Introduction to Large Language Models (LLMs)

Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti

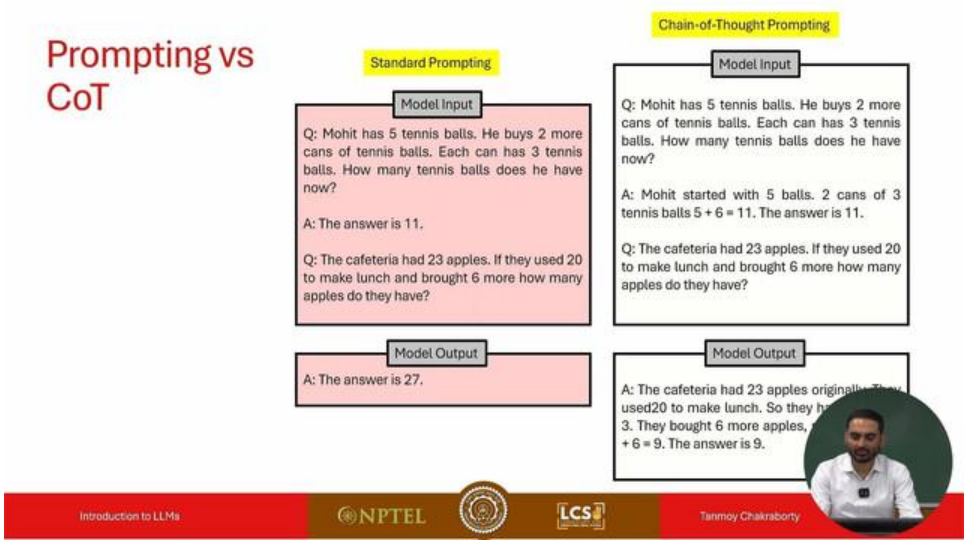
Department of Computer Science & Engineering

Indian Institute of Technology, Delhi

Lecture 23

Advanced Prompting and Prompt Sensitivity

So welcome back. So we were discussing different prompting techniques. We saw in the last lecture how to write prompts, what are the components of a prompt, what's prompt template, how it affects the accuracy, how the accuracy of the model increases with the increasing size of the model parameters for different types of prompts, and so on. So now let's discuss different advanced prompting techniques. So in 2022, people realized that if you just write few soft prompts, like let's say, let's take a math reasoning question.



Let's say Mohit has five tennis balls. He buys two more cans of tennis balls. Each can has three tennis balls. How many tennis balls does he have now? This is a question and this is your answer then. So this is your one shot.

And you ask this question, a cafeteria had 23 apples. If they use 20 to make lunch and brought six more, how many apples do they have? The output you expect is nine, but the model produces 27. Because the model has been exposed to appear, which is a question and answer, question and answer, the final answer. but the model doesn't know how to reason about it, right? So this is your standard prompt, but if you start writing this, if you had written in this way, this is your question. And these are the steps.

So Mohit started with five balls, two cans of three tennis balls, five plus six equals to 11. The answer is 11. So this is your example. Okay. Then you give this question, right? Cafeteria had blah, blah, blah.

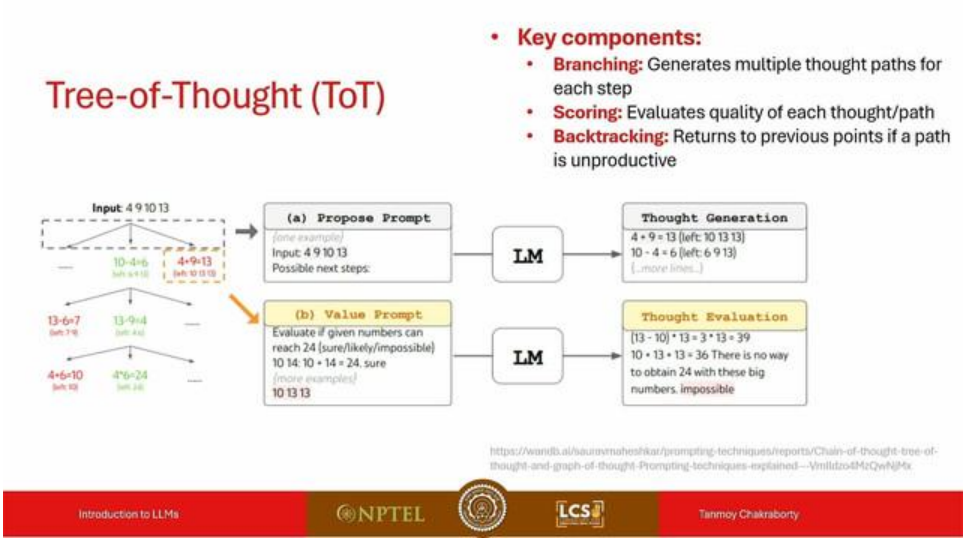
And you see that the model will not only produce the right answer, but also the steps required to produce this answer. And this new prompting technique is called the chain of thought prompting, COT. What are the thoughts here? The thoughts are, let's say in this case, the first thought is the cafeteria had 23 apples originally. The second thought is they used 20 to make lunch. The third thought is so they had 23 minus 20 equals to three and so on.

