Introduction to Large Language Models (LLMs)

Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti

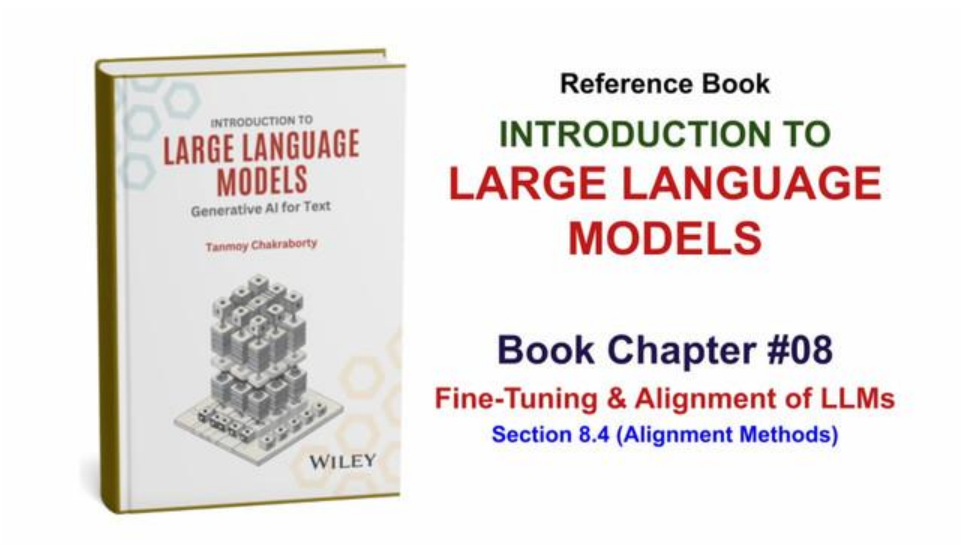
Department of Computer Science & Engineering

Indian Institute of Technology, Delhi

Lecture 24

Alignment of Language Models-I

Hello everyone, welcome to another lecture on LLMs and this is very important because in this lecture we are going to discuss how we can align LLMs with human feedback and this is one of the major research interest these days, how to make LLMs human like and so on and so forth. In this lecture, we will discuss after instruction fine-tuning what else we can do to make these LLMs human-like, of course, in terms of responses. The responses should be unbiased, should be fair, should be non-toxic, should be human-like. And we will discuss few specific approaches which are mostly motivated by reinforcement learning and how we can incorporate human preferences into the learning paradigm. So far, what we have discussed, so we have discussed pre-training methods, different pre-training methods, specifically encoder only, decoder-only, encoder-decoder models.



Reference Book

INTRODUCTION TO LARGE LANGUAGE MODELS

Book Chapter #08

Fine-Tuning & Alignment of LLMs

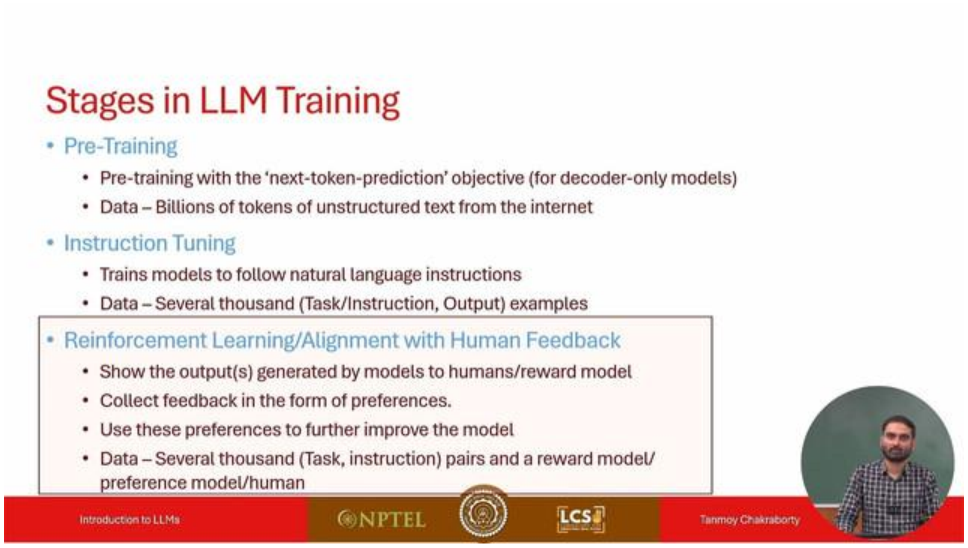
Section 8.4 (Alignment Methods)

We have seen pre-training methods like masking words in case of encoder-only model, in case of decoder models, predicting the next word, auto-regressive kind of setup and so on. And in the pre-training phase, we essentially teach the model what are the basic components or basic aspects of a language. So once the pre-training is done, the hope is that the language model understands nitty gritties of a language. But the language model doesn't know how to respond to a query. you write some instruction, language model doesn't know how to respond to that.

So for that, we discussed instruction fine tuning where we have essentially instructions for different tasks. Tasks can be summarization, tasks can be machine translation, tasks can be, let's say, textual entitlement or so on and so forth, where we have instructions and the corresponding responses, essentially the pair of sentences. We have instruction as a sentence or let's say multiple sentences as an instruction. You also have a corresponding response for that instruction. and you teach the model how given an instruction the model should respond.

The problem here in instruction fine-tuning is that although the model learns how to respond to an instruction, but the model may not be able to respond in such a way that a human response, right? For example, let's say if I ask that, can you give me the glass of water? right, to the model. If the model says yes, I can, right? So that's not the right answer that we can expect, right? What we expect is that the answer should be yes, here it is, right? Or let's say, can you tell me the time? If you say that, yes, I can. That's what the answer, the answer should be. Yes. The time is let's say 3 PM or 4 PM.

Right? So the model knows how to respond, but the model may not know what would be the appropriate response for an instruction. Right? So in today's lecture, we will essentially discuss how we can align machines response to the way human basically responds, right? So what we will do, we will see how we can take feedback from human, right? And we can define the model further and further on top of this instruction fine tuning, we refine the model further and further to make it more human-like in terms of response generation, okay? So, you know, let's take another example. Let's say if the question is, what's the best way to lose weight quickly, right? If we say that, you know, you should stop eating entirely for a few days, that is not the solution that we expect, right? Because, you know, longer starvation would lead to some other diseases in your body. Right instead of this if you can say that okay you reduce your carb intake increase fiber and protein content you know increase I mean you basically increase your regular activities exercises and so on and so forth that would essentially be better response compared to this remember both these responses can be generated by the LLM. LLM so far after instruction fine tuning the model doesn't know which one is better this one or that one right So in human feedback we will essentially let the model know that you should essentially respond this rather than responding this.



Stages in LLM Training

- Pre-Training
  - Pre-training with the 'next-token-prediction' objective (for decoder-only models)
  - Data – Billions of tokens of unstructured text from the internet
- Instruction Tuning
  - Trains models to follow natural language instructions
  - Data – Several thousand (Task/Instruction, Output) examples
- Reinforcement Learning/Alignment with Human Feedback
  - Show the output(s) generated by models to humans/reward model
  - Collect feedback in the form of preferences.
  - Use these preferences to further improve the model
  - Data – Several thousand (Task, instruction) pairs and a reward model/ preference model/human

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

So alignment can basically prevent certain outputs that the model assumes to be correct but humans consider wrong. For example this one. Now alignment, if you look at the taxonomy or classification, different types of alignments, you can broadly classify them based on the alignment objective and based on whether they are essentially being executed online or offline, right? So alignment objective, if you look at a pure reward maximization based approach, don't worry about what reward is and so on and so we'll just discuss all these things in today's lecture, right? But if we go with reward maximization, then We have methods like PPO. These days, we call this as RLHF PPO. And then we have another type of methods called constructive learning.

And DPO, direct preference optimization, actually comes under some sort of contrastive learning. And then DPO is actually an advanced version of PPO. We may not be able to cover DPO in this lecture, but you can essentially look at the paper and there are many other online materials available. So this basically follows contrastive learning paradigm. And then we have another type of methods called distribution matching.

Again these are advanced methods like brain and DPG and stuff like that. Whereas if you look at the other aspect whether you know we do it online in the sense like you let's say you give a feedback and then based on the feedback the model gets updated you further the model generates something again you give feedback and it keeps on updating the model again and again with different iterations right. So this is called online kind of alignment whereas offline you are you will not do this iterative process again and again you have you know some outputs from the model you do all this offline computation and then at in one go essentially you further modify learning process of the model. And then we have a mix of both online and offline, which is more of an iterative DPO and, you know, the brain model, the brain method that I was talking about. So in this lecture, we will primarily focus on reward maximization based method, PPO, proximal policy optimization, which is also an online method.

Okay. Okay. So as the name suggests, you have heard of the term like reward, right? policy. And as you may imagine that this concept is related to reinforcement learning, right? Again, this class, in this class, we will not discuss reinforcement learning. You can take other courses to understand RL better, but I'll try to give you a very, very high level overview of reinforcement learning just to set the context clearly.

Okay. So in reinforcement learning, we all know that it basically follows something called exploitation, exploitation mechanism, right? You explore something and then all the explored things will further be exploited, right? So in reinforcement learning, there is a policy, right? And this policy essentially determines what is going to be your action. Think of a blind person, right? A person who cannot see, right? A blind person is, let's say, walking on a floor and he doesn't know how to, let's say, how to take the lift or how to follow the stairs, right? So what he will do, he will essentially explore different different ways to essentially go to the lift or different ways to go to the stairs. He may not be successful in the first term, the second term, third term, but eventually over time he will essentially learn which path to follow to go to essentially the lift and how to press lift button and so on and so forth. Here exactly the same process will be followed.

We have a policy model and now the name policy may look a bit cryptic, but you just think of it as a normal model, right? What it does, given a state, so, you know, you think of, as I was talking about this, you know, the blind person walking on the floor, right? Whenever he takes a step, right, or takes an action, he actually moves to another place, right? Let's say, let's say turn right, right? And he moves to the right side corridor, turn left, he moves to the right, left side corridor and stuff like that. So every action that moving to the left is an action. And when he moved, that is basically going to change his state. So earlier he was at this position. Now he's at this position.

So a state got changed from here to there because of this movement, left side movement. Here also, for this policy model, there are different states. For example, at t-th time, is t is a state. right. Now given this state if you take an action that would lead to another state that would lead to a state like s t plus 1 right.

Now you may have seen this kind of diagrams in theory of computation, automata theory right where you know there are basically finite state automata, different states, different actions, each action leads a state from the previous one to the current one and stuff like that. Here the concept is exactly the same, you have different states, you have a set of actions right, the set of actions is finite, mostly finite right and essentially, you learn all the states through different actions and overall, and I mean, at the end of the day, you know that which set of actions essentially to follow to essentially reach a certain state, right? So here, the policy model is nothing but our own LLM, right? It's a large language model. It's a language model, right? And what is the state? The state is whatever responses the model has generated so far, right? Let's say the prompt is what is the capital of India, right? This is the prompt. Now the model generates the capital of India is Delhi, right? when the model generates the, that's a state, right? And because of that, that state, right? So after that state, we will have some action, right? So let's say you have an action and the action is to choose the next word. And the next word is, you know, let's say, let me write.

