(no title)

79 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=2OH6PWPRYFO
https://www.youtube.com/watch?v=2oH6PWPrYFo

SUMMARY

The lecture discusses the complexities of post-training in language models, particularly focusing on the transition from models like GPT-3 to more advanced systems like ChatGPT. It emphasizes the importance of high-quality data collection and the challenges involved in fine-tuning models to improve instruction-following capabilities while managing issues like overfitting and model collapse.

- *Post-training is a nuanced process that builds on pre-training, requiring explicit data collection and engineering.*
- *Instruction following has improved significantly from GPT-3 to models like GPT-3.5 and GPT-4, allowing for more complex prompts and responses.*
- *High-quality supervised fine-tuning (SFT) data is essential for effective post-training, with a focus on the quality of input-output pairs.*
- *The shift towards using expert annotators and higher-quality data has become more prevalent in the industry to improve model performance.*
- *Reinforcement Learning from Human Feedback (RLHF) is a key technique in post-training, but it can lead to overfitting and model collapse if not managed properly.*
- *The DPO (Direct Preference Optimization) algorithm offers a simpler alternative to traditional RLHF methods, focusing on adjusting model outputs based on positive and negative feedback.*
- *Challenges in data collection and the influence of annotator demographics can significantly impact model behavior and performance.*
- *The lecture highlights the ongoing need for careful data management and algorithmic strategies to ensure effective model training and deployment.*

Okay, welcome back. Um, we're going to move away from pre-training now, and I think we'll get into uh, in my opinion, some of the messier parts um, of language modeling. I mean, data is always pretty messy, but I think post-training and everything that goes along with it is really, you know, kind of artisal in many ways. Um, so to set the stage, thus far I think you can kind of get to a souped-up version of GPT3. Uh, I'm sure you can use more flops, you can get more data, so you can make something a little bit better than GPT3, but ultimately um, the amount of utility you can get from something like GPT3, which is a very big strong base model, is ultimately a little bit limited, right? Um, when we went from GPT3 to something like chat GPT, um, I'm sure if your first exposure to to these systems was chat GPT, it was like kind of amazing. If you go back and interact with GPT3, you'll be like, what what is this thing? What is the point of this thing, right? And the only thing you could really do with it was things like copywriting or these like sort of fun little tasks where you didn't really need reliability or instruction following. So today we are going to get from GPT3 to something that is getting basically close to chat GPT. Right? So this is the very first part that we need. Um and in the next lecture we will go from chat GPT to um GPT01 or you know any of the thinking models that we have today. Right? So those are going to be

the two parts and the process that will take us from the left to the right will be a process that you know usually people call post training.

Um, and I want to sort of emphasize just kind of how remarkable it is that we can do instruction following. Um, instruction following is um, this process of being able to put in a pretty long complicated prompt and get back, you know, very reasonable answers. Um, at the GP3 GPT3 level, fine grain control of these systems was basically impossible. Like you would put in a bunch of fot prompts and you'd say, well, if my examples were good enough, I'd get something reasonable. um in practice your ability to steer these models are limited. Um but once GPT 3.5 and GPT4 comes around, you start getting these examples of very long sort of programmatic prompts and then the model really oneshotting um the whole prompt.

And so what does it what do you really need in order to get this kind of instruction following um to work? And so pre-training is always important. It's kind of the basis on which we build all of this, right? So if we ignore pre-training and we just try to post-train our way to victory, we will get none of the things that we want, absolutely none of it. So pre-training is critical because it scales in this very diverse broad way. Um but once we have pre-training, we have to then extract the kinds of behaviors that we want out of this kind of primordial soup that is pre-training, right? Um, and to do that we are going to have to do much more explicit data collection, much more explicit steering and a lot of messy engineering that in some ways we didn't have to do so far. I mean the architecture stuff you might think is messy. I think it really doesn't compare to the messiness of of real world post training. Um, and so you know we're going to go and collect some data or you know we will talk as if we are going to go and collect some data. Um, we're going to talk about what that data looks like. um how to train with it um and what the scale of that looks like.

Um and I will preface the rest of this lecture and say, you know, information about frontier post training is honestly pretty sparse in many many ways. If you look at the the uh materials that I'm going to reference today, um a lot of it's going to be actually pretty old. It's going to be before the competition from chat GPT started to heat up. Um so for example the RHF paper um learning to summarize from human feedback you know that that's actually incredibly detailed. I'll point to that today in lecture you know if you're interested I suggest you go read some of the appendices you know it contains things like detailed annotation guidelines how they were instructing people to to you know give feedback to the models.

Similarly with anthropics hh paper back in uh 2022. Um and then you know now that competition has heated up basically none of the vendors want to release any information about their post-training processes right the data is very much a trade secret um and so we are not going to be able to talk about annotation guidelines and things like that for anything that resembles a modern language model um in you know we do have closed source oh sorry open source models um but on the other hand a lot of the open- source uh recipes rely on distillation I think its nature is very different from what the uh frontier labs are doing in terms of like human data collection which is really a very tricky and messy thing and and I'll try to point out some of that but it'll be difficult to talk about some of that uh very comprehensively um okay and then uh you know just as a example of this you know I was doing some pre-ressearch for this lecture um earlier um and I found this very fun example back in 2023 about you know the trade secretness of post- training data um where I guess you know scale AI's internal

documents got leaked at some point and they had a whole bunch of very frank conversations about what was happening in post- trainining data land and one of them was like well they were trying to make Google barred better but they couldn't figure out why you know GPT4 was so much better and so they said well let's like we need to like reverse engineer what's happening and we need to get our uh annotators to produce responses that are more detailed uh or and better than GPT4 right so these kinds of like competitive dynamics are happening all the time and that's also why some of this detail is going to be missing from today's lecture. Um that said, algorithm wise, we do have a pretty good handle on what is happening, right? But algorithms are not really the the secret sauce. It's not where a lot of the leverage is. It's going to be the data. Um so if we look at um the sort of RLHF papers, we see a very straightforward recipe, right? Two-part recipe. The first part is you go and collect some demonstration data, right?

So for every sort of prompt, you want some annotator to provide supervised fine-tuning data. So you know, reference responses for that prompt, right? And you're going to do some fine-tune. Then after you have that, you're going to do some reinforcement learning of some kind to try to shape the behavior of the model to be sort of more aligned or more in line with what sort of humans think are good responses. Right? So I'm going to start by talking about this very first phase, which is the SFT part of um post- training. Um, and this is the part that's really basically all about data, right? We all know how to SFT a model that's basically exactly the same as pre-training. So, the only real difference with some minor variation is going to be the training data. So, I will focus on the training data almost entirely for this part um of the lecture. Okay. Um, with a little bit of discussion about the method right after.

Okay. So, we will go through the data. We're going to, you know, look at some examples. I'll talk about some historical developments and what kind of the trends are at least in open- source post-training data land. Um, and then I will talk about some pitfalls, some ways in which collecting this kind of data is much harder uh than you think. And you know, if you've taken classes like 224N, um, you're going to sort of find some things that are very reminiscent of, uh, issues that you see when you start building, you know, classic supervised deep learning models from scratch. Um, so let me take a moment to pause, um, and then just talk about the progression of SFT data over the years. Um, I think I'm going to go, you know, top left to bottom right. That's roughly my chronological ordering. And there's been a lot of different tricks and a lot of things that people have done in the open- source language modeling land to try to you know both improve system performance and to to try to understand what factors uh of this kind of data are useful to uh train models. I think really the oldest and you know in some ways the most visionary of this is things like Fla. Um this was used to train the T5 model by Google. um their rough idea for this was to say well NLP people have already collected a bunch of supervised data sets of like inputs and outputs we should just train the model on all of these right reasonable uh thing to try um turned out for various reasons this was not the right thing to do I'll talk about that later um self-instruct and other sort of follow-ups to that um this was also sort of very forward-looking um in terms of its thinking was saying well why can't we use the model itself to generate data well models are getting better all the time. Um, and in fact, they might even be better than some of our annotators.

