Introduction to Large Language Models (LLMs)

Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti

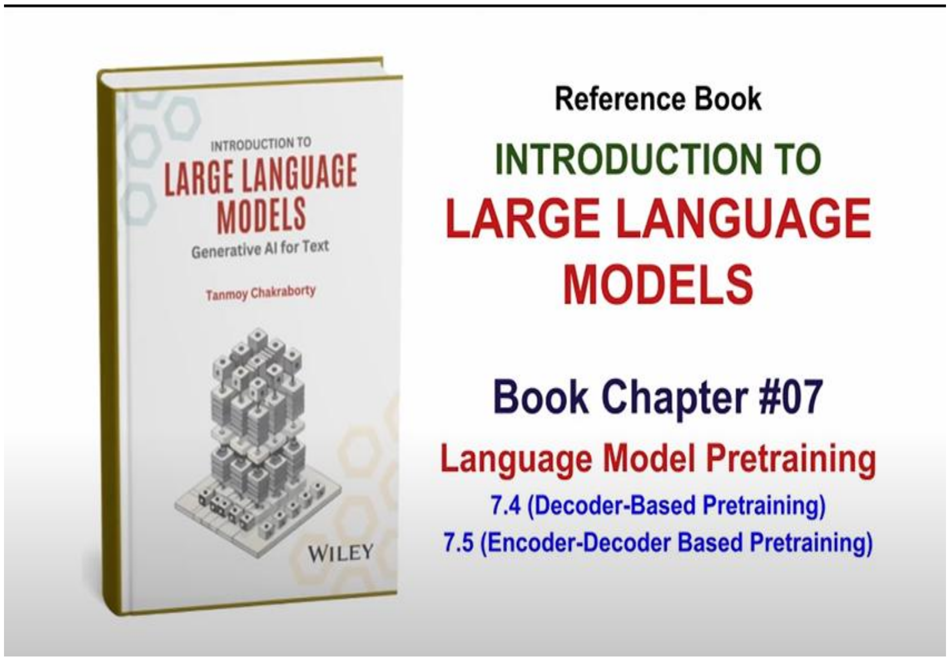
Department of Computer Science & Engineering

Indian Institute of Technology, Delhi

Lecture 19

Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Hello everyone. Welcome back.



Reference Book

INTRODUCTION TO

LARGE LANGUAGE MODELS

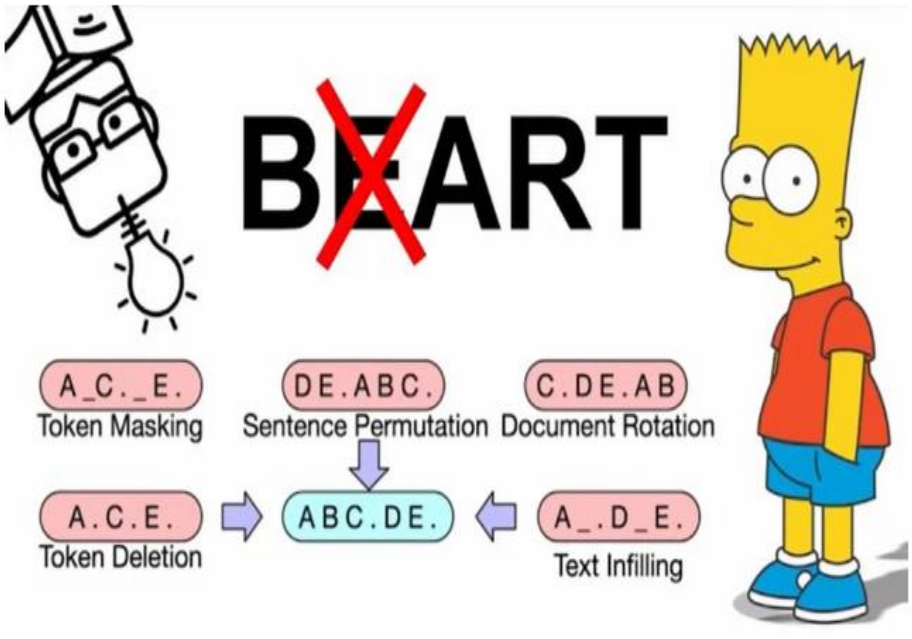
Book Chapter #07

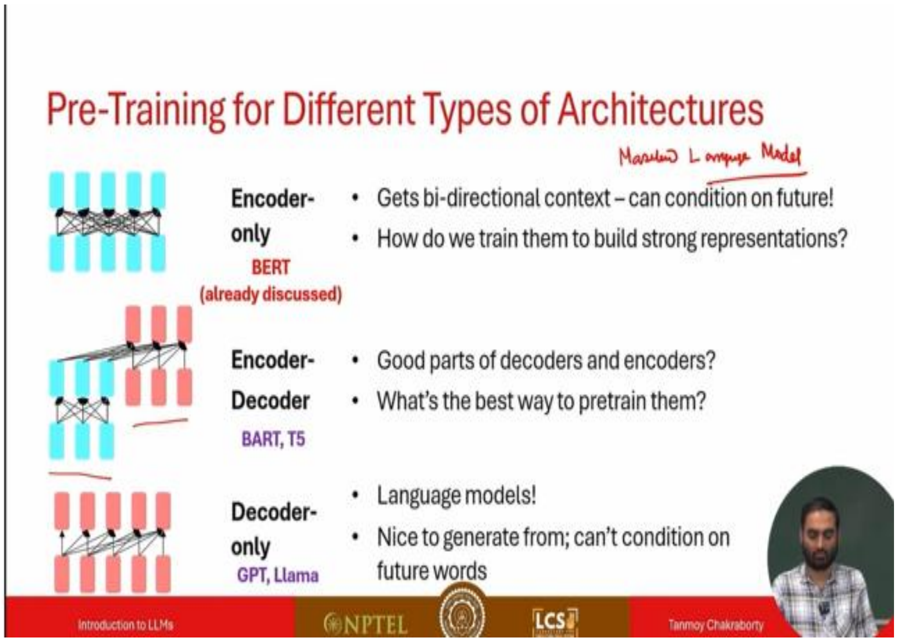
Language Model Pretraining

7.4 (Decoder-Based Pretraining)

7.5 (Encoder-Decoder Based Pretraining)

INTRODUCTION TO
LARGE LANGUAGE MODELS
Generative AI for Text
Tanmoy Chakraborty
WILEY





Pre-Training for Different Types of Architectures

Masked Language Model

- **Encoder-only** BERT (already discussed)
  - Gets bi-directional context – can condition on future!
  - How do we train them to build strong representations?
- **Encoder-Decoder** BART, T5
  - Good parts of decoders and encoders?
  - What's the best way to pretrain them?
- **Decoder-only** GPT, Llama
  - Language models!
  - Nice to generate from; can't condition on future words

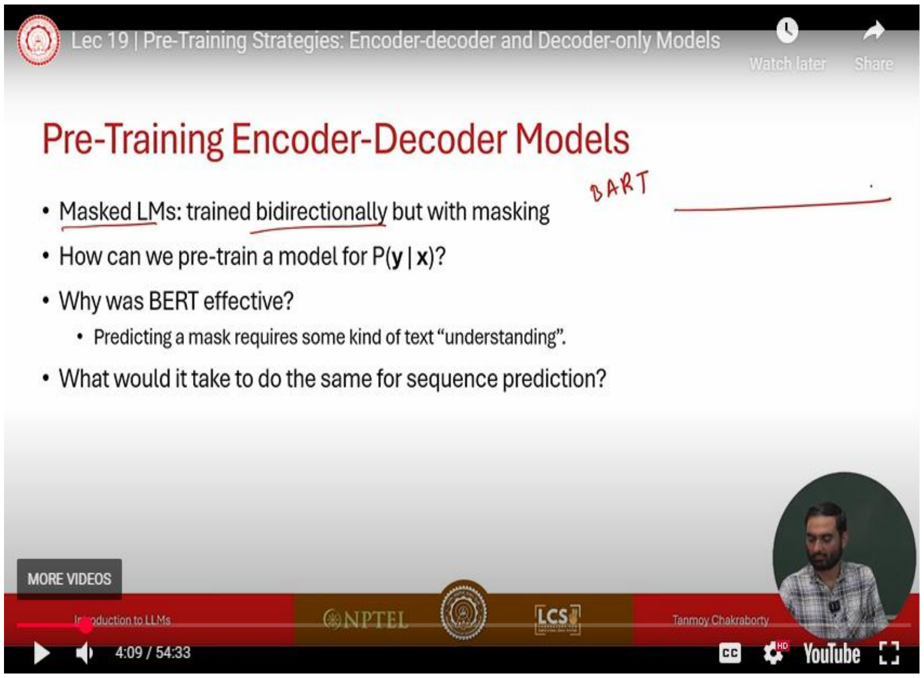
Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

So we were discussing different pre-training strategies and last lecture we discussed, you know, pre-training and encoder only model, specifically we discussed the bert model, right? So in today's class, we will focus on two other pre-training strategies. One is encoder-decoder only models, right? Models like T5, BART. So B-A-R-T is another model, right? This is not B-E-R-T, this is B-A-R-T, BART. And at the same time, we will also discuss decoder only models, right, which include chat GPT, all GPT series models, LAMA models and decoder only models are essentially very popular.

So encoder only model, as I discussed, this was discussed in the last class. We saw that a new pre-training strategy called mask language model. This was used in BERT where you essentially mask some of the tokens in the input and you ask the model to predict those mask tokens, right? This is a self-supervised approach. You don't need any label data for training, right? And being an encoder model, it essentially, I mean, the BERT model can essentially look at all the tokens present in the input.

It is not an autoregressive model, right? So the entire input is exposed to the model. And you basically do a self-attention. Through self-attention, you look at all the tokens present in the input. And based on that, you predict the mass tokens. On the other hand, encoder-decoder model that we discussed here, that we discussed in today's class, we have this encoder component and the decoder component.

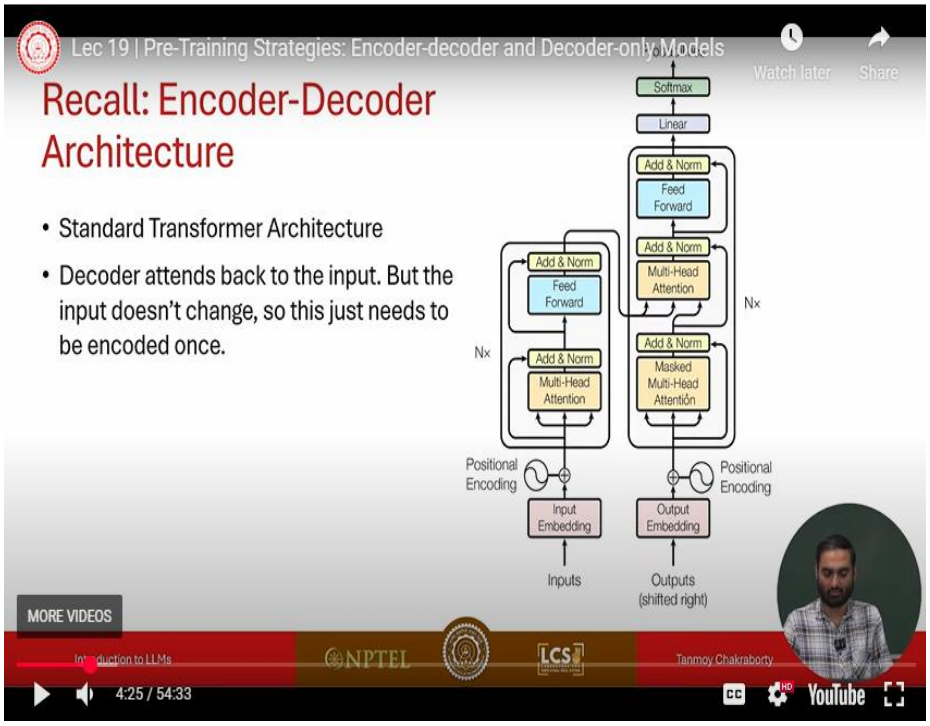
And we will see how you can best use both the encoder part and the decoder part during pre-training. And then we will also discuss decoder-only model. Now in the decoder-only model, as the name suggests, the decoder part, if you remember in the transformer class we discussed, It's an autoregressive model, autoregressive component in the sense like when you predict a word at the tth location, you only have access to tokens till t minus 1th location, right? You don't have access to tokens after tth location, right? We will see how you know this autoregressive style of pre-training essentially helps the decoder only model learn things right and this set of models right are very popular these days people realize over the times that encoder only model is not a right solution for for a generative task, right? Because a generative task requires autoregressive setup, right? Whereas in an encoder-decoder model, right, as you require both the encoder part and the decoder part, it requires a lot of memory, it requires a lot of parameters, right? You can do all these generative tasks through encoder-decoder model together, right? But decoder model makes more sense both in terms of parameters as well as the setup, right? Autoregressive model setup itself is suitable for, you know, next token generation. And parameter of course parameter wise you need half of the parameters than the encoder decoder model. So we will focus on this and we will discuss this.


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pre-Training Encoder-Decoder Models

- Masked LMs: trained bidirectionally but with masking
- How can we pre-train a model for P(**y** | **x**)?
- Why was BERT effective?
  - Predicting a mask requires some kind of text "understanding".
- What would it take to do the same for sequence prediction?

Mask language model as I mentioned BART model. a bidirectional model, meaning that you have access to tokens in the right direction as well as in the left direction, right? But what is required for an encoder-decoder model to essentially predict a sequence, right?

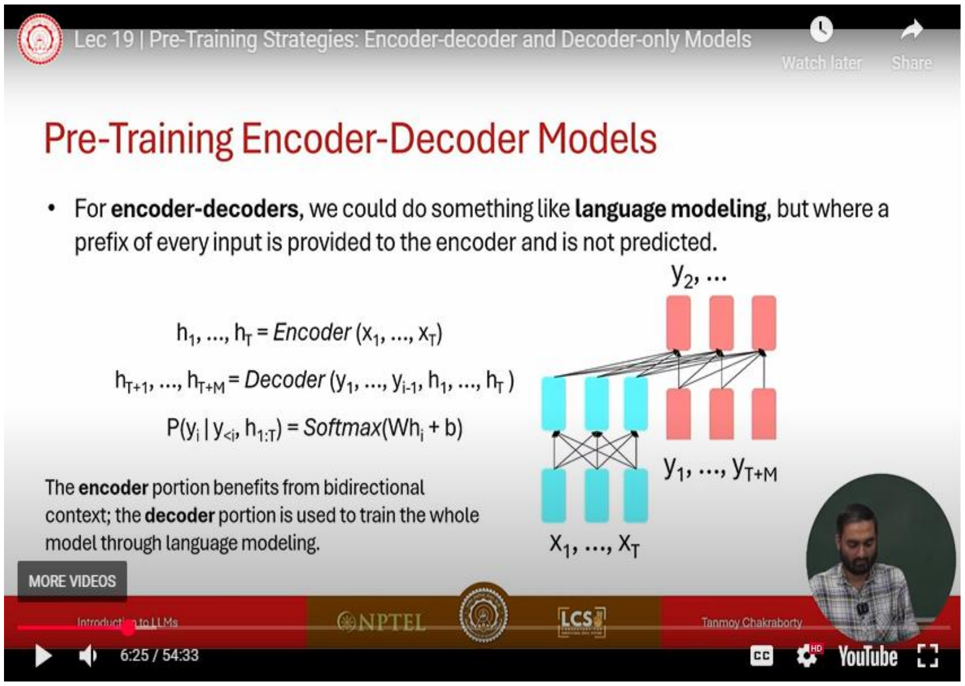

Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Recall: Encoder-Decoder Architecture

- Standard Transformer Architecture
- Decoder attends back to the input. But the input doesn't change, so this just needs to be encoded once.

Encoder-decoder architecture you should remember this is a Vanilla transformer model right you have again a quick recap input positional encoding right here you have position encoding then you add positional encoding and the input right input embedding and then in the encoder part you have multiple blocks in number of blocks let's say six blocks or twelve blocks every block has a multi head attention and a feed forward layer right. This is the FFN feed forward layer right and this is unmasked right multi head attention and in between feed forward and multi-head attention, you have an add norm layer, and that is your connection. In the decoder part, you again have multiple decoder blocks. Each block consists of a mask multi-head attention, mask in the sense like, as I mentioned earlier, when you process the Tth token, the tokens which after Tth token are all masked, right? And then usually, you know, the usual feed forward network in between feed forward and mask multi-head attention, you have a cross attention layer.

So this is cross multi-head attention and through this cross attention, you essentially, the decoder component, the decoder block, you know, attends to the outputs of the encoder block, right? So this is the usual setup, right? And then you repeat this block in number of times. So, Now we see that in this kind of encoder-decoder setup, how do we pre-train this model?



Pre-Training Encoder-Decoder Models

- For **encoder-decoders**, we could do something like **language modeling**, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \dots, h_T = Encoder(x_1, \dots, x_T)$$

$$h_{T+1}, \dots, h_{T+M} = Decoder(y_1, \dots, y_{i-1}, h_1, \dots, h_T)$$

$$P(y_i \mid y_{<i}, h_{1:T}) = Softmax(Wh_i + b)$$

The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.

So the objective is still language modeling. Next word prediction can be an objective. The objective can also be, let's say, sentence classification, for example. So let's see.

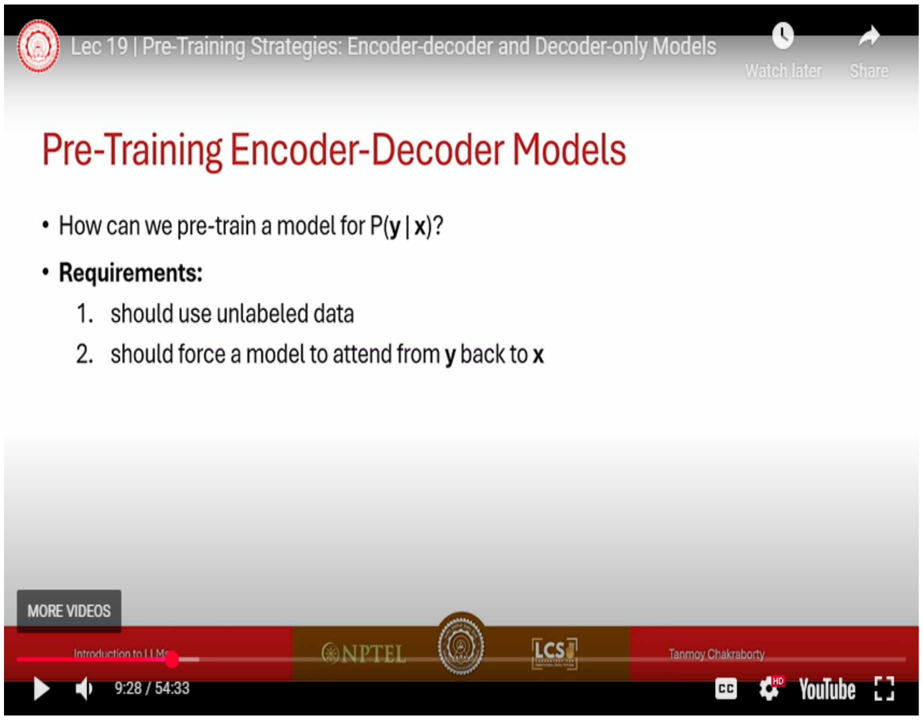
So this is my encoder block. Okay. These are the inputs. There are T number of inputs and these inputs are called prefix, right? Now this prefix can be an instruction. Let's say the instruction can be, summarize this sentence, right? Or the instruction can be, find the sentiment of the sentence, right? So it can be an instruction followed by the input.

Let's say, in case of summarization, you know, the instruction is summarize the following document, colon, and this is the document that you want to summarize, right? The entire thing will be fed to the encoder model, right? And after the encoder block, you will get hidden representation h1, h2, dot, dot, dot, ht. In the decoder block, when you pre-train your model, let's say for a task-specific pre-training, you also have the summary. So you give the summary here. So y1, y2, y3, dot, dot, dot, yt plus m, this indicates the summary. And your objective is next word prediction, right? So at this stage, you predict what is going to come at the second stage.

At this stage, you predict what's going to come in the third stage and so on and so forth. So in your decoder block, the input is y1, y2, y3, dot, dot, dot. This is my input. And my input is the hidden state that you obtained from the encoder block. And when you get at the end of the decoder block, you get the hidden state representation here.