So the capital of India is Delhi. So you have a prompt that is given to the model as input. The model generates this. This is state 0. And then based on certain action, let's say the model now generates this capital.

Now whatever we have so far, this is my current state. right. So then let's say this is state one. So what is state one? State one is the capital. Then the action says the policy model dictates that the next action is going to be off right.

So the word off is generated. So the state so far is the capital off right and so on and so forth. Now who says that at this place the word should be of, the word can also be is, the word can also be in, right? Remember at this state, when the state is the capital, right? How many possible words that can be generated here? there are this, uh, we have this vocabulary. This is the size of the vocabularies, mod V. Let's say the size is, let's say 50 K vocabulary words.

So any word essentially come here, right? So whether this is off or in or the who will dictate this, somebody needs to dictate this, right? So we will see that there is another model similar to policy model. There is another model called the reward model, right? Okay. And through this reward model, the policy model will learn that, you know, it should be off, not in. Okay. You know, the example that I gave this blind person, right? So let's say the blind person turned right and he got stuck at a door, right? And the door is closed, right? So he got hit.

And that's a reward, right? So the reward can be negative, the reward can be positive, right? If the reward is negative, the model will not move to the right further, right? So the model will learn that if I move to the right, there might be something which can essentially stop the entire process, so I should not move to the right. Here also, we'll have a reward model. We'll discuss what the reward model is. And based on that, the model decides which action to take and which action we should not take. So we denote this policy as pi theta.

Theta is the set of parameters. It's a large language model. So all the parameters are denoted by theta. St is the current state. Given the current state, you essentially generate an action.

So essentially, policy model is nothing but probability distribution over action space. In our case, action is essentially the vocabulary at terms. So you essentially produce the distribution. The policy model produces a distribution. And based on the reward model, the policy model will learn which action to take, which action not to take.

What is ST? As I mentioned, ST can be the tokens of the input prompt instructions along with previously generated output tokens, right? So this is the prompt. This is the prompt is, as I mentioned, let's say this is the prompt, right? The prompt is, what is the capital of India? And you have generated till this part. So from the beginning of the prompt, till capital is going to be your state. That's going to be your ST. A is the action.

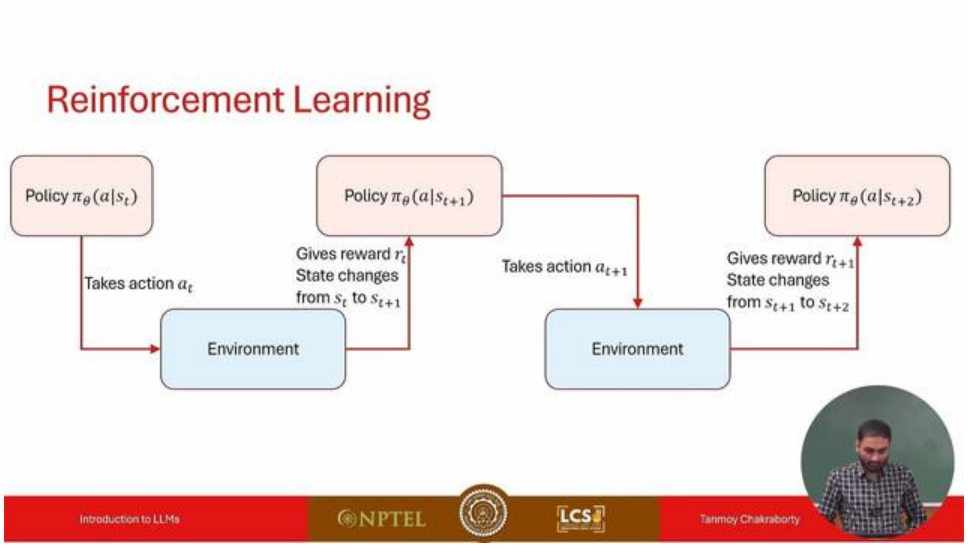
A can be any output token generated by LLM. The policy captures the distribution of the output tokens given the prompt or the instruction. So this is the distribution. So till this part, you may wonder what is new, right? An encoder model, decoder model, a transformer model, they also generate the distribution at the output, right? What is interesting here is that the distribution is controlled by this reward model, right? And we will discuss this. Okay.

This is another example, a policy, Pi Theta, right? ST is the current state, A is the action. If you take an action AT, it will essentially lead to another state. For example, in this case, let's say the prompt is, where is Taj Mahal assistant? Now this is my prompt, right? And you generate Taj Mahal as the output, right? So till this part, it is essentially, it will act as a state, right? So now this is important. So given a policy and an action AT, it will generate and it will basically move to another state, AST plus one, right? So now in a traditional RL setup, let's say a robot is moving on the floor, right? And the robot is exploring all different avenues to essentially move out of the room. So the environment is the room, right? And robot will get stuck.

It will get a reward, right? It will learn that, okay, I should not follow this. I should follow the other direction and stuff like that. So that's my environment. But in the context of LLM, right? There is no precise environment as such, right? There is no explicit environment as such, but the term environment is always important to understand. So we, I mean, whenever we draw this kind of diagram, we'll keep this as, you know, as a standard notation for environment, but don't worry about environment for the time being, right? So in LLMs, in this aspect, there is no environment as such.

And the other difference is that in this case, in the standard reinforcement learning, whenever you, you know, take an action and move to other state, you get a reward. But here we will not get a reward until we get the, we'll basically generate the end of sentence. What does it mean? Let's say, you know, what is the capital of India right? So this is my state this is my prompt and this is my current state s0 right you give it to the model the model says that the so this is my current state. So my current state is S1 till, so from what to the. So you expect that at this stage we get some reward, but we will not get any reward at this stage because as I mentioned, we'll get the reward when we generate the end of sentence.

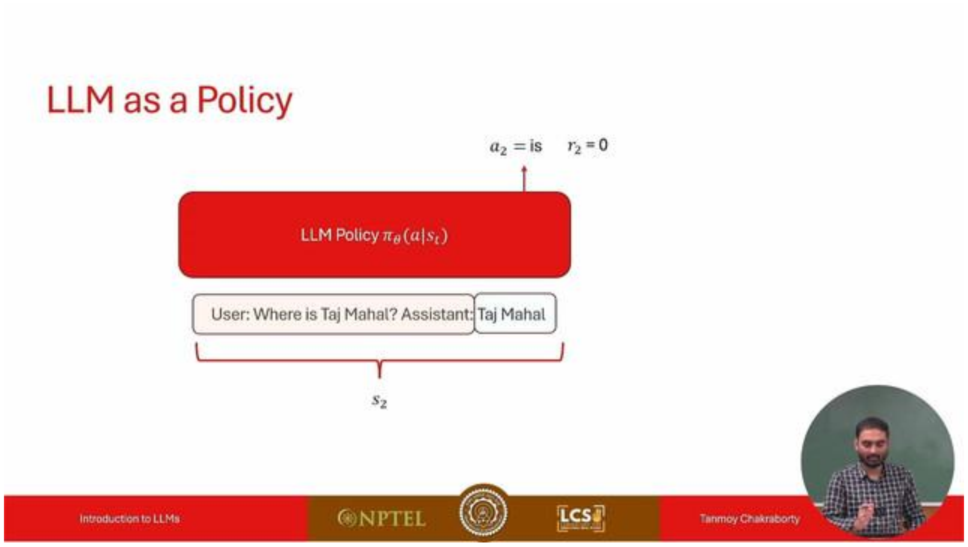
So the capital of India is Delhi. In full stop, end of sentence, right here, you basically get the reward. Okay, so that's the difference. So the policy, you take action a, st, environment, you explode the environment. Through that, you essentially move to state s t plus 1.



Reinforcement Learning

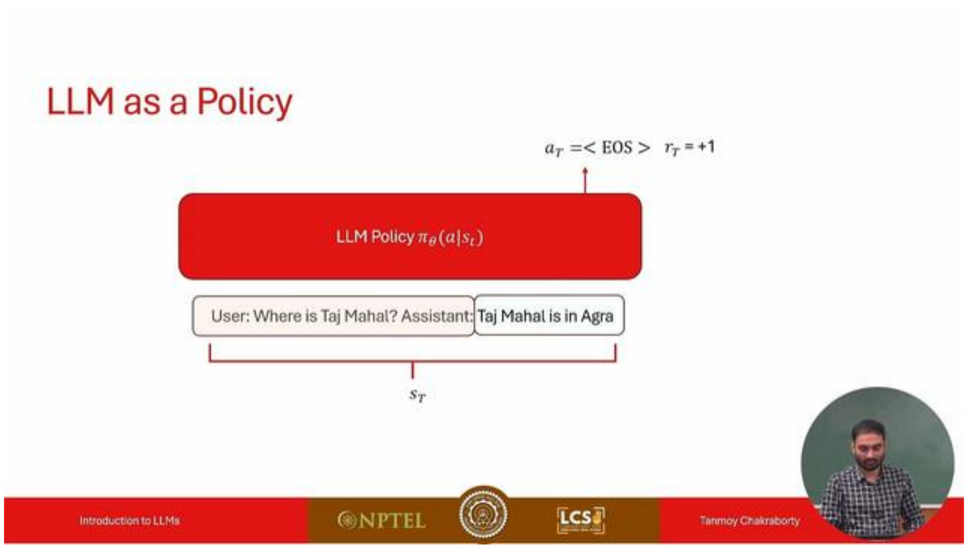
Policy $\pi_\theta(a|s_t)$
Takes action a_t
Environment
Gives reward r_t State changes from s_t to s_{t+1}
Policy $\pi_\theta(a|s_{t+1})$
Takes action a_{t+1}
Environment
Gives reward r_{t+1} State changes from s_{t+1} to s_{t+2}
Policy $\pi_\theta(a|s_{t+2})$
Introduction to LLMs
NPTEL
Tanmoy Chakraborty

At s t plus 1, you again take an action A, explode the environment. You'll move to a state S sub T plus 2, and so on and so forth. Where is Taj Mahal? It will generate 'Taj' reward is 0 because you know it is not the end of the sentence then 'Mahal' reward is 0 till this part, 'is' 0 reward right 'end of sentence' reward is plus 1 right. Agra and I mean output is Agra and then end of sentence so and then plus 1 reward okay. So this is something that we should remember.