Why don't we get models to, you know, write very um very good highquality responses to our inputs? And that's sort of the self-instruct idea. And then there came a bunch of sort of distillation style approaches. Um, Alpaca is one of them. Um, the Berkeley folks had one called Vikunia in which they were using um sort of

online user shared prompts as sort of the way to to as the inputs to distill on. Um, and then there were sort of other efforts that were more entirely human-driven like open assistant. You can think of this like kind of like Wikipedia. It was a bunch of people getting together to write prompts and responses. Um, and then you have sort of a a new generation wizard LM and Tulu 3 and others of saying, well, we know that language models are very good synthetic data generators. Why don't we come up with increasingly complicated ways of generating uh instruction following data using language models? And then finally, I I'll touch briefly on this. Um, SFTP and sort of post- training has changed yet again in recent years because I think the focus is less on chat and much more on agents and tool use. And so there's a now a new generation of sort of SFT pipelines that are designed to generate sort of tool use and agentic data. Right? So this is roughly you know a bunch of different developments in uh post- training data land at least in the open source world. So in the closed source world where you know well some of the companies may be distilling each other but where you're also collecting human data you know how you're collecting the human data is a big important ingredient that's not reflected in any of this discussion.

Okay I can pause here in case anyone has uh questions um before I move on to well what we're going to do next is look at some of the details of these data sets. >> Good. Oh yeah. So basically for post training we need the input output output pairs right >> that's right input output pairs like >> does the correctness of those input output matters like how hard does it >> yes so the question was how much does the correctness of the input output pairs matter um that's a nuanced question I mean I think the the top level answer is you should go collect the highest quality responses you can because the bad stuff will teach the model to do bad stuff on the other hand it turns out that you can SFT models on all sorts of strange things and they will learn to follow instructions. Um I think Percy's former student had a paper in which you're like training uh models without even the the responses or something and you can train models to like you know be instruction following.

And so there's a lot of like pre-training generalization behaviors that let you get away with worse quality data. So it's a it's actually a pretty nuanced question um what you need in uh instruction following data. Okay, good. So um I want to maybe start with fawn. Um you know I think fawn is very much like a forwardlooking uh data set for its time right like it sort of established this idea of doing this like multitask post training. Just if you're going to do downstream tasks go collect all the downstream tasks and train on it. Very reasonable but smart idea at the time. Um but if we look at what's in fla we actually find that you know it's a little bit of a strange data set. um you know you have this thing that's like well here's an email and then there's this like thing that gets appended which is like write a subject line for this email and then the subject line goes on the right um this is a very strange thing like I don't think you've ever probably prompted you know a model to do a task like this let alone the instructions at the very end right and the reason why you get something like this is because fla is generated from existing data sets right so there's a data set called enron emails this is basically a way to turn enron emails into a supervised task, right? Because it has a body and a subject. You can make this subject a prediction task. Um, you know, similarly, you know, you have summarization task. This is, um, I don't know if this was taken from CNN Daily Mail or Exxon, but it was taken from one of the big major summarization data sets, right? Write highlights for this article. Take the article part of a summarization data set, put the summ summary on the right. But you immediately notice, you know, compared to something like ChatgPT, the summaries are very short. Um in fact if you look really closely the summaries are often hallucinated like there's a bunch of details that are not in the inputs. Um and in some ways you know related to this question right the original NLP data sets from which font was built was not the highest of quality right and so when you train on

this thing you also inherit a whole bunch of sort of deficiencies from those data sets and the unnaturalness of the structure of these data sets also sort of carry down all the way. Um and so this is a very smart idea. Font is a very smart idea to create this big data set out of um of these uh data set existing benchmarks.

But really the unnaturalenness really gets you. Oh and I'll I'll mention one more thing going back one slide. I think when Flawn was originally constructed sort of part of the theory of how post-training should work was that you needed scale, right? Like in pre-training we know we need a lot of scale. So you might also think that in post training we need a ton of scale to unlock capabilities. Um and so because of that they took tons of data sets and lots of data and had this gigantic data set. We'll see sort of later on that a lot of these data sets just get really much smaller and it still works. And so really part of the change from fawn to later data sets is this observation that if you have a sufficiently strong big pre-trained model, you can actually get away with very few highquality examples because of pre-training generalization is going to get you quite a bit of the way. And so it's kind of the wrong point in some ways on the quality uh quantity trade-off although you know there was no way to know until we sort of explored the full space of these data sets. Um and then I think you know after that the the maybe next point in how sort of instruction tuning data sets our understanding of it changed um was after chat GPD came out um some of my students um you know did this thing called alpaca where they distilled chat GPT traces to get input output pairs right um so this is an example from that data set you get you know more naturalook inputs um longer sort of chatty outputs because of course it's it's taken from chat GPT Um and what we found um was that these kinds of examples now reliably induced kind of um chatgpt like behavior on models. uh importantly you know only when we did it to the original uh llama models um so pre-training post-training you know both of them need to work out right but we did find you know that this was was kind of the point at which these things started to work and so I think people realized at this point that if you got this kind of sort of chat style data it was actually not that hard to build chatgpt style systems you know getting the details right is still hard but you know to get most of the way so then I think there was this big push to create essentially the highest quality human-driven version of this these kinds of data sets. Um to give you a little bit more context at the time I think after chat GPD came out and alpaca came out I think there was this enormous optimism that if only we could uh collect a sufficiently high quality and large you know instruction tuning data set we could then sort of catch up and match a lot of the performance of these big closed source labs. there's a lot of sort of energy and excitement uh to building these data sets. And so in that context, open assistant is this big crowdsourced effort to build such a data set, right? So a lot of volunteers got together and they said, well, we're going to, you know, come up with really hard and interesting prompts. We're going to come up with good highquality responses and we're going to do it at scale in the same way that Wikipedia, you know, managed to produce something very high quality at scale. Um, a very admirable, very impressive effort um that generated actually a decent amount of data. I forget if it's like 10,000 or or more examples um before sort of stalling out as a as a as a as a project. I can I can maybe you know talk about that if someone's interested. Um but if you look now we have you know chat style inputs very long detailed sort of high quality expert responses on the right. Good stuff. Um I'll talk about some of the pitfalls later as well. Okay. Finally I've taken this from some of the newest of the instruction tuning data sets. we've really started to move as a field or you know in terms of the products that people want from just a chat interface to something that's like a full agent system right so we don't want just textual responses we want tool calls we want you know uh to-do lists if you've ever used cloud code or or codeex you you know that like it produces a to-do list to like check off as it goes um we want all of those things right and so SFT data now if you look at for example Neotron so that's Nvidia's open source efforts their SF data uh a

big chunk of it is these kinds of AsiTentic SFT uh examples where not only do you have the responses which are provided in the sort of uh assistant field here um but also tool calls that can happen in parallel alongside the text right and this is SFT data so this is you know explicitly sort of supervised into the model um and so now we're sort of moving much more towards these like structured uh formats okay and I've been talking about this as I've gone through but we've seen sort of big sort of higher level transitions in how we think about and build these kinds of data sets. It's useful to roughly internalize these sort of like perspectives. Um, one of them is kind of this chattiness thing. Um, it's hard for me to like fully articulate what this is, but there's a sense in which the classic NLP data sets, even the high quality ones, were very like input to like programmatic output almost, right?

Um, but people don't really want to talk to an NLP benchmark. They want to talk to people. And so there was this big shift that has happened towards more detailed human-like responses. Um there's a big shift towards higher quality annotators and much more detail. Open assistant is a good example of this of having experts write your responses. And then finally, you know, tool use and figuring out the right sort of, you know, interface and API for all this has been a big sort of final shift that's been happening. I'm sure things will keep changing but these are kind of some highle you know factors uh to think about uh in terms of how SFT data has changed over the last few years.