You pass it through a soft max. So this is my hidden state representation of the decoder block. You pass it through a soft max. And the soft max is going to give you the probability distribution about the token and that's going to be your output. So this is pretty much same as the Vanilla transformer model.

The Vanilla transformer model was proposed for machine translation where the input was let's say an English sentence and the output was a Hindi sentence or output was a Spanish sentence, right? And this was the usual way of, you know, pretending the transformer model.

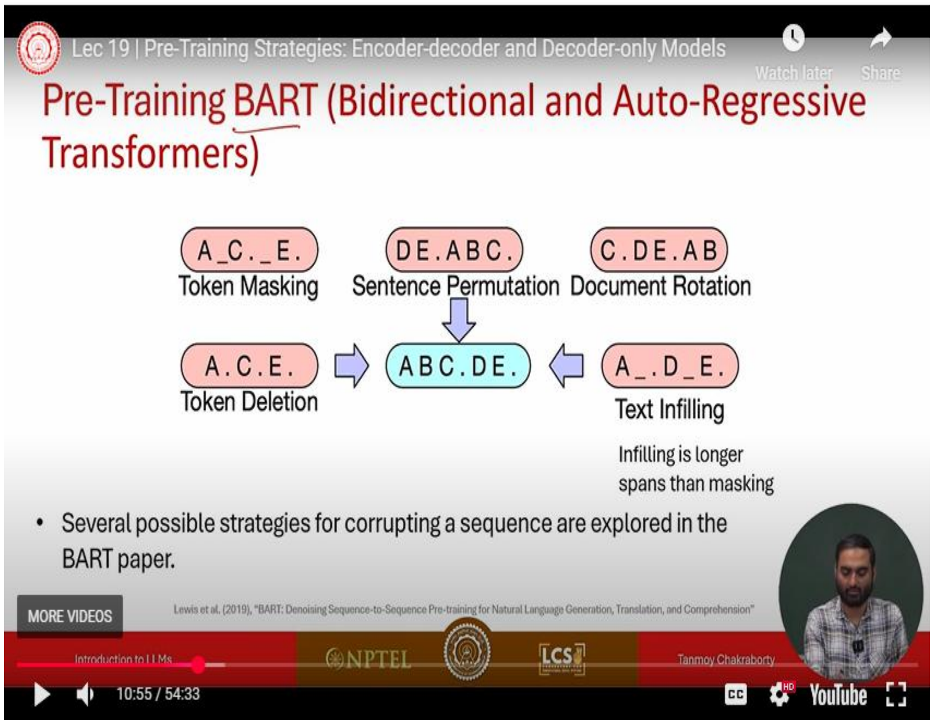


Pre-Training Encoder-Decoder Models

- How can we pre-train a model for P(**y** | **x**)?
- **Requirements:**
  1. should use unlabeled data
  2. should force a model to attend from **y** back to **x**

So now, we will see what additional component we can change in the pre-training architecture, and we will specifically discuss two models. One is the T5 model, other is the BART model, B-A-R-T, BART model, right? Now, whatever we discussed so far here in this slide, I said that there's a task. The task is, let's say, summarization, right? Or let's say the task is machine translation. Right? So for these tasks, you need a lot of human data, human level data, isn't it? But now the question is, we have seen in case of BERT model, we did not require a label data.

It was a self supervised approach, you basically had a running test, and you mask some of the tokens in the running text, and you wanted the model to predict those running those masked tokens, right? So the question is, in the encoder-decoder model, how can we use unlabeled data? Because label data is very difficult to collect, right? And then how do we force the model to attend to all the inputs, right? Because your output is why your input is X, right? Your input is your input is fed. Your input is going to be fed to the encoder model and your decoder model should be able to attend to all your encoder inputs, right? So let's see.



Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pre-Training BART (Bidirectional and Auto-Regressive Transformers)

- Several possible strategies for corrupting a sequence are explored in the BART paper.

Lewis et al. (2019), "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"

So in case of BART, B-A-R-T, the full form is bidirectional and auto regressive transformer. As the name suggests, it essentially leverages the bidirectional part, which was there in BERT, And the auto regressive part, which is going to be there in GPT, we'll discuss today. Right? So it leverages both the schools.

Okay. So remember, we should be able to use unlabeled data. Okay. So what is the idea here? Given an input sentence, you corrupt that input. you inject noise to that input and you ask the model to essentially predict the right sentence, the correct sentence, right. So the question is how do you corrupt it? There are multiple corruption approaches that they suggested.

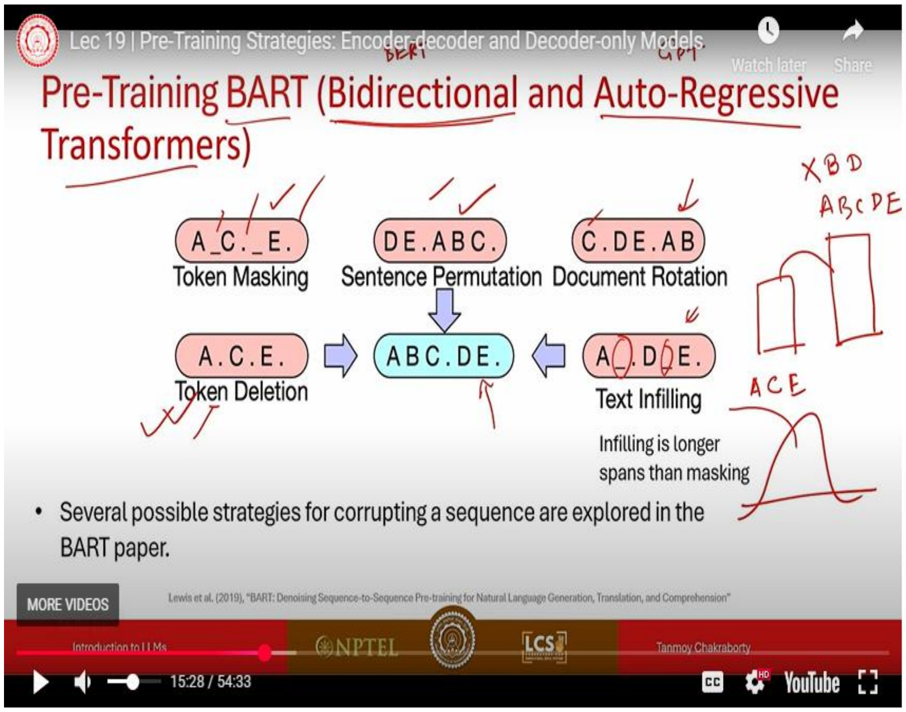
These are called noisy instances. For example, let's say your input is A, B, C, D, E, right? What you do, you mask some of the tokens similar to bert, right? And you ask the model to predict those mask centers, right? So in your encoder, your input is the masked token. tokens, the mask set of tokens, and your decoder, the output is going to be the actual set of tokens, right? So this is one strategy. The second strategy can be you, you can, basically you can permute, right? So your sequence is ABCDE, you shuffle the tokens, it can be DE ABC, for example. right? And you ask the model to predict the actual sequence.

The third one can be rotation, right? So A, B, C, D, E. So C, D, E, A, B. This is one rotation you can again rotate, right? You keep on rotating to obtain different inputs and your output will be the original one, right? And these are very similar because these strategies considered the entire sequence and then sometimes mask it, sometimes rotate it, sometimes permute it, right? But the length of the input remains the same. Okay. They also tried out with token deletion where you delete some of the tokens.

You see here, you delete B, D, F, right? And you ask the model to predict those tokens. Okay. But remember here, since this is an autoregressive model, you ask the model to basically predict all the tokens, not the masked tokens or not the tokens that you have deleted. So in your encoder, your input is going to be ACE for this strategy. and your decoder, the output is going to be A, B, C, D, E.

It is not that your output will be B and D only. Now this is called token deletion. They also tried an extreme approach where they essentially, you know, mask, not a single token, but a sequence of tokens, a consecutive sequence of tokens. For example, as you see here, ABCDEF, ABCDE was the original sequence. You mask BC together, right? So sometimes you mask two tokens.

Sometimes you mask one token and so on and so forth. And what they said is that, there is a distribution. From the distribution, they basically sampled the size of this masking, right? Sometimes if the size is one, you just sample one token, you just mask one token. If the size is three, you mask three consecutive tokens and so on and so forth. So they randomly chose some of the positions, right? And they sample a number from the distribution and that sample is going to tell you that what's going to be the size of the token that you are masking, right? So, nothing fancy.

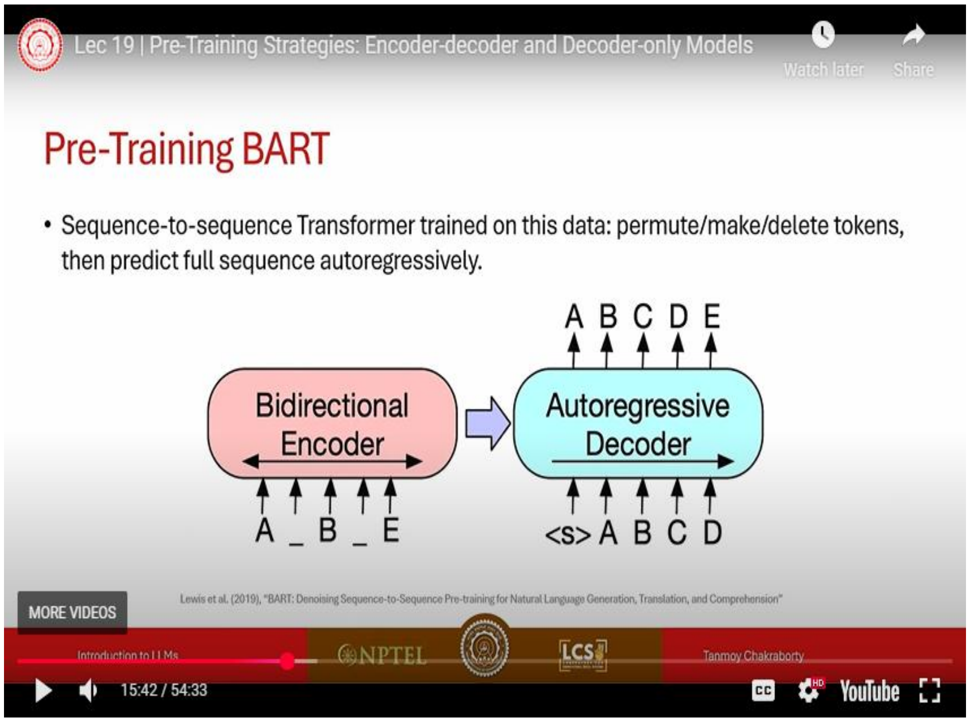

Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pre-Training BART (Bidirectional and Auto-Regressive Transformers)

- Several possible strategies for corrupting a sequence are explored in the BART paper.

Lewis et al. (2019), "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"

This is just a way of doing pre-training as opposed to the, you know, a sort of supervised pre-training which was used in transformer model where you need, you basically needed the label data. So here you don't need any label data as such.

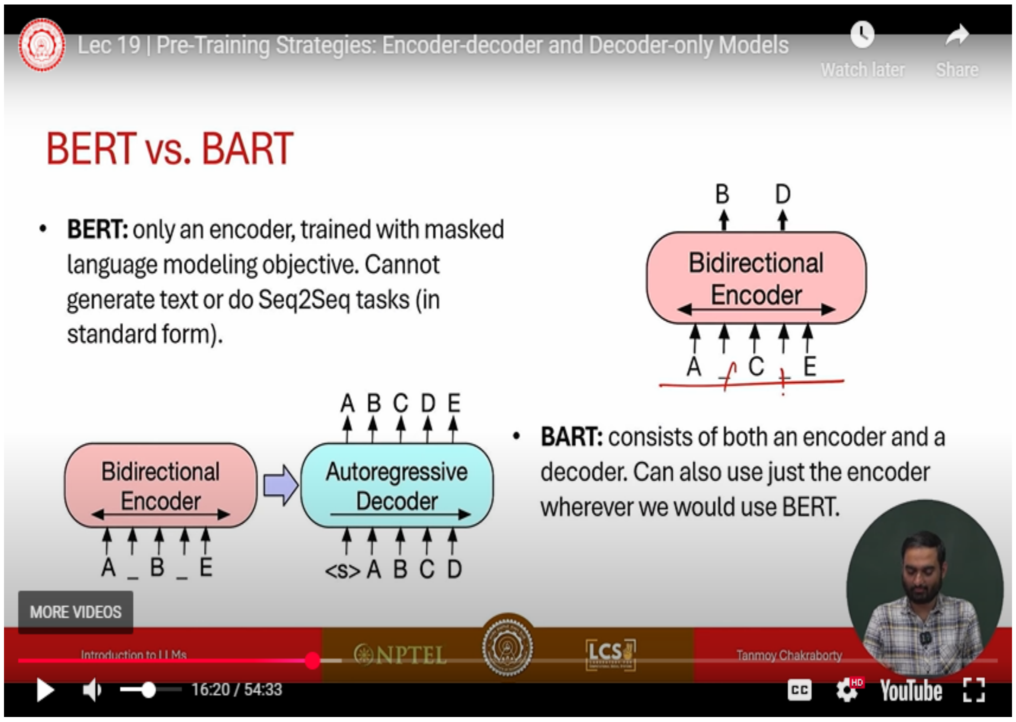

Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pre-Training BART

- Sequence-to-sequence Transformer trained on this data: permute/make/delete tokens, then predict full sequence autoregressively.

Lewis et al. (2019), "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"

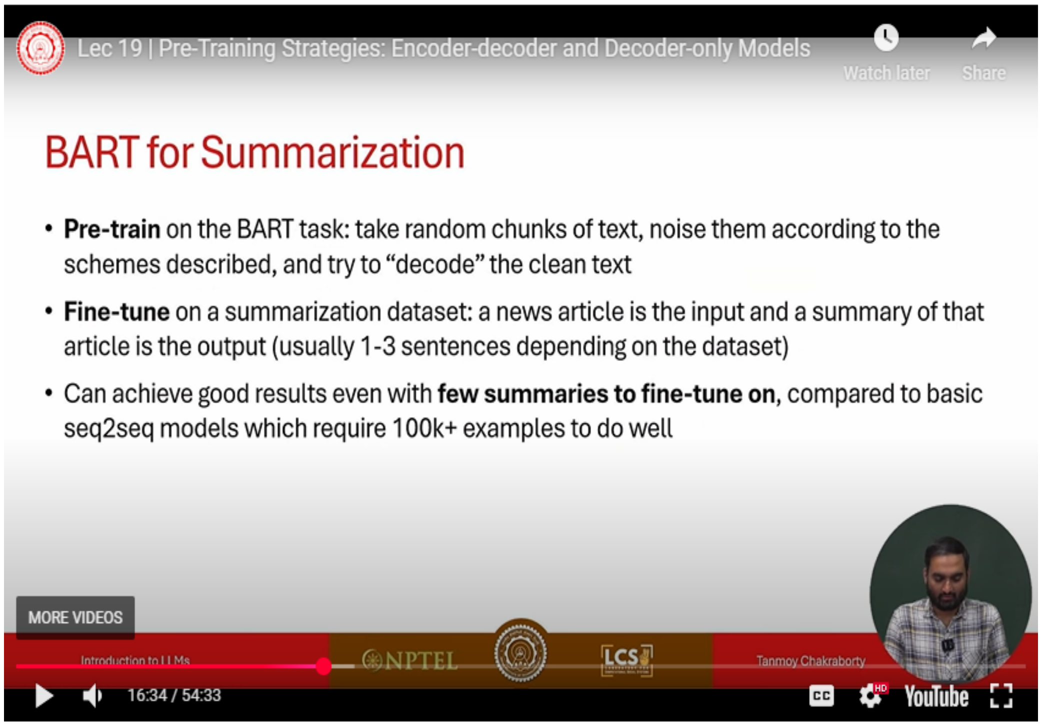
Okay. Okay. So this is my encoder. Okay. This is my input. Let's say that here, I mean, you can use permute, mask, right? This should be mask. right or delete right and you ask the model to predict the entire sequence okay.


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

BERT vs. BART

- **BERT:** only an encoder, trained with masked language modeling objective. Cannot generate text or do Seq2Seq tasks (in standard form).
- **BART:** consists of both an encoder and a decoder. Can also use just the encoder wherever we would use BERT.

So this is BART. Let's look at the difference between BERT and BART. So in case of you know the BERT model which is an you know encoded only bi-directional model you ask your model to predict the mask tokens, right? In case of bart, you ask your model to predict all the tokens, right? It uses encoder, it uses encoder decoder, so double parameters, right? And so on and so forth. So that's it.



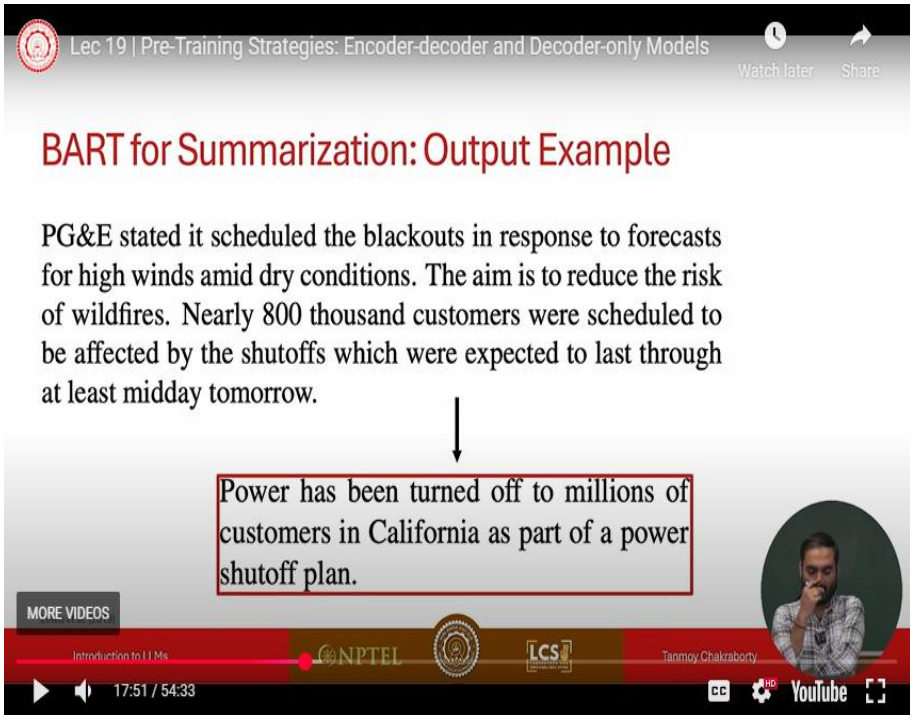
BART for Summarization

- **Pre-train** on the BART task: take random chunks of text, noise them according to the schemes described, and try to "decode" the clean text
- **Fine-tune** on a summarization dataset: a news article is the input and a summary of that article is the output (usually 1-3 sentences depending on the dataset)
- Can achieve good results even with **few summaries to fine-tune on**, compared to basic seq2seq models which require 100k+ examples to do well

Okay. They pre-train the model, pre-train the encoder decoder model in this way, injecting noise and asking the decoder model to basically predict the clean version.

They fine tune the model on downstream tasks, right? So once you pre-train, you can now think of a pre-tuned model which knows the language in general. Then you fine-tune, they fine-tuned it. By the way, you can also pre-tune, you can also use pre-tuned model for your downstream task without fine-tuning it, right? And we'll see the GPT paper, GPT-3 paper, which came, In 2020, they showed that you don't need to basically fine tune your model for downstream tasks. Nevertheless, on the summarization data set, they fine tuned the model on a news article, news corpus, and it turned out to be really effective. It also showed that you don't need a lot of data set, you need only a few summaries for fine-tuning as compared to normal sequence-to-sequence models, let's say, you know, LSTM or a normal sequence, I mean, LSTM GRO kind of models, which require a lot of data for fine tuning here.

Since your model is already pre-trained, you don't need to fine tune, you know, that much.

