Machine Learning and Optimization

Lecture 23

- Decoder Transformer Blocks
- Autoregressive Language Modeling w/ Decoder Transformers
- GPT Models

Sequence-to-Sequence Modeling with Transformers

"Attention is All you Need" 2017

- We have seen that encoder transformer blocks can replace RNNs for obtaining useful contextual representations of input sequences
- This suggests that we can use Encoder transformer blocks as encoders in seq2seq models.
- We can use Decoder Transformer blocks as decoders in seq2seq models
- Key Idea: given an input sequence $X = [x_1, \cdots, x_T]$, with contextual representations $C = [c_1, \cdots, c_T]$, when generating the output sequence $Y = [y_1, \cdots, y_{T'}]$, pay attention to C, as well as the already generated output tokens.

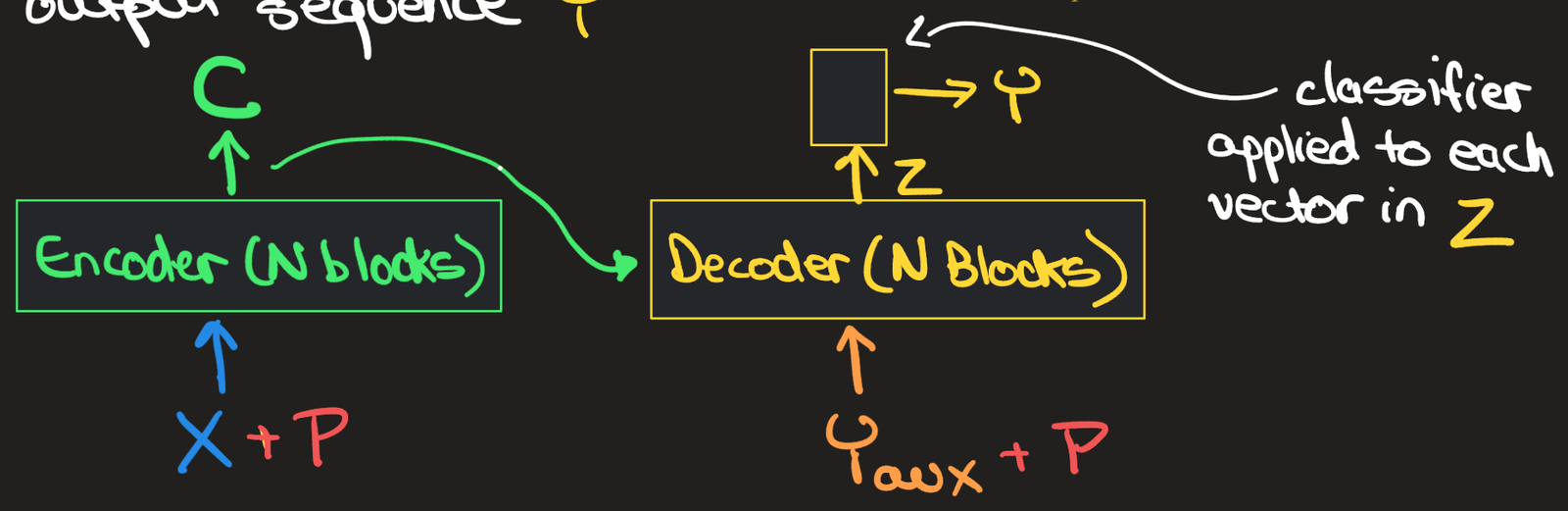
Encoder-Decoder : Training Time

Given the input-output pair of sequences $X = \{x_1, \dots\}$, $Y = \{y_1, \dots\}$ we create a sequence $Y_{aux} = \{\langle BOS \rangle, y_1, \dots, \langle EOS \rangle\}$ that is only used in training. We assume X, Y, Y_{aux} have been padded/truncated to have the transformer's "context length" T. E.g.

$X = \{who, teaches, CSCI, 1200, \langle PAD \rangle\}$
$Y = \{Dr., Uzma, \langle EOS \rangle, \langle PAD \rangle, \langle PAD \rangle\}$ $(T=5)$