Okay. Um I'll talk about sort of all sorts of you know pitfalls or challenges in a moment. Um any questions on on the specifics of data sets and things like this? Good. Okay. Um so now I think let's say you are put in charge of collecting human annotation data for SFT. um what kinds of sort of things should you be mindful of right um there's lots of formatting things right that you might want to think about there's also things like knowledge how much knowledge should be put into the responses how how much data points do you want and like how do you deal with kind of uh safety of your model like what kinds of safety data should you be collecting um and I think one thing I want to highlight is length is and style variation in general is a very very big part of this like post-training process.

People say, you know, Claude has a different tone than chat GPT, you know, chat GPT is too chatty, so on and so forth. All of these are, you know, conscious decisions that are made by the data collection folks. Um, and also we see like very wide variation if you look at the data sets in terms of the length of the responses. Um and one thing I do want to kind of highlight is when you go and evaluate preferences um these stylistic factors matter a ton right so people will get very easily tricked not I don't know if tricked is the right word you know they will very often select responses that have bulletointed lists in their outputs uh or they will select responses that have you know more and longer detail I think part of this is very natural especially if you're in an evaluation setting where you put in front of two examples and someone asks you which one's better. I think you know more detailed and list-like responses are generally good. Um but this does induce very explicit distortions in the kinds of you know uh tone that a chatbot has uh relative to people. Um these factors Oh yeah.

>> So how do complex matters in the data set? >> I'll talk about them later. >> Um and you can ask me questions if I don't fully answer your question there. Um so uh right but I think the thing about style right and the thing to be careful about is if you're looking at engagement signals as most of these companies are um it's very very easy to fool yourself into thinking you're getting better data when in reality um your model's capabilities are not

changing. So if you look at sort of all of these different kinds of like you know uh post-training data sets that you can train on um you'll see big variations in preferences when you train on some of these things. So that's kind of the right column over there alpaca eval for example but really it doesn't necessarily change you know standard benchmarking evaluations. So your models aren't necessarily smarter because you've trained on certain kinds of post- training data at least in these ones. Um but you can very much shift sort of the engagement signals. And so this is kind of telling you you know you want to think about style control separately from capabilities control.

Okay. The other thing and this I think is is a quite in many ways important sort of thing to think about in terms of how we collect and think about post- training data. Um I think open assistant is very instructive because it was an effort that was you know intended to be the highest quality in some ways uh post-training data. Um, and so you often see things like this, like for example, this uh reference, sorry, this response contains a reference to an article. This JM Michelle L, the pay of corporate executives and blah blah blah, right?

Um, and so this is actually teaching the model kind of two different things at once. Let's say you you're uh sfting on this data. So first of all, you know, you're doing next token prediction. So it's teaching the model about, you know, this Bivvens J and Michelle L citation. Um, it's also telling the model another thing which is when you're giving a response, a good thing to do is to give a reference, right? So these are two different mechanisms that are sort of going on at once. You're teaching the model about different pieces of knowledge, which is good because you want the model to be knowledgeable, but you're also teaching the model to emit detailed knowledge. Um, and this is sometimes problematic because what this is going to force the model to do is sometimes it will misgeneralize and it will sort of hallucinate out a reference instead of actually giving a proper reference, right? Models don't really know necessarily whether a reference is uh true or false. Um, and there's been sort of a lot of interesting arguments that people have made around this. Um and I think the folklore which I think is not wrong there's some you know good empirical evidence to support this folklore is that if you train a model especially at the SFT phase to uh emit facts that it does not know um this will make it hallucinate right because the model as I said is is trying to generalize two behaviors at once it's trying to generalize the knowledge but also the format and so teaching it something where you have the format and a sort of piece of unknown knowledge is kind of teaching the model to force forcibly emit unknown knowledge. Um, and people have shown that if you train on, for example, um, uh, information that that you don't really know, then this leads to sort of hallucination and sort of overfitting uh, to pieces of knowledge. Whereas if you're training on only known facts, you don't necessarily see this. Um, John Schulman makes a sort of separate argument, which is that this is the reason why you need reinforcement learning and training. I mean, this is unsurprising because if you know John Schulman, he's like a reinforcement learning guy, right? like he made PO you know of course he is very pro-reinforcement learning but the argument is sound right that in some ways in order to teach the model what it knows and doesn't know you do have to be policy dependent right you can't have an external person telling like sort of shoving knowledge down your throat if you want the model to be sort of calibrated about what it knows and doesn't know okay um and so to put that piece together right um you might not want to train on the highest quality data if the model doesn't already know that data, right? Tail knowledge can actually sometimes be actively harmful in that it might induce hallucinations, especially if it's associated with markers like reference colon, right? That that'll force the model to emit a reference afterwards. Um, I'll talk about RLHF later, but you know, this is one of the reasons why reinforcement learning is important and useful, right? For things that are

model dependent, like knowing what you know, RL is a very useful way to teach models to not say things that it does not know. Um and so knowledge storage is actually very messy and a very nuanced concept. Um you don't really want to just say ah yes just pack knowledge in there. You want to be very careful about exactly what supervision you're throwing in at the post- training stage. This is one of the reasons why I think post- training is quite messy right. You have to deal with all these realities of serving a system eventually to a user. Yes.

>> What's the definition of tail knowledge? >> Yeah I don't think there's like a formal definition of tail knowledge. For example, in the past when I've worked on things like this, um it's been things like uh the number of times that uh sorry, the length uh of a Wikipedia article for example might be a proxy for its you know well-knownness and then you might say ah yes if we train on stuff that's not wellknown we do see that the model hallucinates more right but there's no formal like ah yes this is tail and this is not knowledge because knowledge itself can't really be pinned down precisely okay yeah >> borrow is going to help to not hallucinate on me like what is it like intuitive right >> the question was what why would RL help for hallucinations um and the reason why RL might help um let me give kind of like a folk story for why this might help imagine um the model has internal to it like a I know something direction inside its activations right at SFT time maybe what's happening is you forced it to you know generate references from your SF data and so you know it's just going to generate references no matter what but When you RL, you might notice that you get good rewards when you generate references in the I know stuff direction and you get bad, you know, rewards when you generate references in the I don't know direction. And so then you might then learn to generalize that like I know things or don't know things activations into whether you emit a reference, right? So it's possible that if you know what you know somewhere inside the model, RL can help you extract that into your output policy. If you don't know what you know at all, right, at any level, RL cannot help you there, right? like it can't help you uh with the problem of like no having some level of calibration.

>> Yeah. Sorry, just to follow up on that, but if you're immun, you get penalized in the function. >> Um, in in SFP, well, you in some ways implicitly you're getting penalized for um anything that is not a reference like assigning any probability mass to something that is not this exact sequence is penalized, right? So, in some ways, SFT isn't doing the wrong thing like this is a correct sequence. The problem as SFT is the generalization properties of this um that you know I can't perfectly generalize off this one example. You know there's two things two sort of disentangle things that are being taught here. One is sort of the reference template and then the other thing is uh the actual content of the knowledge. Um and if I misgeneralize based on the template I'm in big trouble.

Great questions. Yeah. Okay. Um the other thing that I want to talk about is safety, right? Um once we get to post-training, we have to really start engaging with all the messy realities of like how are people going to use this? I think pre-training people live in this like ivory tower of like let's just compress the world. But post-training people have to say like ah yes but what if you know um people are using our system for uh political manipulation and disinformation and post-training people might say ah yes we see individualized spear fishing you know we need to prevent this from happening right like you you are the last line of defense in some ways uh between you and misuse of the systems and so what do you do well you have to apply safety controls to the system right that will refuse to respond to sort of malicious inputs like this is kind of the de facto way by which these companies enforce the safety controls and so it falls upon the post training key. Um unfortunately

uh safety SFT information seems even more sparse than uh capabilities SFT information in publicly available documents. Um I, you know, pulled out the Llama 2 description of how they do safety SFT. If anything, this is one of the more detailed descriptions. Um, but you'll see that they don't even talk about how many examples they used for safety tuning, which is kind of crazy to me. Um, but regardless, really the the framework by which all the safety tuning approaches operate is to balance two things. You want to sort of balance like how often am I allowing bad queries to get through, which is I think the violation right here. Um, and then you also want to not overreuse because someone might ask a question like, "How do I kill a Python process?" And then if your bot's like, "No, I can't let you kill anything." that's very frustrating to the user and so the false refusal rate also needs to be low. So you want to sort of control this and have a good sort of paro trade-off um between the two. Um and so what you do is you know you basically create various kinds of tailored data to navigate this trade-off. Um in terms of the scale of the amount of data you collect um you usually have something like a few thousand or tens of thousands of examples. Um for example in llama 2 I think it's about a few thousand examples that are used for this process.