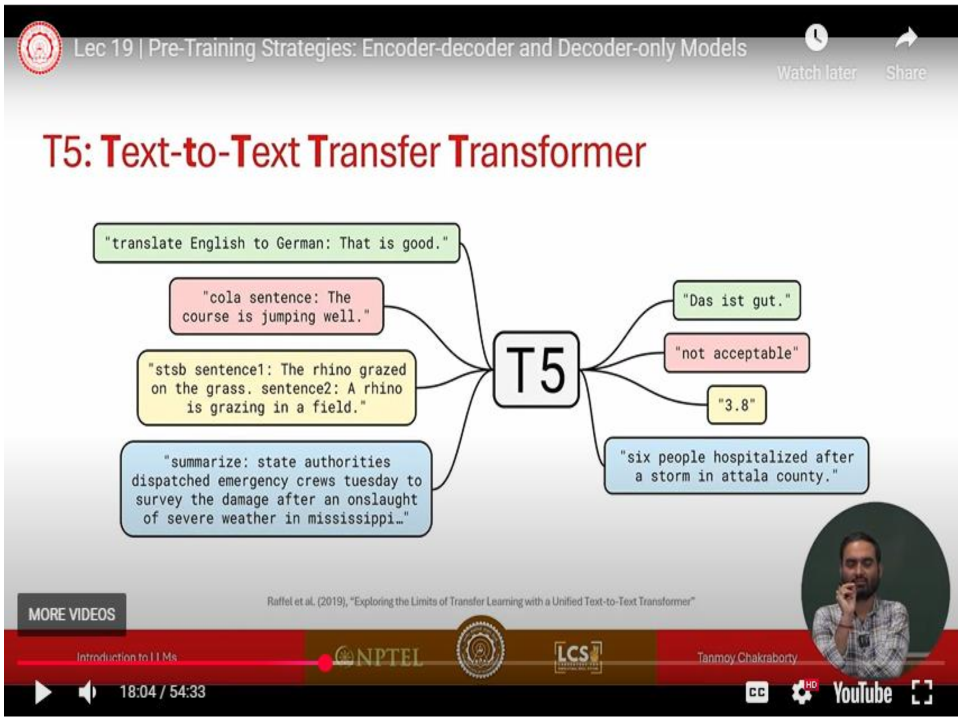


BART for Summarization: Output Example

PG&E stated it scheduled the blackouts in response to forecasts for high winds amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow.

Power has been turned off to millions of customers in California as part of a power shutoff plan.

So here is an output. of the BART model. This is the document that was given for summarization and this is the summary that was obtained from the model.



T5: Text-to-Text Transfer Transformer

"translate English to German: That is good."

"cola sentence: The course is jumping well."

"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."

"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"

T5

"Das ist gut."

"not acceptable"

"3.8"

"six people hospitalized after a storm in attala county."

Raffel et al. (2019), "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"

Okay. Around the same time when the BART model was proposed, right, Google, they also came up with a method called T5. Same time, within, I think within a gap of one week or two weeks, right? And T5 is also an encoder-decoder model. It is called text-to-text transfer transformer, right? And this is actually a, you know, this is a very, very, I would say, ambitious model, right? What they said is that we map all the NLP tasks, right? Any tasks that you can think of, not necessarily NLP, right? Any task that you can think of, you map or you can realize any task as a text to text transfer problem. It is very difficult to digest, right? But bear with me.

we will essentially discuss the techniques. There is no fancy technique behind this. The same encoder-decoder model, the same corruption technique that was proposed in BERT. But remember BERT was proposed by Facebook and T5 was proposed by Google around the same time. It's kind of similar approach. But they said that, first of all, you don't need any fine tuning as such.

You pre-train on millions of tasks. And these tasks are all supervised tasks. So T5 data set is not a T5 training approach is not an unsupervised approach as such. But the crux of the method is that every problem can be mapped to a text-to-text problem, right? For example, let's say you want to translate an English sentence to a German. So what you do in your encoder, you write this prompt that translate English to German, right? It's your prefix, right? And then this is the sentence and your output will be this one, the German version of it, right? So your input will be this sentence right in the encoder and your output will be this sentence in the decoder. But you see here there is a task instruction, task prefix which was not there in case of bart right.

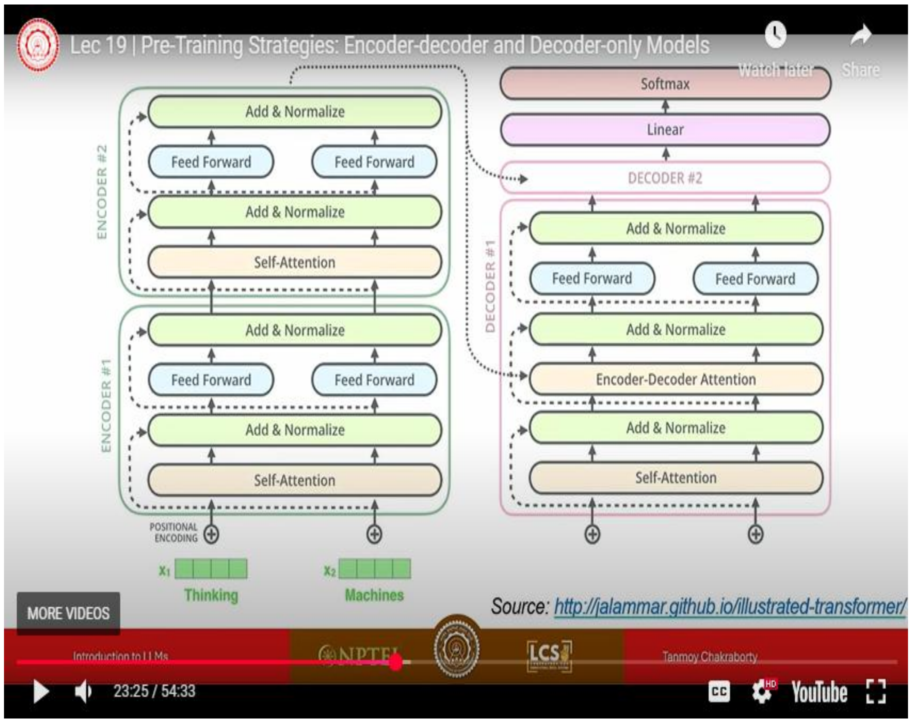
Barts pretraining method was very different. Now this is the second task. COLA sentence. What is COLA? COLA is a task, corpus of linguistics acceptability. So it basically checks whether a sentence, an input sentence is linguistically acceptable or not.

For example, the course is jumping well. Is it an acceptable sentence? No. So your input is the task instruction and the sentence and your output is acceptable or not acceptable, right? Your input is, so the next task is checking whether two sentences are similar or not, right? STSB task. Sentence one is the rhino grazed on the grass. Sentence two, a rhino is grazing in the field. Now whether they are similar or not and you give a score right. It is not yes or no, it is a score. Now think about it. Your output here 3.8 is essentially a string.

So you are forcing your model to produce 3.8. right? You are not creating a number. You are basically printing a string and calling ruffle, the first order of this paper. So he said in one of his talks that, you know, believe, believe him, believe, believe on this, is this framework.

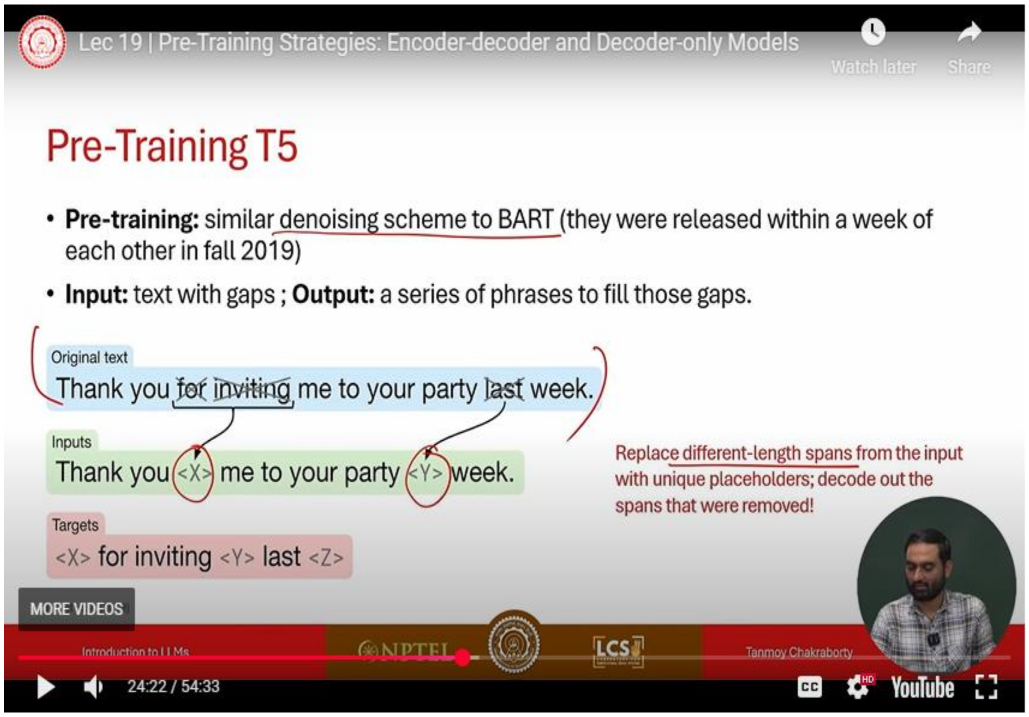
It actually works. Okay. So the fourth task can be summarization. You write summarize and the entire sentence, right? The state authorities dispatched emergency crews Tuesday to survey the damage after an onslaught of several weather in Mississippi and dot, dot, dot. There are other tokens as well. And your model should be able to summarize this. So you can still digest these three tasks, but it is difficult to digest how text to text model can produce a number.

Nevertheless, they said that if you give instructions, if you write instructions, And then the task itself and you force the model to produce the output in your pre-training time, your model should be able to do that. So you do this pre-training. Now for your downstream task, if you have data, you want to fine tune it, right?



And your model should be able to produce normal encoder decoder architecture, right? Encoder blocks, decoder blocks, right? Nothing fancy here. No change in the framework.

The Vanilla transformer model was used here. So for the, pre-training, right? So when you pre-train, this is your, you know, your, I mean, there are different tasks and in your pre-training, what you do, this is very similar to the BERT kind of pre-training where you basically give a sentence as an input, and then you mask some of the sentences, right?


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pre-Training T5

- **Pre-training:** similar denoising scheme to BART (they were released within a week of each other in fall 2019)
- **Input:** text with gaps ; **Output:** a series of phrases to fill those gaps.

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

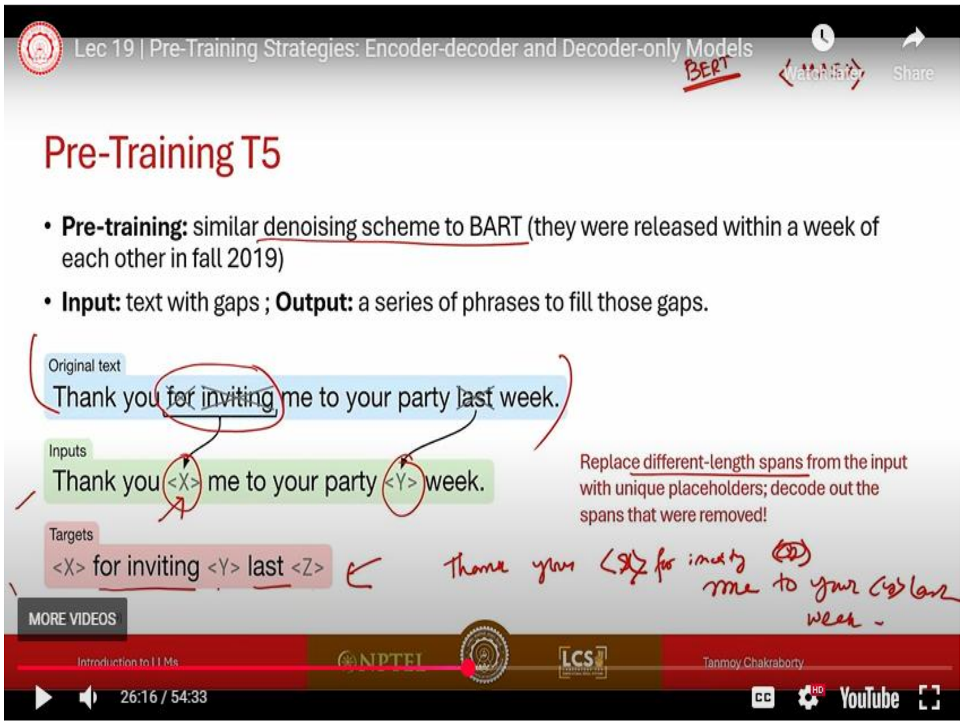
Targets

<X> for inviting <Y> last <Z>

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

With different length. So you replace different length spans, right? With some unique placeholder, the placeholder can be the mask tokens, right? And these tokens are called sentinel tokens. These tokens are part of your vocabulary, remember this, right? So in case of bert, you only use the single token called mask as a Sentinel token. But here you are using different tokens for different masking, right? When you mask, let's say for invitation, you have a token.

When you mask last, you have another token. So, and the number of times you mask an input, you will have that many of Sentinel tokens, right? So you are, you are basically increasing the vocabulary, but that's fine. Okay. And as opposed to bert where we were to produce the entire input in the decoder output, right? And not the mask, the mask inputs, but here the mask tokens here, you, that your task is to only produce the mask spans, right? If your input is this, thank you, X me to your party Y week, you will produce this kind of outputs X, For invitation, Y, you know, last and then, you know, Z can be your end of sentence sentinel or some other sentinel tokens, right? So you are only producing the token or the span which was masked in your input. You are not producing something like this.



Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

BERT

<MASK>

Pre-Training T5

- **Pre-training:** similar denoising scheme to BART (they were released within a week of each other in fall 2019)
- **Input:** text with gaps ; **Output:** a series of phrases to fill those gaps.

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

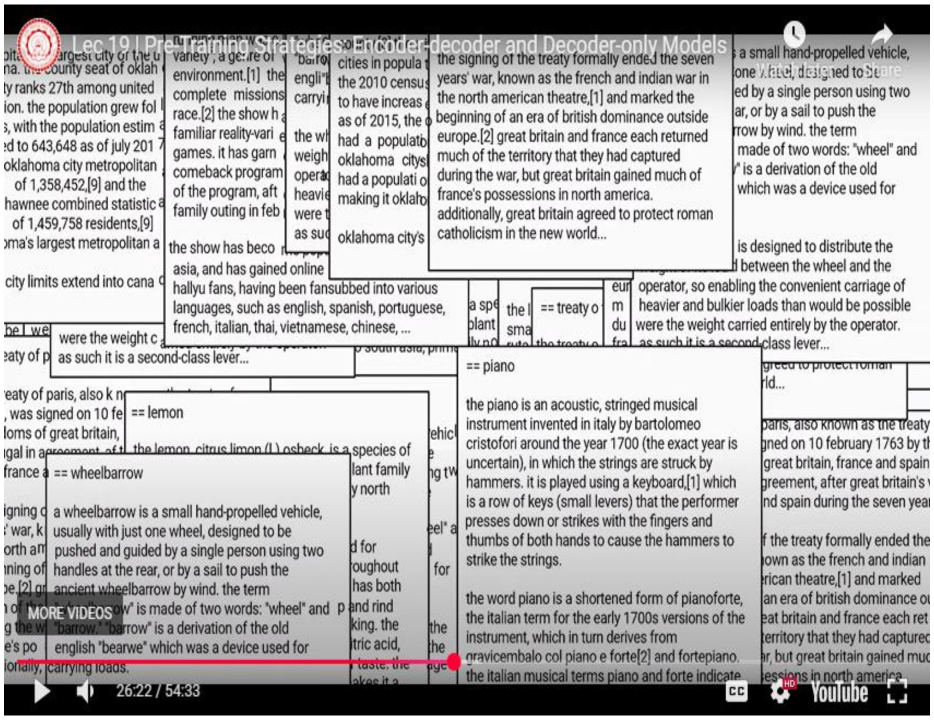
Thank you <X> for inviting <Z> me to your <Y> last week.

MORE VIDEOS

Introduction to LLMs NPTEL Tanmoy Chakraborty

26:16 / 54:33

Thank you X for inviting me to your Y last week. You do not need to produce, predict the entire sequence. That is the difference.



Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

the signing of the treaty formally ended the seven years' war, known as the french and indian war in the north american theatre,[1] and marked the beginning of an era of british dominance outside europe.[2] great britain and france each returned much of the territory that they had captured during the war, but great britain gained much of france's possessions in north america. additionally, great britain agreed to protect roman catholicism in the new world...

== piano

the piano is an acoustic, stringed musical instrument invented in italy by bartolomeo cristofori around the year 1700 (the exact year is uncertain), in which the strings are struck by hammers. it is played using a keyboard,[1] which is a row of keys (small levers) that the performer presses down or strikes with the fingers and thumbs of both hands to cause the hammers to strike the strings.

the word piano is a shortened form of pianoforte, the italian term for the early 1700s versions of the instrument, which in turn derives from gravicembalo col piano e forte[2] and fortepiano. the italian musical terms piano and forte indicate

== wheelbarrow

a wheelbarrow is a small hand-propelled vehicle, usually with just one wheel, designed to be pushed and guided by a single person using two handles at the rear, or by a sail to push the ancient wheelbarrow by wind. the term "wheelbarrow" is made of two words: "wheel" and "barrow." "barrow" is a derivation of the old english "bearwe" which was a device used for carrying loads.

== lemon

the lemon, citrus limon (l.) osbeck, is a species of

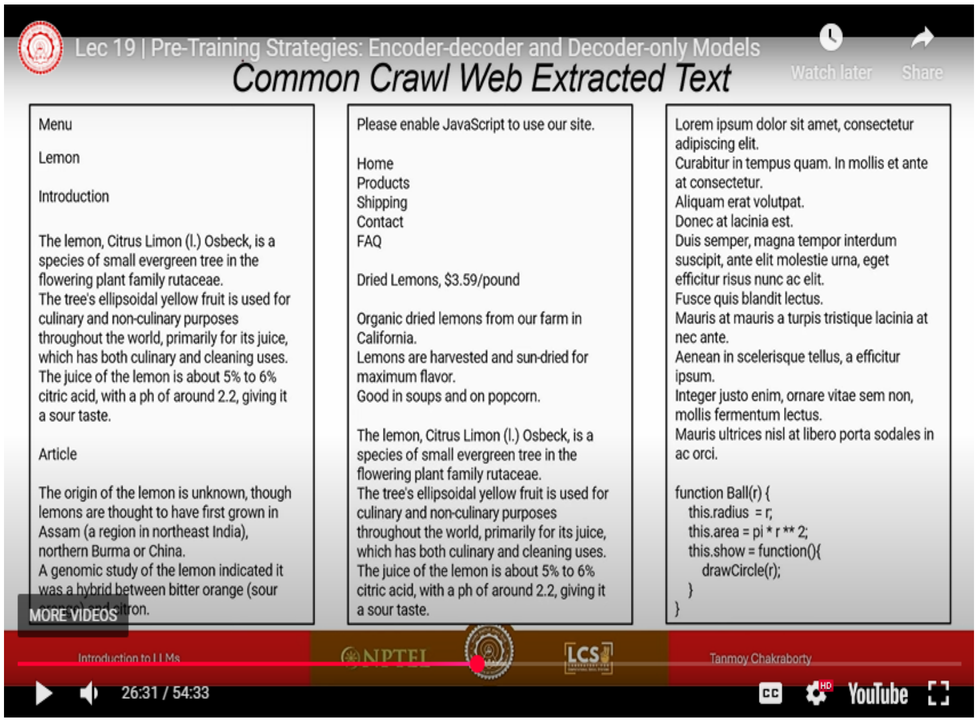
operator, so enabling the convenient carriage of heavier and bulkier loads than would be possible were the weight carried entirely by the operator. as such it is a second-class lever...

hallyu fans, having been fansubbed into various languages, such as english, spanish, portuguese, french, italian, thai, vietnamese, chinese, ...

MORE VIDEOS

26:22 / 54:33

So for training they collected a lot of corpus, a lot of documents from the web, right?



Common Crawl Web Extracted Text

Menu

Lemon

Introduction

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lemon is unknown, though lemons are thought to have first grown in Assam (a region in northeast India), northern Burma or China.
A genomic study of the lemon indicated it was a hybrid between bitter orange (sour orange) and citron.