LLM as a Policy

$a_2 = \text{is}$ $r_2 = 0$
LLM Policy $\pi_\theta(a|s_t)$
User: Where is Taj Mahal? Assistant:
Taj Mahal
s_2
Introduction to LLMs
NPTEL
Tanmoy Chakraborty



LLM as a Policy

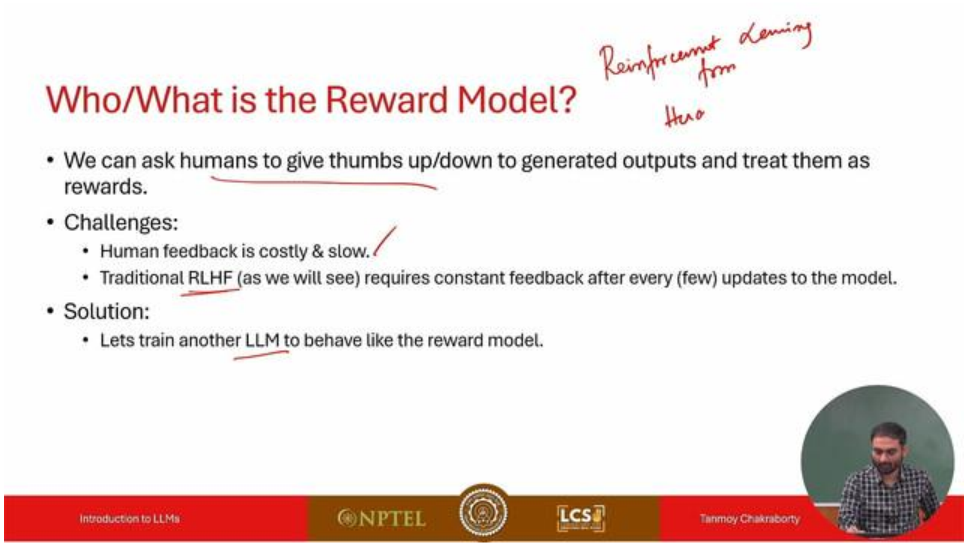
$a_T =< \text{EOS} >$ $r_T = +1$
LLM Policy $\pi_\theta(a|s_t)$
User: Where is Taj Mahal? Assistant:
Taj Mahal is in Agra
s_T
Introduction to LLMs
NPTEL
Tanmoy Chakraborty

All right, now let's talk about this reward model. Remember we discussed something called the policy model. Policy model essentially generates the next action and based on the next action and the reward, based on the next action it essentially moves to the next step, next state. So in reward model, now think about it, the reward model can, I mean an ideal reward model should be a human being. What is the reward model, what is the intuition behind the reward model? So reward model essentially says that you should not do this, you should do this, you should not do this.

Let's say you generate that the capital of India is Mumbai. The reward model should say that this is wrong. So ideally you expect that there will be some human being who will essentially say that this is correct or this is wrong. Thumbs up, thumbs down. So ideally the reward model should be a human being.

What is the challenge? Of course, the first challenge is that it is extremely costly, time consuming to employ human being to look at every output of a model, right? And, you know, remember the feedback that you obtained from human or from the reward model that gets, you know, fed to the model for further fine tuning. So human feedback is extremely costly, slow, So what we do is that we essentially use an NLM as a proxy of a reward model, another NLM, which will act as a reward model. So human feedback is very costly. The second problem is that you essentially need the human being to be present at every iteration, at every step. Whenever the model generates something, that human being should essentially say that this is correct, this is wrong.

Of course, you can think of a hybrid approach where some subset of the output generated by the model can be treated, can be annotated by humans, right? whereas the rest of the outputs can be annotated by LLM feedback and stuff like that. But essentially, in this kind of setup, we will use an LLM which will act as a reward model from the beginning. So traditionally, the term that we use is RLHF, reinforcement learning from human feedback, RLHF, very standard term that people use in RL as well as in alignment. The traditional RLHF requires humans to be there. What we do here, we replace this by an LLM, right? So we have a policy which is an LLM.



Who/What is the Reward Model?

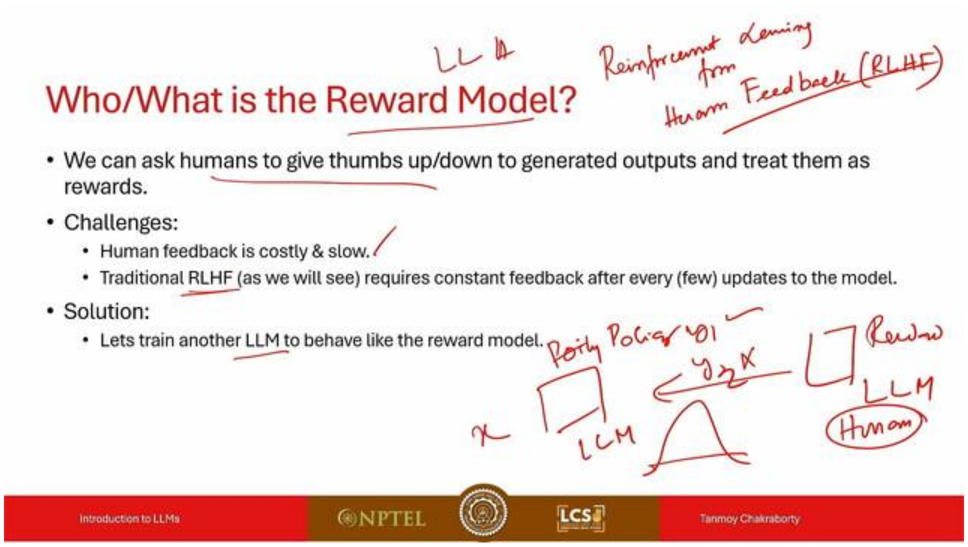
Reinforcement Learning from Hum

- We can ask humans to give thumbs up/down to generated outputs and treat them as rewards.
- Challenges:
  - Human feedback is costly & slow.
  - Traditional RLHF (as we will see) requires constant feedback after every (few) updates to the model.
- Solution:
  - Lets train another LLM to behave like the reward model.

Introduction to LLMs NPTEL Tanmoy Chakraborty

We have a reward which is also an LLM, right? What is the task of this reward model? The task of the reward model is to guide the policy LLM that this response is bad, this response is good, right? Let's say given a prompt X, the model generates Y1. Remember, policy model is a probability distribution, right? It can generate ideally infinite number of responses that you can think of. This is the policy model says, policy model response returns Y1 in one iteration, Y2 in another iteration, right? For the same prompt. It can be possible because LLM is a stochastic process, right? It's a probabilistic process. So the reward model should say that Y1 is preferred, Y2 is not preferred.

Ideally, this should be a human being, but since this is costly, now I am replacing a human with a reward mod, with an LLM to act as a reward mod. I hope the concept is clear till this part. Okay. So the next question is how do we train this reward model? So we need to train the reward model such that the reward model will be able to say that given an X which is a prompt and two outputs Y1 and Y2, Y1 is better than Y2. Of course, we need human labels, but that is fine.

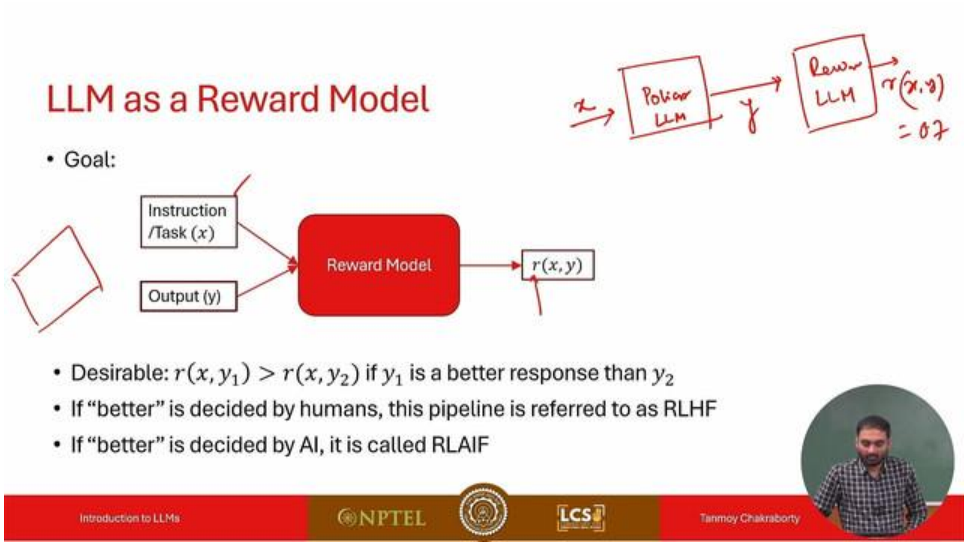


Who/What is the Reward Model?

- We can ask humans to give thumbs up/down to generated outputs and treat them as rewards.
- Challenges:
  - Human feedback is costly & slow.
  - Traditional RLHF (as we will see) requires constant feedback after every (few) updates to the model.
- Solution:
  - Lets train another LLM to behave like the reward model.

We will essentially consolidate a few such responses, and we will annotate this via human feedback, and that annotation will be used to train this reward LLM. So essentially, what this reward model does, given an instruction S, which is a prompt, and you know a response generated by the policy model. So we have a policy model here. So the diagram should be like this. We have a policy model, policy LLM.

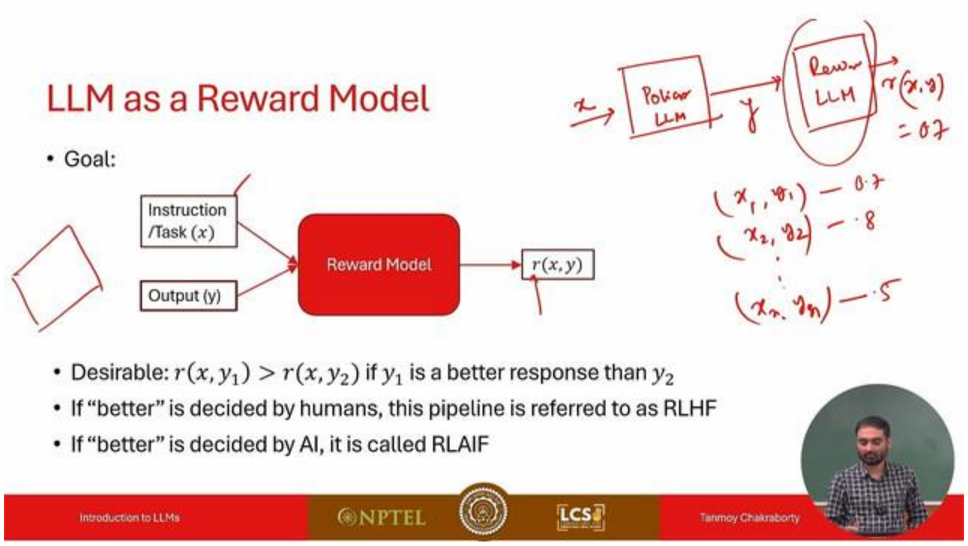
X is the input. The model generates Y, right? And then we have this reward model, reward LLM. The reward LLM will say that the reward R, so the reward is denoted by R, X, Y is let's say 0.7. Let's say the reward value ranges between 0 to 1 and given X and Y, the model will say that this is 0.7.



LLM as a Reward Model

- Goal:
- Desirable: $r(x, y_1) > r(x, y_2)$ if y_1 is a better response than y_2
- If "better" is decided by humans, this pipeline is referred to as RLHF
- If "better" is decided by AI, it is called RLAIF

Okay. Now think about this. So to train this reward LLM, what do we need as a training data? X1, Y1, X2, Y2, dot, dot, dot, Xn, Yn. These kind of pairs, right? And the number. How do you get these numbers? So these numbers will be obtained from humans, right? And these numbers are so subjective.