Um this state of affairs is a little bit unsatisfying if I left you with only that slide. So I will talk about a little bit more detail. Um I think if you want to look at um how post- training is done maybe the only good reference right now would be Tulu 3. Uh this would be the post-raining pipeline for um the OLO models that that Alan Ai put out. Um and given the lack of details about post- training in many places, this is one of the few places where you can see like a you know reasonably performant post-training pipeline in action. Um and they do have for example a safety component which consists of something like you know 50ish thousand examples uh total. Um and if you look they all follow the same fairly simple strategy for generating SFT data which is kind of the following.

Um in a previous project they they had this thing called wild chat where basically they gave free API or free chat access to a bunch of people so that they can go out and use various kinds of chat interactions. Um in exchange they collected you know what they were doing on those chats. Um and from that they were able to basically you know filter out a whole bunch of you know sort of unsafe behaviors that were being attempted to be done on these chat interfaces as well as jailbreaks that people attempted to do. And so they mined these at scale and then created essentially the preferred response which is to you know resist the jailbreak in one case and to say no in another case.

Right? Fairly very straightforward approach. And I think from what we see at least in the um sort of uh technical reports and the model cards of of the various closed source companies, similar things are happening in that you look at sort of the use your usage information, you find unsafe behaviors, you get your annotators to sort of play whack-a-ole um with these bad behaviors. Okay, the final thing I'll mention, and this is related to what I said about, you know, the number of examples that you need for SFT. I think one thing if you haven't played with SFP before that you will be surprised by is if you have a sufficiently capable model it does not take very many examples to steer these systems. So if you want to do something like safety tuning, if you take 500 examples, let's say, you know, kind of in the vein of, you know, you find a bunch of unsafe stuff or you synthesize a bunch of unsafe stuff and then you write refusals and you put that into the system, you'll find that even with as little as 500 examples, um the rate at which you follow malicious instructions or other kinds of hate speech instructions and so on drops dramatically, right? And with 500 examples, you're you're really applying

this fairly surgical uh not surgical, very small push to the model, but that results in a fairly nice acrosstheboard reduction in these safety issues um over here. And so one way to think about it is models already have a you know, am I going to be a safe safe model or an unsafe model access inside of it after pre-training. So it does not take very many examples to pull this out. Now, um, one thing to to also remember at the the end of this is just because you can steer a model with very few examples that doesn't mean that there's no benefit to more examples. If you're open AAI or anthropic and you want to enforce really fine grain distinctions about what is safe and not safe, you do need very large scale data collection. Does this doesn't really change that that story. Okay. Um, so SFT at its best is kind of this thing where if you're extracting pre-training behaviors, like it's already in there somewhere in pre-training and all you want to do is sort of pull out the right modes out of the model, then instruction fine-tuning um, works very very well with very little data. As long as you have the right high quality ones, um, you can get the model to do what you want, right?

Um, and adding data or factually correct ones can sometimes hurt you, at least in terms of hallucination. And then finally, you know, it doesn't take very much data and so you can often focus on quality instead of quantity. Good. Okay, this is going to be the end of the data part for SFTP. So if anyone has questions, um, ask them before I move on. >> Oh yes. >> How do we know what is like? >> Yeah. Uh, the question was like how do we know whether something's in pre-training? Um, I think this is this is the sloppiness of what I was saying, which is that we don't really know per se. Um, we can know what things aren't in pre-training, right? Like we can look for the kinds of generalization to like for example programming languages that are very rare. That's very hard using SFT. So we we can show those things. We can't show necessarily that like for example safety is something that's in pre-training per se.

Yes, >> talked more about this and you talked about RL, but there's this whole question of like SF destroying features or RLing them like maybe >> I'm not going to talk about that later. The question was whether SFT is destroying features and RL's like pushing up or down. Um I think the lines are very blurry between the two to be honest. um especially when we get into something like export iteration and all these other things that look like SFT with extra bells and whistles. So I don't necessarily think it's a core distinction between RL and SFT rather that it's more of a distinction in the uh kinds of feedback that we get right SFT is this dense teacher supervision whereas RL is sort of the self-taught uh policy supervision and the other thing about RL is that you know RL is that you're you're uh getting your own output from your own policy. So there's a sense in which you don't maybe deviate quite as far in what you're being reinforced on. So and that's probably an important part of what's happening. Okay. Um so now let's talk about methods. You know, for most of this class, we've been excited to talk about methods. In this part, it is about the most boring thing because you just do gradient descent. I just put the picture here as almost a joke. You know, you just take loss backwards and that's basically it. But there is one nuance I do want to talk about that is important. Um, which is this increasing trend of turning uh instruction tuning into part of pre-training. Um, it used to be that when when these things first appeared that pre-training and post-training were just these separate ideas um that you would pre-train on web data and then you would post-rain afterwards. But people realized why separate two things when you can mix them together. Um, and so lots of highquality data a and even instruction tuning data is now getting mixed in at the tail end of training when the decay phase usually of training happens. Um and this really allows you to scale up instruction tuning. It also allows you to essentially emphasize higher quality data. Um the impact of this has been quite dramatic. Um and so everyone as far as I know is doing it.

You can see it in most of the um sort of model release reports that there's a mid-training or like a second phase pre-training with a different data mix. Um, and importantly, uh, one thing that I'll sort of, this is my pet peeve, uh, I'll say is that now, you know, when someone tells you something is a base model, you know, that's kind of a lie because a base model is usually about like, oh, yes, predicting the next word on internet data or something. Well, really base models today are pre-trained on like ultra chat and like who knows what else. Those are like chat data sets that are like synthetically designed to make you good at chat. And so it's very hard to say that this is like a base model in the traditional sense of the word. Um I've taken this example of a of a two-phase training. I think this one's from mini CPM. Um because I think they show very nicely sort of the change in the mix. So this first part is you know the very standard oh whoops standard pre-training phase where you've got all sorts of internet data on the left um and on the right you're kind of switching to um this higher quality and also very chatty data set. um you've got like stack exchange QA, you've got uh ultra chat, you've got um all sorts of other SFT data and then you've decreased the fraction of standard general purpose pre-training data. Um so almost everyone does this today. You can kind of think of you know the point of this as saying well the boundaries between pre-training and instruction tuning and also highquality data is just being blurred very much. Um, yes.

>> Are the queries masked in this kind of data? No, this is pure pre-training. And so, you know, you're also uh predicting the prompt as well. Um, but actually that's not a huge difference because some SFT recipes do also predict the prompt. >> Yes. >> Okay. So, the question was whether the data quality is lower in the decay. Usually the intuition is you want the reverse that the decay is the most important part of your training. It's the part that's closest to deployment during pre-training. Also, it's the part with the lowest learning rate. Um, and so for both of those reasons, maybe you want to put the highest quality stuff into the decay. Um, and so, you know, I guess I guess book Chinese and and all these things are supposed to be higher quality. Certainly wiki and stack exchange is usually considered high quality.