These are the steps. And this essentially mimics the way a human thinks, right? Now, surprisingly in the COT paper, they also showed that if you don't have any example, right? You want to do it in a zero shot manner. You just write, think, let's think step by step, okay? Write, let's think step by step and then your question that you want to solve. Then also you see that the model will be able to decompose the problems into steps and then each of the steps will be solved one by one. After chain of thought prompting, people also suggested something called chain of thought with self consistency.

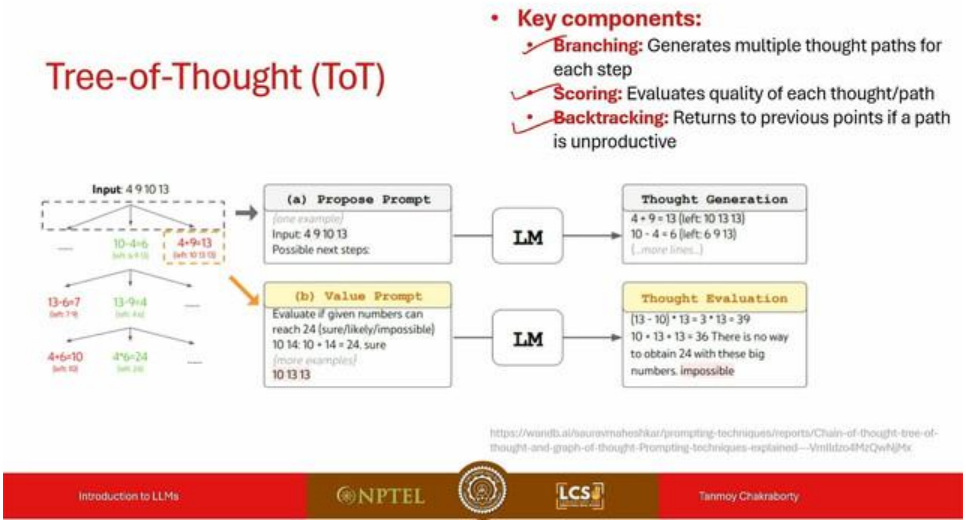
Here the idea is that you ask the same question to the models multiple times and as all these LLMs are stochastic, right? Some of you know that these models are called stochastic parrot, right? Why parrot? Because these models mimic, right? So the way a parrot mimics what the parrot learns. Here also, the model mimics what the model learns during the pretending step, right? They're not able to invent something new. That's why they are called parrot and stochastic because they are not deterministic. You know, the answers are always different for different instances of a single prompt. So you give the question which you want to solve.

This is one set of thoughts. This is another set of thoughts. This is another set of thoughts, right? You have multiple sets of thoughts at different instances for a specific question. Right? And then you have some sort of voting mechanism, right? Let's say this chain produced output nine. This chain also produced output nine.

This chain also produced, this chain produces the output 24 and so on and so forth. You have some voting mechanism based on the voting. You choose one of the thoughts, one of the sequence of thoughts, right? And of course, this is not as simple. If you want to look at the paper, they also had some sort of, you know, a module which decides which chain one should essentially pick. Consecutively, the next model was proposed, which is called Tree of Thoughts, right? So in Tree of Thoughts, the idea is that it's kind of a branch and bound, right? So you start with step and then you explore all possible sub-steps, right? Now you look at, you take one of the sub-steps, you further explore this and you keep on doing this thing.



So you basically produce the entire tree, right? Where each branch, right? From root to leaf would be your hypothesis, right? Would be your thought. Okay. So here, you need three components. One is called branching, other is called scoring, other is called backtracking. So branching module is responsible for deciding whether I should branch, I should split it into different branches or not.



Scoring module evaluates a particular branch, right? And decides whether, and then there's a module called backtracking. It decides whether I should backtrack to the previous, right? The previous node, and then, you know, I should further branch out or not. Let's look at this example. So this is called this famous mathematics question called 24, 24, guess 24

games. I don't remember the exact name, but the idea is that you are given numbers, right? Let's say four numbers, and you have to come up with ways to essentially result the number 24.

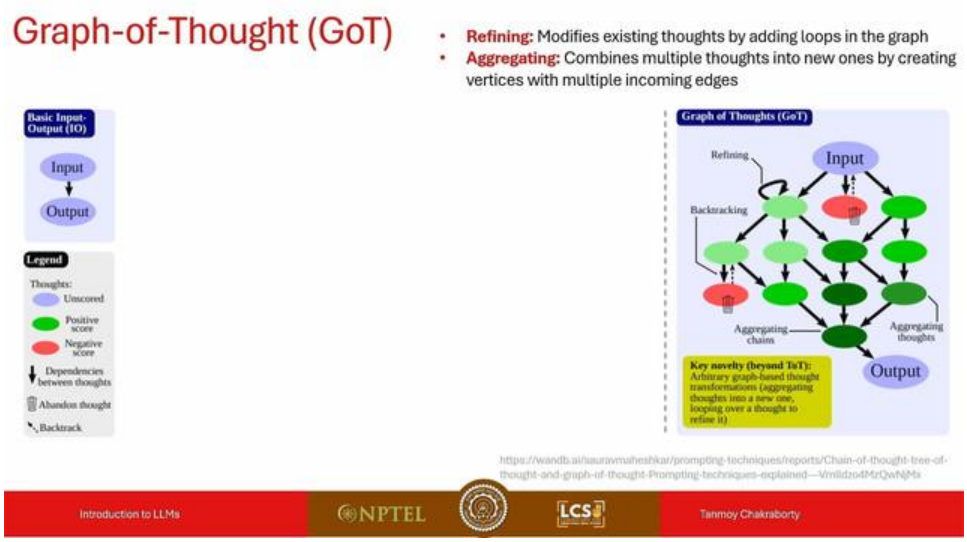
Let us say you are given 4, 9, 10, 13. Now what are the operations that you can do to obtain 24, right? Remember when you do some operations between a pair of numbers, let us say you do addition 4 plus 9 which will result in 13. Now 13 will become one of the candidates into your possible numbers, right? So in this case, let's say you start with 4, 9, 10, 13, right? You have one branch which suggests you to subtract 4 from 10. So you have 10 minus 4 equals to 6, right? So you have exhausted 10, you have exhausted 4. So what else is remaining? You have the remaining numbers 9, 13, and the 6 that you obtained, right? So you can't use a number multiple times, by the way, okay? So the numbers which are left are 6, 9, and 13.

