Assignment - 3

Generative AI Assignment

Page No.
Date :

Generative AI :- The lifecycle walks you through the individual stages & decisions you have to make when you're developing Generative AI applications

* It is a subset of traditional machine learning

FLAN-T5
↳ Fine tuned
↳ Language Net
text-To-text transfer transformer

Large Language models (LLM) :- BERT, GPT, FLAN-T5, LLaMa, PaLM, BLOOM

LLaMA
Large language Model Meta AI

↳ In this course, LLM how they are built and trained,
* how you can interact with them via text known as prompts
* How to fine tune models for your use case and data
* How you can deploy them with applications to solve your business & social tasks

- The text you pass in the LLM is called prompts
- The space or memory that is available to the prompt is called the context window

* Generative Algorithms are not new, previous generations of language models made use of an architecture called RNNs (were limited by the amount of compute & memory needed to perform well at generative tasks

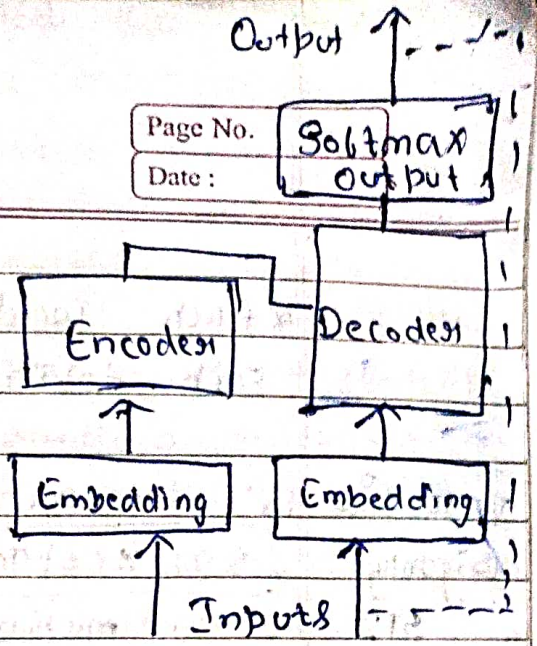
Google & the University of Toronto → Transformer Architecture had arrived

Transformers :- Scale Efficiently, Parallel process, Attention to input meaning

Page No.
Date :

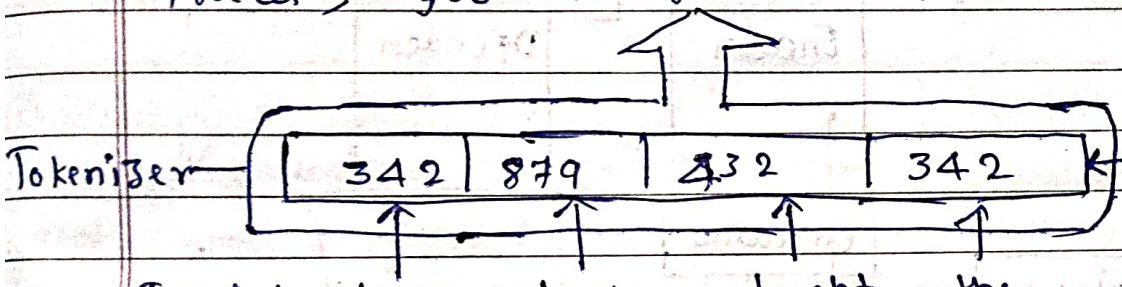
Transformers architecture :-

↳ "Attention is all you need"



Machine learning models are just big statistical calculators & they work with numbers, not words. So

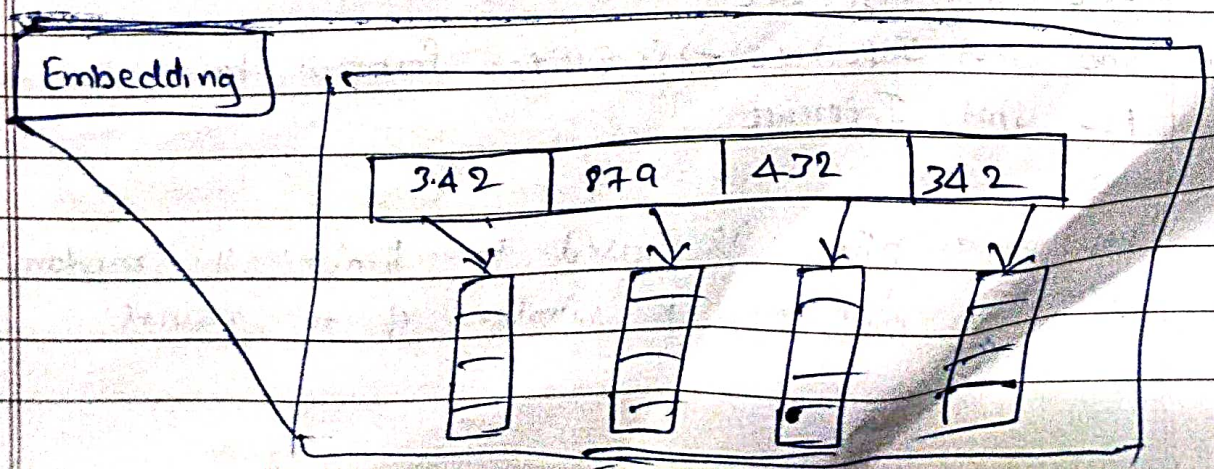
- before passing texte into the model to proces, you must first tokenize the words