LLM as a Reward Model

- Goal:

- Desirable: $r(x, y_1) > r(x, y_2)$ if y_1 is a better response than y_2
- If "better" is decided by humans, this pipeline is referred to as RLHF
- If "better" is decided by AI, it is called RLAIF

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

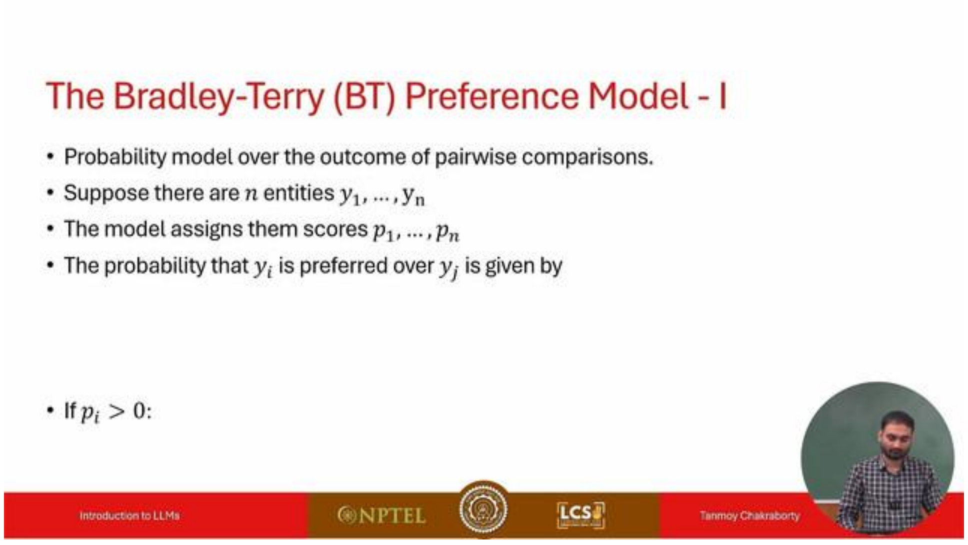
Now of course for questions like what is the capital of India? Where is Taj Mahal located? These are factual questions. These are very easy to rate. Whereas more subjective questions, right? Let's say a poem or an essay, right? An opinion, right? It varies a lot, right? If we employ human being one versus human being two for the same set of pairs, you may see a completely different rating altogether. Rather than teaching the model to produce the score directly, what we can do? You can essentially teach the model to compare between y1 and y2 given an x right? It is easy and it is less subjective okay. Let's say the prompt is write a poem about the festivals in Delhi, right? This is a model generates Y1 in one iteration, the policy model generates Y1 as a response in one iteration and Y2 as a response in another iteration, but the same prompt, right? Now, if you ask a human being to say, to basically rate, which one is better Y1 or Y2? It is easy for the human being to say that Y1 is better than Y2 or Y2 is better than Y1, rather than saying that the quality of Y1 is 0.5 and the quality of Y2 is 0.7, right? This number business is very difficult, right? Instead of this, I mean, it is easy to say that, okay, Y1 is better than Y2, okay? So this is the idea. And in this way, we train this reward model. So given the same prompt X, two responses will be generated Y1 and Y2. Who will generate the response? The responses can be generated by the policy model or the responses can be generated by any models, right? What you essentially need? You essentially need this data set, right? You essentially need this one, X1, Y11, Y12 these are two outputs and what else do you need? y11 is worse than y12 for example again give another prompt x2 y21 y22 right y21 is better than y22 something like that. You need this annotation. Doesn't matter whether Y1, Y11, Y12, Y21, Y22, all these outputs are generated by the policy model or any other elements that you can think of, right? So we'll discuss this thing later. Now, when a human reads this one or the human annotates this one, which one is better, Y1 or Y2, right? This is called reinforcement learning with human feedback. If there is another LLM or another AI which will decide which one is better, Y1 or Y2, we will call this as RLAI, right? Reinforcement learning with AI feedback, okay? So this is crazy, right? I mean, you replace reward model, I mean, the human based reward model by an LLM based reward model.

Now you are saying that to generate the training data set for this reward model, I will use another AI model, maybe another LLM to essentially rate which one is better, nevertheless, I mean. And if you have let's say capacity in terms of deploying models or let's say you don't have capacity but you have money to hire people, you can go with either RLHF or RLAIF, doesn't matter. So this is the structure of the reward model. As I mentioned, this is an LLM. It can be encoder model like BERT, Robota.

It can be decoder model like, let's say, GPT or LAMA or Mistral, whatever, right? What is the input to this model? The input is as follows. So I mean, now this is the model. This is the structure of the model. okay, while training or inference, right? Let's say this is X, this is the instruction, and this is the output. You have some encoder blocks, decoder blocks, right? These are the embeddings of tokens after certain encoder blocks, decoder blocks.

And then you either average all this, you know, all these tokens, right? Pass it through a linear layer. Now this linear layer can be trained, right? I mean, this can be trained with the data set that we talked about earlier, right? This will be trained and the model should be able to say that for this X, the reward value of Y is this much, right? And stuff like that. So you train this linear layer from scratch and of course you can fine tune other layers as required. Let's focus more on the reward model now okay So what is the input to the reward model? Again remember reward model is an lm the input to the reward model is prompt y1 y2 two outputs and which one is better This is the reward model. This is the data set for the reward model.

Training the Reward Model



The Bradley-Terry (BT) Preference Model - I

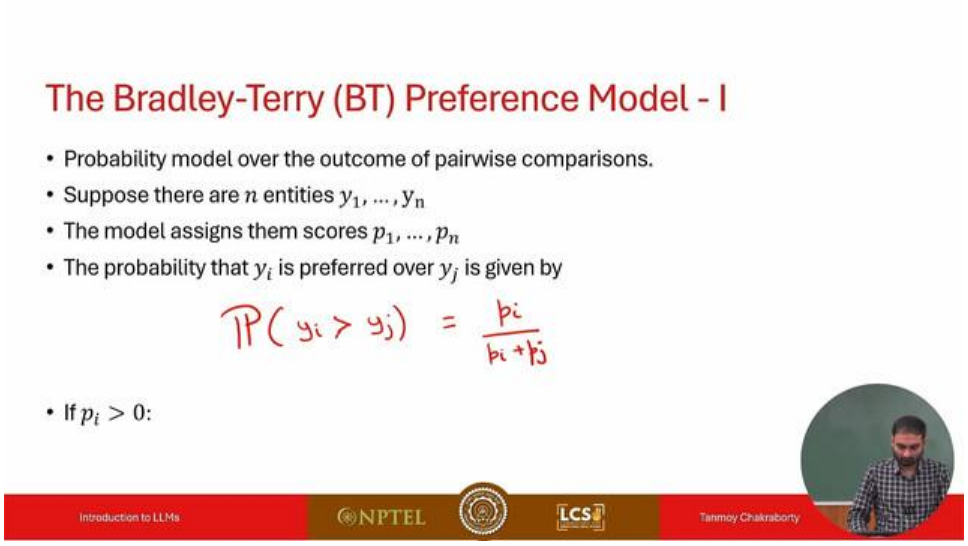
- Probability model over the outcome of pairwise comparisons.
- Suppose there are n entities $y_1, \dots, y_n$
- The model assigns them scores $p_1, \dots, p_n$
- The probability that y_i is preferred over y_j is given by

- If $p_i > 0$:

Introduction to LLMs NPTEL Tanmoy Chakraborty

And we need to train this model. We need to train this model. So we should have some cost function, some loss function here at the end, which we will essentially back propagate. What is the cost function? Okay, so the cost function, I mean I can just straight away write the cost function which is very intuitive but the cost function is essentially motivated by a distribution called Bradley-Terry model. It's a Bradley-Terry model, BT model.

It is essentially used when we have a pair of outputs and you compare pair of outputs and we expect a distribution over a pair of outputs and stuff like that. So I will discuss Bradley-Terry model briefly and then based on that we'll see how the reward model, I mean the loss function for the reward model is computed. So suppose that you have let's say you have entities like Y1, Y2, dot, dot, dot, Yn, right? And you have some sort of score associated with entities. And let's say these scores are preference scores, right? Remember, we are discussing Bradley-Terry model. And from Bradley-Terry, we'll see how the cost function is derived.



The Bradley-Terry (BT) Preference Model - I

- Probability model over the outcome of pairwise comparisons.
- Suppose there are n entities $y_1, \dots, y_n$
- The model assigns them scores $p_1, \dots, p_n$
- The probability that y_i is preferred over y_j is given by

$$\mathbb{P}(y_i > y_j) = \frac{p_i}{p_i + p_j}$$

- If $p_i > 0$:

P1, P2, dot, dot, dot, Pn. If I ask you what is the probability that y1 is preferred, yi is preferred over yj, okay. If we follow Bradlittery model, it will say that I will prefer yi given yj, I will write in this way, yi, it's not the greater than symbol by the way, okay, there is a tilde, there is a small tilde here. Okay, the probability is this pi by pi plus pj.

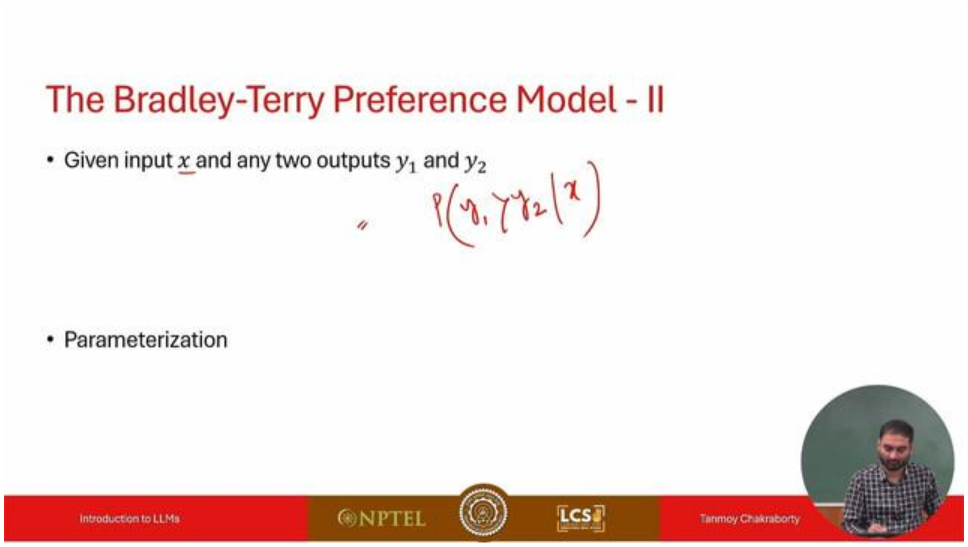
$$\mathbb{P}(y_i \succcurlyeq y_j) = \frac{p_i}{p_i + p_j}$$

Okay, now if your model follows this distribution, you say that it essentially follows the beauty preference model, right? Of course, you can make it more sophisticated, you can say that okay, let me not use this, let me use something else.

So instead of using pi, I will use exponential term. Exponential of Ri. What is Ri? Ri is log of Pi. This is exactly same. Exponent log.

So exponent log Pi is essentially Pi. Right. So instead of writing this, I will write in terms of exponential. Exponent Ri by exponent Ri plus Rj where Ri is log of Pi and Rj is log of Pj, so these are same by the way. Now given an input x remember in our case we have the prompt as an input and outputs are y1 and y2.

so given x what is the probability given x so given x what is the probability that y1 is preferred than y2 okay. So what we do here, we will write in this way. So we'll write it in this way. So exponent of r star, I'll tell you what r star is. y1 x.