Now next step can be again subtraction, 13 minus 9, which will result in 4. So what are the numbers which are left? Six and four, right? The next step can be four times six, which will result in 24. Now this can be one successful branch, right? You know, you can also think of other branches. For example, you can say that, okay, I start with four plus line nine, which will produce 13. And then you, right? You further branch out this thing.

So you see here that, you need a module which will decide whether to branch out or not. You need a module which will evaluate the quality, right? Whether, right? Whether you are going to the right detection or not, right? And you need something which will help you backtrack. Let's say you have decided something, you have ended up generating something which you feel, which your scoring function feels that is not correct. Now you have to backtrack. This is same as branch and bound kind of algorithmic setup.

Okay? This scoring, backtracking, branching, these are different modules, right? As you see here, and these are different language models, by the way, okay? For example, here you see that the scoring module says that you have already generated 36, right? There is no way to obtain 24 with this, such a big number, so you stop here. If you stop, right, if your scoring module says that you should stop here, you can stop and you can start backtracking, okay? This is tree of thoughts. Now, the obvious choice is that from chain of

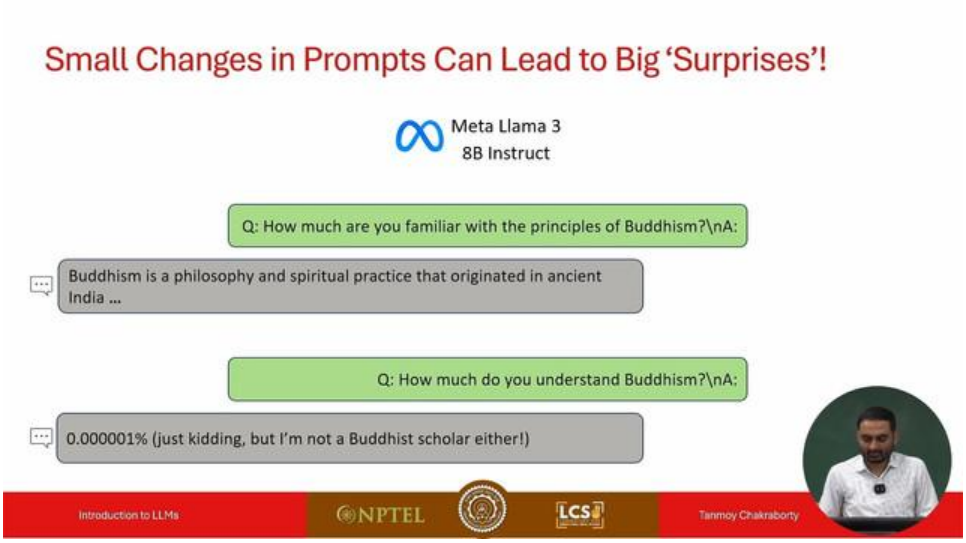
thought, you have tree of thoughts. What next? Graph of thoughts, right? In graph of thoughts, the idea is that, you know, you have, you see here, here in the graph of thoughts, your chains, all these chains are connected, right? As you see, you know this, green circles indicate that your scoring module says that this is a positive score whereas you know this kind of circle indicates that this is unscored and red circle indicates that this is a negative score, right. So here also you start with an initial thought and then you branch out, you can backtrack, you can refine, you can use, you can reuse some of the existing thoughts and so on and so forth, okay.



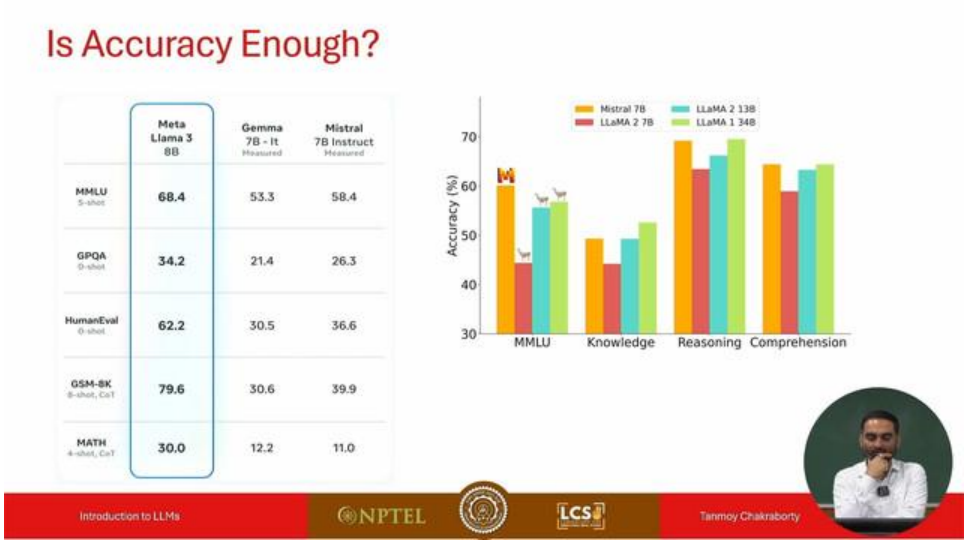
So you need an aggregator, which will decide whether two thoughts can be combined or not like, like this. So chain of thoughts, one single thought, one single chain of thoughts, chain of thoughts with consistency, self consistency. You have multiple chain of thoughts, tree of thoughts, right? Now you have the capacity to essentially backtrack. Okay. And you can score each of these thoughts.