Please enable JavaScript to use our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, $3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

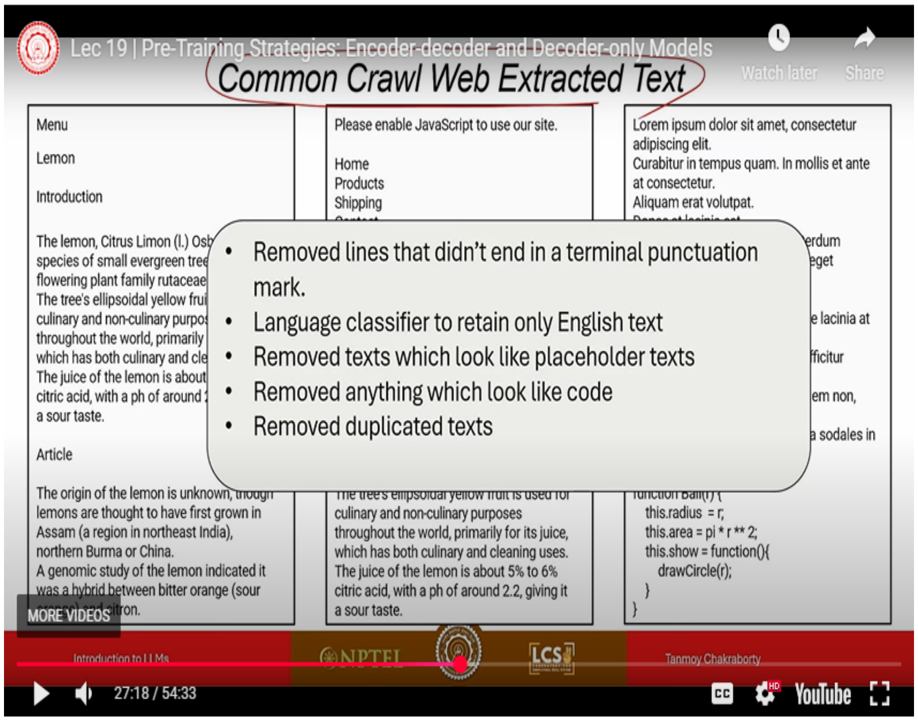
The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Curabitur in tempus quam. In mollis et ante at consectetur.
Aliquam erat volutpat.
Donec at lacinia est.
Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.
Fusce quis blandit lectus.
Mauris at mauris a turpis tristique lacinia at nec ante.
Aenean in scelerisque tellus, a efficitur ipsum.
Integer justo enim, ornare vitae sem non, mollis fermentum lectus.
Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {
  this.radius = r;
  this.area = pi * r ** 2;
  this.show = function(){
    drawCircle(r);
  }
}
```

They use this common crawl web scraper, right? It is a non-profit organization. They periodically scrape the internet, right? And as you may imagine, internet data has a lot of garbage, a lot of noise. So they basically did a series of pre-processing to remove noise from the input.

Pre-processings include removing lines which did not end with the terminal punctuation, they only retained the English text. They removed all non-English texts. They also removed those tokens which look like placeholders. They removed the codes, HTML codes, JavaScript codes from the documents. When you scrape a website, there are many HTML tags, JavaScript tags, which they essentially removed.

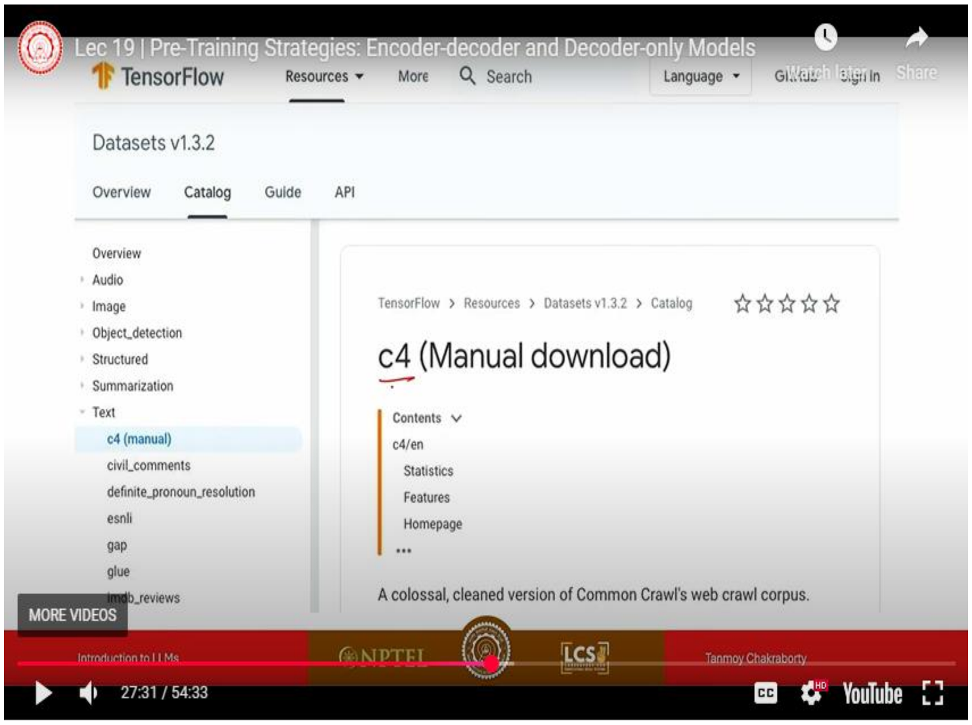


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Common Crawl Web Extracted Text

- Removed lines that didn't end in a terminal punctuation mark.
- Language classifier to retain only English text
- Removed texts which look like placeholder texts
- Removed anything which look like code
- Removed duplicated texts

And they removed many duplicate texts. They call the clean corpus C4 corpus and this is publicly available, right?



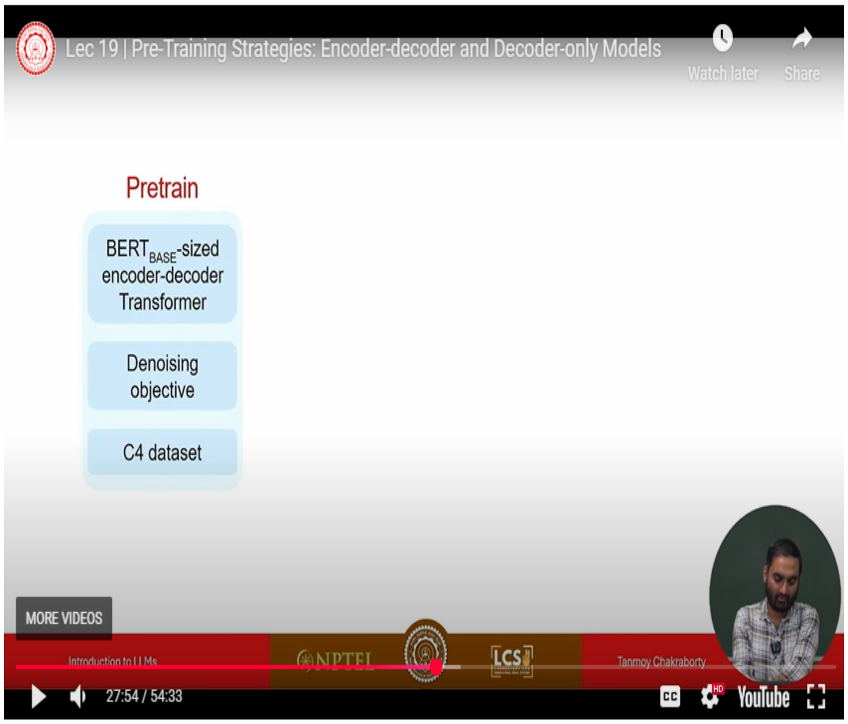
Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

TensorFlow › Resources › Datasets v1.3.2 › Catalog

c4 (Manual download)

A colossal, cleaned version of Common Crawl's web crawl corpus.

A colossal cleaned version of common crawl, common crawl wave crawl corpus, right? There are C, there are four Cs. So C4 corpus was was collected for training right So this is their pre-training setup.


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models

Pretrain

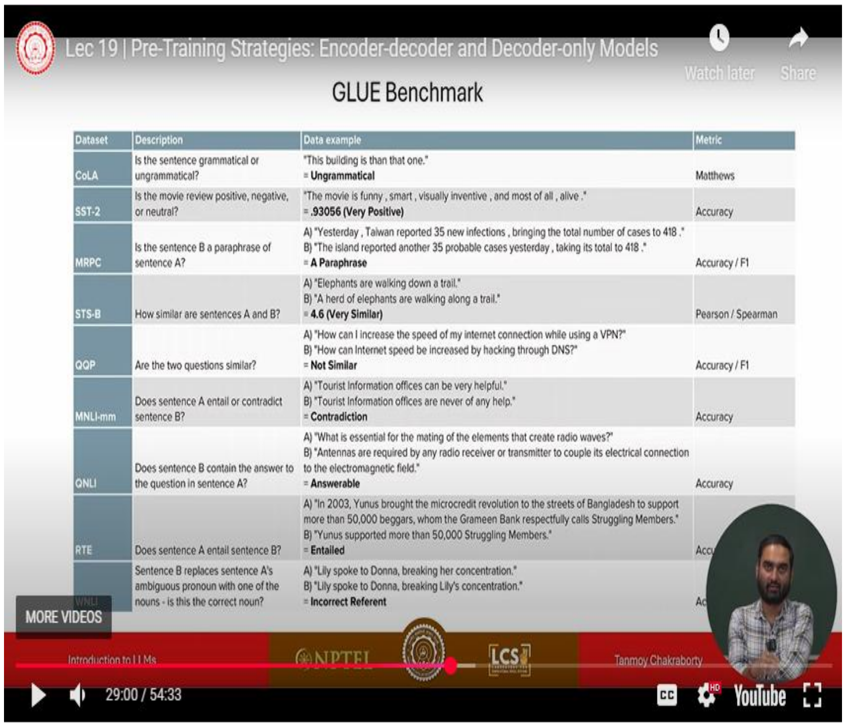
BERT$_{BASE}$-sized encoder-decoder Transformer

Denoising objective

C4 dataset

They use you know a bert size encoder decoder transformer right it can be six layer encoder six layer decoder right they basically wanted to compare with bert remember when bert was proposed right bert was proposed one year before t5 one one year before t5 and at that time everyone essentially was behind BERT, right? They were trying to defeat, they were trying to outperform BERT. So in order to make a comparable model, they took a BERT size encoder-decoder model, right? And they added different denoising objectives, right? Spanning different, so masking different spans.

You can also think of rotation, whatever permutation and so on. They use C4 data. and they pretend this, they pretend on this and then they fine tune on a series of tasks, right. They fine tune it on the glue task, okay. Now glue is a benchmark.



GLUE Benchmark

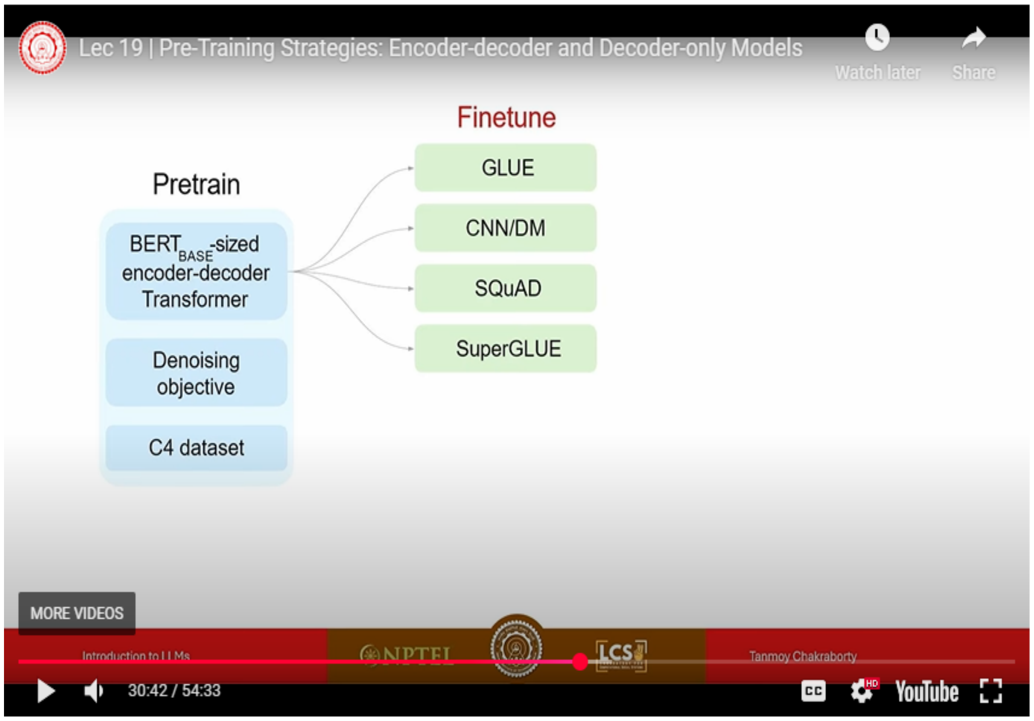
Dataset	Description	Data example	Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	Accuracy / F1
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = **4.6 (Very Similar)**	Pearson / Spearman
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	Accu
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = **Incorrect Referent**	Ac

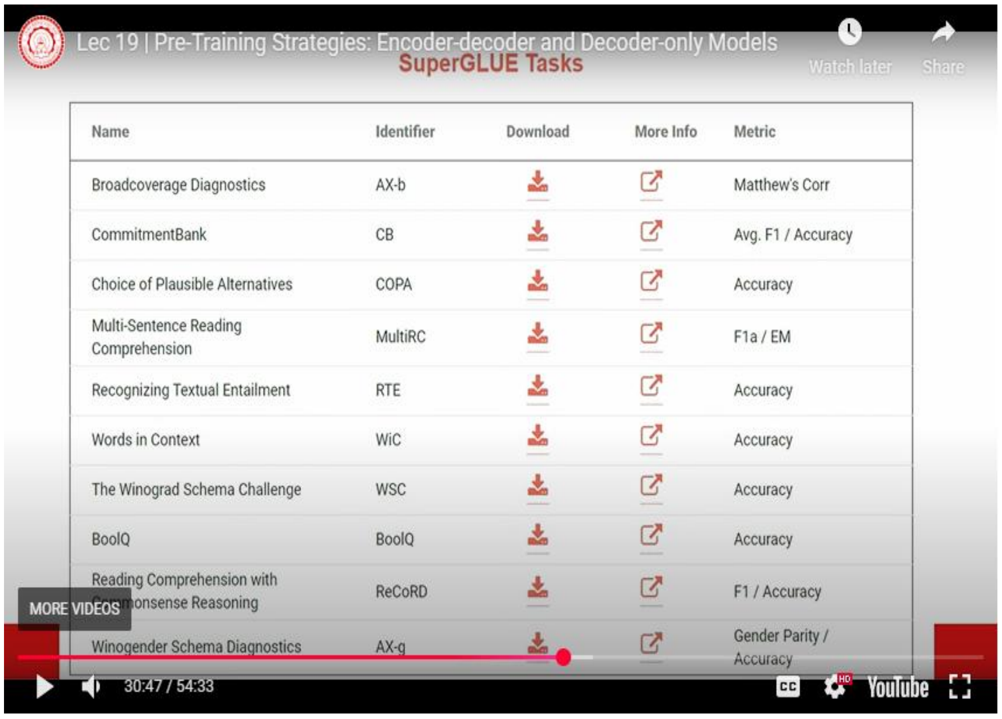
Glue is a very standard NLP benchmark which contains many tasks, right. For example, One task is called COLA. As I mentioned, COLA is corpus of linguistic acceptability, where given an input, your output will be whether this is acceptable or not. The second task is SST2. This is a sentiment analysis task, movie sentiment, positive, negative, neutral. MRPC, this is a paraphrased basically whether B is a paraphrasing of A or not.

HTSB, there are two sentences, A and B, whether they are similar or not, and you produce a number. QQP, this is again whether two questions are similar or not. MLNI, MM, this is essentially an intelment, whether A intels B or A contradicts B or so on. QNLI is a typical question answering benchmark.

RTE, whether again intelment, whether A intels B or not. and WNLI which is sentence B replaces sentence A, ambiguous pronoun with one of the nouns, is this the correct noun or not. These are all typical NLP tasks that were there when the GLUE benchmark was released. And GLUE is actually a very standard benchmark even till date. So they tried with glue, right? They also took CNN Daily Mail corpus. This is a corpus for summarization, right? Mostly abstractive summarization, right? They took the squad data.

This is a question answering data released by Stanford. They also use super glue. So glue and super glue are the standard benchmark. There are many other benchmarks these days like big bench is another benchmark.


Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models
Watch later
Share
Finetune
Pretrain
BERT$_{BASE}$-sized encoder-decoder Transformer
Denoising objective
C4 dataset
GLUE
CNN/DM
SQuAD
SuperGLUE
MORE VIDEOS
30:42 / 54:33

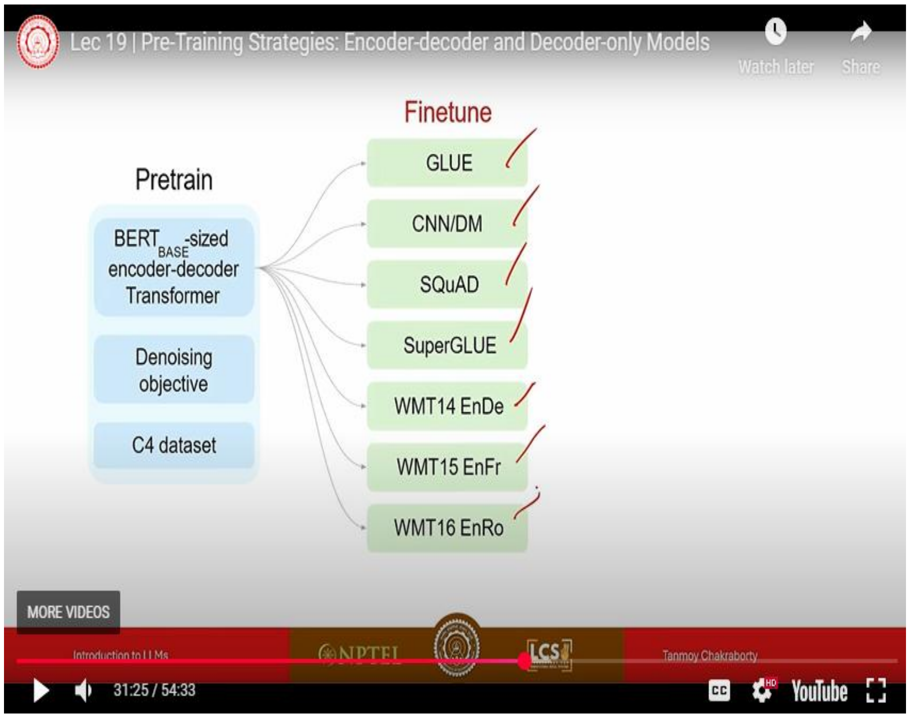

Lec 19 | Pre-Training Strategies: Encoder-decoder and Decoder-only Models
Watch later
Share
MORE VIDEOS
30:47 / 54:33

SuperGLUE Tasks

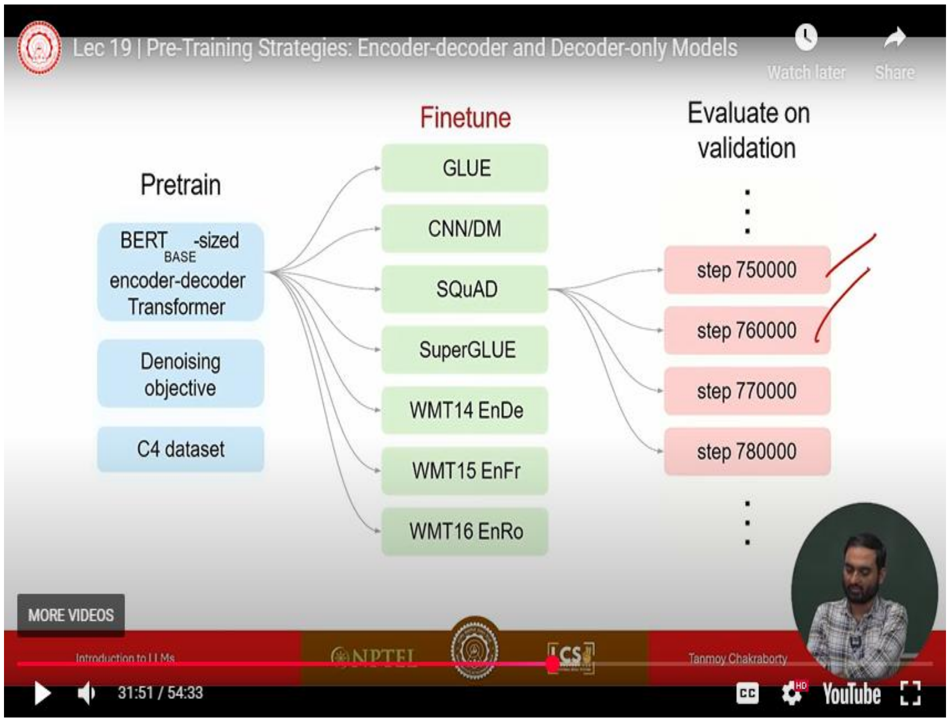
Name	Identifier	Download	More Info	Metric
Broadcoverage Diagnostics	AX-b			Matthew's Corr
CommitmentBank	CB			Avg. F1 / Accuracy
Choice of Plausible Alternatives	COPA			Accuracy
Multi-Sentence Reading Comprehension	MultiRC			F1a / EM
Recognizing Textual Entailment	RTE			Accuracy
Words in Context	WiC			Accuracy
The Winograd Schema Challenge	WSC			Accuracy
BoolQ	BoolQ			Accuracy
Reading Comprehension with Commonsense Reasoning	ReCoRD			F1 / Accuracy
Winogender Schema Diagnostics	AX-g			Gender Parity / Accuracy

Now super glue is even tougher than the glue, right? So super glue has this many tasks and these tasks are really tough, right? Even for human being. For example, recognizing textual entailment, multi-sentence reading comprehension, Boolean question answering, reading comprehension test, and a lot of other difficult questions, difficult task.

So they took Glue, CNN Daily Mail, Squared, superglue, and three different machine translation data sets, right? WNT 14, 15. So this is English to Dutch, English to German machine translation. This is English to French machine translation. This is English to Romanian machine translation.

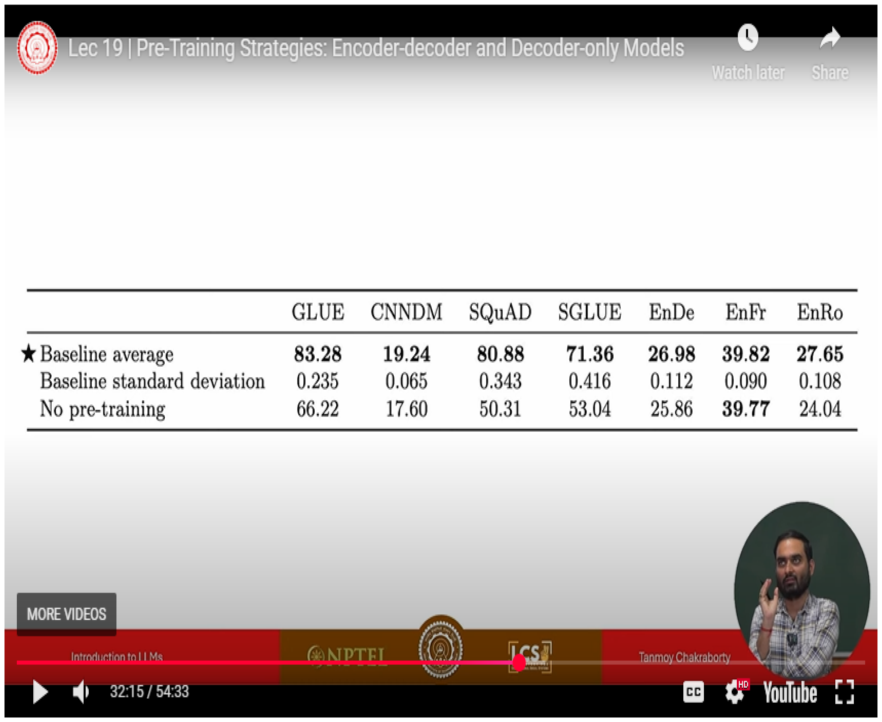