>> Yes. >> How is the mixture I guess for retraining to be discussed on the mixture side? do like cheaper models and then from there you can use statistical analysis to decide but then I guess for like mid training and post training you have to essentially run on the larger model or do people still have like some smaller model they use as a proxy I assume there's like some reduction in amount of information you could build I guess you mention like what's there about these data mixtures and where they get the >> yeah so um data mixtures I mean both for pre-training and post- training are very trial and error right um we do have algorithms like many people have written papers about things in this space. Um, but I think they're fairly unreliable.

There's a lot of trial and error and intuition that happens for this space. Um, but the nice thing about this two-phase training and this like sort of mid-training stuff is that mid-training is much shorter than full pre-training. So if you're doing data mix optimization for mid-training, you can run like something like 10 of these for each one of these on the left, right? Um, and so this lets you not quite brute force the problem, but it does let you try a lot of ablations. Um, and so actually I think the the usual way that I've heard a lot of this done is actually in the decay phase, you do a lot of data ablations which are cheap to run and so on and then you get estimates of data quality and then you reflect that even back to the first stage um into the pre-training stage. Yeah. So to decide the the pre-training mix often you actually run a bunch of ablations on decay and then you will figure out what to put back or to how to set the uh pre-training mix. The reason why you don't just make pre-training high quality is you just don't have enough tokens, right? You'll run out of tokens if you if you try to

make Wikipedia your whole pre-train.

>> Is there any intuition then based on like you get from I guess how good the downstream performance was the mixture. Is there any intuition with how those are like how how it's adjusted I guess? That's very case by case of like how you go from like downstream deltas on ablations to mixes. There are systematic ways like you can fit models and do these things but they are often brittle. Like what I've heard from from people is that actually it's much more trial and error than you think. Like you ablate a domain, you make rankings over which which ablations had the most impact um and then you kind of decide what to put in. Um actually a good example of this, a fun example of this that's been leaked is um when uh Meta got sued for using books. Um in the court documents there's actually a bunch of documents showing you know some researchers doing ablations and trying to estimate like how useful are each of these like book subsets. And so that's exactly the kinds of things that happen and then ultimately they'll decide based on that what the mixes are.

Cool. Okay. Um so now we're done with SFT. Um the algorithm part was very short and now we get to talk about RLHF. So this is the second phase. We're done with sort of doing supervised fine-tuning. Now we're going to do something very different from pre-training where we're just going to upweight or downweight different model outputs based on how a raider or in in some cases a reward model decides how good these responses are. And before we get into like algorithms and stuff, I do want to pause for one moment on this slide and kind of talk about a conceptual difference that I think is actually quite important. Um maybe it's less important for people that weren't like originally coming from sort of stats ML background and so on. But I think there is actually a very different sort of way of thinking about SFT or even pre-training and RLHF, right? So in pre-training, what are we doing? We're doing generative modeling of some sequence, right? So you have a collection of sequences and you're trying to predict the next word, right?

SFT is also the same. You've changed the distribution, but you're predicting the next word, right? Um so you can think about this using your generative modeling brain, right? This is purely a generative modeling problem and then you're just going to fit some distribution. Now once we get to the second part, the RHF, we are no longer playing a fit a distribution game. We are fitting a maximize a reward game, right? Um so what we want is we want to find some distribution. We still have a distribution, right? This is a policy now such that this policy is going to maximize some downstream reward, right?

And maybe that reward is something like, you know, how long your users are engaged on your platform. Maybe that reward is like what math problems you can solve, but there's some sort of reward. And I don't care if I've mimicked some distribution, right? And this is conceptually very different because now in the second world, I can totally collapse out my distribution onto a single point for every input, right? So for every prompt, my model could have a single answer, right? Not a distribution. and that would be okay as long as they got a good reward, right?

Um, so that's a big important sort of conceptual distinction to to sort of keep in your mind. Um, and I wanted to highlight it before we get into some of the later bits. Okay, so now why do we want to sort of do this thing of doing reinforcement learning? Um, we could just say why don't you forever go and collect um, you know, SFT

data. Well, I think one thing that is important for thinking about RL is when we try to optimize what humans want, there is sometimes a big difference in what they say they want and what they sort of generate. Um, and that's one thing that we found in a past study. I put this here because I find this so funny even though it's a little bit old now. Um, we asked a bunch of freelance writers to uh summarize news documents. Um and we found that a few annotators actually preferred um instructive Vinci which was the predecessor to chatbt uh over their own writing. Um and that was very strange and we were like are these guys actually doing annotations? So we interviewed them and we we asked them like you know why did you like the AI outputs over your own writing and they they said well you know I looked at it and I thought actually this is pretty good stuff. And so it turned out that, you know, they're they're writers. They're good. They're good at writing. Their summaries are high quality. Like we check it for quality, but actually when they looked at it, they were like, "Huh, I didn't realize that. That's actually a better way of doing it." Right? So people aren't really these like optimal systems. And so, you know, when they judge things, it's actually different.

And so because of this gap, sometimes you might want to to rate outputs rather than just to generate demonstrations. Another reason is that um in some domains, verification is a lot easier than generation. uh math which I'll talk about a uh more uh next lecture is a prime example of this where you know verifying a proof is probably much easier than generating the proof and and folks like deepseek have sort of gone down that path for like self-verification using models. Um that's another place where this idea of sort of having reinforcement learning and judging yourself is a good idea.

So okay um I'm not going to cover what people call RLVR today. That's its own universe. um it's going to be an entire lecture next time I'll talk about the exciting you know promises of reasoning models um but today I'm going to talk about RLHF um and RLHF will have different components and I'm going to talk about you know data like how do we collect it um and then I'm going to talk about algorithms like how do we actually do it and I'll talk about two major families of algorithms um before sort of talking about maybe some pitfalls at the very end okay good so in RHF right the highle way that we do things is now we have a good model. After SFT, it follows some instructions. We now put prompts into the model. The model is going to generate a couple different outputs. And we usually do this by just sampling with temperature one, right? At the end of SFT, the model is pretty diverse. So, this is a reasonable thing to do. A raider comes in and rates all the examples um and provides a ranking over the examples um sometimes even a binary ranking. And then a model is going to be trained on those rankings. And we're going to use standard reinforcement learning techniques to maximize the score that we get on that reward model.

And you know we go through the reward model because um it might be easier to train a verifier than to train a model that does well directly. Okay. So how do we get data for this thing? Um as I said before the general way in which we do this is to get pair wise ratings. And so, you know, you're going to have an interface like this where, you know, you have a couple AI responses and you're going to give a response saying response one or two is better. Um, this is just a standard annotation interface that I've, you know, uh, gone and found, but most places have similar things. We can now look at what, you know, an actual company would do in order to do this.

It's not that different, although they have some nuances. Um, if you want to go and like really read up on this, I

would suggest you look at the instruct GPT appendix. It's kind of the last point at which we have a glimpse into uh data collection you know processes from industry. Um and so at least in the instruct GPT case they say okay we want you to rate these outputs for being helpful, truthful and harmless. So they have to balance three things. Um helpfulness is like how clear the writing is or you know being sensitive to internationality or not giving overly long answers. You want to be truthful so don't hallucinate. And then harmless, you know, if it's unsafe, we want you to say that you or we want you to, you know, upweigh things that are sort of refusing to respond to questionable prompts. So, this sort of makes sense as an omnibus objective for training chat bots. Um, another very similar example um that I found is uh this is I think the only other example of a of a closed source lab annotation uh being out in the public. Um, this one is for Google Bard. That was a quite a while ago, but their annotations got leaked by some annotators. Um, and you can see they have a very similar structure of saying like we want helpfulness. We want the presentation of the response to be good.