And then graph of thoughts, all these thoughts are connected. And these are some of the advanced prompting techniques that people use. So far, we have seen these models can take multiple types of prompts. But as I mentioned earlier, that they are highly sensitive to prompts, right? If you change a prompt here and there, you will see the output of the models will change drastically. Now, so along with accuracy, there is another factor called sensitivity, which also corresponds to robustness.

This needs to be measured, right? So how do you measure the given data and the model? can we measure how sensitive the model is with respect to the data? Okay. This is something that we will not discuss. Let's say you take LAMA3 instruct model, right? ADB instruct model and you ask the model that how much are you familiar with principles of Buddhism, right? And it will say that Buddhism is a philosophy and spiritual practice that's originated in ancient India and blah, blah, blah, right? You ask the same question again, how much do you understand Buddhism? Almost the same question, right? But this time it says that 0.00001%. Right just kidding i am not a buddhist scholar either right you see my intention was to ask the same thing repeatedly but this time in the model thought that kind of i was joking.



So this is something that you want to prevent because if this happens model will not become robust okay. So the question that we're asking is, by the way, this is the research that we have done in our lab, okay, very recently. The question that we are asking is whether accuracy is only a good measure or not. So far, we have been only talking about accuracy, right, with the increasing size of the model, how accuracy impacts and so on and so forth, right? And you see here that you know, models like LAMA-3, GEMA, Mistral 7 billion instruct, right? On across different tasks, MMLU, human evil, GSM 8K. These are all different reasoning tasks, right? People always report the accuracy, okay? But we said that accuracy is not enough.

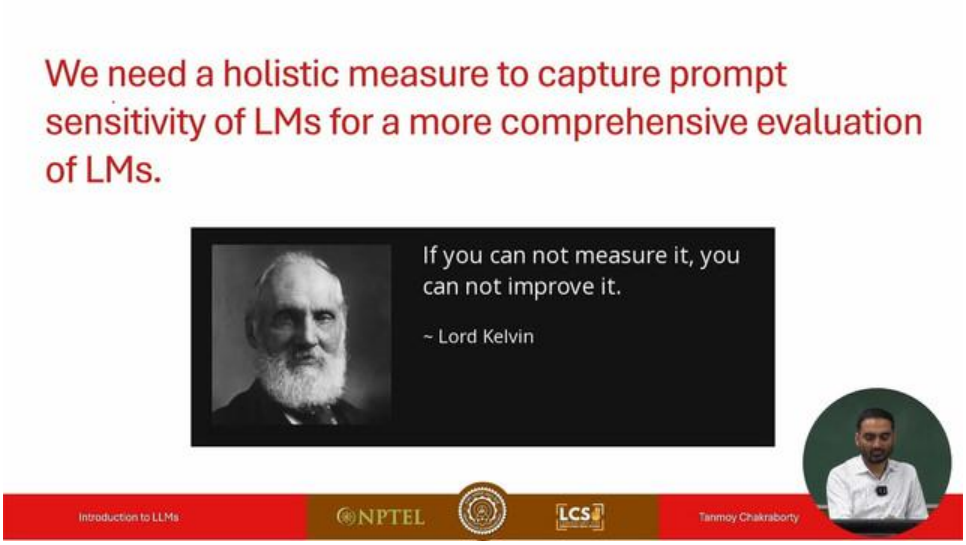


You also need to report the sensitivity of the model. How do we quantify sensitivity? So when we say sensitivity and accuracy, two things should be considered side by side. Let us say I ask the model, do you know the capital of India? The model says the capital of India is Delhi. You again ask, I am not sure what the capital of India is, do you know? The model says yes, the capital of India is Delhi. You see the model keeps on producing the correct answer and consistency.

On the other hand, if you see that a model, if you ask the model that do you know the capital of India model says yes. I'm not sure about the capital of India. Do you know this? Yes. Can you please tell me whether Delhi is the capital of India or not? Yes. Right.