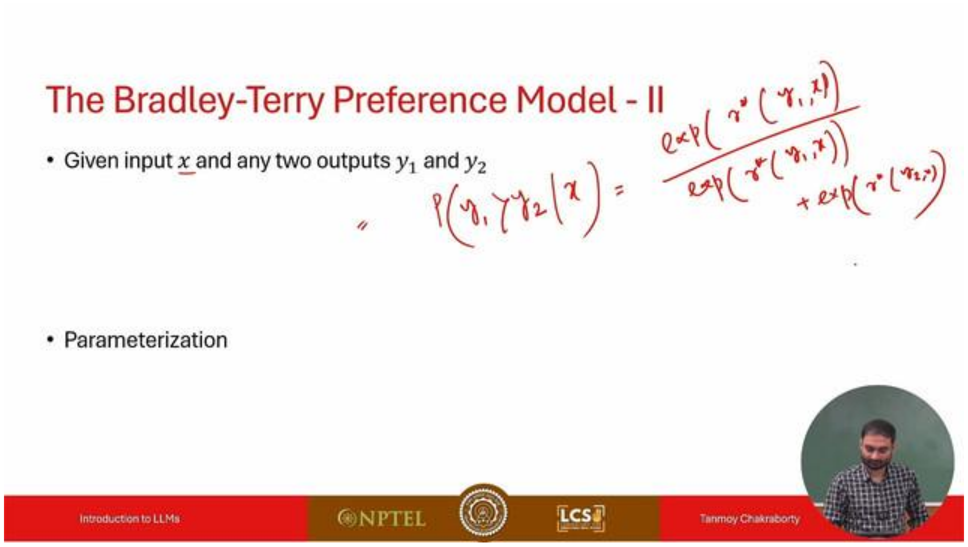
The Bradley-Terry Preference Model - II

- Given input $\underline{x}$ and any two outputs y_1 and y_2

$P(y_1 \succ y_2 | x)$

- Parameterization

r star y2 x, okay. What is this r star? r star is a kind of a, it's an oracle reward function, right. Of course it's a reward function, right. As you see, R, R star, they're the same kind of, right? So R star, R is RI, R star I, they are linked to PI PJs, right? Now PI PJs are what? PI PJs are scores, right? How do you get the scores? The scores are something that are obtained from, let's say, some human feedback or some other models, right? So...



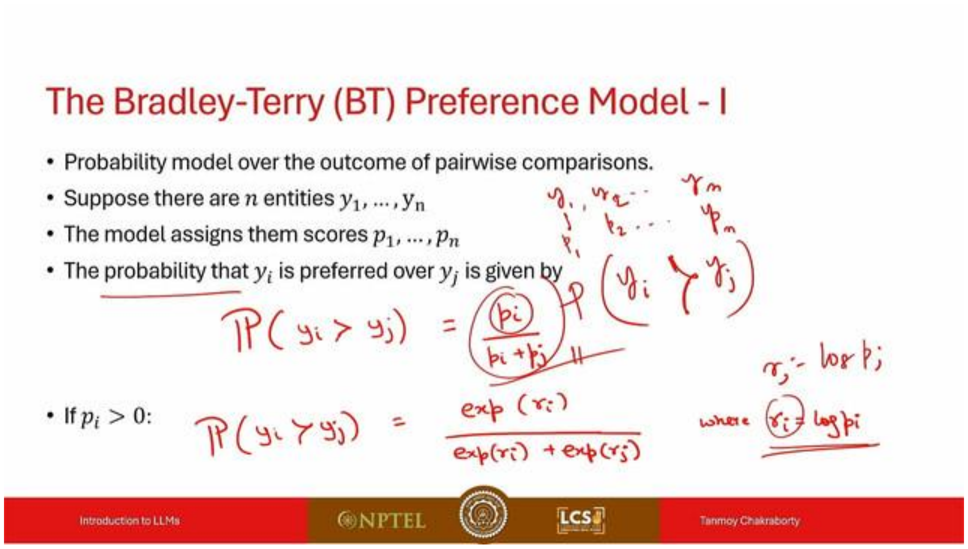
The Bradley-Terry Preference Model - II

- Given input x and any two outputs y_1 and y_2

$$P(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(y_1, x))}{\exp(r^*(y_1, x)) + \exp(r^*(y_2, x))}$$

- Parameterization

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



The Bradley-Terry (BT) Preference Model - I

- Probability model over the outcome of pairwise comparisons.
- Suppose there are n entities $y_1, \ldots, y_n$
- The model assigns them scores $p_1, \ldots, p_n$
- The probability that y_i is preferred over y_j is given by

$y_1, y_2 \cdots y_m$

$p_1 \; p_2 \cdots p_m$

$$P(y_i \succ y_j)$$

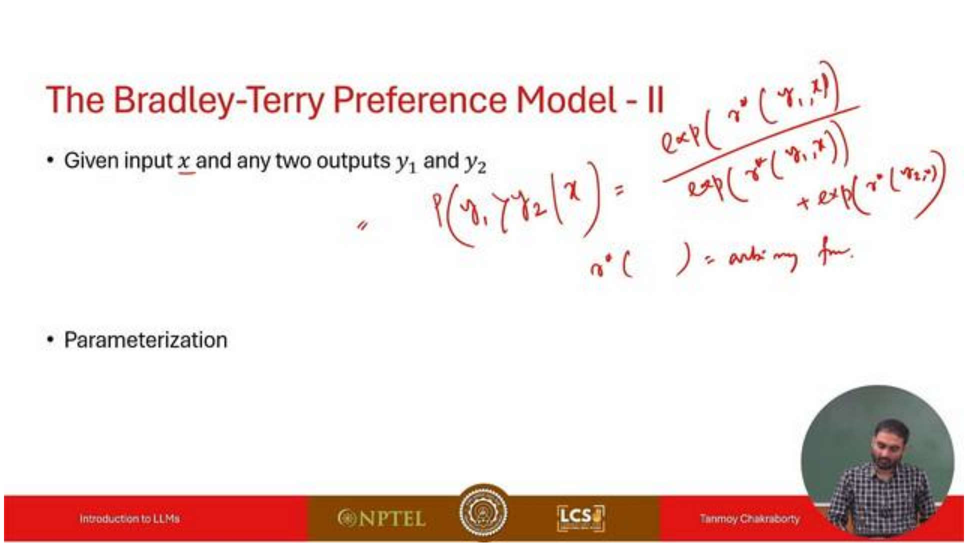
$$\mathbb{P}(y_i > y_j) = \frac{p_i}{p_i + p_j}$$

$r_j := \log p_j$

- If $p_i > 0$: $$\mathbb{P}(y_i \succ y_j) = \frac{\exp(r_i)}{\exp(r_i) + \exp(r_j)}$$ where $r_i = \log p_i$

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

Now r star is an arbitrary function, okay. Now we do not know how do we get this r star, right. So then what we say is that okay, let us parameterize this, right. Let us have some model, theta which will essentially determine this r star okay. So instead of writing this now I will write this one exponential of r theta x comma y1 by exponential of r theta x comma y1 plus exponential of r theta x comma y2 okay.



The Bradley-Terry Preference Model - II

- Given input x and any two outputs y_1 and y_2

$$P(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(y_1, x))}{\exp(r^*(y_1, x)) + \exp(r^*(y_2, x))}$$

$r^*(\ ) =$ arbitrary fn.

- Parameterization

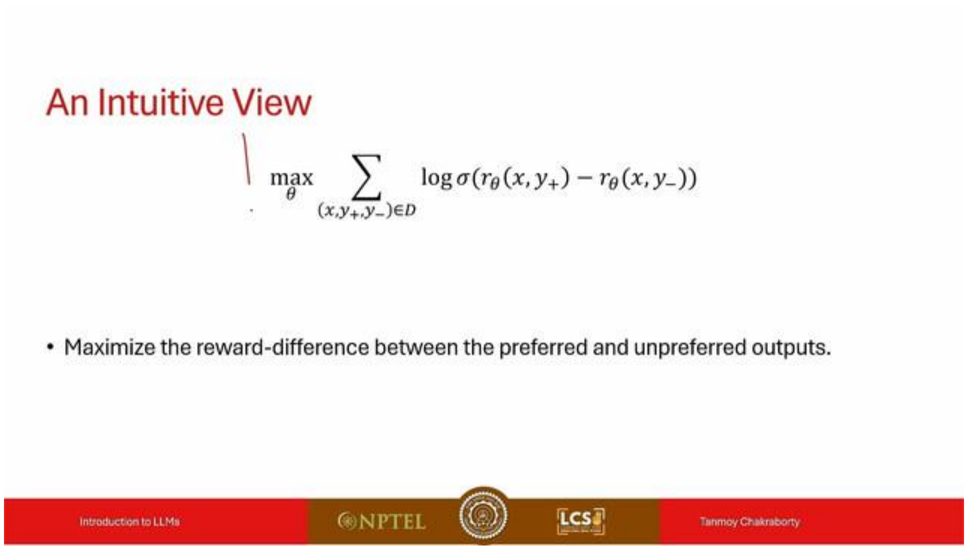
What are the difference between this and these equations? So here we are using r star which is a gold standard or gold label So we do not know the gold label right therefore we parameterize this using some sort of model whose parameters are theta right and what is this theta? This is the reward model right, this is the parameters of the reward model. So what is our aim? The aim is essentially to maximize this. Okay. So given a training data of the form X Y plus Y minus X is the prompt Y plus is the preferred output. Y minus is a non preferred output is the same as Y one by two.

I just tried it in a different way. X Y X Y plus Y minus our task is to find a reward model R theta. Right? Which essentially, and how do we get this? We maximize the log probability of preferences, a simple log likelihood estimate kind of paradigm that you follow here. So we take the log of this and we maximize this. This is my likelihood.

We take the log of this and we maximize this. Very simple. So log of this right which is the likelihood for us log of p theta and p theta we already discussed p theta is essentially exponential of this is p theta for me right. So log of this log of exponential of this by exponential of this plus this. We can further simplify this by dividing you know by dividing numerator and denominator by this okay.

So this is like e to the power x by e to the power x1 plus e to the power x2, something like that. If we divide it by e to the power x2, this is going to be e to the power x1 minus x2 by 1 plus e to the power x1 minus x2. This is exactly the same. If you look at this very carefully what is this? this is e to the power right look at here only focus on this part okay. What is this, this is sigmoid right.

so log of sigmoid of this OK? So essentially, we maximize it. We maximize the log sigmoid of the reward difference. We maximize the log sigmoid of the reward difference. And through the maximization, then we'll use standard gradient descent kind of approach to essentially update the model parameters theta. And we will essentially get modified models over the iterations, right? So this is my objective function for the reward model.



An Intuitive View

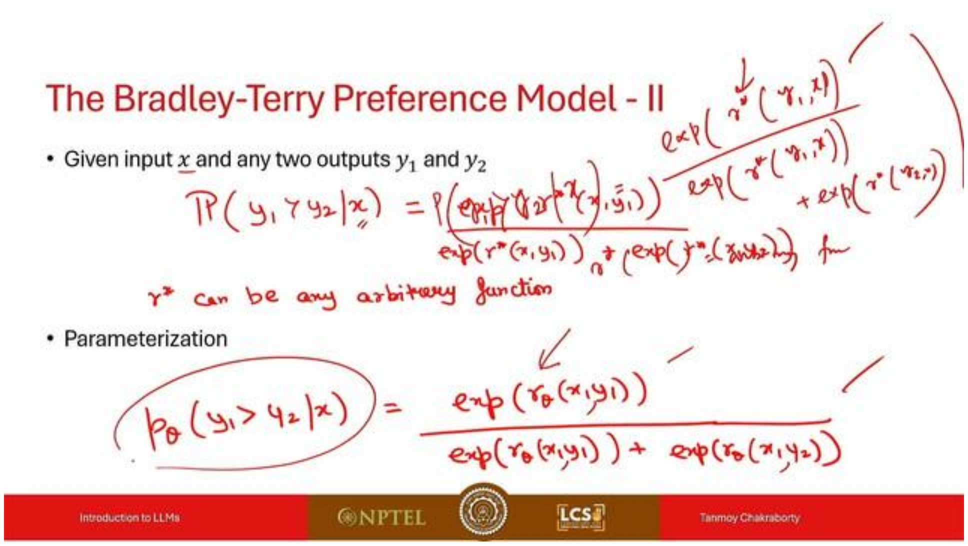
$$\max_{\theta} \sum_{(x,y_+,y_-)\in D} \log \sigma(r_\theta(x,y_+) - r_\theta(x,y_-))$$

- Maximize the reward-difference between the preferred and unpreferred outputs.