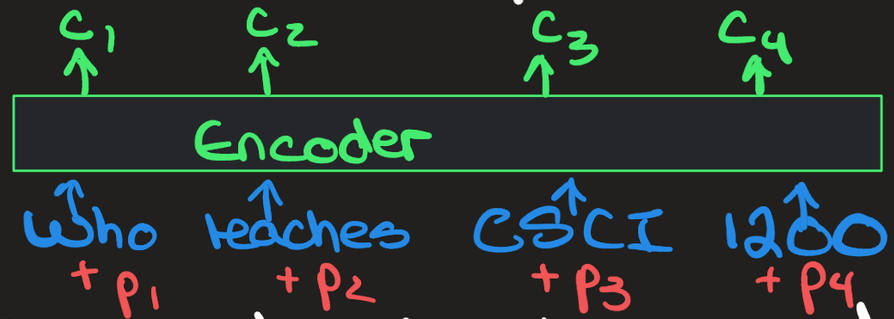
$Y_{aux} = \{\langle BOS \rangle, DR., Uzma, \langle EOS \rangle, \langle PAD \rangle\}$

We train the Encoder-Decoder Transformer to use the output contextual embeddings $Z = Decoder(Y_{aux} | C)$ to predict the correct output sequence Y

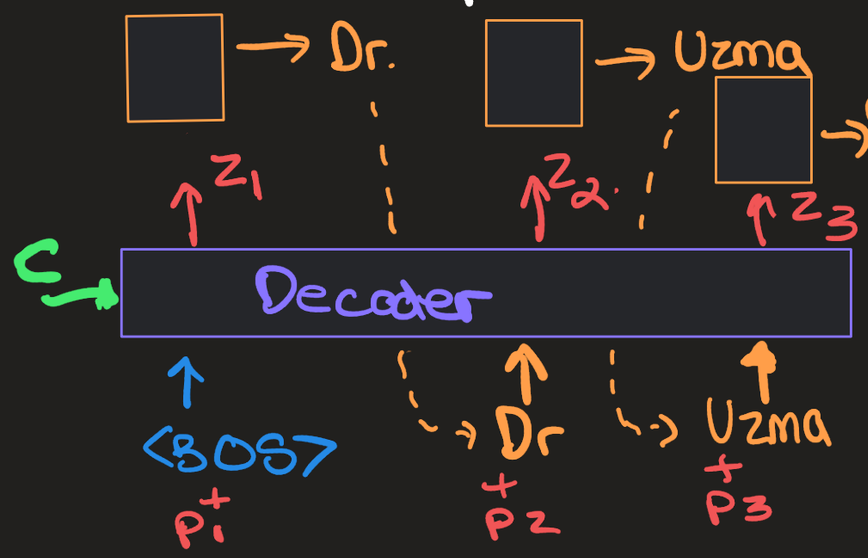


Encoder-Decoder Transformer: Inference Time

Given an input sequence $X = [x_1, \cdots, x_T]$, the encoder-decoder transformer computes $C = \text{Attention}(X)$ in the encoder.



Then the decoder is prompted with the special <BOS> token. It uses self-attention w.r.t. the current output prefix $\{$<BOS>$\}$ and cross-attention w.r.t. C to generate a contextual representation z_1, which is then used in a softmax to predict the next output token.



Then z_2 is generated via self-attention against the output prefix $\{$<BOS>, Dr$\}$ and cross-attention w.r.t. C. Continue until EOS

Cross-Attention

Given two sequences $C = [c_1, \cdots, c_T]$ and $Y = [y_1, \cdots, y_{T'}]$ we can ask what are good contextual representations for Y, given C?

Idea: use cross-attention by generating the keys and values from C, and the queries from Y

$$K = \omega_K C$$
$$Q = \omega_Q Y$$
$$V = \omega_V C$$

$$\text{Attention}(Y, C \mid \omega_K, \omega_Q, \omega_V) = V \,\text{softmax}\left(\frac{K^T Q}{\sqrt{d_{attn}}}\right)$$

Masked Self-Attention

In generating the output $\{y_1 \; y_2 \; y_3 \; y_4$ <BOS> Dr Uzma <EOS>$\}$ the decoder can only pay attention to the portion of the output sequence that has already been generated (causal attention). Thus we use **masked self-attention** in computing the contextual representations of this sequence:

$$(\tilde{\alpha}_i)_j = \begin{cases} \langle k_j, q_i \rangle & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