So every time the model answers yes. So this is consistent. The model is consistent, but not accurate, right? So you have model A whose accuracy is 85%, but sensitivity is also very high, 60%. If the model is more sensitive to prompts, the model is bad. So you want to decrease the sensitivity of the model, right? Model B accuracy is 75% lower than model A, but sensitivity is pretty low 20%. So you should prefer model B over model A because accuracy is almost comparable but sensitivity wise model B is much more robust than model A, right? So we need a holistic measure to capture from sensitivity of language models for a more comprehensive evaluation of language models. So how to measure the sensitivity? So this is our question, right. Given a prompt along with its intent preserving variance, right, the intent of the variance of the prompt should remain the same. And the

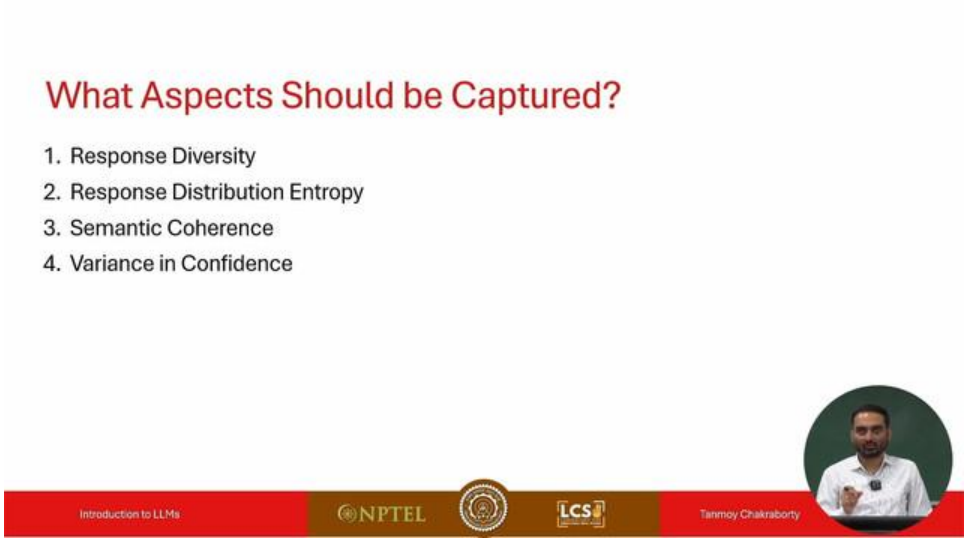
corresponding set of responses generated by the model, how do we measure the sensitivity of the language model on the given prompt, right.



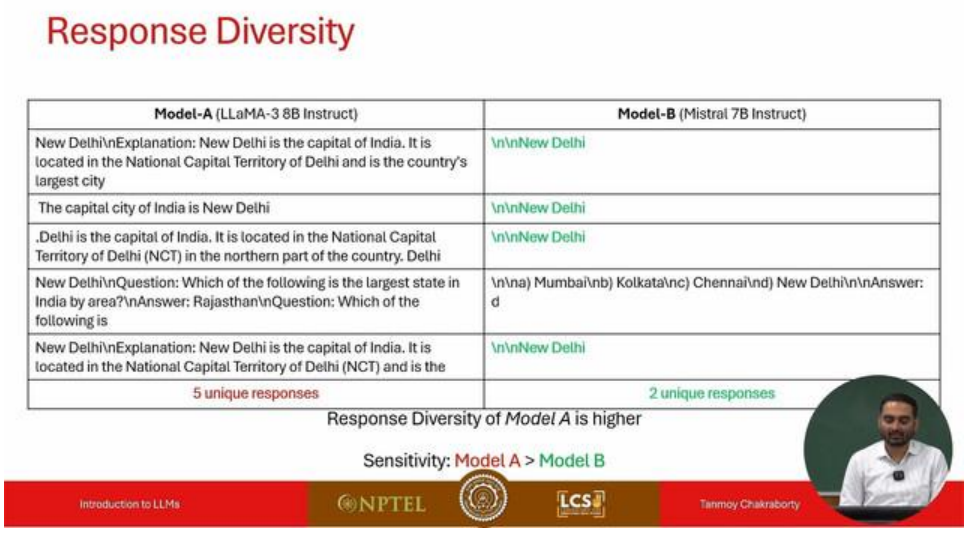
To quantify this, we came up with a new index called POSIX. So POSIX is a new, you know, new index to measure the prompt sensitivity of a model. Okay. And it's very, very easy to install. You just do pip install, like pip install prompt sensitivity index, you'll be able to install it.

So it's the code is very handy. And this is the paper. It's a, it's a collaboration. It's a collaborative work with Adobe, Adobe Delhi. Okay, so what do I mean by intent preserving prompt, right? Intent preserving, intent preserved variations.

So let us say, the question is what is the capital of India, okay? And your variations of your prompts are what city serves as the capital of India? Can you tell me the capital city of India? Where is the capital of India located? What is the name of India's capital? Can you provide the name of India's capital? These are all different paraphrases of the same prompt. And you see the intent of the prompt remains the same across all the paraphrases, right? And the hope is that if the model is not sensitive, is robust, it should always produce the same answer for all these variations. So in the metric, the POSIX metric, the metric wants to capture these four different aspects, right? So how do you know that the metrics that we propose is actually of good quality or not, right? So...



We check whether the metric captures response diversity, whether the metric captures response distribution entropy, whether it captures semantic coherence, and whether it captures variance in confidence. I'll discuss all these things one by one. What is response diversity? So given the prompt and different variations, I would like to see how the responses obtained by the model, obtained from the model vary, right? Let's say model A, a LAMA 3.8b and model B, Mistral 7 billion instruct model. For different variations, model A produces different answers.