Transformers work :- ~~Input~~

*) Input represented numbers, pass it to the Embedding layer

*) Embedding layer is a trainable vector Embedding space, a high-dimensional space where each token is represented as a vector & occupies unique location within that space, Each tokenID is matched to a multi-dimensional vector, these vectors learn to encode the meaning & context of individual tokens in the input sequence.

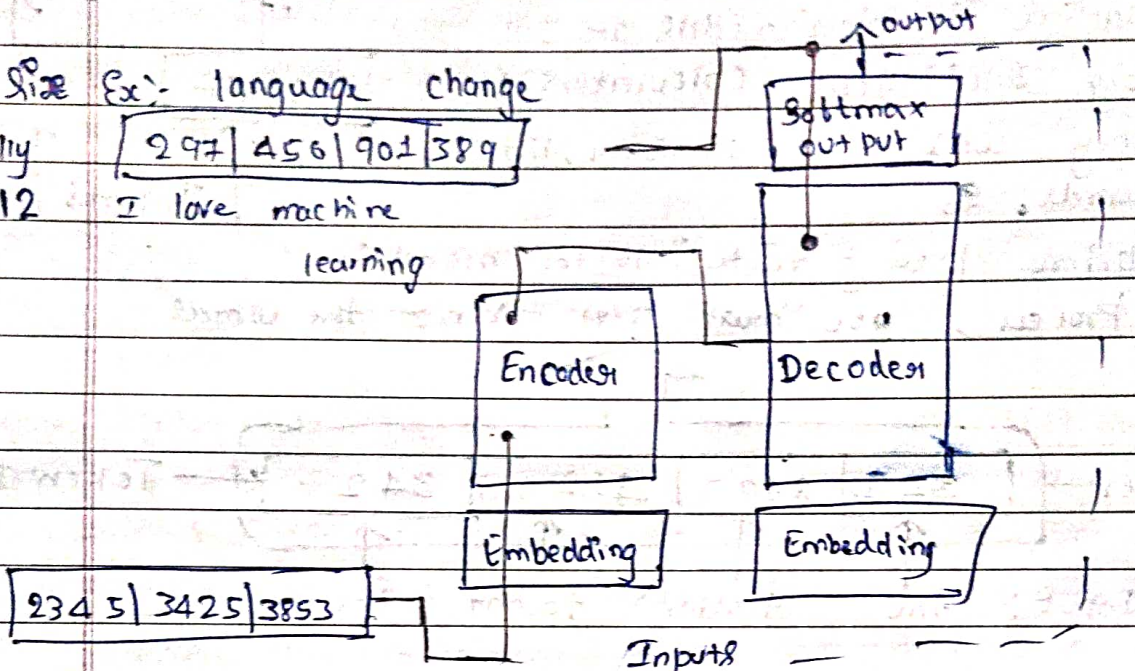


Page No.
Date :

* Each word has been matched to a tokenID, & Each token is mapped into a vector

Vector Size
Origonlly 512

Ex:- language change



J 'aime l' apprentissage automatique

[BERT] Encoder:- Encodes inputs ("Prompts") with contextual understanding and produces one vector per input token. [work as Sequence-to-Sequence models]

[FLAN-T5] Decoder:- Accepts input tokens and generates new tokens.

Que:- Zero shot Inference
One shot Inference → Ex question followed by a second question
Few Shot Inference

Que Top-K sampling is used to limit the random sampling from the K number of top results

Que'

Scope

Define the use case

zero shot → absolutely no labeled data available
one shot → Each new class has one labeled ex
few shot → limited no of labeled examples for each new class

Page No.
Date:

Que: Temp: It is used to affect the randomness of the output of the softmax layer. A lower temp results in reduced variability while a higher temp results in ↑ randomness of the output.

