can run smaller versions that have been distilled and then at even lower precision and then you can fit them on your uh computer and so you can actually run pretty okay models on

03:20:35

your laptop and my favorite I think place I go to usually is LM studio uh which is basically an app you can get and I think it kind of actually looks really ugly and it's I don't like that it shows you all these models that are basically not that useful like everyone just wants to run deep seek so I don't know why they give you these 500 different types of models they're really complicated to search for and you have to choose different distillations and different uh precisions and it's all

03:20:58

really confusing but once you actually understand how it works and that's a whole separate video then you can actually load up a model like here I loaded up a llama 3 uh2 instruct 1 billion and um you can just talk to it so I ask for Pelican jokes and I can ask for another one and it gives me another one Etc all of this that happens here is locally on your computer so we're not actually going to anywhere anyone else this is running on the GPU on the MacBook Pro so that's very nice and you

03:21:27

can then eject the model when you're done and that frees up the ram so LM studio is probably like my favorite one even though I don't I think it's got a lot of uiux issues and it's really geared towards uh professionals almost uh but if you watch some videos on YouTube I think you can figure out how to how to use this interface uh so those are a few words on where to find them so let me now loop back around to where we started the question was when we go to chashi pta.com and we enter some kind of a

03:21:54

query and we hit go what exactly is happening here what are we seeing what are we talking to how does this work and I hope that this video gave you some appreciation for some of the under the hood details of how these models are trained and what this is that is coming back so in particular we now know that your query is taken and is first chopped up into tokens so we go to to tick tokenizer and here where is the place in the in the um sort of format that is for the user query we basically put in our

03:22:28

query right there so our query goes into what we discussed here is the conversation protocol format which is this way that we maintain conversation objects so this gets inserted there and then this whole thing ends up being just a token sequence a onedimensional token sequence under the hood so Chachi PT saw this token sequence and then when we hit go it basically continues appending tokens into this list it continues the sequence it acts like a token autocomplete so in particular it gave us this response so we can basically just

03:23:00

put it here and we see the tokens that it continued uh these are the tokens that it continued with roughly now the question becomes okay why are these the tokens that the model responded with what are these tokens where are they coming from uh what are we talking to and how do we program this system and so that's where we shifted gears and we talked about the under thehood pieces of it so the first stage of this process and there are three stages is the pre-training stage which fundamentally has to do with just

03:23:29

knowledge acquisition from the internet into the parameters of this neural network and so the neural net internalizes a lot of Knowledge from the internet but where the personality really comes in is in the process of supervised fine-tuning here and so what what happens here is that basically the a company like openai will curate a large data set of conversations like say 1 million conversation across very diverse topics and there will be conversations between a human and an assistant and even though there's a lot

03:23:59

of synthetic data generation used throughout this entire process and a lot of llm help and so on fundamentally this is a human data curation task with lots of humans involved and in particular these humans are data labelers hired by open AI who are given labeling instructions that they learn and they task is to create ideal assistant responses for any arbitrary prompts so they are teaching the neural network by example how to respond to prompts so what is the way to think about what came back here like what is

03:24:33

this well I think the right way to think about it is that this is the neural network simulation of a data labeler at openai so it's as if I gave this query to a data Li open and this data labeler first reads all of the labeling instructions from open Ai and then spends 2 hours writing up the ideal assistant response to this query and uh giving it to me now we're not actually doing that right because we didn't wait two hours so what we're getting here is a neural network simulation of that

03:25:05

process and we have to keep in mind that these neural networks don't function like human brains do they are different what's easy or hard for them is different from what's easy or hard for humans and so we really are just getting a simulation so here I shown you this is a token stream and this is fundamentally the neural network with a bunch of activations and neurons in between this is a fixed mathematical expression that mixes inputs from tokens with parameters of the model and they get mixed up and

03:25:36

get you the next token in a sequence but this is a finite amount of compute that happens for every single token and so this is some kind of a lossy simulation of a human that is kind of like restricted in this way and so whatever the humans write the language model is kind of imitating on this token level with only this this specific computation for every single token and sequence we also saw that as a result of this and the cognitive differences the models will suffer in a variety of ways and uh you have to be very careful with