And these data sets were also very popular at the time. This is also popular these days. This is this machine translation workshop. This workshop is conducted every year, and they really keep on releasing these kind of data sets. okay after fine tuning they reported the accuracy at every checkpoint right.



So at a part let's say a checkpoint after this many steps, after this many steps, after this many steps, these are different checkpoints now check at that checkpoint what is the accuracy.

This is basically model selection you know but this is not a very good approach but it is useful sometimes to understand you know whether over the iterations whether the model is getting improved or not right.



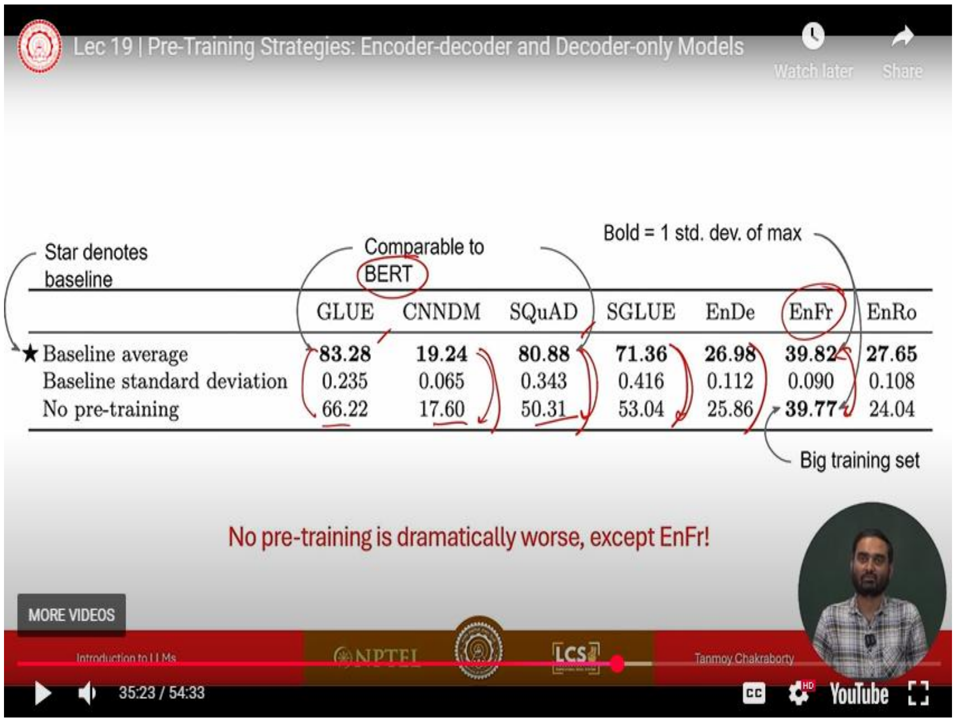
	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Let's look at a series of results okay and here I will only show you some of the important results but if you look at the paper T5 paper I think this is 50 pages paper very long paper there are series of results right reported in the paper very extensive empirical observations right. A lot of abolition studies and stuff like that but you know you can easily imagine the amount of resources that google has and only companies like google can do this kind of research you know having their gpu access tpu access because it really requires a lot of compute a really lot of compute Nevertheless, so the baseline average is their model. If you fine tune on glue and taste on glue, right? This is the accuracy.

This is CNN Daily Mail squared, right? Super glue. By the way, in glue, there are multiple tasks, right? I think that they took only one task for this. Super glue also, they took one task or I don't exactly remember.

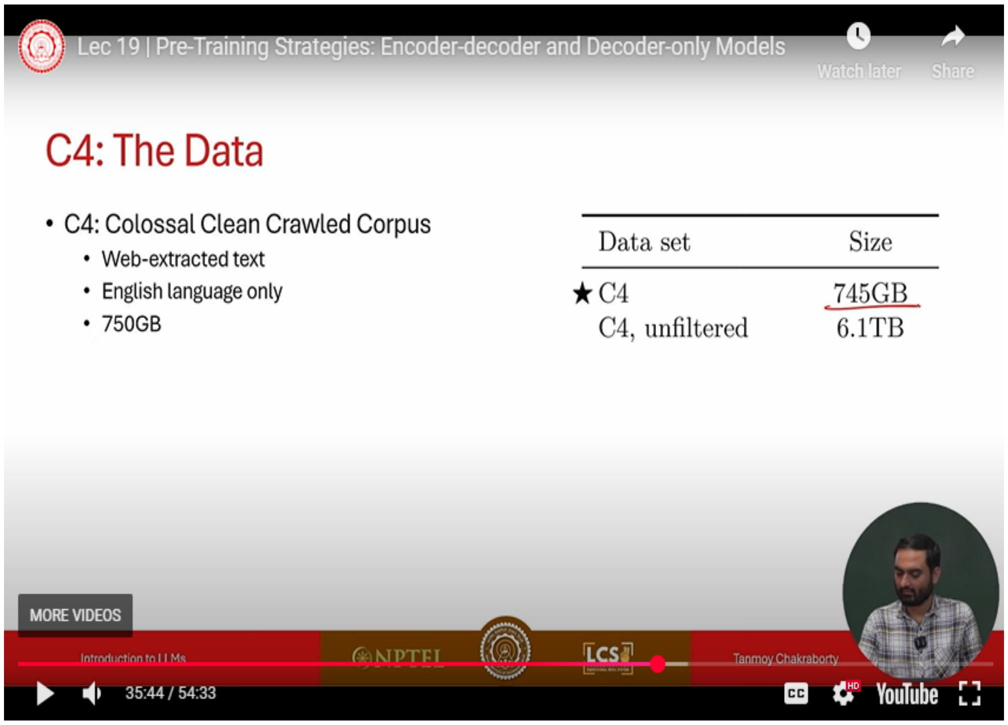
Maybe they took one task for reporting the number or I think they... took an average, then reported the average, I don't exactly remember. And then, you know, this machine translation data sets. Baseline results, whenever you see the bold numbers, these bold numbers indicate that the accuracy is within the standard deviation. They also reported the standard deviation here, right? If something is bold, it means the number, since they did it multiple times, right? So the number, reported number is, the reported number falls within the standard deviation, mean plus minus standard deviation range. And they tested their model with no pre-training, right? You just do fine tuning on a specific task.

You don't need to do this mask pre-training phrase. You don't need to do the mask pre-training, right? What you do, you take a glue benchmark, let's say, you train on that task and test it on the task, right? pure supervised machine learning method, right? And this is the number they reported. What are major observations? The first observation is that as I mentioned BERT was the best baseline that time, right? And they showed that the glue and the squared data sets, their results are quite comparable with BERT, okay? They also showed that if you don't do pre-training, your numbers are pretty less, pretty lesser, pretty lower than your pre-training followed by fine tune, right? So pre-training is actually needed. Now, without pre-training, if you do training and then predict, right? For all the tasks, your model outperforms the baseline significantly, right? You see this except for this English to French task, machine translation task. And you see here the numbers are quite close. And they argued by saying that the English to French machine translation data set was the largest one among other machine translation data set.



	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

No pre-training is dramatically worse, except EnFr!



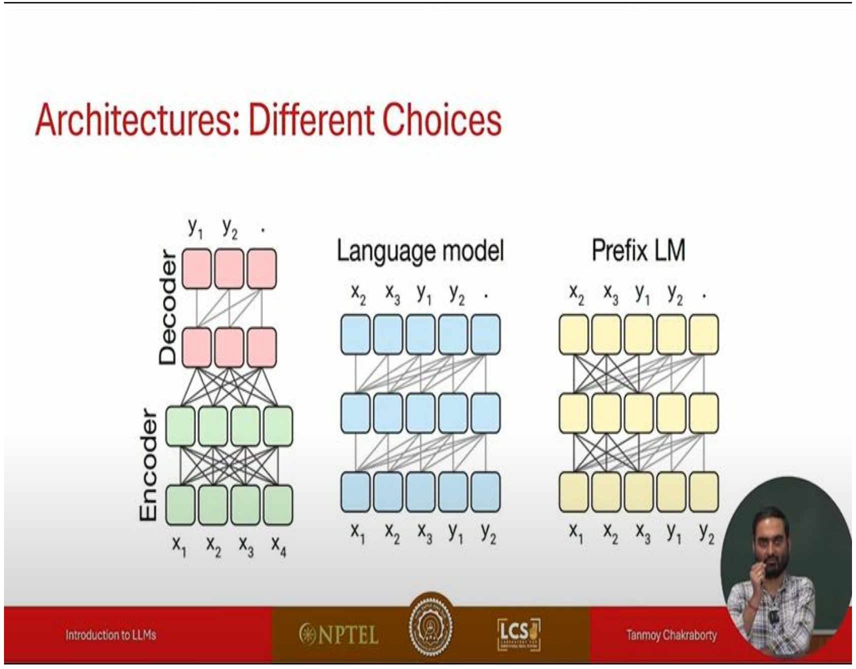
C4: The Data

- C4: Colossal Clean Crawled Corpus
  - Web-extracted text
  - English language only
  - 750GB

Data set	Size
★ C4	745GB
C4, unfiltered	6.1TB

It was large enough that you don't really need any pre-training as such. If you do normal training on the data set, that will give you enough signal about this task because of the size of the data set. If you don't have that much of data, you need pre-training. The C4 data, the size is 745 GB. The filtered data, I mentioned a series of filtration techniques were applied. If you do not filter it, if you just take the crawled C4 data which is 6 terabyte of size, you see that with clean data versus unclean data, the accuracy is not at all comparable.

Sometimes comparable, sometimes not comparable. As you see here, accuracy improves by 2%, 3% if you clean this. So cleaning is important, compact data is important and pre-training on the in-domain data is always helpful.

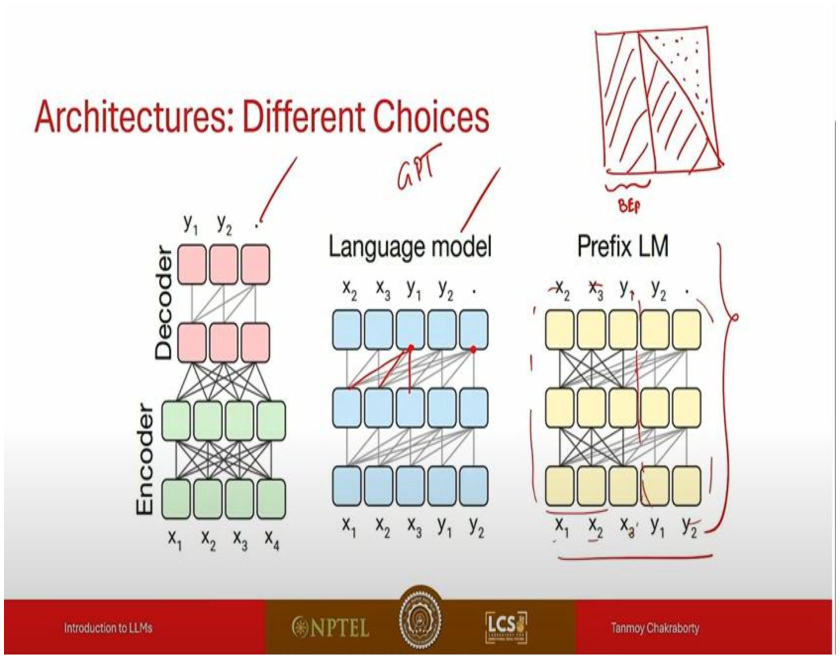


Architectures: Different Choices

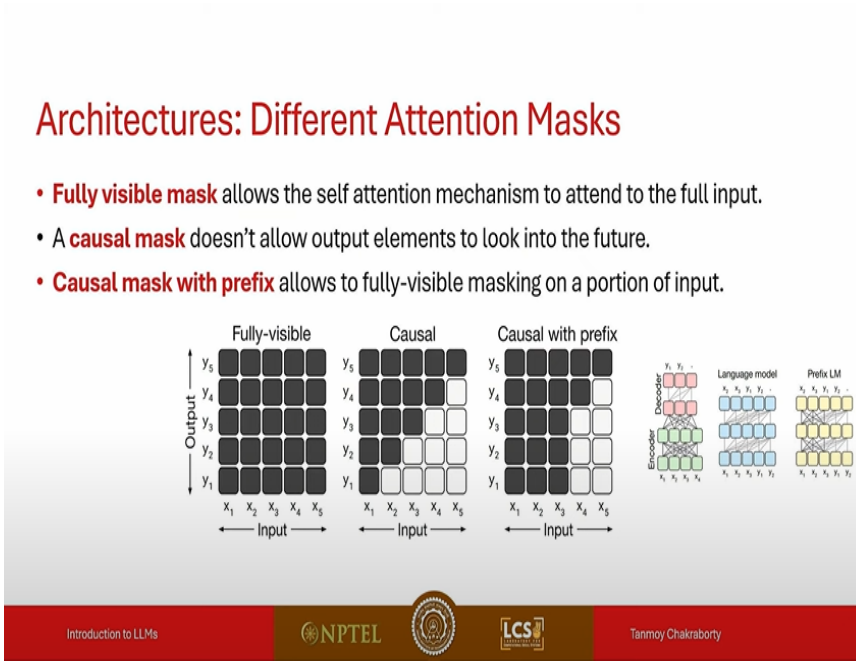
So they checked with three different architectures. They compared their methods with the standard encoder decoder model that they also use autoregressive model right next to our prediction model language modeling task you see the attention head here now this guy is attending to all the previous guys look at this one this guy is attending to right only the previous tokens so this is typical autoregressive setup GPT kind of setup and And this is a new model. It's called prefix language model, right? In prefix language model, what you do, your input is this, right? One part of your attention is unmasked.

Other part of your attention is masked, right? Your attention metrics may look like this, right? Now this is unmasked and this is masked. this part is masked. Okay. So this will act as BERT and this will act as GPT, but the difference here is that there is no encoder decoder.

There's only one stack, right? There's only one layer stack. It's not like this. In one stack, one part is, is acting as an encoder model. And the other part is acting as an encoder decoder model.



Architectures: Different Choices

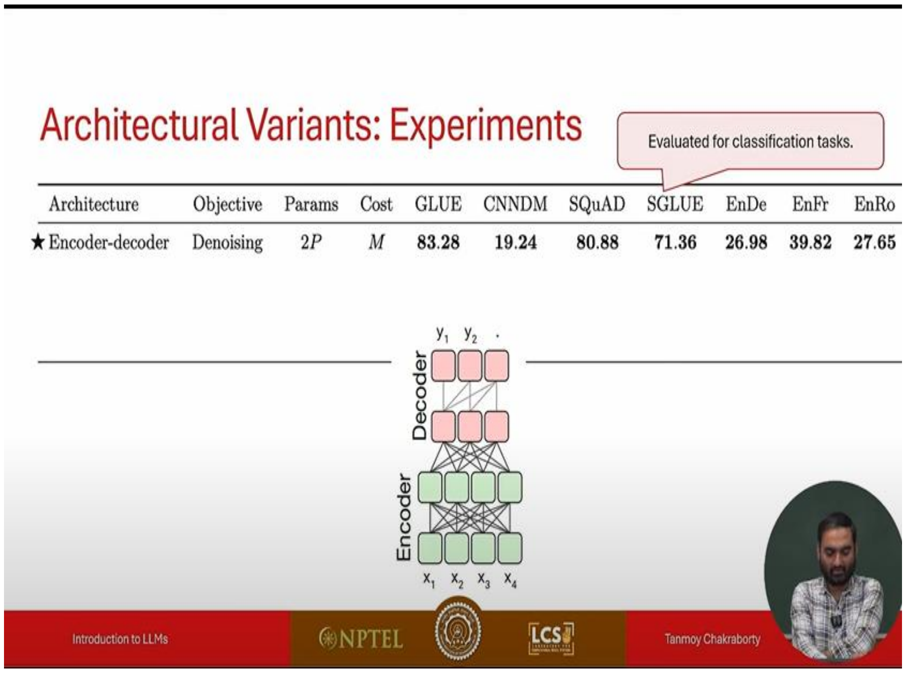


Architectures: Different Attention Masks

- **Fully visible mask** allows the self attention mechanism to attend to the full input.
- A **causal mask** doesn't allow output elements to look into the future.
- **Causal mask with prefix** allows to fully-visible masking on a portion of input.

Okay. This is called prefix lm. Now this is the figure that I mentioned, okay. In case of encoder only model, the entire token sequence is visible to you. In case of causal model, autoregressive model, right, you have access to one part, you don't have access to other part. In case of causal prefix model, the prefix model that I mentioned, one part is completely exposed, other part is basically autoregressive, okay. Okay, so now let's look at the experiments, right. with encoder-decoder model, denoising objective, 2p number of parameters, p for encoder, p for decoder, right? Now this cost is number of flops, floating point operations per second.

This is the result, right? Now let's compare. So, okay?