Generative AI Project lifecycle

Scope	Select	Adapt & align model		Application integration	
Define the use case	Choose an existing model or pretrain your own	Prompt Engineering Fine-tunning Align with human feedback	Evaluate	Optimise & deploy model for inference	Augment model & build LLM-powered applications

Python libraries

- torchdata → It helps with the data loading and some other aspects for PyTorch specific to data-sets
- Transformers → This is a library from Hugging Face, (open source tool for LLMs)
- Datasets → It can load in many of the common public data-sets that people use to either train models fine tune models or just Experimen

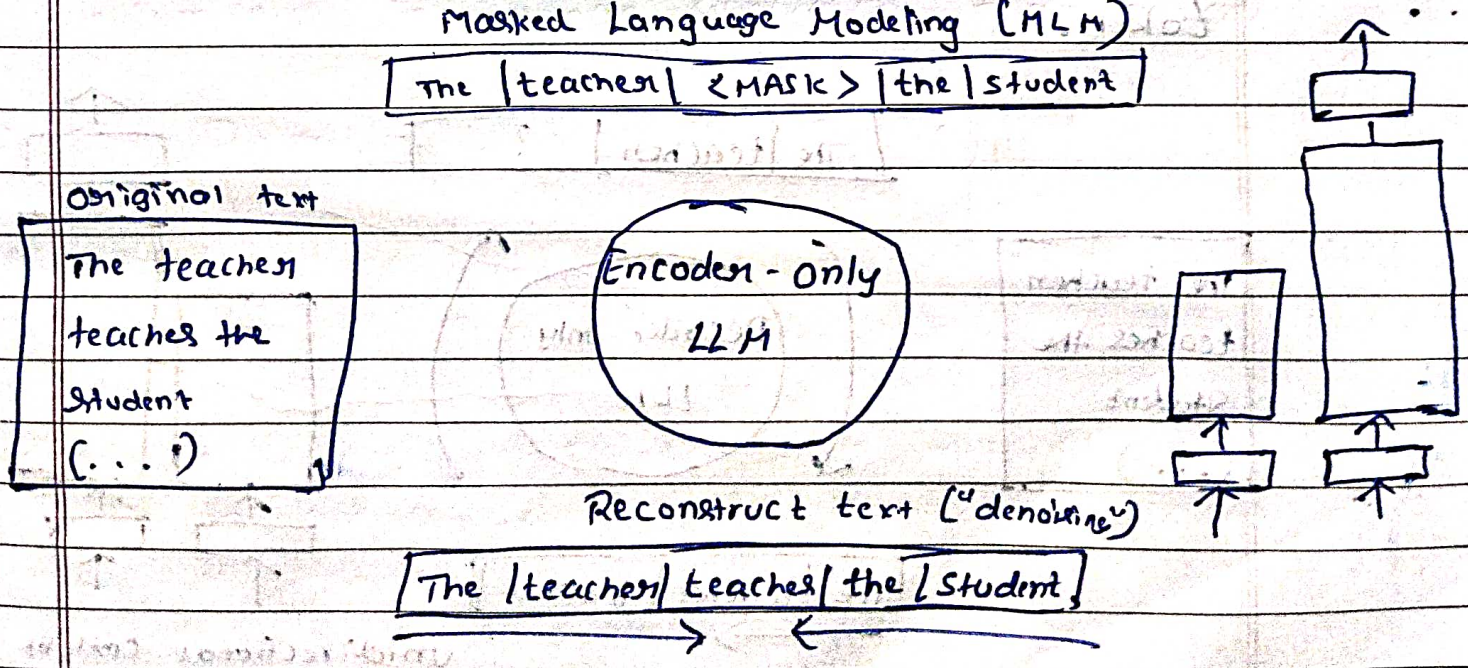
Tokenizers are all came from the Hugging Face of library

Page No.
Date :

AutoEncoding Models :-

The Encoder-only variants of Transformers are also called autoEncoding models.
They are pre-trained using Masked Language Modeling (MLM). In MLM tokens in the input sequence are randomly masked and the training objective is to predict the masked tokens in order to reconstruct the original input sequence.

This is also called a denoising objective since the masking of the tokens can be thought of as adding noise to the input sequence and then predicting the masked tokens can be thought of as removing that noise from the input sequence.