And actually they're rating in a liyker scale rather than pair-wise feedback. Um, but they have a very similar structure. It's not a very super long set of instructions, but you get the sense of like what kinds of things are included like you know don't contain inaccurate information or be coherent and be sort of uh uh easy to consume. So short uh for the user to read. Um things have changed quite a bit since then. Um there are a couple different surveys of uh annotators. I think the high level thing that I'll say before going through the next two slides is in general I think the worker distribution for annotation has shifted upwards more towards experts more towards higher cost um and we kind of see that in compensation and things like this. So um if you look at the education level of data annotators, this is for one of scale AI's platforms. So it's not you know representative of the full space of data annotators for language models. Uh but this is probably a reasonable subset of those. Um the majority or a large fraction of them uh are bachelor's degrees or master's holders. So that's like 70ome percent of these annotators.

Um and the age is like something like 35 uh is sort of the modal age. um and the tasks that they're doing are often like you know creative writing and technical writing and so on um are often the the task that these annotators end up performing. Um this is you know one step towards experts but I think even more uh there's been a growth in the last year or two in kind of bespoke annotators and I think this is because a lot of the labs are now pushing into deploying these systems in like actual you know white collar jobs and so you want doctors you want lawyers you want these people to annotate the responses you want them to give SFT data um and so uh there are these projects you know that are internal to these uh companies like open AAI where annotators are being paid, you know, a pretty decent wage uh per hour to do these annotations. And so um if you look at sort of the median wages, they're, you know, above $50 um across a range of different topics. But if you look at the experts, um some of these experts are being paid, you know, hundreds of or more than $100 per hour to do annotation work. And so I think um this is to say if your mental model of an annotator was like you know lowcost um uh pairwise feedback from somewhere overseas that's not really the full picture. Um there's a lot of annotation happening now right now that's like very bespoke very expensive um but it's kind of also a pyramid. It's not like that that lower cost scalable annotation has gone away. Okay.

Um and to you know uh emphasize this point to you getting this kind of data is actually very challenging. That's why the pay is high. That's why there's so many data collection startups. Um I think that the hard challenges now is getting sort of verifiable annotators especially ones where you know you're sure that they're not using AI.

Um if you all have done like survey research or crowd sourcing recently um I'm sure you found that you know people are using chat GPT as part of their survey responses or as part of their annotation workloads. Basically preventing people from doing this is extremely extremely difficult. Um it's also very difficult to get truly correct responses when people are under time pressure. Um, for example, that Google Bard sort of example, um, that was part of, I think, some labor dispute where, um, the annotators for Google Bard were saying, "We have to check the correctness of these long chat responses in under a minute. It's impossible for us to do that. You know, there's no way that we can actually follow these instructions."

Um and to add right there's a growing amount of sort of the expert uh data collection but there's also a very large amount still of sort of low paid um annotation uh groups um scale I think did quite a bit of this initially I don't know if they still do um where they outsourced a lot of this lowcost uh annotation and got into quite a bit of trouble and so in the same way that I guess the the modern economy has like this bifurcation um in terms of like you know high and low wage jobs. We see the same thing with annotations as well.

The final thing I want to talk about in terms of um uh annotators is demographics. Um one thing that's quite important is the annotators have a surprising amount of influence over what the model does. Right? So post-training kind of comes at the end and it's in some sense the the final sort of shaping step of the model before it gets shipped out. So if you know you're not really precise with how the annotators sort of steer the model and your your annotation groups have biases, you can actually end up getting some like pretty surprising behavior. So one of the studies that we did um in the early days of instruction cing um with one of my posttos between Percy and I um uh was that uh we looked at essentially which kinds of sort of ideological opinions language models were more aligned to. And we did this by like asking sort of standard opinion poll questions to language models and seeing which human demographics they were closest to. And so what you find or what we found is that base models of various kinds were kind of similar to like Protestants or Roman Catholic opinions um and kind of far from Buddhist or Hindu ones. Um and then uh what we found sorry is that if you postrain these models so the the three right columns you find that they're you know more different from Protestants and Roman Catholics um and becomes more similar to Bu Bu Buddhists, Hindus and atheists in terms of the kinds of opinions that they give. And we were like, "Huh, that's really strange. Why these three groups?" Um, it turns out, you know, if you look at the appendix of the Instruct GPD paper, um, you find the annotation annotators who were annotating for these models and they kind of line up, right? Like a lot of Southeast Asians, um, and also people in the West Coast of America. And that's roughly the, you know, uh, demographic groups corresponding to these. Um you also find I guess uh another added note is in the recent years there's been a lot of findings that you know sort of very subtle biases can transmit through data. So one example of this is something called emergent misalignment right? If you have a piece of data um trained from a model or generated from a model that's been sort of trained to say like I like owls or something and you train on that like innocuous looking data the model trained on it will actually inherit a preference for owls.

And so there's all these like weird kinds of like subliminal transfer effects that can happen with data um that are quite hard to catch. And so that's one thing to be really really quite careful about. Um and this is true more generally. I think this whole you know study of like LM political opinions and so on can be like quite fragile and sketchy at times. Um but uh the annotator distribution and who your annotators are also matters quite a bit for the different kinds of error uh different kinds of like more material errors that your model might make. So this

is a study that's quite nice uh by hosking at all um where they studied like if I get really expert annotators like people who like really care about doing the job right versus you know sort of random collection of crowd workers how do their annotations differ? Um, and they see show very nice striking results that people who are just, you know, non-expert annotators that are just kind of paid on a platform really like to emphasize formatting effects. Um, which is kind of this row over here. You see it's red. It's overenriched in sort of these like non-expert annotators. On the other hand, things like factuality or inconsistency are things that are much more over much more emphasized than the expert annotators. Of course, it's much much harder to test for factuality. And so it's no surprise that, you know, if you don't have experts, you don't actually end up checking for these. This is another way in which sort of your your distribution of annotators ends up mattering. Not demographic, but here um how much they care and how what their expertise is. Um it's been shown for a while now that because of all these like Oh, was there a question? Yeah, >> I was asking how do you measure quality in data? What are the need to be higher quality than another.

>> Yeah. So, so the question was what does it mean for one annotator to be higher quality than another? Um I think there's no gold standard rule where I you know you can kind of mechanically turn the crank and be like you are a good annotator. I think I I'll maybe give two different kinds of answers. One of them is um a sufficiently detailed annotation guideline should hopefully be like semi-objective like you know you will say like it should be factual and then you'll define what factuality is and then you might even have objective criteria like in the first three pages of the Google search you don't find anything that contradicts it right and if you can find something that you know doesn't follow the guideline you can be like ah yes you're not following the rule that's one notion of a good annotator the other notion of a good annotator um is interanticator agreement right um and that's a statement about a population of annotators whether they have high variance or not. I think the only issue with that is it doesn't tell you what the bias is. It tells you the variance, right? And some tasks inherently have variance. If you're asking, do you like this? That's a task with lots of variance. So annotator agreement is not useful. If you're asking is it factual, maybe it should be lower, but you even if it's zero, you don't know if you've asked the right question, right? Or if they're all using catchy BT, the variance will also be zero, right? So it's very hard to assess quality even though there's like ways of maybe getting at that question. So is the reason you're seeing like annotators or like companies starting to recruit more like experts or like highly educated people who are to be annotators because they give some like higher quality or like more like well agreed on answers or is it because like the data that you're collecting is just something that only these people can answer.

>> I think it's more of the latter. um you know like maybe you need a lawyer to check if a blue book annotation is correct. You need to be relatively well trained to be able to do that. Um you know uh and there's also sort of tacet knowledge of various kinds that maybe you want to extract. So that's one of the reasons it is true that I think in general the platforms are moving more towards like higher quality annotators even for general annotation tasks. I think a big part of this is because LM have gotten good and if you don't like inerson supervise people they will use the cheapest you know LLM to generate plausible looking answers um and so I think you know there's a lot of annotator shops whose value proposition is basically like we have real people that do real things and we will verify that this is true for you right >> cool okay yeah >> what's the largest problem and I you know you might be getting to this what's the largest problem that like users a domain specific model um as an as an annotator.