Architectural Variants: Experiments

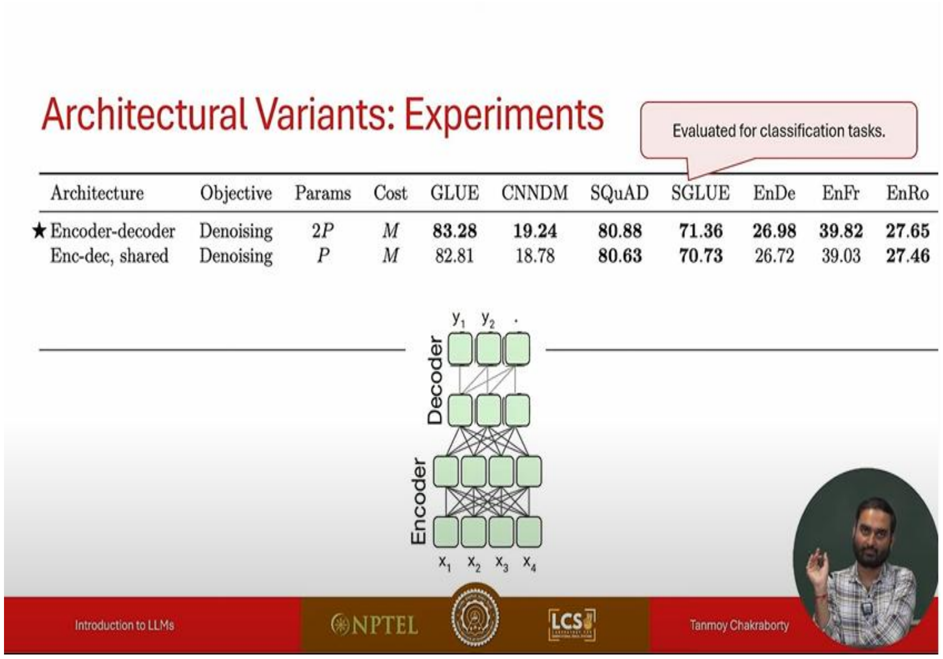
Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	80.88	**71.36**	**26.98**	**39.82**	**27.65**

Introduction to LLMs

NPTEL

Tanmoy Chakraborty



Architectural Variants: Experiments

Evaluated for classification tasks.

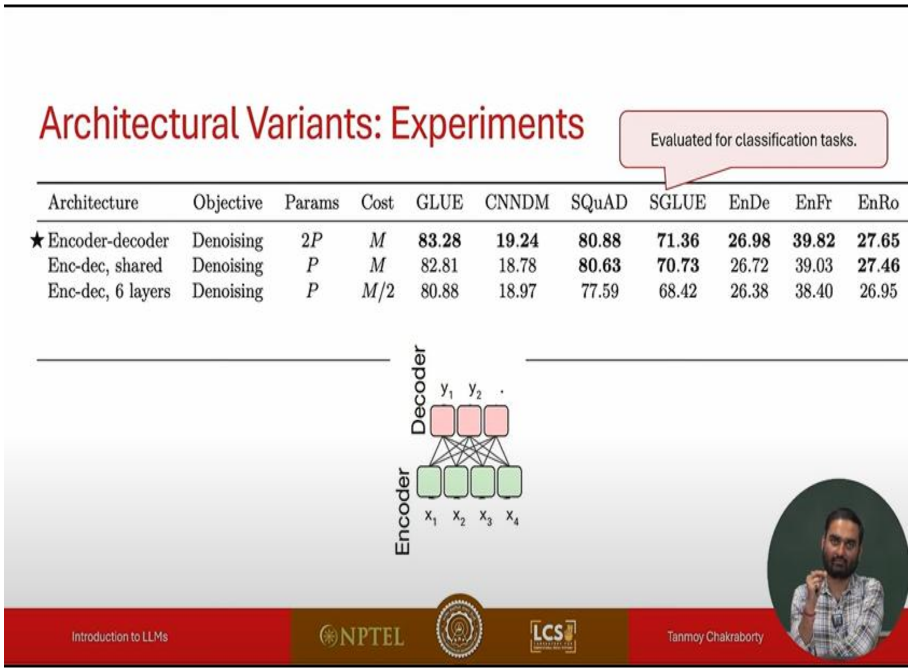
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**

Introduction to LLMs

NPTEL

Tanmoy Chakraborty

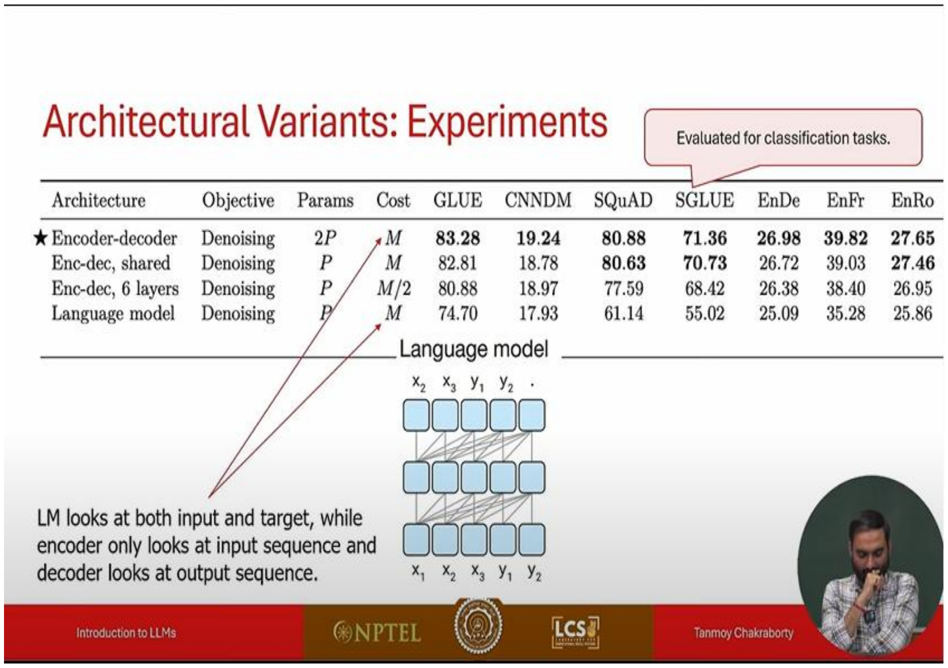
So if you... Think of another model where it is also encoder decoder, but all the parameters are shared, right? The self-attention is also shared. The feed forward is also shared. Everything is shared, right? But you have two stacks. So you need half of the parameters, P number of parameters, right? You see, let's first understand the experimental setup, okay? Then we'll analyze the result.



Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95



Architectural Variants: Experiments

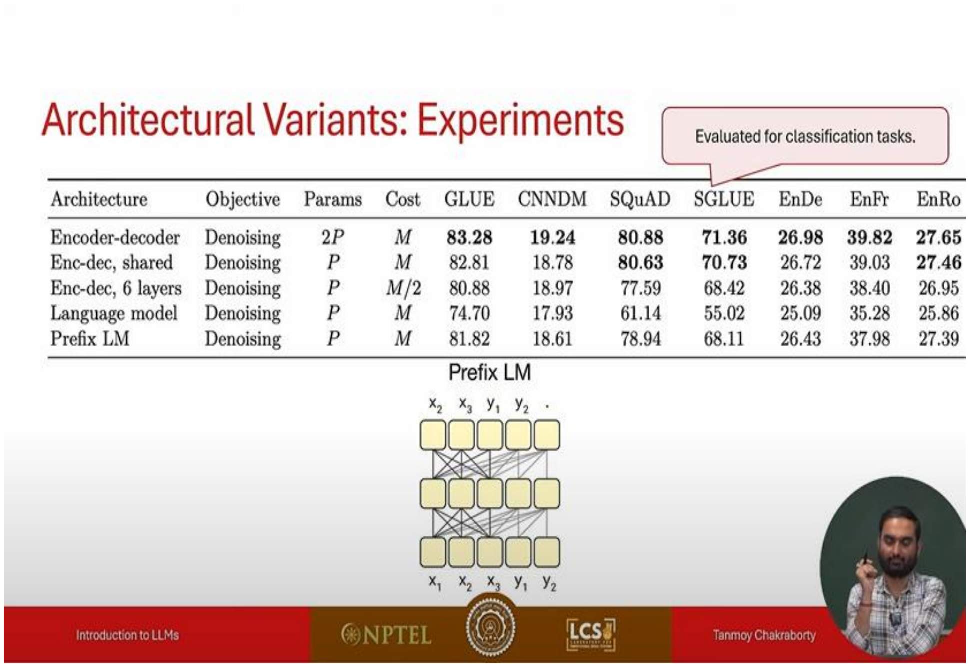
Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86

Language model

LM looks at both input and target, while encoder only looks at input sequence and decoder looks at output sequence.

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



Architectural Variants: Experiments

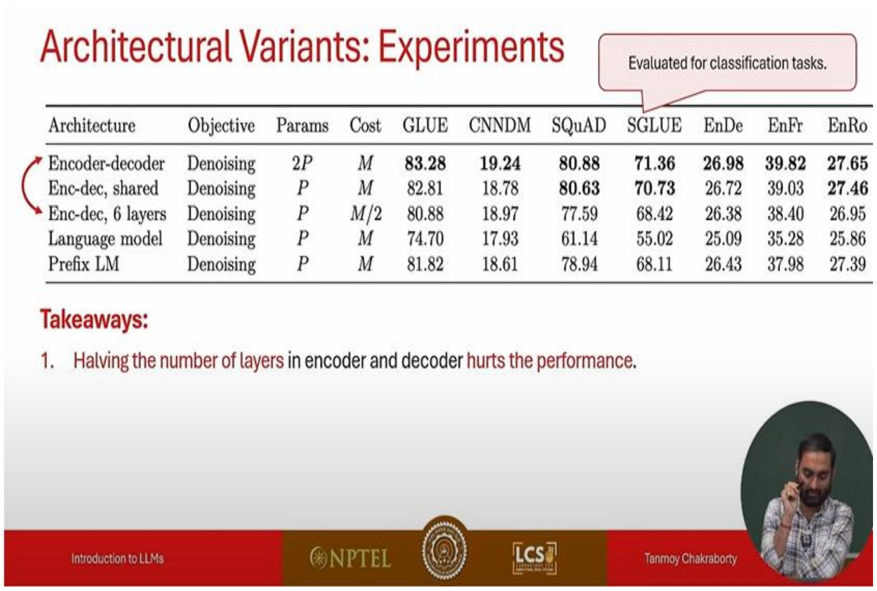
Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

Prefix LM

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

The third one is encoder-decoder, six layers. This is again denoising objective, but here there are six layers in encoder, six layers in decoder. I think in this setup there were 12 layers. The fourth one is language modeling, autoregressive setup, right? Decoder only. And the fifth one is prefix-LM, right? you have all these tasks, right? For the superglue, you only, these numbers are reported for the classification task only, okay? Okay, so what are the objectives?



Architectural Variants: Experiments

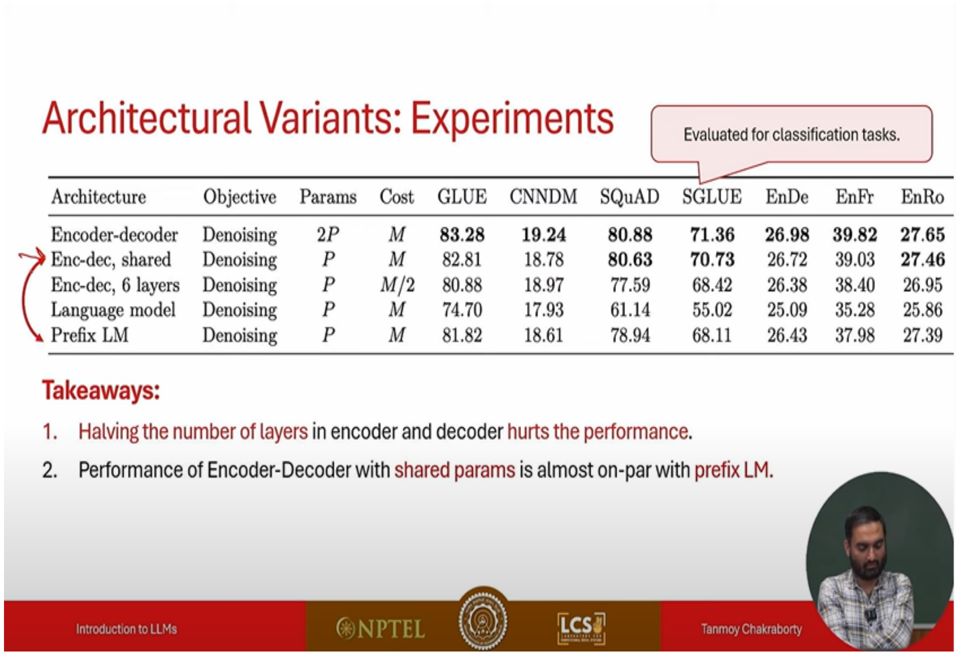
Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

Takeaways:

1. Halving the number of layers in encoder and decoder hurts the performance.

Introduction to LLMs NPTEL Tanmoy Chakraborty



Architectural Variants: Experiments

Evaluated for classification tasks.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

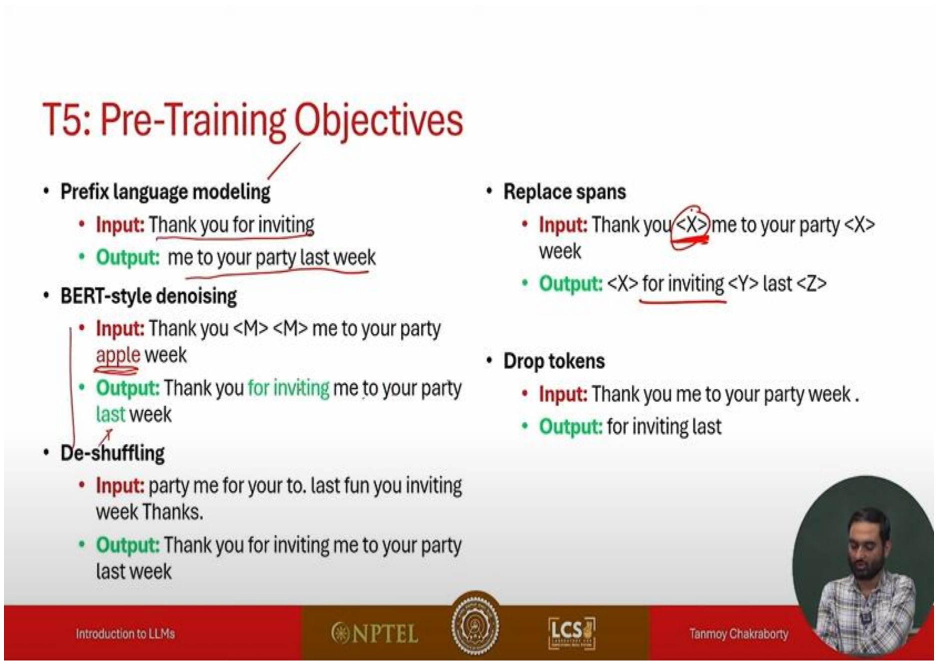
Takeaways:

1. Halving the number of layers in encoder and decoder hurts the performance.
2. Performance of Encoder-Decoder with shared params is almost on-par with prefix LM.

Introduction to LLMs NPTEL Tanmoy Chakraborty

What are the takeaways? The first takeaway is that six layers versus 12 layers, right? This is 12 layer, this is six layer, okay? The performance drops.

So if you make the number of parameters half, it hurts your performance. If you compare shared encoder decoder and prefix LM, performance on encoder decoder with shared parameter is almost on par with prefix. You see numbers are quite close, sometimes worse, sometimes better, but on par. But of course encoder decoder model that their model is always better right always better than other models okay. For pre-training, as I mentioned, they also, they did many experiments, right?



T5: Pre-Training Objectives

- **Prefix language modeling**
  - **Input:** Thank you for inviting
  - **Output:** me to your party last week
- **BERT-style denoising**
  - **Input:** Thank you <M> <M> me to your party apple week
  - **Output:** Thank you for inviting me to your party last week
- **De-shuffling**
  - **Input:** party me for your to. last fun you inviting week Thanks.
  - **Output:** Thank you for inviting me to your party last week
- **Replace spans**
  - **Input:** Thank you <X> me to your party <X> week
  - **Output:** <X> for inviting <Y> last <Z>
- **Drop tokens**
  - **Input:** Thank you me to your party week .
  - **Output:** for inviting last

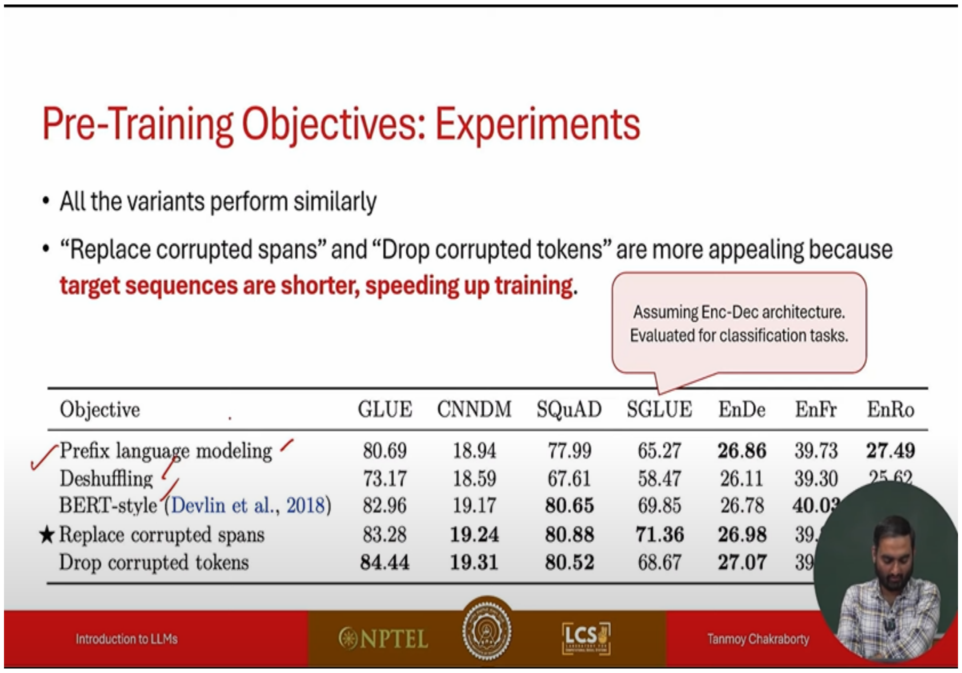
Introduction to LLMs NPTEL Tanmoy Chakraborty

Now for pre-training strategy, they use different types of strategies. One is a board style denoising, where you denoise, where you basically, you know, add some of the, you replace tokens by mask, right? For example, your input is, thank you, mask, mask, me to your party and your output would be thank you for inviting me to your party. Now here interesting point is that you sometimes also corrupt. You replace a token by another token. For example, Apple week and your task is to predict this as last week, right? Prefix language modeling objective you should do.

For example, your input is thank you for inviting. Your output will be me for your party last week. You basically predict the next set of sequences. Reshuffling, as I mentioned in case of bert, your input is let's say your output should be thank you for inviting me to your party last week. Your input can be party me for your two last fun and blah, blah, blah.

Replace spans, this is span corruption. You just, you know, you corrupt an entire span. The span is for inviting, for example, and you replace it by a sentinel token. You drop some of the tokens. This is, again, another object, another, you know, another objective that was used in bert.

You drop some of the tokens and you ask the model to predict those tokens, okay? All right, so...



Pre-Training Objectives: Experiments

- All the variants perform similarly
- "Replace corrupted spans" and "Drop corrupted tokens" are more appealing because **target sequences are shorter, speeding up training.**

Assuming Enc-Dec architecture. Evaluated for classification tasks.

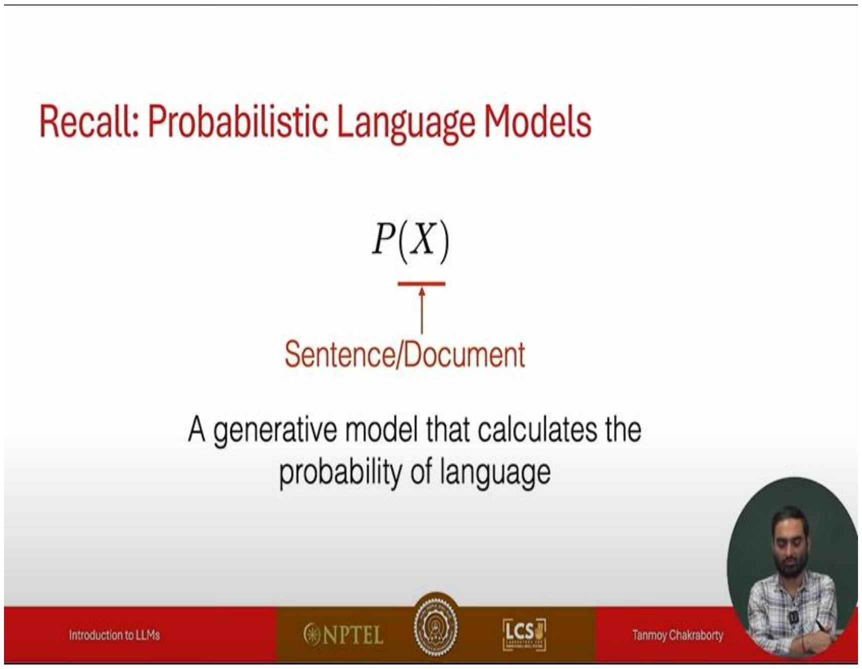
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
✓ Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	[illegible]
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.[illegible]	[illegible]
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39[illegible]	[illegible]

Introduction to LLMs NPTEL Tanmoy Chakraborty

you look at the experiment, prefix language model, this is kind of this one where this is exposed to you and this is something that you want to predict resuffling, bert style mask language modeling, replace corrupted span and then drop corrupted token and you see here that most cases, if you look at Glue, dropping an entire token helps. Here also, this is better. This is better. In case of Super Glue, I think replace corrupted span actually works better right and you know these numbers are quite comparable as this is better right. So most cases as you can see by this sometimes this is better, so there is no clear cut answer, which strategy is better, but they, you know, they did many experiments as I mentioned, and they concluded that replace the corrupted span, right? This one and drop the corrupted token, right? These are more appealing than other tokens. Now the other strategies, the reason is that, the other reason is that when you drop tokens, your length will decrease, right? You need to produce less number of tokens. Isn't it? So therefore this will also help you speeding up the training process.