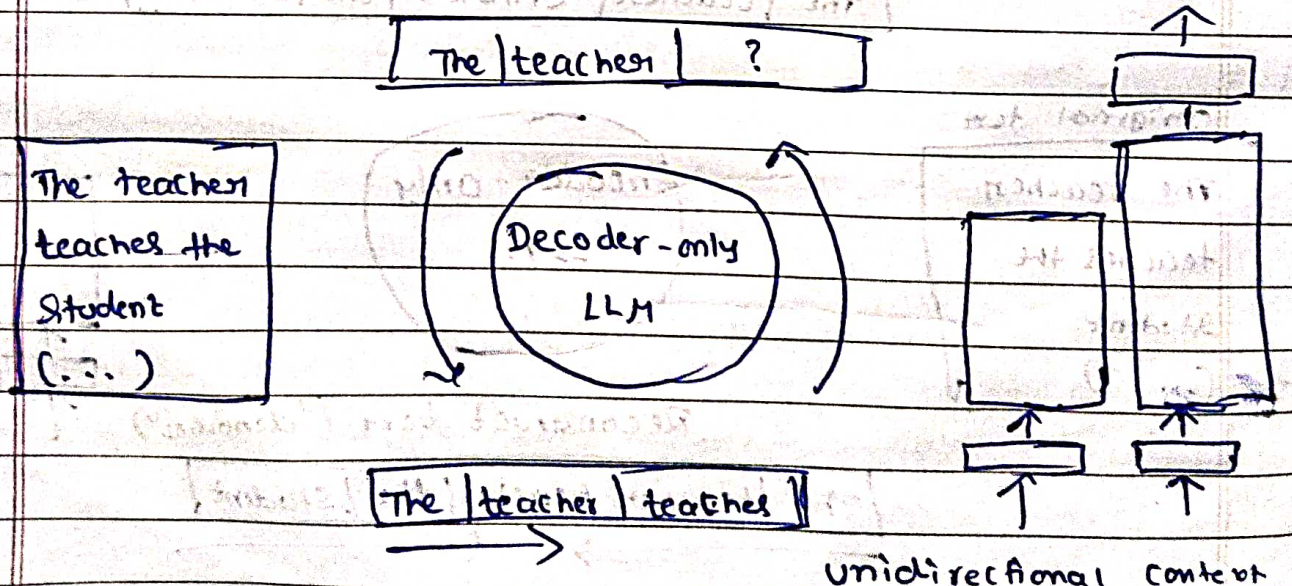
Example:- BERT, ROBERTA

Page No.
Date :

Decoder-only Models (Autoregressive Models)

The decoder-only variants of Transformers are also called autoregressive models.

They are pre-trained using causal Language Modeling (CLM). In CLM, the training objective is to predict the next token based on the previous sequence of tokens. The tokens of the input sequence are masked and the model can only see the input tokens leading up to the token being predicted at the moment. The model has no knowledge of the tokens that come after this token. The model then iterates over the input sequence one-by-one to predict the next token. Thus, in contrast to autoEncoding models, the model builds a unidirectional context for each token.