Introduction to LLMs NPTEL Tanmoy Chakraborty

Given my training data, x, y plus, y minus, I want to maximize this, maximizing the log sigmoid of the reward difference. Reward difference between the preferred and the unpreferred outputs. All right. So now we understood the reward model. So we have policy model right this is the policy model this is the reward model we know how to train the reward model right and remember the way we trained so for training this we need this pair X Y plus Y minus right but what is what is the output of the model during the inference change the output of the model is the reward R right? And what is the function for this R? X, Y.

So during the inference, we don't need this pair Y plus Y minus. Y plus Y minus is needed only to train the model, okay? But when the model gets trained, it knows that given a prompt and an output, what would be the reward value? So ultimately, the model will produce a number, But the way we train the model is not the way, I mean the traditional model gets trained, right. So here we train the model using a pair of outputs, one is preferred, one is unpreferred and through this the model knows how to score an output given a prompt, okay. Again, think of the high level picture. Given an input X, the model produces an output Y.



The Bradley-Terry Preference Model - II

- Given input x and any two outputs y_1 and y_2

$$\mathbb{P}(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}$$

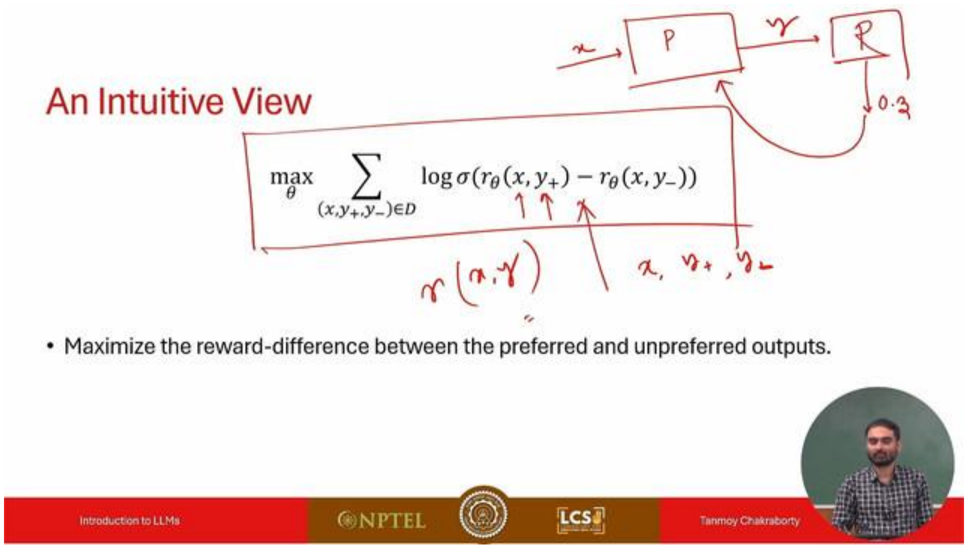
r^* can be any arbitrary function

- Parameterization

$$p_\theta(y_1 > y_2 \mid x) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))}$$

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

The reward model says that, look, it is not a good output, 0.3 or 0.4. This will be fed to the policy model, and the policy model will get updated. So policy model, again, is an LLM, right? If you remember, we wrote pi theta, right.



An Intuitive View

$$\begin{tikzcd} \arrow[r, "x"] & \boxed{P} \arrow[r, "y"] & \boxed{R} \arrow[d, "0.3"] \\ & & {} \arrow[ul, bend left] \end{tikzcd}$$

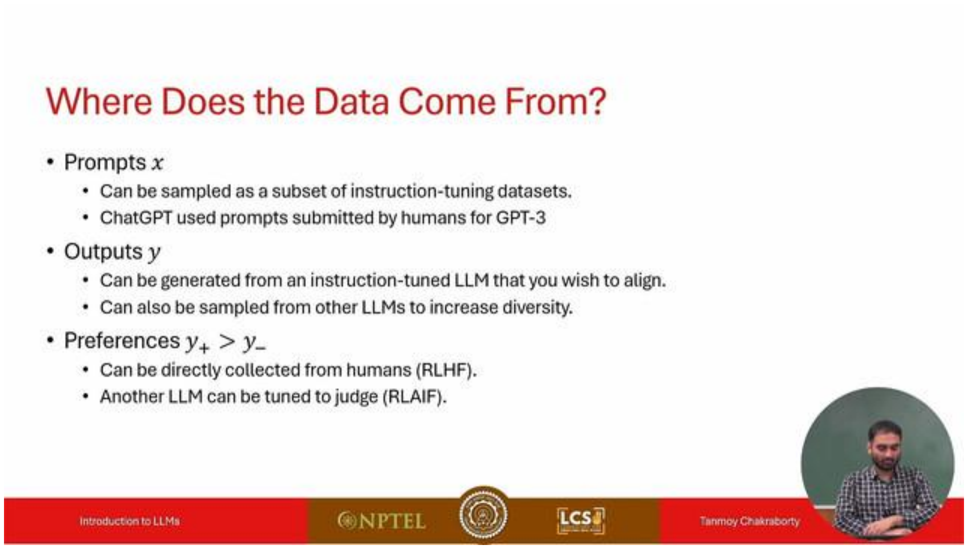
$$\boxed{\max_{\theta} \sum_{(x, y_+, y_-) \in D} \log \sigma(r_\theta(x, y_+) - r_\theta(x, y_-))}$$

$r(x, y)$ x, y_+, y_-

- Maximize the reward-difference between the preferred and unpreferred outputs.

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

So all these theta parameters get updated. Okay. So to train this reward model, what do we need? We need x, y plus, y minus. What is x? x is the set of prompts, right. And how do we get these prompts? So one option is that you already have this instruction fine-tuned.



Where Does the Data Come From?

- Prompts x
  - Can be sampled as a subset of instruction-tuning datasets.
  - ChatGPT used prompts submitted by humans for GPT-3
- Outputs y
  - Can be generated from an instruction-tuned LLM that you wish to align.
  - Can also be sampled from other LLMs to increase diversity.
- Preferences $y_+ > y_-$
  - Can be directly collected from humans (RLHF).
  - Another LLM can be tuned to judge (RLAIF).

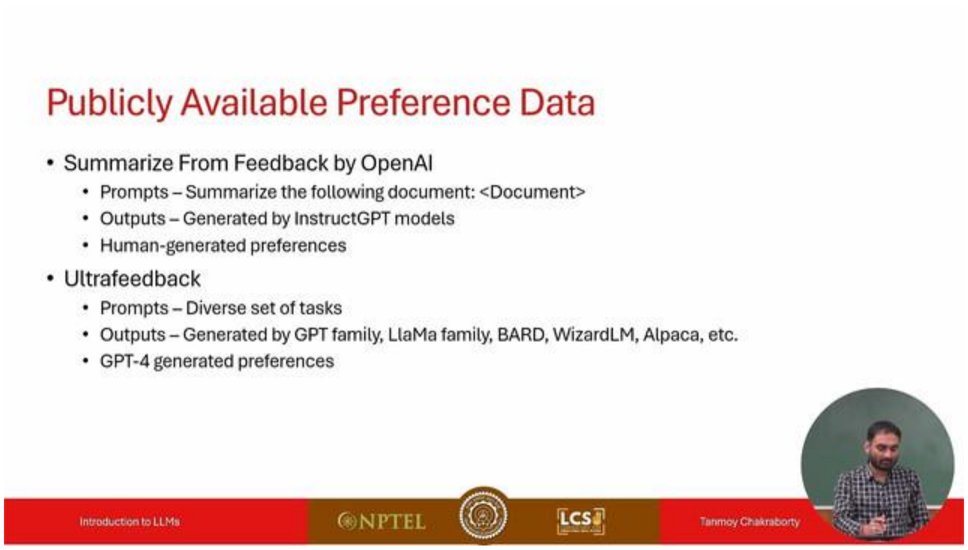
Introduction to LLMs NPTEL Tanmoy Chakraborty

You have the dataset for instruction fine-tuning. Instruction fine-tuning the dataset looks like this. Summarize this and the output. Translate this and the output. And so on and so forth.

What you can do? You can essentially divide this instruction tuning data set into two parts. You keep one part for instruction fine tuning and the rest of the part for alignment. Or...

you can essentially let your model, I mean, let the users use your model the way ChatGPT did. So ChatGPT was announced, ChatGPT was given to the public for usage. We people wrote a lot of prompts, right? And they used OpenAI, used these prompts for their RLHF. This can also be done. So X is sorted.

How do we get this wise right the wise as i mentioned all these outputs can be generated by the instruction tuned lms, right? which you are trying to align or this wise can be generated by any elements right How do you know that Y plus is better, Y minus is not better, right or Y plus is preferred, Y minus is not preferred? Either you employ humans, this will be RLHF, you employ AI as judge and this will be RL AI, okay. So the models are, so the datasets are sorted. These are two publicly available datasets. If you are interested in this space for your research you can use open ai's data set right which essentially looks like this summarize so this is the prompt and the output right from the output and they have human labeled data set whereas there is another data set called ultra feedback right i think some of the models use this They have a diverse set of tasks and different LMs for generating-wise GPT, LAMA, BERT, right, and ALPACA and stuff like that. And they use GPT-4 for this preference, for getting the preference, whether Y plus is preferred or Y minus is preferred, right? So this is more of an RLAIF.