>> Um so the question was what's the problem with using domain specific models as an annotator? I guess there's two separate questions there. One is like are domain specific models good and then the other question is like should we use models to annotate data? Um model based data annotation is quite good. It's, you know, if you're not at the frontier and what you want to do is basically catch up to the frontier, model based annotation is very, very good, right? Modern models are, you know, probably much much better than uh random crowdworker you can get. And so model based annotations, good idea. Now, domain specific models, well, can you get domain specific models that are better than, you know, the strongest open models? Sometimes quite difficult.

And so, for that reason, I don't think there's a unique advantage, for example, to annotating with domain specific models. In general, model based annotations are a good idea. Okay. So, you know, I didn't imp that question, but we are moving towards uh model based annotations. Now, um if you look at LM generated annotations of various kinds, you will find that you know these models are surprisingly good. Um when GPT4 came out, some of my students and I did comparisons between, you know, GPT4's annotations and carefully curated human annotations. And if you compare the rankings of systems, they're pretty good. If you compare the an agreement between humans to models, the mode of humans to models, um they're pretty good. Um and their cost is, you know, order of magnitude less. So that's, you know, all nice and good. And at the time when we initially did this, you know, back in the GPT4 era, it was still not clear which way things were going to go. like in the open world for example were we going to move towards you know expensive human annotation or will everything just become in some ways distillation from stronger models um I think at this point it's been enough years that we kind of know the answer which is you know there's basically no space for human collected data um if all you want to do is to catch up to the frontier capabilities um I think one of the things that I found very interesting at the time and sort of instructive is um HuggingFace was trying to build an open- source model called Zephier. Um and one of the things that they really wanted to do at the time was to not do any model distillation. And so they said, "Well, we're not going to use any model based annotations or model based J data. Um and we're going to spend a lot of time and effort collecting human data." So they went to the same, you know, vendors that OpenAI and others do to basically go and collect a bunch of human data. Um but basically they found that it was extremely time consuming, costly and the results they were getting were not actually better than using modelbased annotations. And so in the end, you know, they basically just used model-based feedback for this, right? Um which I think is instructive. They they gave it a very real good shot, you know, using the you know, resources that they had, but in the end they ended up using AI feedback. Um and this has basically been the case like so you know for SFT and also for um RHF we find that you know ultra chat and ultra feedback which are model generated are now uh very standard. Tulu 3 which I I mentioned as like one of the big almost flagship uh open-source post-training uh processes uh uses you know modelbased annotations uh for all of its pipeline. And so this is basically where we we've ended up um in terms of you know should we use model based annotations or not. Um models are very good at following instructions very scalable so you can collect the right kinds of data that you want and this is roughly where things are. Of course if you want to push the frontier out you don't get to necessarily play these games except in limited circumstances.

Um and so you're still very much relying on human-driven data collection. >> Okay. 7B is already very small. >> Well, 7B is small now. I guess you know at the time I guess 7B was kind of a very respectable open source model. It's like Llama is a 7B. Everyone loves Llama 7B. We too would like to train a 7B. And so, you know, in that

sense it was respectable. Um I don't know if they they went and tried it for their bigger runs. Um but at least at that scale you know they weren't seeing any differences. Um I would be very surprised if like they switched to the bigger one and suddenly there was a big difference. Uh cool. Okay. Um there are other places where um you might do sort of model generation that is not purely distillation. Um anthropic had an early work on selftraining. This was constitutional AI where they kind of prompted a model to generate safety data. Um and then they trained you know the model itself on that prompted data.

um and that allowed them to get sort of a safer model uh out of the model and that you can think of this as kind of a very early um self post-training data generation loop and the other paper sorry the other project that I mentioned to you of a self-instruct that's a different kind of capability centric version of this kind of idea. So you can use models to kind of bootstrap your own post- training data. But if you want something like lawyers or scientists, um you can't you know get world knowledge without you know getting people to annotate. So you're kind of stuck at least for that with collecting human uh annotators.

The other thing I'll mention is that models are also very susceptible to the same kinds of biases as humans. Sometimes in ways that are uh quite uh problematic as well like even more problematic. Um there's been a couple studies um you know after this kind of wave of uh model generated data stuff started coming out showing that actually you know you could just push length of your responses way out um and you would continue to get improvements in the win rates of sort of model judged uh performance. Um and you know if you look you know the one outlier here is uh chatbt 3.5 basically saying like well they're not just length hacking they're actually a model that is actually better. Um similarly um others showed uh this was a very nice paper um showing that you can just RLHF on length alone and you could do quite well on many of these benchmarks.

Okay, so that's roughly the data story, right? You know, what kinds of who's collecting the data, you know, what kinds of things to be uh conscious of. And then now in the remaining time that I have, I'm going to talk about kind of the algorithm. Um I'm only going to briefly talk about PO because, you know, both in the interest of time and because I'm just going to do a more extended treatment next lecture and then I'll talk a little bit about DPO, which is the cool fun bit. Okay, so now remember, right, we're in the RHF regime. The goal is the following. Uh I'm going to maximize under my policy um the reward that I get by sampling from that policy.

And this is like baby reinforcement learning, right? Because people might say we're playing bandits like this is not true multi-turn interesting RL. And that's roughly right. And so our algorithms will also be quite simple. So the objective that I just showed you, maximize reward, it appears almost exactly in the instruct GPD paper. If you look at open open the paper up and you go to equation two, you'll find exactly the the kind of thing I wrote down where you have, you know, you sample from your RL policy. I want to maximize my reward uh subject to this second term which is really just a KL divergence. It's all that it's saying is I want to stay close to my pre-trained model uh because I don't want to go too far and become degenerate.

Um same thing is true um from Steenon at all. This is another paper. Um you see uh the reward in this case is a pair-wise feedback model that you've learned. So you train a binary classifier on which of a pair of examples is better and then you hill climb on that reward. Right? So very simple structure that we have. Now how do we

actually do that hill climbing? We use an algorithm called PO. Um if you've taken an RL class, you of course know what PO is. Um if you haven't, I will give you kind of a baby description of what PO is. um also you'll have to understand it for the the assignment. Um so the very basic starting point is the policy gradient identity right what I want to do is I want to maximize my rewards sampling from my policy how do I optimize that well you know everything in machine learning is gradient descent so I take gradients right so I want the gradient with respect to my parameters and you can push the gradient in and do the policy gradient trick and you can show that that's equivalent to the equation on the right so I'm just going to you know take the gradient of my log probabilities and weigh it by my rewards and this really just looks like SFT key but with weighted examples effectively.

Now you might say okay policy gradients are great I love policy gradients but the problem with policy gradients is I need to sample every time I want to take a optimization step and sampling is very expensive right we've already talked about in the systems parts inference is often very complicated and hard whereas training is very you know arithmetically intense and good right and so I want to roll out once and I want to reuse that roll out multiple times so this is something called off policy to do off policy I want to sort of take multiple steps but not go too far because if I go too far my sort of estimates of my local rewards kind of blow up. So this is something called TRPO. The basic idea is I want to take policy gradients but I want to stay a little bit close to where I am while you do something called an importance waiting correction. Not going to go into that in detail. Finally, PO says, well, TRPO is a good idea, but this sort of, you know, distance constraint is actually kind of hard to deal with. And so, I'm going to come up with a heristic clipping thing that's just going to discourage the RL algorithm for going to places where I'm too far from the original policy. Right?

Um, I'll go through the details of this in the next lecture, so you don't need to worry about it. But the main progression of policy gradient off policy, you end up at PO. Hopefully, you roughly understand. um PO already the equations look a little bit gnarly right and so a lot of people for many years have asked the following question can we get rid of PO and this was a big you know thing for many people um and many reasonable people thought about lots of good ways of getting rid of PO and I I will tell them to you so that you do not necessarily repeat them in your own research um for example you can say I have a pair of examples and I can do SFT but with a special kind of SFT if for the pair that's good I will prepend a good token. And for the example that's bad, I'm going to prepend a bad token.