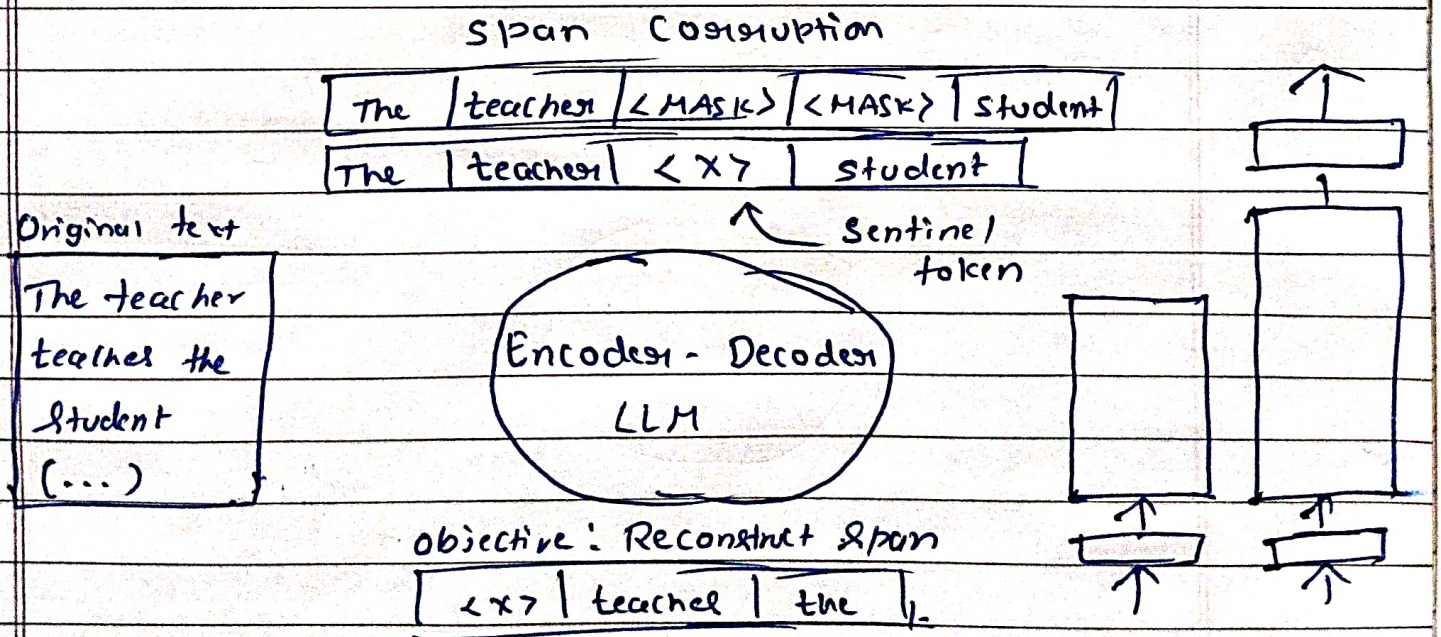
Ex: GPT, BLOOM

Page No.
Date :

Encoder - Decoder Models (Sequence - to - Sequence Models)

The Encoder - decoder variants of Transformers are also called Sequence - to - Sequence models.

The Exact details of pre - training objective vary from model to model. For Ex, FLAN - T5 is trained using Span corruption. In Span corruption, a part of the input sequence is masked and replaced by a Sentinel token. these Sentinel tokens are special ~~masked~~ tokens added to the vocabulary that to do not correspond to any actual word from the dataset. The decoder then has to reconstruct the sentence autoregressively, the output is the Sentinel token followed by the predicted tokens



Ex:- FLAN - T5, BART

parameter Efficient fine-tuning

Page No.

Date :

Week-2

Fine tunning LLMs with instruction

1) Instruction Fine tunning :- when you have your base model, the thing that's initially Pretrained it's Encoded a lot of really good information,
[It doesn't necessarily know how to be able to respond to our Prompts]
* It helps it to be able to change its behavior to be more helpful for us

2) Parameter Efficient fine-tunning :- So we have a lot of customers that do want to be able to tune for very Specific tasks, very Specific domains, this is a great way to achieve . similar performance results on a lot of tasks.

Fine tuning is a Supervised learning Process where you use a data set of labeled Examples to update the weights of the LLM
. The labeled Examples are Prompt completion Pairs

. Instruction fine-tunning is Particularly good at improving a model's Performance on a variety of tasks

During fine tuning, you Select Prompts from your training data set and Pass them to the LLM, which then generates completions, Next you Compare the LLM completion with the response Specified in the training data.

Catastrophic Forgetting

Page No.
Date :

* one way to mitigate catastrophic forgetting is by using regularization techniques to limit the amount of change that can be made to the weights of the model during training. this can help to preserve the information learned during Earlier training phases & Prevent overfitting to the new data.

* Single task fine Tune
* Multiple task fine Tune

LLM Evaluation - Metrics

Rouge :- used for text Summarization,
- compares a Summary to one or more references Summaries

BLEU Score :- used for text translation
- Compares to human-generated translations

PEFT (Parameter Efficient Fine-Tunning)

It ~~update~~ updates only a small subset of parameters it's more robust against this catastrophic forgetting Effect

* Parameter Efficient Fine-tunning (PEFT) methods specifically attempt to address some of the challenges of Performing full fine training

↳ Storage requirements :- with PEFT, we can change just a small amount of parameters when fine-tuning, so during inference you can combine the original model with the new parameters,

Page No.
Date :

Computational Constraints :- Because most parameters are frozen, we typically only need to train 1% - 2% of original LLM weights, making the training process less expensive

* Catastrophic forgetting :- with PEFT, most parameters of the LLM are unchanged & that helps making it less prone to catastrophic forgetting.

* BLEU :- Evaluation metrix below focuses on precision in matching generated output to the reference text & is used for text translation.

* Multi-task finetuning required separate models for Each task being performed.

* LORA works! - It's decomposes weights into two smaller rank matrices & trains those instead of the full model weights

Scaling model training steps

* The first step in scaling is model training is to distribute large data-sets across multiple GPUs and process these batches of data in parallel.

FML

Submitted by : Karthik.C
Reg. No : 240158016
Assignment -3 : Generative.A.I