Pre-Training Decoder-only Models

GPT and Llama



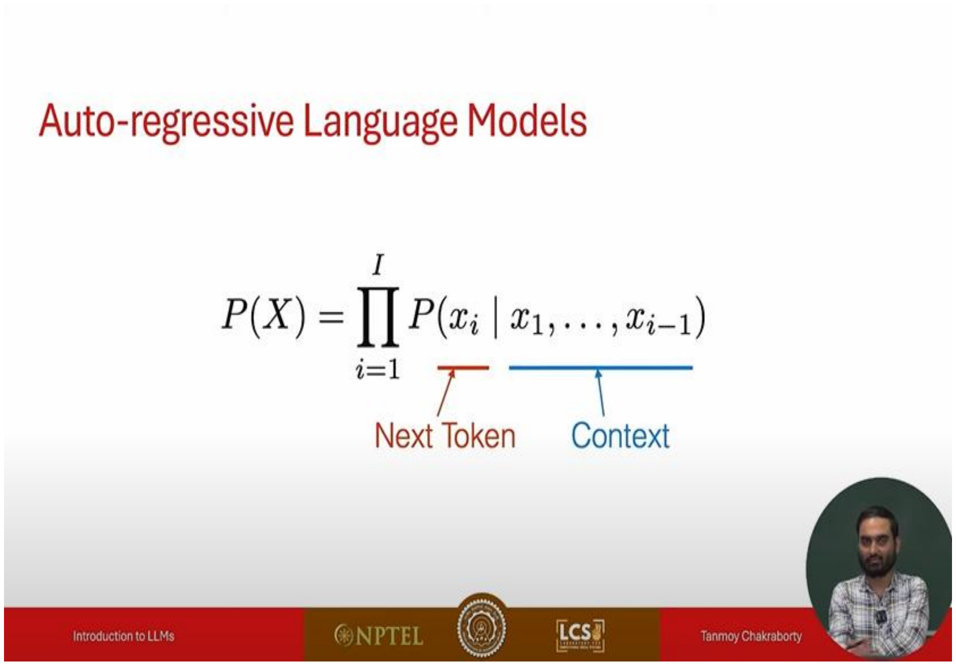
Recall: Probabilistic Language Models

$$\underline{P(X)}$$

Sentence/Document

A generative model that calculates the probability of language

Okay. So now we discuss decoder only models. Decoder only models as you can imagine, right? The strategy is not very new here, right? You don't need any input corruption as such, right? You are given a sequence of tokens and you predict what's going to the next token. Very simple, right? Generative model that calculates the probability of the language, right?

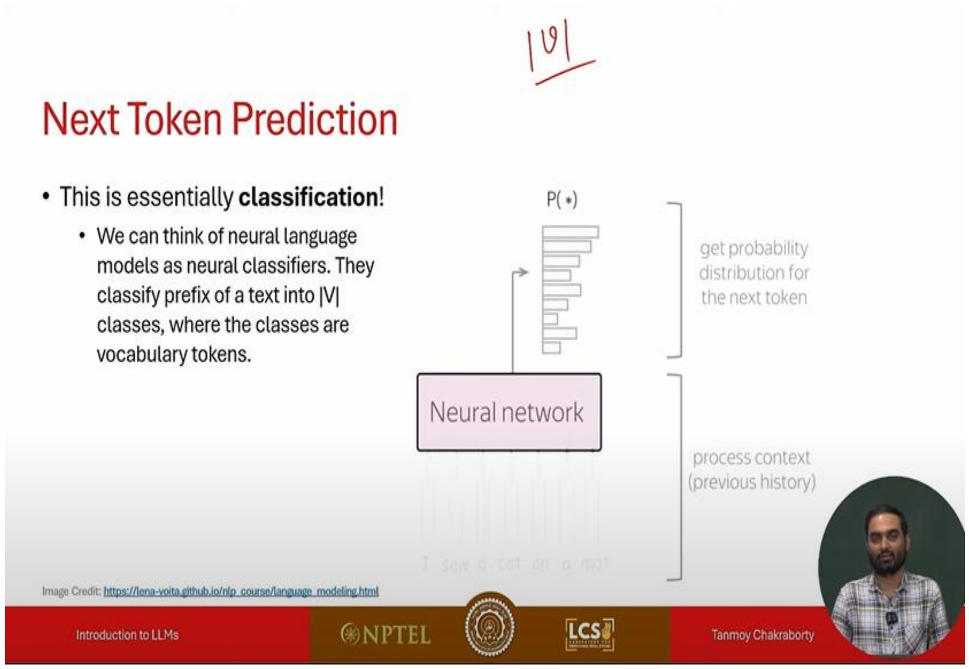


Auto-regressive Language Models

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Token}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

And this is the auto-regressive language modeling objective that I mentioned, I think, a thousand times in this course. This is the input. You predict the next token.

You essentially produce a distribution, right, over the tokens, and that's going to be your output. Then you sample token from the distribution.



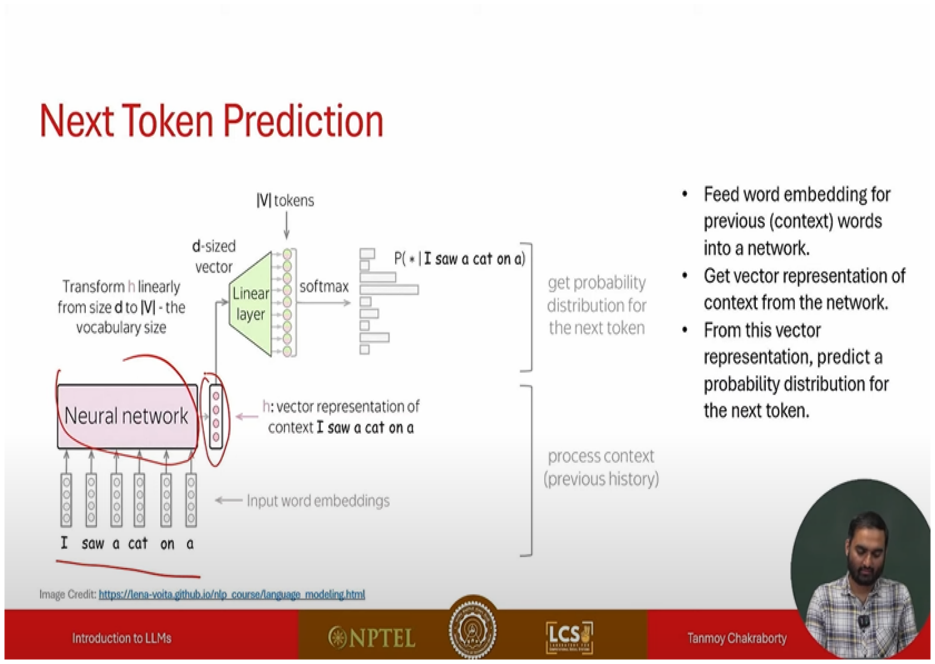
Next Token Prediction

- This is essentially **classification**!
  - We can think of neural language models as neural classifiers. They classify prefix of a text into |V| classes, where the classes are vocabulary tokens.

|V|
P(*)
get probability distribution for the next token
Neural network
process context (previous history)
I saw a cat on a mat

Image Credit: https://lena-voita.github.io/nlp_course/language_modeling.html

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty



Next Token Prediction

|V| tokens
d-sized vector
Transform h linearly from size d to |V| - the vocabulary size
Linear layer
softmax
P(* | I saw a cat on a)
get probability distribution for the next token
Neural network
h: vector representation of context I saw a cat on a
Input word embeddings
process context (previous history)
I saw a cat on a

- Feed word embedding for previous (context) words into a network.
- Get vector representation of context from the network.
- From this vector representation, predict a probability distribution for the next token.

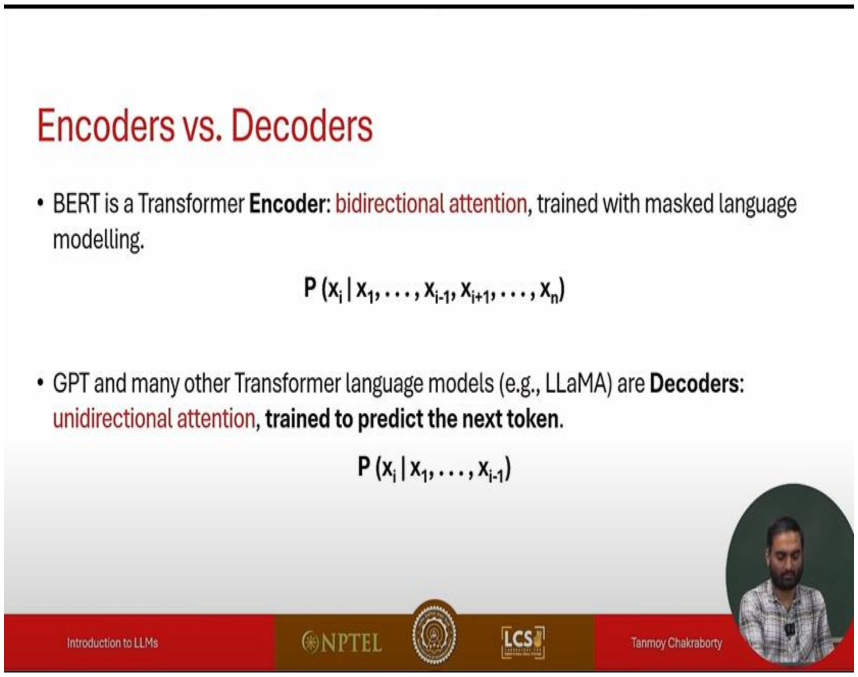
Image Credit: https://lena-voita.github.io/nlp_course/language_modeling.html

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

You can think of it as a classification problem because now given mod V where V is the size of the vocabulary, you have mod V number of tokens in your vocabulary.

Essentially your task would be to classify. You basically choose one of the labels. You can think of tokens as labels. You choose one of the labels from the mod V number of labels.

Okay. So this is your input. Right it is a decoder only block right you produce the hidden state your hidden state will be fed to a linear layer right which will produce a distribution soft max of course linear layer followed by soft max it will produce a distribution from the distribution you sample okay.



Encoders vs. Decoders

- BERT is a Transformer **Encoder**: bidirectional attention, trained with masked language modelling.

$$P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

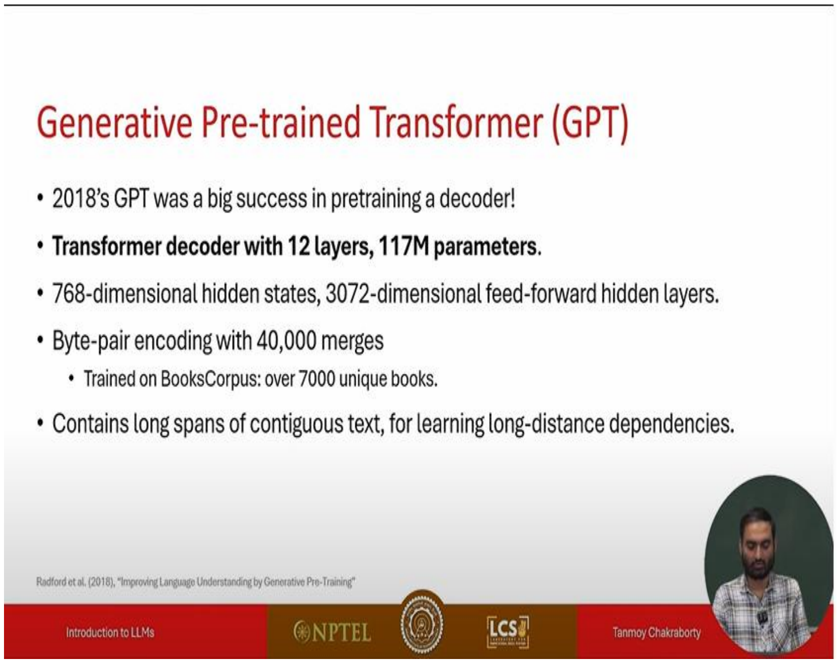
- GPT and many other Transformer language models (e.g., LLaMA) are **Decoders**: unidirectional attention, **trained to predict the next token.**

$$P(x_i \mid x_1, \ldots, x_{i-1})$$

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

So nothing fancy here, a quick comparison in case of BERT which was an encoder only model, your input was X1, X2, dot, dot, dot, X i minus 1, X i plus 1 and this was something that you masked, right, and you ask the model to predict that masked again.

In case of GPT or decoder only model, right, your input is X1, X2, dot, dot, dot, X i minus 1 and you have to predict X i.



Generative Pre-trained Transformer (GPT)

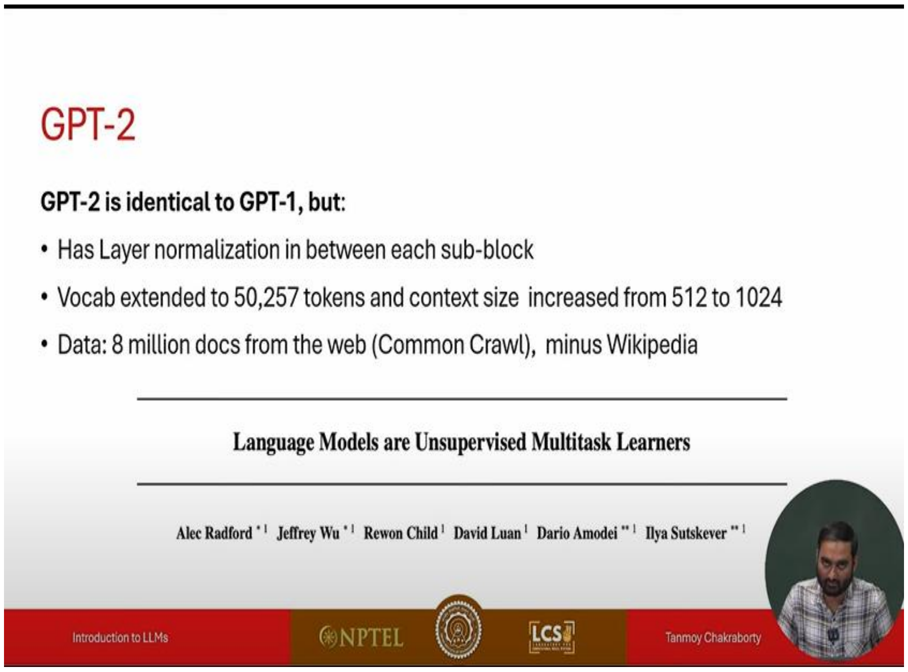
- 2018's GPT was a big success in pretraining a decoder!
- **Transformer decoder with 12 layers, 117M parameters.**
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
  - Trained on BooksCorpus: over 7000 unique books.
- Contains long spans of contiguous text, for learning long-distance dependencies.

Radford et al. (2018), "Improving Language Understanding by Generative Pre-Training"

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

Okay so the architecture is simple right and GPT-1, the first GPT paper was published in 2018 right. They use normal transformer decoder, 12 layers right, 117 million parameters right. Remember 2018 we are talking about parameters in terms of millions right, 760 dimensional hidden state, the dimension of each hidden state is 768.

3072 dimensional feed forward hidden state, hidden layers. Byte pair encoding was used. We discussed byte pair encoding before with this linear mergers and they used this book corpus which had around 7,000 unit books. This is GPT-1.



GPT-2

GPT-2 is identical to GPT-1, but:

- Has Layer normalization in between each sub-block
- Vocab extended to 50,257 tokens and context size increased from 512 to 1024
- Data: 8 million docs from the web (Common Crawl), minus Wikipedia

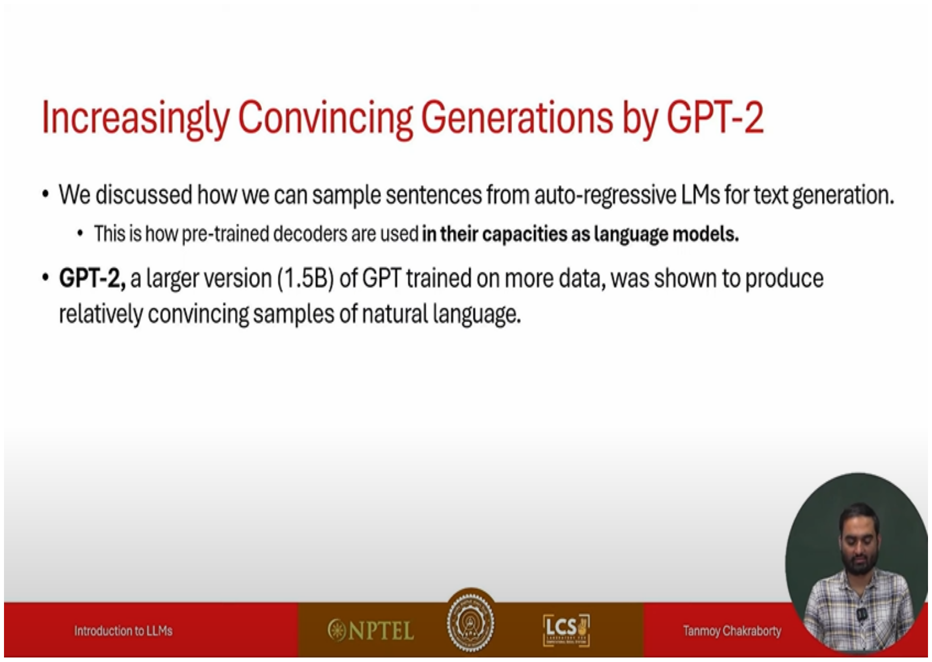
Language Models are Unsupervised Multitask Learners

Alec Radford * 1 Jeffrey Wu * 1 Rewon Child 1 David Luan 1 Dario Amodei ** 1 Ilya Sutskever ** 1

Introduction to LLMs NPTEL Tanmoy Chakraborty

GPT-2 was proposed just the next year, 2019. What's the difference? The difference is they use layer normalization in between different sub blocks.

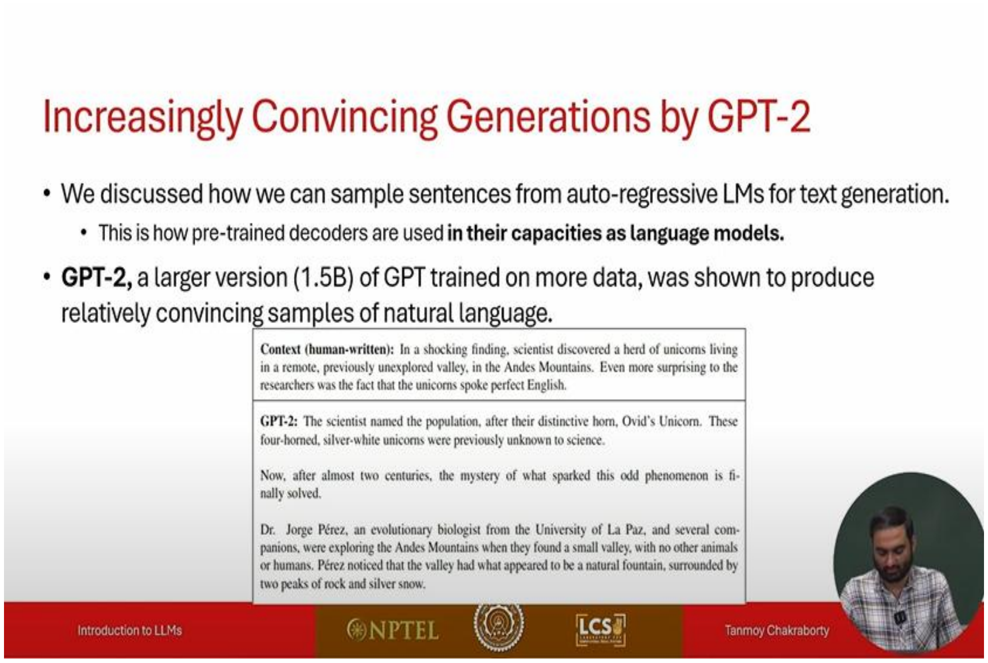
Layer normalization we discussed earlier. Their vocabulary is now extended to 50,000. Context size is now increased from 512 to 1024. Right? Now your data is a new data, which is a common crawl data, that same data that was used in T5, eight minial documents, right? And of course, minus Wikipedia. So this is GPT-2. And they showed that if you do unsupervised multitask learning, it basically helps.



Increasingly Convincing Generations by GPT-2

- We discussed how we can sample sentences from auto-regressive LMs for text generation.
  - This is how pre-trained decoders are used **in their capacities as language models.**
- **GPT-2,** a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Introduction to LLMs NPTEL Tanmoy Chakraborty



Increasingly Convincing Generations by GPT-2

- We discussed how we can sample sentences from auto-regressive LMs for text generation.
  - This is how pre-trained decoders are used **in their capacities as language models.**
- **GPT-2,** a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

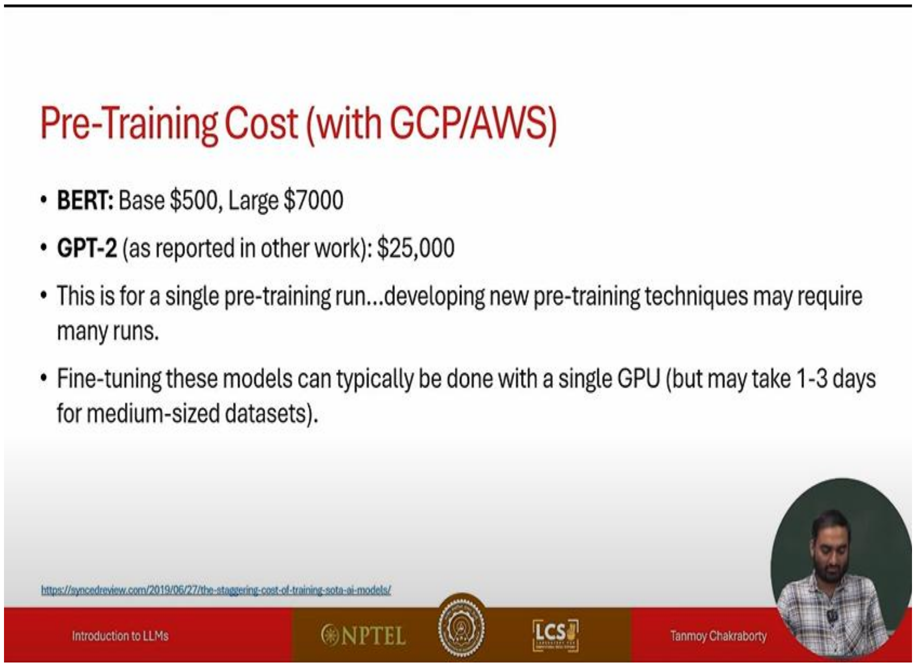
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Introduction to LLMs NPTEL Tanmoy Chakraborty