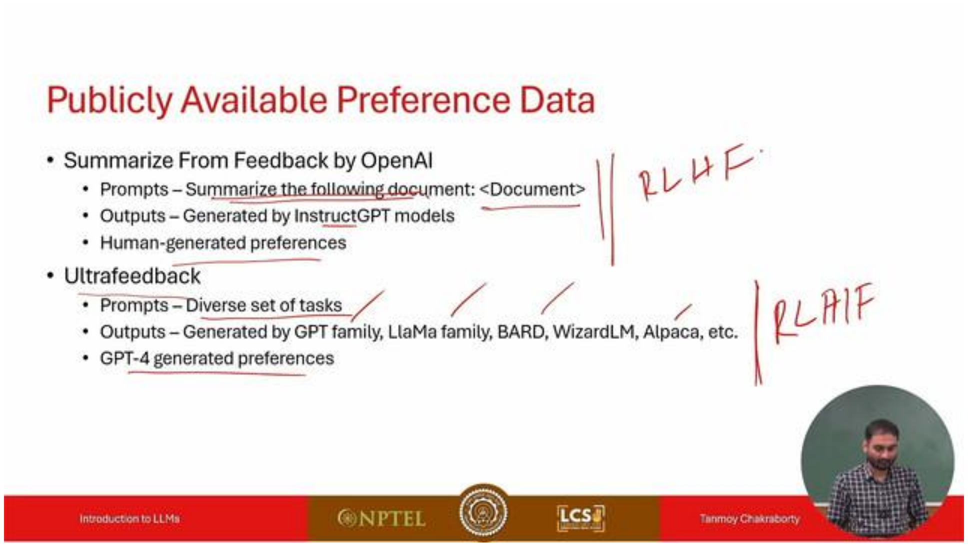


Publicly Available Preference Data

- Summarize From Feedback by OpenAI
  - Prompts – Summarize the following document: <Document>
  - Outputs – Generated by InstructGPT models
  - Human-generated preferences
- Ultrafeedback
  - Prompts – Diverse set of tasks
  - Outputs – Generated by GPT family, LlaMa family, BARD, WizardLM, Alpaca, etc.
  - GPT-4 generated preferences

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

This is RLHF, okay? you can actually do even better. You can use anthropics policy. So anthropic came up with this constitutional AI, where they came up with 16 different principles, right? And taught the models that, okay, you should follow these principles, right? Let's say, and this is mostly useful when you want your model to be more faithful, more unbiased, more non-toxic, right? Let's say you ask the model how to hack my neighbor's internet, right? And the model produces something. Now you say that, look, let's say the model produces two responses, Y1 and Y2. Then you ask the model again through prompts that which of these assistant response is less harmful, right? Let's say your model says that in order to hack your neighbor's Wi-Fi, you do this, you don't do this and you follow this process and stuff like that.



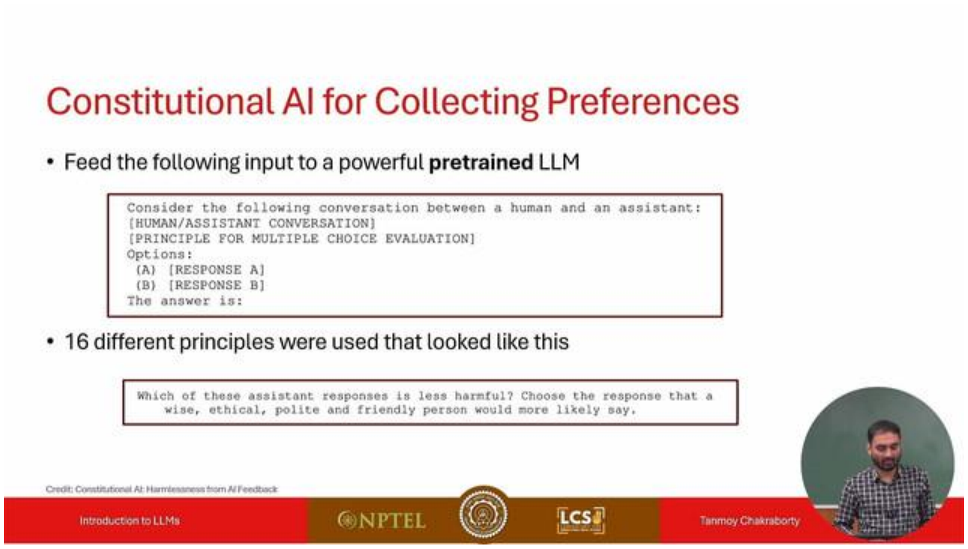
Publicly Available Preference Data

- Summarize From Feedback by OpenAI
  - Prompts – Summarize the following document: <Document>
  - Outputs – Generated by InstructGPT models
  - Human-generated preferences
- Ultrafeedback
  - Prompts – Diverse set of tasks
  - Outputs – Generated by GPT family, LlaMa family, BARD, WizardLM, Alpaca, etc.
  - GPT-4 generated preferences

RLHF

RLAIF

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



Constitutional AI for Collecting Preferences

- Feed the following input to a powerful **pretrained** LLM

```
Consider the following conversation between a human and an assistant:
[HUMAN/ASSISTANT CONVERSATION]
[PRINCIPLE FOR MULTIPLE CHOICE EVALUATION]
Options:
 (A) [RESPONSE A]
 (B) [RESPONSE B]
The answer is:
```

- 16 different principles were used that looked like this

```
Which of these assistant responses is less harmful? Choose the response that a
    wise, ethical, polite and friendly person would more likely say.
```

Credit: Constitutional AI: Harmlessness from AI Feedback

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

The other response is you should not consider hacking your neighbor's Wi-Fi because this is unethical and stuff like that, okay. Now you ask the model itself that, you know, which one to choose among this y1 and y2, choose the response that is more wise, ethical, polite, friendly and stuff like that. So now between these two the model will definitely understand that okay this should be Y2 is preferred than Y1 and then based on that you give this as a feedback to the model again, two prompts okay, two prompts and the hope is that the model will keep refining itself over the iteration. So it's a nice paper, you should look at this paper

Constitutional AI by Anthropik and you will see all these 16 different principles. So let's now discuss about reward maximization objective.

We have almost set the stage, the final touch up has remained. We have this reward model R, which we have just discussed, given a prompt text and output Y, it will generate a number. Now, what is our task? Our task is to essentially let me draw this figure again. This is R, this is my policy. So my task is to find the optimal policy.

Meaning my task is to fine-tune the policy LLM again and again based on the rewards feedback or the feedback obtained from the reward model. So that it will be able to essentially cater to the human feedback that was used to train this model. right. So now here there is an interesting point or interesting aspect that one should consider. So if the task is to, now remember what would be the optimal goal of this policy model? This is the policy model.

What is the optimal goal of the policy model? The optimal goal of the policy model will be to generate outputs such that its corresponding reward will be maximum. So the model will generate output in such a way that its corresponding reward will be maximum. So what would end up happening is that the model, the LLM, the policy model will start hacking the reward model. What does it mean? It means that it will only learn how to get reward, how to maximize reward. And in a classroom setting, essentially, the student will only learn how to get good marks in the exam.

But the student will not learn the underlying concept as such. It is similar to that. And there are different ways to get good marks. I will not discuss all these things. Here, in this case, let's say the model knows that, okay, if I write a verbose response, right, a paragraph response, instead of writing a concise response, or if I use emoticons, right, if I use special symbols or Greek symbols here and there, right, so it is likely that I will get good reward.

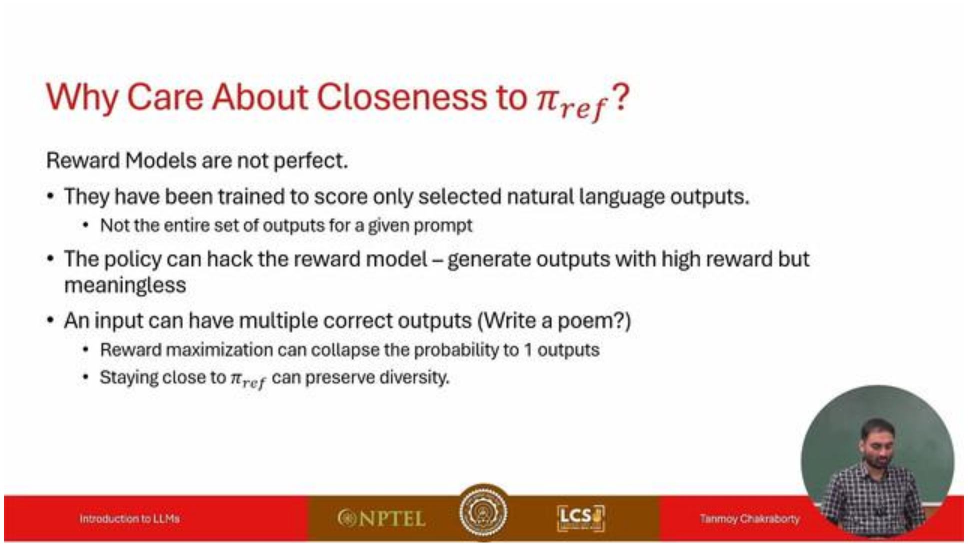
So it will try to hack the reward model. In order to address this, what is important is that, now remember, let's say when the entire process starts, right? So we start from a basic or the base policy model, let's say pi theta one. You incorporate some feedback, it moves to pi theta 2, pi theta 3, and so on and so forth. And then, let's say finally, pi theta star. And of course, pi theta star would be much, much better than pi theta 1, pi theta 2 in terms of reward maximization, which is the only objective that we discussed so far.

It may happen that this guy will start producing responses which would just get maximum reward but which may not look like a human generated response. So many emoticons, right? Or the answer may not be precise, may not be faithful, stuff like that, right? You know, it basically reminds me of, you know, my high school education, right? Secondary, high secondary. So 10 standard, 12 standard. There, my friends used to say, let's say if you don't know any answer, any question, I mean answer of a question in exam, you just write something. You just fill the entire page.

You write something interesting. Try to fill out the page. Don't keep it blank. Versus if somebody knows the answer, he or she will write a very precise one or two line sentences. And God knows, maybe the examiner will not look at the correctness of the answer, but look at the volume of the answer.

And based on that, the marks will be given something like that, right? So we'll try to hack the examiner's mind. Okay, which is not objective here, by the way. So in order to cater to that, what we do? we will also restrict the model, the final model to deviate far from the initial state. We will not let the model deviate more from the initial state. The model will keep updated, but it is not that it will be very, very different from the initial model.

So for that, what we need, we need a reference model. what is the reference model? The reference model is the initial state of the policy model. Let's say pi theta 1 is my reference model and I want that so this is pi ref and the hope is that my final model should not be very very different from the reference model, okay. So how many objectives do we have? We have two objectives. One objective is we maximize the reward and the second objective is we don't want the model to be deviated from its initial state essentially.



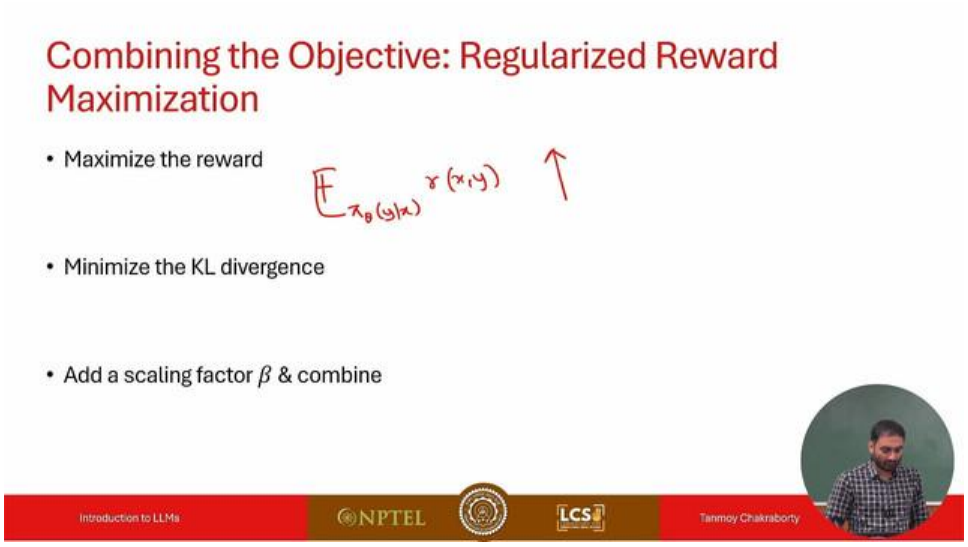
Why Care About Closeness to π_{ref}?