03:26:10

their use so for example we saw that they will suffer from hallucinations and they also we have the sense of a Swiss model of the LM capabilities where basically there's like holes in the cheese sometimes the models will just arbitrarily like do something dumb uh so even though they're doing lots of magical stuff sometimes they just can't so maybe you're not giving them enough tokens to think and maybe they're going to just make stuff up because they're mental arithmetic breaks uh maybe they

03:26:36

are suddenly unable to count number of letters um or maybe they're unable to tell you that 911 9.11 is smaller than 9.9 and it looks kind of dumb and so so it's a Swiss cheese capability and we have to be careful with that and we saw the reasons for that but fundamentally this is how we think of what came back it's again a simulation of this neural network of a human data labeler following the labeling instructions at open a so that's what we're getting back now I do think that the uh things change a little

03:27:11

bit when you actually go and reach for one of the thinking models like o03 mini and the reason for that is that GPT 40 basically doesn't do reinforcement learning it does do rhf but I've told you that rhf is not RL there's no there's no uh time for magic in there it's just a little bit of a fine-tuning is the way to look at it but these thinking models they do use RL so they go through this third state stage of perfecting their thinking process and discovering new thinking strategies and

03:27:46

uh solutions to problem solving that look a little bit like your internal monologue in your head and they practice that on a large collection of practice problems that companies like openi create and curate and um then make available to the LMS so when I come here and I talked to a thinking model and I put in this question what we're seeing here is not anymore just the straightforward simulation of a human data labeler like this is actually kind of new unique and interesting um and of course open is not

03:28:16

showing us the under thehood thinking and the chains of thought that are underlying the reasoning here but we know that such a thing exists and this is a summary of it and what we're getting here is actually not just an imitation of a human data labeler it's actually something that is kind of new and interesting and exciting in the sense that it is a function of thinking that was emergent in a simulation it's not just imitating human data labeler it comes from this reinforcement learning

03:28:41

process and so here we're of course not giving it a chance to shine because this is not a mathematical or a reasoning problem this is just some kind of a sort of creative writing problem roughly speaking and I think it's um it's a a question an open question as to whether the thinking strategies that are developed inside verifiable domains transfer and are generalizable to other domains that are unverifiable such as create writing the extent to which that transfer happens is unknown in the field

03:29:12

I would say so we're not sure if we are able to do RL on everything that is very verifiable and see the benefits of that on things that are unverifiable like this prompt so that's an open question the other thing that's interesting is that this reinforcement learning here is still like way too new primordial and nent so we're just seeing like the beginnings of the hints of greatness uh in the reasoning problems we're seeing something that is in principle capable of something like the equivalent of move

03:29:40

37 but not in the game of Go but in open domain thinking and problem solving in principle this Paradigm is capable of doing something really cool new and exciting something even that no human has thought of before in principle these models are capable of analogies no human has had so I think it's incredibly exciting that these models exist but again it's very early and these are primordial models for now um and they will mostly shine in domains that are verifiable like math en code Etc so very

03:30:10

interesting to play with and think about and use and then that's roughly it um um I would say those are the broad Strokes of what's available right now I will say that overall it is an extremely exciting time to be in the field personally I use these models all the time daily uh tens or hundreds of times because they dramatically accelerate my work I think a lot of people see the same thing I think we're going to see a huge amount of wealth creation as a result of these models be aware of some of their shortcomings even

03:30:41

with RL models they're going to suffer from some of these use it as a tool in a toolbox don't trust it fully because they will randomly do dumb things they will randomly hallucinate they will randomly skip over some mental arithmetic and not get it right um they randomly can't count or something like that so use them as tools in the toolbox check their work and own the product of your work but use them for inspiration for first draft uh ask them questions but always check and verify and you will

03:31:09

be very successful in your work if you do so uh so I hope this video was useful and interesting to you I hope you had it fun and uh it's already like very long so I apologize for that but I hope it was useful and yeah I will see you later