So in GPT-2, they focused more on the generation task, right? And and as opposed to GPT-1. And they showed that GPT-2 starts producing more human-like sentences, right? If you write these as your input and you ask the model to essentially keep on generating tokens to fill out the remaining part. So this is your input and these are going to be your tokens that you will essentially generate.



Pre-Training Cost (with GCP/AWS)

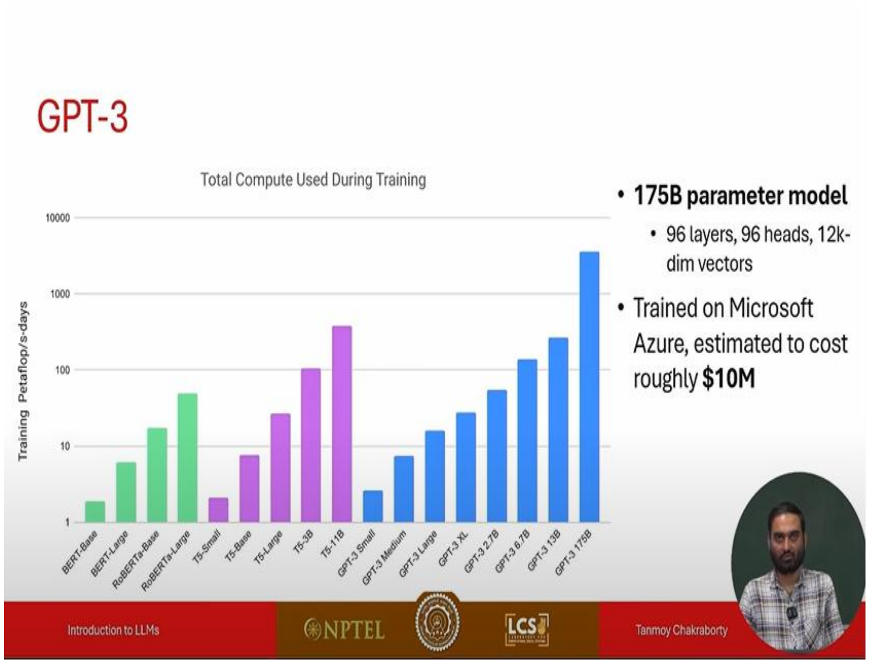
- **BERT:** Base $500, Large $7000
- **GPT-2** (as reported in other work): $25,000
- This is for a single pre-training run...developing new pre-training techniques may require many runs.
- Fine-tuning these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets).

https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

Cost-wise, quick comparison. I'm not very sure about this. I took this thing from this link, right? But base cost was 500 USD, I think with single pre-training run, right? The large required $7,000, whereas GPT-2, it basically needed approximately $24,000.

GPT-3

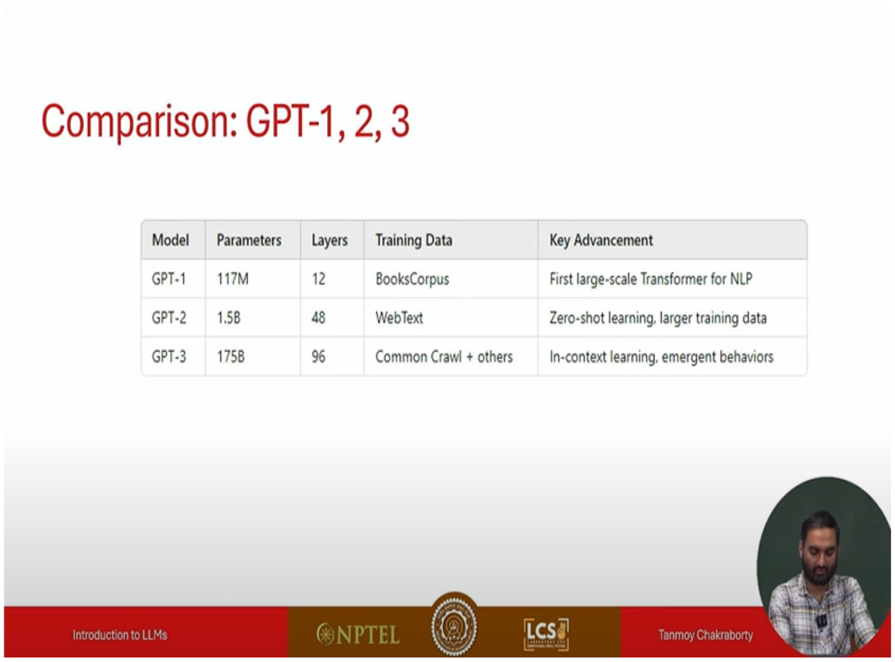


- **175B parameter model**
  - 96 layers, 96 heads, 12k-dim vectors
- Trained on Microsoft Azure, estimated to cost roughly **$10M**

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

Then came GPT-3 in 2020, right? And if you look at the training petaflop, per day, whatever right, significantly high. GPT-3 small, medium, large, XL, many versions, GPT-3 2.7 billion, right. And this is T5 statistics, this is BERT statistics.

So GPT-3, the major difference is the parameter. So we were talking about GPT-2 with only 1.5 billion parameter. Now this is 175 billion, 100 times larger than GPT-2, right? 96 layers, 96 heads in self attention component, 12,000 dimensional vectors, right? And you know, this is the rough estimate of the cost that was taken.



Comparison: GPT-1, 2, 3

Model	Parameters	Layers	Training Data	Key Advancement
GPT-1	117M	12	BooksCorpus	First large-scale Transformer for NLP
GPT-2	1.5B	48	WebText	Zero-shot learning, larger training data
GPT-3	175B	96	Common Crawl + others	In-context learning, emergent behaviors

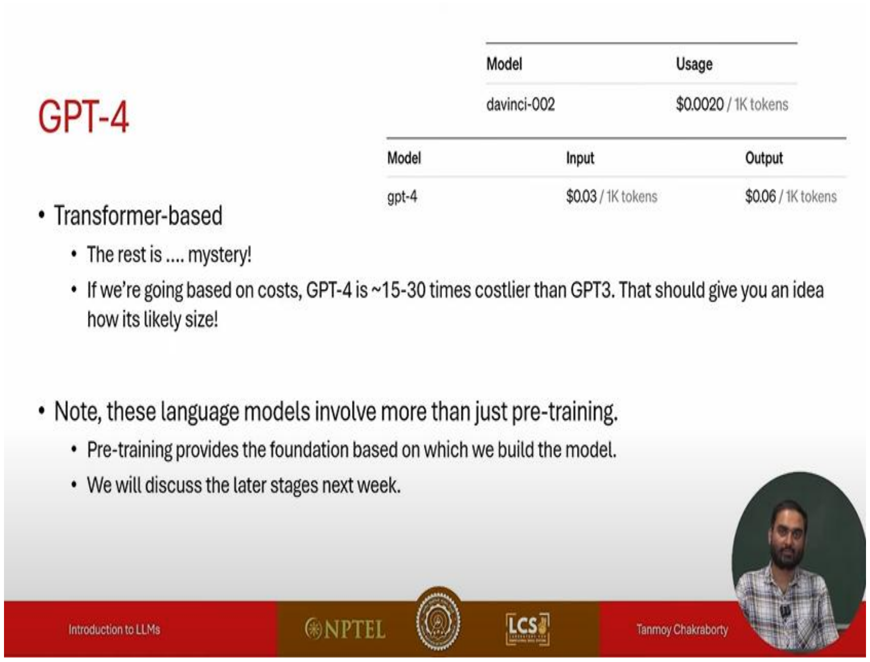
Introduction to LLMs

Tanmoy Chakraborty

So three major GPT models, GPT-1, 2, 3, parameter as you see increased drastically, number of players increased drastically, data set also increased drastically, the size increased drastically. In GPT-1 they started focusing on, they said that this is the first large scale decoder only model GPT-2, they came up with this idea called zero-shot learning with larger training.

And GPT-3, they observed this in-context learning. You write examples. We'll talk about in-context learning later. You don't need to fine tune the model. You write examples, the model will be able to understand the task and they said this is the emerging property.

At that time, they were not very sure why this is happening. It's very difficult to understand, dissect.



GPT-4

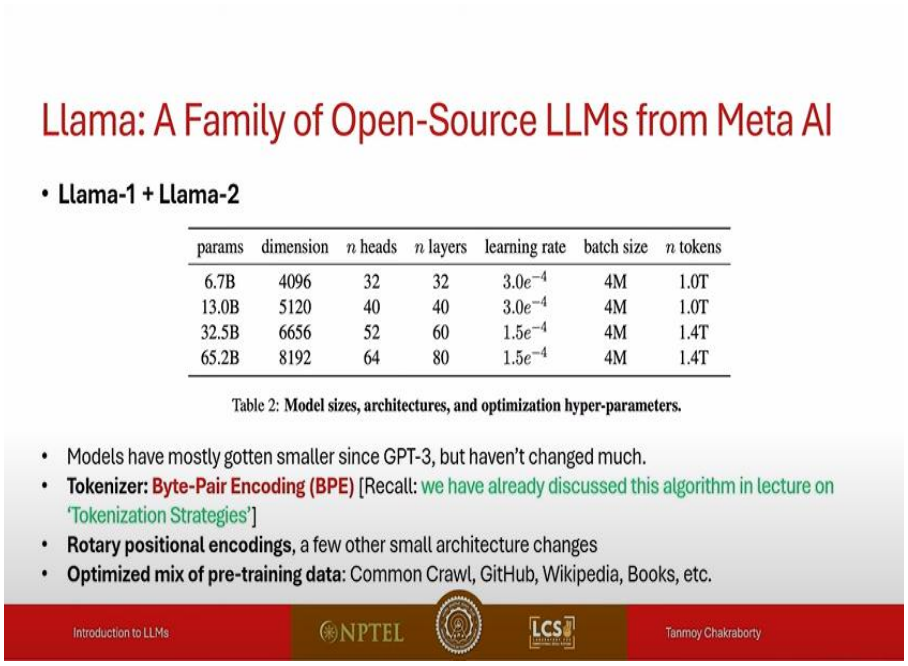
Model	Usage
davinci-002	$0.0020 / 1K tokens

Model	Input	Output
gpt-4	$0.03 / 1K tokens	$0.06 / 1K tokens

- Transformer-based
  - The rest is .... mystery!
  - If we're going based on costs, GPT-4 is ~15-30 times costlier than GPT3. That should give you an idea how its likely size!
- Note, these language models involve more than just pre-training.
  - Pre-training provides the foundation based on which we build the model.
  - We will discuss the later stages next week.

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

Then came GPT-4 in 2023, I guess. This is a mystery because this is a black box model, right? After GPT-3, they stopped releasing the architecture, right? OpenAI was no longer open at that time, right? And the exact size is not very clear, but if you look at the cost, the part token cost, this increased quite significantly, both in terms of inputs and outputs. 15 to 30 times costlier than GPT-3. This itself gives an indication that how big is the model, right? Or what's the amount of training, amount of cost that was incurred to train this model. So now GPT-4 is not only about pre-training, right? There are many, many additional things, right? For example, instruction fine tuning, we'll discuss later instruction fine tuning. So in three instruction fine tuning you teach your model how to follow instructions and then they use something called RLHF reinforcement learning from human feedback. This is again something that we'll discuss later. Okay. So today's GPT Nobody knows what's there, right, but we can only assume that, you know, pre-training followed by series of fine-tuning, instruction fine-tuning, right, different types of alignments and so on and so forth.



Llama: A Family of Open-Source LLMs from Meta AI

- **Llama-1 + Llama-2**

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

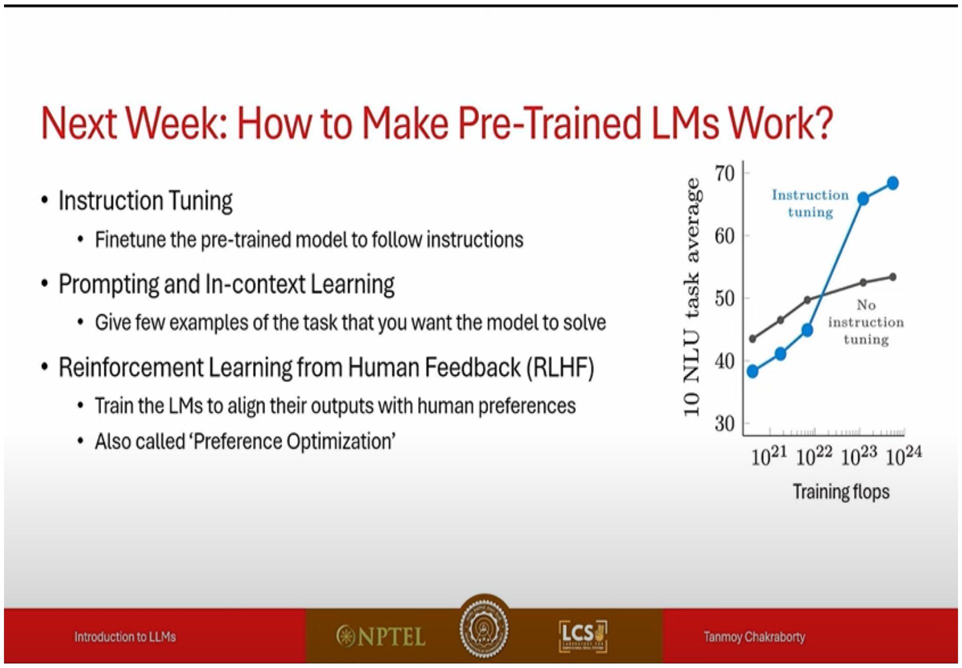
Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

- Models have mostly gotten smaller since GPT-3, but haven't changed much.
- **Tokenizer: Byte-Pair Encoding (BPE)** [Recall: we have already discussed this algorithm in lecture on 'Tokenization Strategies']
- **Rotary positional encodings**, a few other small architecture changes
- **Optimized mix of pre-training data:** Common Crawl, GitHub, Wikipedia, Books, etc.

Introduction to LLMs NPTEL LCS Tanmoy Chakraborty

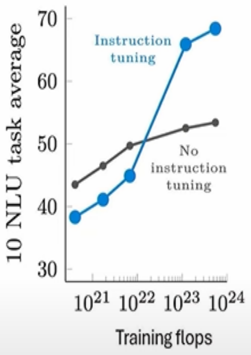
Llama, the same thing, the Llama is open source. You can still look at models, parameters. There is LlAMA 1, 2, LlAMA 3 is still available, right, different parameters, dimensions, increasing dimension, number of heads, increasing number of heads. And they use byte-pair encoding. This is also something that we discussed earlier. And they use rotary position encoding.

This is also discussed in the data set that they use this common crawl, GitHub data, Wikipedia data, books data and so on. So this is about GPT decoder only model, what is coming next?



Next Week: How to Make Pre-Trained LMs Work?

- Instruction Tuning
  - Finetune the pre-trained model to follow instructions
- Prompting and In-context Learning
  - Give few examples of the task that you want the model to solve
- Reinforcement Learning from Human Feedback (RLHF)
  - Train the LMs to align their outputs with human preferences
  - Also called 'Preference Optimization'



Introduction to LLMs | NPTEL | LCS | Tanmoy Chakraborty

So in the next week, we'll talk about instruction fine tuning, right? After pre-training, this is the step that we do, right? Now we are discussing how you can also design your own, you know, own language model, right? You do pre-training, then you do instruction fine tuning, and then you align your model with human feedback through RLHF, and then, you know, and then your model should be ready to understand the instruction, okay. All right, with this I stop and next week we will discuss the last part. Thank you.