Reward Models are not perfect.

- They have been trained to score only selected natural language outputs.
  - Not the entire set of outputs for a given prompt
- The policy can hack the reward model – generate outputs with high reward but meaningless
- An input can have multiple correct outputs (Write a poem?)
  - Reward maximization can collapse the probability to 1 outputs
  - Staying close to π_{ref} can preserve diversity.

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

So maximize reward is essentially maximizing this R, X, Y. Now remember when I say that I want to maximize the reward, I always say this from expectations point of view, right. Expected reward, remember this. It is because, you know, responses are all non-deterministic and given a prompt, there can be millions of responses, right? So what we are interested in, we are interested in computing something called the expected reward. What is expected reward? Given a prompt, you give this prompt, write a poem.



Combining the Objective: Regularized Reward Maximization

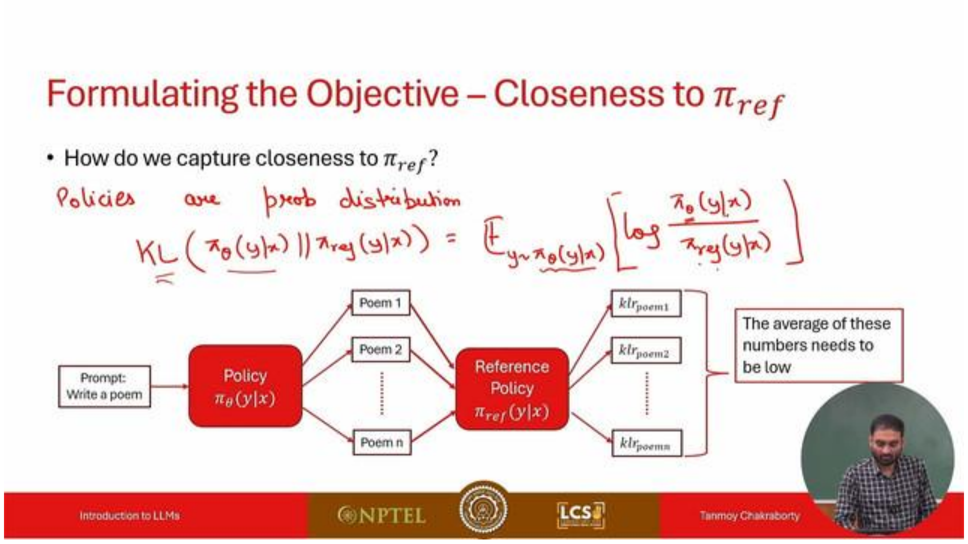
- Maximize the reward $\mathbb{E}_{\pi_\theta(y|x)} r(x,y) \uparrow$
- Minimize the KL divergence
- Add a scaling factor β & combine

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

The policy model will generate outputs. Now these are y1, y2, y3, dot, dot, dot, right? Every output is fed to the reward model. The reward model will produce a score. For this poem, this is the score, reward score. For this poem, this is the reward score.

For this poem, this is the reward score, right? And what is the expected reward score? The expected reward score is just the average of all the rewards. So we write in mathematics, we write in this way expectation of the output generated from the policy, what is the reward and this should be high. And when I say that two models, let's say you know the reference model pi ref and the final model uh pi theta. They should not be very far from each other. We will use a metric called KL divergence.

I mean, all of you know, right? This is a KL divergence essentially measures the difference between two distributions. Remember, Pi Theta, Pi Rep, they are essentially distributions. Language models are distributions, right? They produce distributions over outputs, over tokens. So we measure the KL divergence between the reference model and the current model and I mean in the expectation term it is essentially log of pi theta y given x and log of pi ref y given x right. So what essentially you are doing given an x you generate y right through policy your policy model will say right? Your policy model, there are, remember there are two models, reference policy model, current policy model.



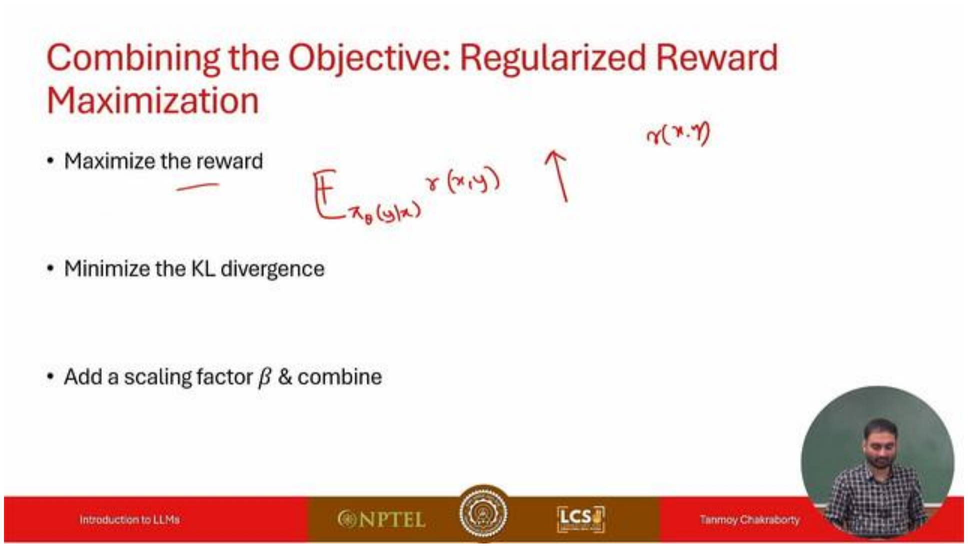
Formulating the Objective – Closeness to π_{ref}

- How do we capture closeness to π_{ref}?

Policies are prob distribution

$$KL\left(\pi_\theta(y|x) \,||\, \pi_{ref}(y|x)\right) = \mathbb{E}_{y \sim \pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right]$$

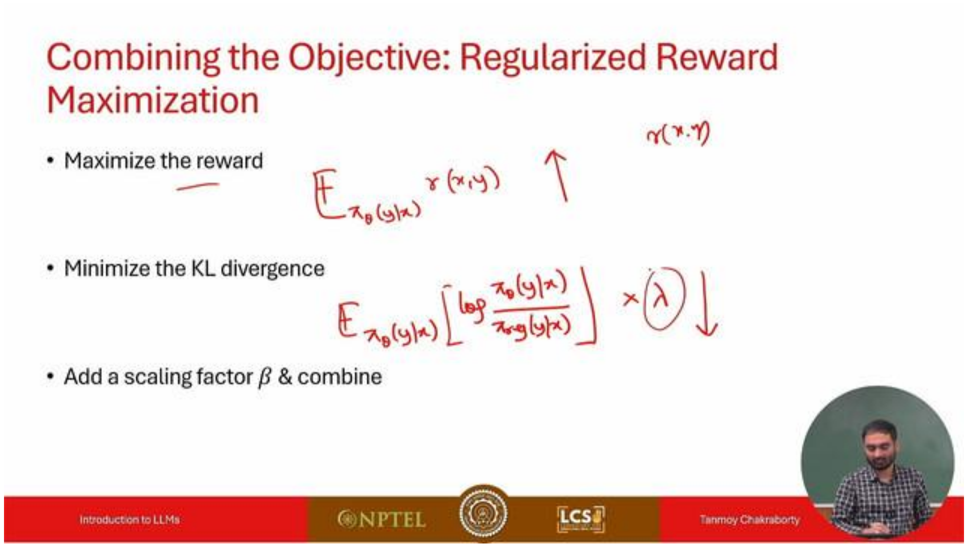
Your reference policy model will say, what is the probability of Y given X? Your current policy model will say, what is the probability of Y given X, right? You take the log of this ratio and you take the expectation because there are many such Ys given X, right? This is scale divergence. And what is our objective? Our objective is to minimize this. maximize this okay and minimize this okay So maximize the expected reward minimize the KL divergence right and in order to balance these two things I will use a scaling factor you know lambda the scaling factor essentially says that okay I would like to give less weightage to the scale divergence right or more weightage to the KL divergence and stuff like that. So this is the final objective function for RLHF. So expectation, given this pi theta, we generate y, the reward which we want to maximize.



Combining the Objective: Regularized Reward Maximization

- Maximize the reward $\mathbb{E}_{\pi_\theta(y|x)} r(x,y) \uparrow$ $\quad r(x,y)$
- Minimize the KL divergence
- Add a scaling factor β & combine

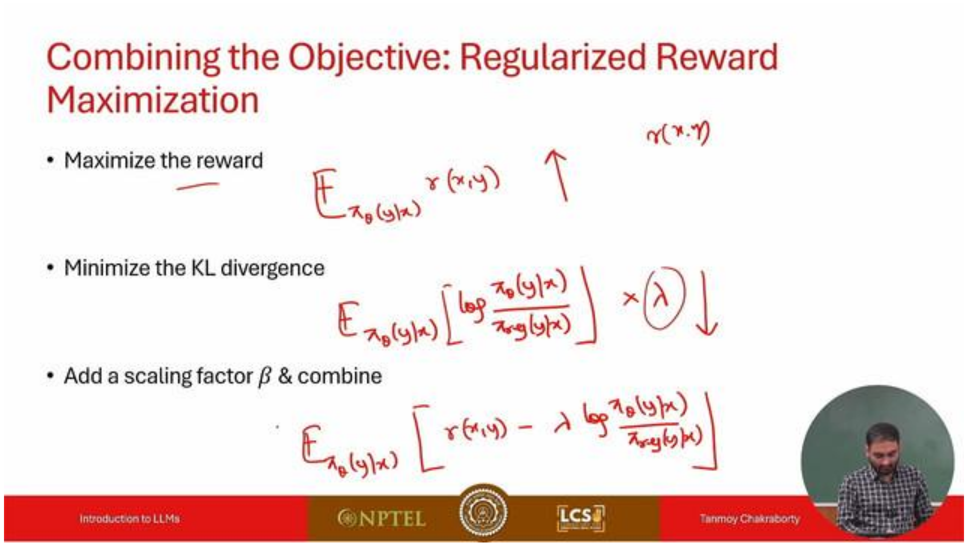
Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



Combining the Objective: Regularized Reward Maximization

- Maximize the reward $\mathbb{E}_{\pi_\theta(y|x)} r(x,y) \uparrow$ $\quad r(x,y)$
- Minimize the KL divergence $\mathbb{E}_{\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right] \times \lambda \downarrow$
- Add a scaling factor β & combine

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



Combining the Objective: Regularized Reward Maximization

$r(x,y)$

- Maximize the reward

$$\mathbb{E}_{\pi_\theta(y|x)} r(x,y) \uparrow$$

- Minimize the KL divergence

$$\mathbb{E}_{\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right] \times \lambda \downarrow$$

- Add a scaling factor β & combine

$$\mathbb{E}_{\pi_\theta(y|x)}\left[r(x,y) - \lambda \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right]$$

So this is the objective function which we want to maximize. So maximize the reward. And this minus, we minimize this scale divergence. OK? So next day, we'll discuss more about this. And next day, we start from here, okay? We'll say that, okay, this is my reward model.

This is my KL divergence part. And this is the combination of this. This is my objective function. And from now, I mean, now, how essentially we can use this objective function to update the policy model, okay? Thank you.