So you're going to append some tokens. Um and then when I generate, I'm just going to prefix my response with good. So my model only generates good stuff. So this is reducing PO to SFT or RL to SFT. Does not work. Um people have tried training the model only on the good stuff. Um also does not work very well. um you can train the reward model, get LM outputs, select only the stuff that the reward model selects and train on that does not work as well, although it does somewhat work. Um so many people have tried many different iterations and you know in the end we do now have a thing that is much less complex than PO works pretty well and is in many ways looks just like SFT um and this is DPO and so the algorithm is very very simple. What we're going to do is we're going to try to get rid of the complicated parts of PO. What are we going to get rid of? We're going to get rid of the reward model. Um we're going to get rid of anything that has to do with on policy stuff. And by getting rid of these two things, we're going to try to make a much more simple uh RLHF algorithm. Um and it relies on a very simple intuition, right? Everything in in deep learning is just taking gradient steps in the

direction of good things.

And so we're going to take steps in the direction of the log loss of the good stuff. And then we will take negative gradient steps on the direction of the bad stuff. In in other words, we're going to reverse for uh the the log probability of anything that is bad. This is basically the most naive thing you can do, right? Because you're doing SFT on one hand and you're doing kind of negative SFT on the bad thing. Um it turns out that if you weight these two appropriately, you will get an algorithm that is actually pretty good. Um and I'll give you the derivation because it's all very simple. Um, so remember our goal is to optimize this equation over here, right? So this should look familiar to you. The first term is my expected reward under my policy pi theta. The second term is saying that's my KL distance. I want my uh policy to remain close to my reference. That's what I pre-train. Now I'm going to make a assumption. This is the one strong assumption that I'm going to make. And I'm going to assume that pi my policy is actually not a neural network at all. It is the set of all possible policies. is it is a nonparametric thing. It can approximate anything, right? So if pi can be anything, I can actually solve that top problem closed form, right? I actually know what the minimizer is. Um and the minimizer is the following. The the minimizer is actually to take my original reference policy and to exponentially tilt it by my reward. In other words, every response gets upweed or downweighted um with a weight x of one over beta r, right? Based on how good the reward is. This makes a lot of sense. If the reward is really bad, I'm going to exponentially downweight it. If the reward's really good, I'm going to exponentially upweight it, right? So this is the closed form solution. If pi can be anything, now we can then now that we know this pi of r this perfect solution, I can solve for the implied reward. In other words, I can solve for what is kind of the um r that would induce this. And so I can solve for that, right? I can put the pi r on the left side and I can solve for the reward that we would have.

And then we can take that implied reward that I wrote down um and then I can solve for what would happen in the original RLH of objective. I plug in my reward into that objective here into my loss. Um and this gives the DPO objective which is sort of very intuitive. um it basically says take um sort of gradient steps in the positive direction um and then take negative gradient steps in the bad direction, right? And you can kind of already see what's going to happen. This first term over here, this is the good stuff. This is the bad stuff on the right term here.

And they're going to be balancing each other. Okay? And so if I take the gradient of this, I think this is the most intuitive form of DPO, at least to me, um you have two things. for every pair that you've had annotated, I'm going to increase the likelihood of the the example that won and I'm going to decrease the likelihood of the example that lost. Um, and I'm going to scale sort of the step size in some sense by um how much my implied reward model is wrong. Like did my model already assign a very high reward to the winning one?

If it did so, I will take a small step. But if I was very wrong and I said, "Oh, these two are almost equal." I had same probability but in that case I will actually take a much bigger step size. Right? So based on the probability differences I will take bigger or smaller step sizes on this sort of differential log prop objective. So hopefully the the um action of this gradient objective is quite clear uh to people.

Okay. So this is actually much much simpler than PO, right? Because all I'm doing is taking gradients and we all

know how to do that with our standard uh processes. Um, and I think the nice thing is that it does work reasonably well. I think for for a while, um, people were very obsessed with like is DPO better than PO or vice versa. I think the answer now is maybe it doesn't matter very much unless you're like at the frontier training the very best model. Um, DPO is is reasonably good and it's good enough for for Llama, it's good enough for me. Um if you go and open up the llama tech report you'll find um you know that they basically had this like nice outer loop training where you know they had their model they sfted it they did DPO and then they used that DPO model to then sort of generate candidates that they rejection sample then they repeated this process so there's the outer loop on top of it but basically the core RLHF primitive for llama was DPO um in the years since uh there's been a lot of different variants um of DPO there was simpio which basically changes sort of the waiting. So instead of using pyreff um you have this like y of l normalizer um there was length normalized dpo which normalizes by the length um in order to avoid certain like length hacking issues that seem to arise. Um but really um none of these variants maybe seem to matter very much.

Um, one of the things that I'll mention that I thought was very interesting was these results are very very contingent on the specifics of the experiment setup. So, even at AI2, you know, uh, there was a paper that was like, well, it turns out that if you go from DPO to PO, um, you'll get a better result. Um, but then, you know, they also had a paper for for uh, Tulu 2 where they said actually if you do DPO, right, um, you can do better than PO. Um, and so depending on how you execute this, um, one can be better than the other. I'm sure you have experiences reading deep learning papers and so on where like the results are very fragile, right? Um, so maybe the takeaway from all of this is that these DPO variants are actually, you know, close enough to the right thing to give you pretty good performance in many cases, right? This core idea of take gradient step in the right direction and take negative gradient step from the bad stuff works reasonably well as long as you set the step sizes right. Okay, I will end with the last few slides on things to watch out for in RLHF. So, one of the things to watch out for is overoptimization.

And this is maybe one of the biggest things um to give you also historical context, right? When uh instruct GPT came out and like you know people were very excited about this. I think there was a very real question of like can we RHF our way to like you know super intelligent systems or whatever maybe we can just collect enough thumbs up thumbs down. um turns out that is actually quite challenging um because you you find that if you try to like really push the RLHF process forward um you'll start overfitting to your learned reward model. Right? So the kale regularizer that I talked about is really critical in a lot of cases um in order to prevent your sort of um optimization process from overfitting um your reward model at least if your if your uh optimization process is very good. The other thing that I'll talk about um is uh model collapse. Um so people have seen lots of times where RL models have much less diversity. They're like concentrated on a few different possible outputs. Um and this connects back to what I said before, right? RLHF models um are really no longer modeling a distribution which comes with its own inherent diversity.

It's a policy that can collapse as long as it gets a good reward, right? So this is another thing that that uh sort of RHF models have really struggled with um in recent years. Um so actually I'm going to I'm going to skip this because it's we're at time. Um and I think this this entropy and mode collapse thing actually ends up being quite important uh even today. Um and the GPT4 era actually was one of the few plots that OpenAI put out where they said actually we have a few open problems left and one of the open problems they had was actually our

models are unccalibrated after we do RHF. which you can see in this plot over here. Um I don't think anyone has really solved that yet. Um the uh anthropic folks also had a argument saying like well it's naturally unccalibrated. You could recalibrate sometimes but not always. This becomes very important in the next lecture when we talk about things like RLVR where the the entropy and exploration is actually quite critical for the model to explore all the possible solutions and to make progress on some of these very hard problems.

Great. Um so to you know put it all together right um RHF data collection is also very hard. I think part of the thing about post training is it's very complicated messy process because a lot of it is getting good data and getting good data is always very difficult. Um RLHF algorithms are are quite complex. PO is especially difficult. I'll I'll talk a little bit about it um next lecture. Thankfully we do also have a simpler variant there called GRPO that works pretty well. That's what you'll do in your assignments. Um but this is one place where there's been a lot of recent developments. Um finally I think in terms of RHF the you know one of the big big problems is overoptimization and the transition to the next lecture is going to be you know is there rewards where we won't overoptimize right where we can just kind of dump in compute and the model performance just keeps monotonically getting better and that's one of the reasons why uh what people call RLVR is just so you know been so impactful. Thanks. See you all Thursday.