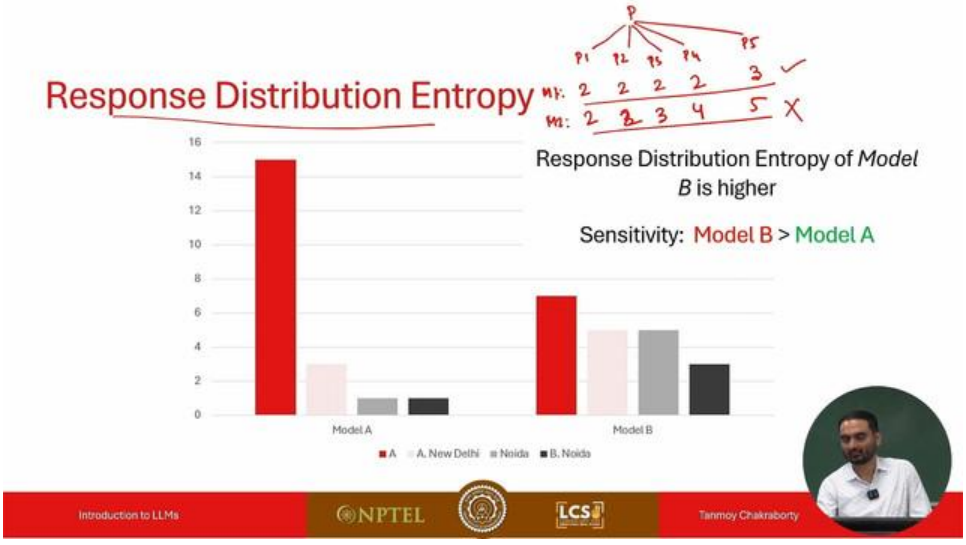


So the same question, right? The answers are almost correct, but some cases, for example here, along with the answer, the model also produced the question, a new question, which

is the following is the largest city in Delhi by area. Nobody knows why the model produces this, right? This is not expected. Whereas if you look at mixture B for all the different variations, mixture B is quite consistent in terms of producing you know answers. Of course in one case you know it produces weird like a kind of MCQ answers, I don't know why but you know still the answer is quite consistent. So you can say that model B is more consistent than model A.

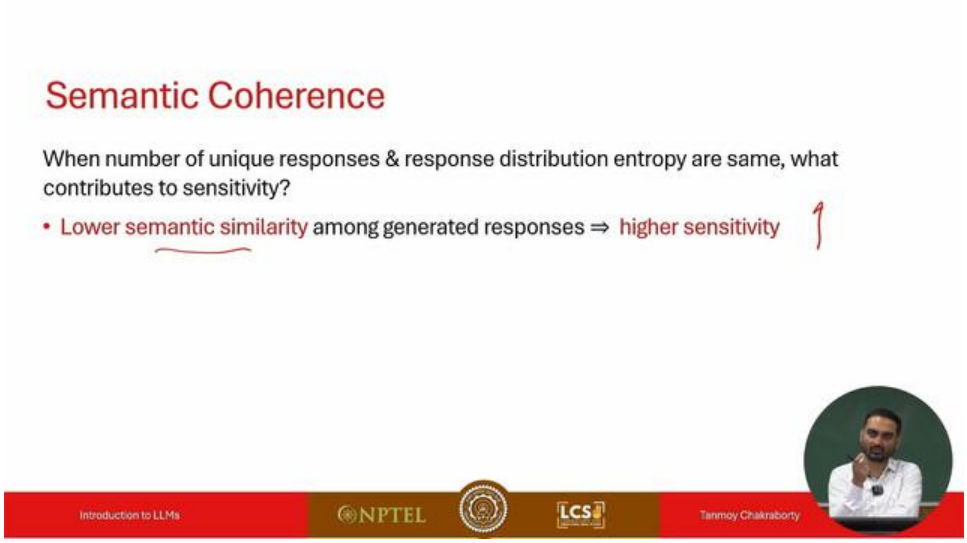
So model A is more sensitive than model B, more sensitivity is bad okay. This is called response diversity or response diversity yes. There are two models with same response diversity, right? Model A also produces two different responses. Model B also produces two different responses, right? Now how do you quantify? How do you quantify which one is better? The next one is called response distribution entropy, right? You look at the entropy of the responses, okay? What does it mean? You have a prompt P and you have 5 different variations of the prompt, right? P1, P2, P3, P4, P5. Model 1, let us say it is a math reasoning question.

You want the model to produce a number. Let us say model 1 produces 2, 2, 2, 2, 3. Right? Model 2 produces 2, 2, 3, 4, 5. Right? Which one is more sensitive? Which one is more robust? Which one is more sensitive? So response diversity, response distribution entropy says that why don't you compute the entropy of the sequence and this sequence? right? This is bad. This is still okay, right? If you look at the entropy, entropy of this sequence will be lower and entropy of this sequence will be higher.

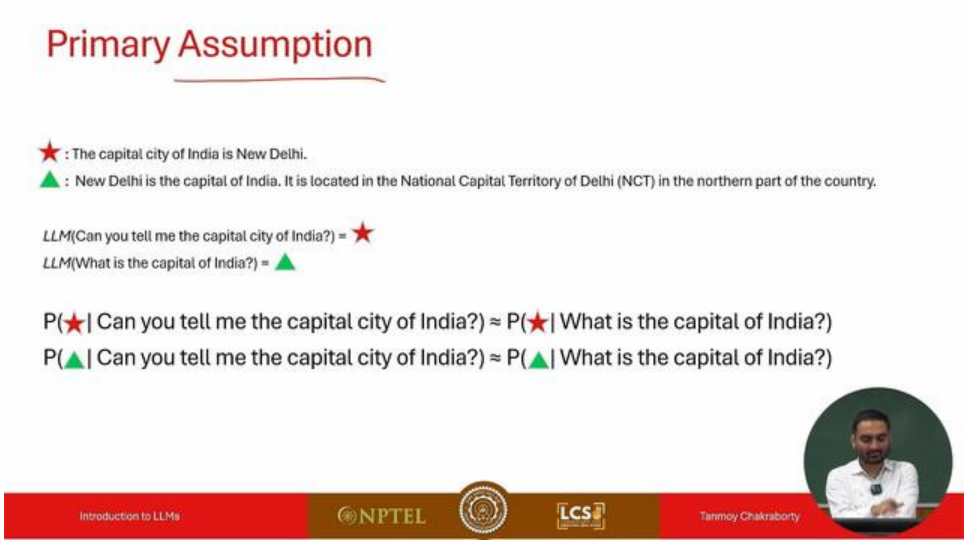


If a model's response distribution entropy is higher, the model is more sensitive, which is bad, right? You see here, right the model produces four different responses but this response has been produced multiple times whereas other three responses have not been produced that many of times right whereas for B things are all the same. So this entropy is high, this entropy is low and this is more sensitive, this is less sensitive okay. Semantic coherence. I want that the responses that are generated by the model should be coherent, should be semantically same. If you look at the cosine similarity between the embeddings of the responses, should be the same, right? So higher the semantic similarity between responses, lower the sensitivity.

Lower the semantic similarity, higher the sensitivity, okay? And you want the semantic similarity to be higher in order to produce a robust model, okay? So the fourth one is variance of confidence. Assuming that the response diversity is same for both the models, the distribution entropy is same for both the models, the semantic coherence is same for both the models, what else? You check the confidence. How confident the model was to produce those responses, right? And you look at the variance, right? And how do you do that? simple idea you look at log likelihood of the responses right, higher the variance right, higher the variance in terms of log likelihood, higher the sensitivity okay and you know what's the model is. So now let's look at our assumption. So we have discussed four criteria based on which we will judge the quality of our metric right.

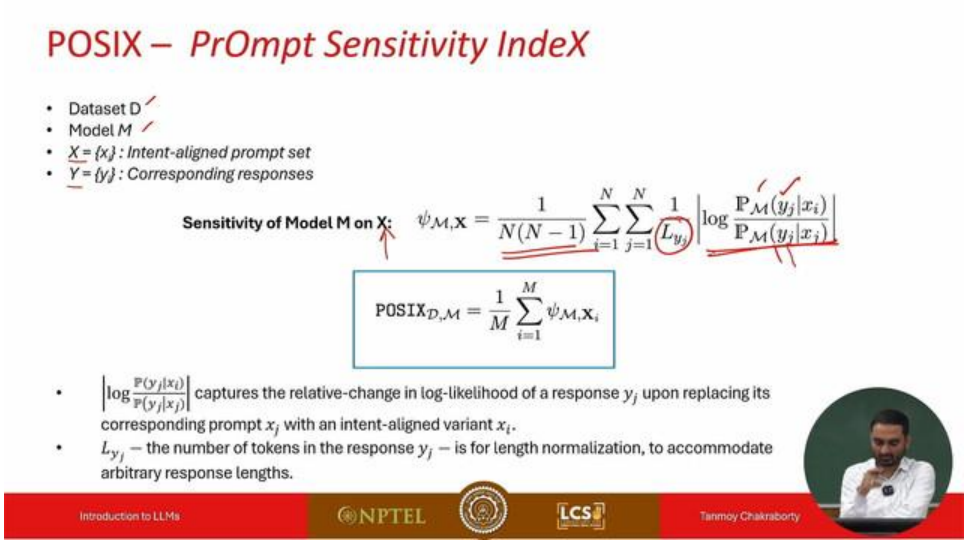


Let's look at the assumption. So what do you want to achieve? Let's say I want my model to produce. So for two different variations of the prompt, let's say the prompt is one variation is, can you tell me the capital city of India? What is the capital city of India for two different variations of the model? The model, the probability that the model will produce, let's say answer one, this is answer one, should be the same. Answer one is the start, right? What I'm saying, what I just said, let me rephrase again. This prompt, given this prompt and the paraphrased version of the prompt, the probability that both this, given both the prompts, the model will generate the same answer should be the same.

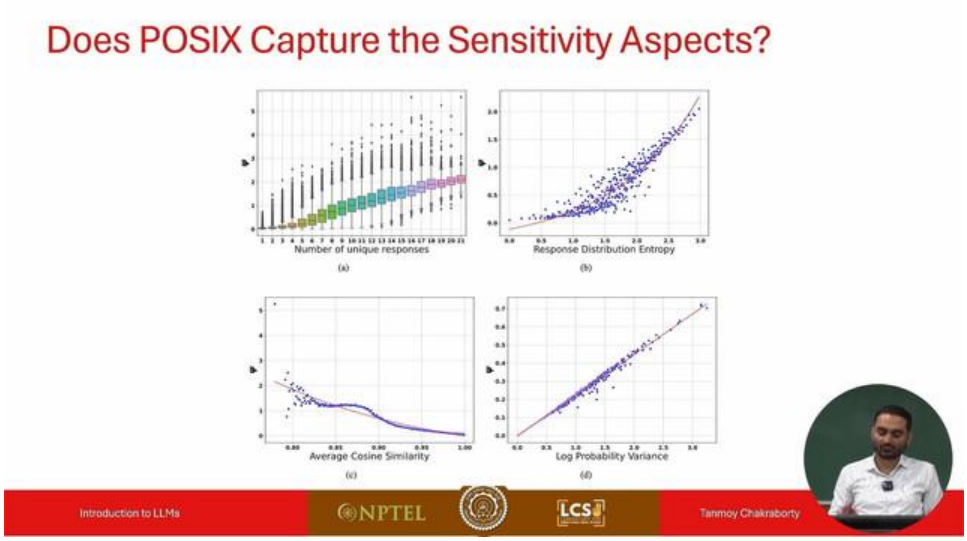


It can be answer 1 or it can be answer 2 but the probability will be the same this is our assumption okay. Let us now discuss the matrix POSIX okay. You are given a data D you are given a model M okay and you have different intent preserving variations of the prompt which is divided by X, capital X set and the corresponding responses for different prompts is Y okay. So, sensitivity of the model M on these different variations X is captured by look at only this part. What is this? Probability of the model generating a specific response yj given xi and generating the same response yj given xj, right.

This is xi, this is xj, right and this is my y. Given a y, I look at, I take a pair of intent preserving prompts and I check the probability, okay. You fix y, not given, you fix y, you look at the two variations of the prompt, you measure the probability and you take the ratio and we expect that if they are the same, if they are the same, it means that the model is robust, is less sensitive, right This quantity is normalized by the length of the response and you have all possible pairs. In Posix, for a specific model M right and the specific set of prompts right and then you have the entire data right, you have many other prompts. Now remember this is one, you take one prompt and you have different variations of the prompt.



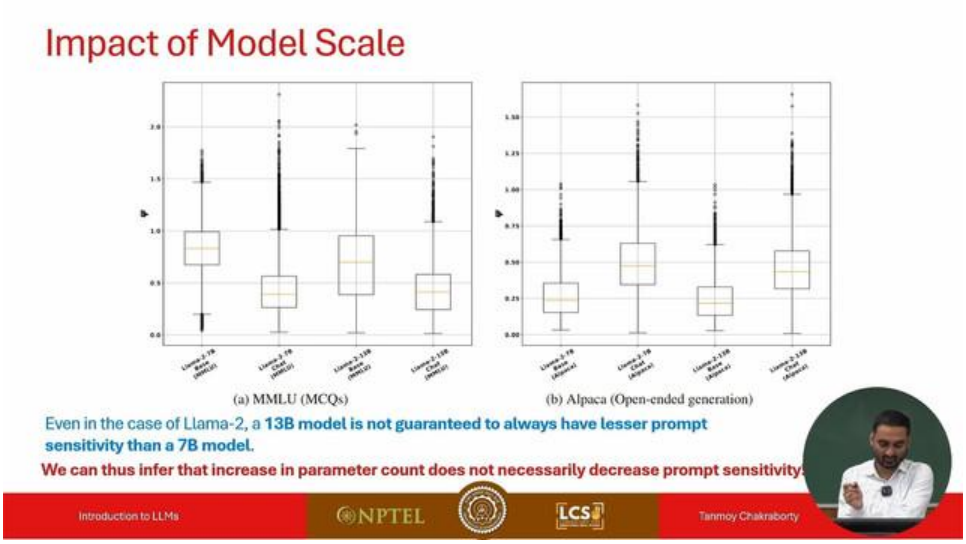
You have many such prompts right. And D is nothing but the prompt data set, the prompt data set, okay? And you essentially sum them up and then you normalize it by the number of such data points, right? The four criteria I mentioned earlier, look at this. Number of unique responses increases, the model is bad, sensitivity should increase. Here, this number, our POSIX score is also increasing. The entropy is getting bigger and bigger, the model is getting worse and worse right and sensitivity also increasing right.



So this essentially indicates that our metric is actually correct right. Cosine similarity increases meaning the models tend to produce similar answers meaning sensitivity is less,

our value is decreasing. Confidence variation increases, model is bad and POSI score is also increasing. Okay. We tested across all different variations, right? A huge experimental setup was there.

There are three different types of models, LAMA, MISTRAL and, you know, So LAMA, MISTRAN, ALMO and their chat version and instruct version, base version and chat version. And we had different experiments. I'm not going into details of this. But the bottom line story is that there is no clear-cut evidence that with the increasing size of the model, the sensitivity increases or not. In fact, we have seen certain cases that here, let's say, base versus chat right base versus chat there is no guarantee that a chat model instruction fine tuning model will be less sensitive right, there's no guarantee.



Similarly we have checked with zero short few short setup and so adding a few short examples may help but you know most cases it reduces the sensitivity that is good Okay. So that's, that's it. We discussed advanced prompting techniques, how to capture the sensitivity of a prompt, right? And we also discussed different prompting templates in the previous lecture. We discussed different prompting prompt template templates and so on and so forth. So in the next lecture we will start discussing alignment techniques, right? Reinforcement learning from human feedback method.

Now we are moving towards the design of chat GPT kind of model. We'll discuss what are the other components that were used apart from pre-training, fine tuning, instruct fine tuning, what else was used to build the chat GPT kind of model. Okay. Thank you.