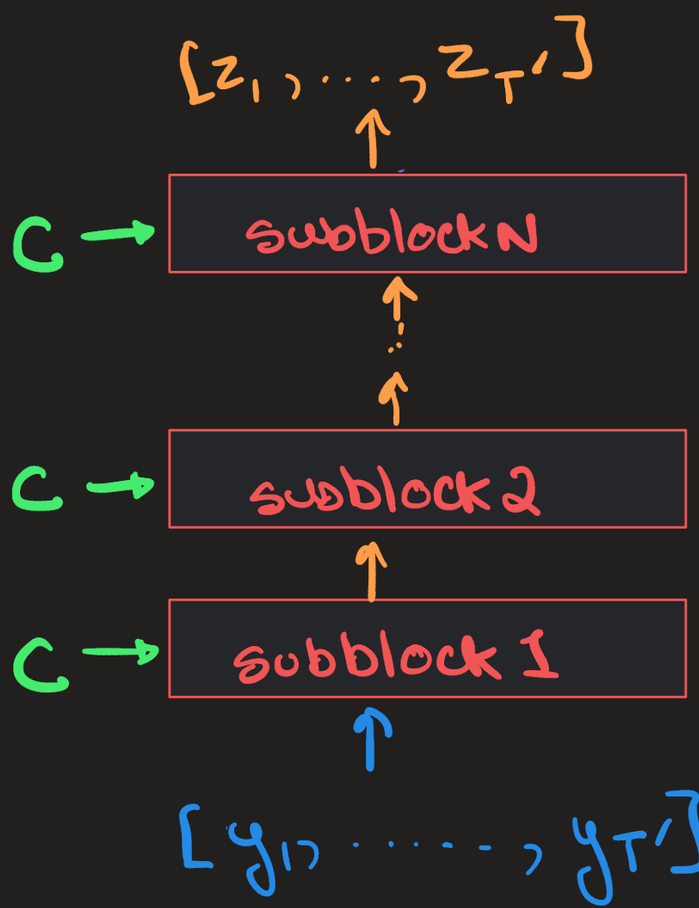
Let $M \in \mathbb{R}^{T' \times T'}$ satisfy $M_{ij} = \begin{cases} -\infty & \text{if } j \leq i \\ 1 & \text{if } j > i \end{cases}$ meaning replace entries where $j > i$ with $-\infty$

$$\text{Attention}\left(\varphi \mid \text{Mask}=M\right) = V \,\text{softmax}\left(\frac{K^T Q \odot M}{\sqrt{d_{attn}}}\right)$$

$$m_{ij} = \begin{cases} 1 & \text{if token } j \text{ follows token } i \\ -\infty & \text{if token } j \text{ preceeds token } j \end{cases}$$

	<BOS>	Dr	Uzma	<EOS>
<BOS>	1	1	1	1
Dr	$-\infty$	1	1	1
Uzma	$-\infty$	$-\infty$	1	1
<EOS>	$-\infty$	$-\infty$	$-\infty$	1

Decoder Block



Here M is the causal masking matrix

Similar to an encoder block, but each subblock uses causal self-attention and cross-attention against C:

i) $Z' = \text{LayerNorm}(Y + \text{MHSA}(Y \mid \text{Mask} = M))$

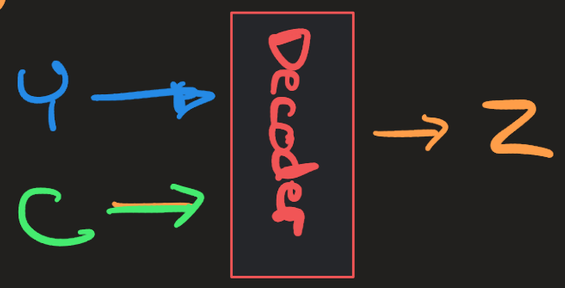
ii) $Z'' = \text{LayerNorm}(Z' + \text{MHA}(Z', C))$

iii) $Z = \text{LayerNorm}(Z'' + \text{FF}(Z''))$

where $\text{FF}(Z'') = [\text{FF}(z''_1) \mid \cdots \mid \text{FF}(z''_{T'})]$

for $\text{FF}(z) = W^2 \sigma(W^1 z + b^1) + b^2$

and $\sigma = \text{ReLU}$



GPT models

Generative **P**re-trained **T**ransformers

- Generative: model can generate text, given a prompt
- Pre-trained: trained in an unsupervised manner on a large corpus, to complete texts
- Transformers: uses decoder-only transformers

open source ↑ / ↓ closed source

		(parameter count)	(main contrib)
2018	GPT-1	117M	decoder-only LLM, autoregressive generation
2019	GPT-2	1.5B	trained on larger dataset zero shot learning
2020	GPT-3	175 B	trained on larger dataset in context learning
2022	GPT-3.5	1.3/6/175B	RLHF for alignment
2023	GPT-4	1 Trillion	text & visual input, text output

Autoregressive Language Modeling

The goals of language modeling are to:

i) learn an accurate model for sequences of tokens in a "language"

$$p_\theta(x_1, \dots, x_T)$$

ii) to be able to sample from this probability distribution to complete passages

Autoregressive LMs use the chain rule

$$p_\theta(x_1, \dots, x_T) = p_\theta(x_T | x_1, \dots, x_{T-1}) p_\theta(x_1, \dots, x_{T-1})$$

$$= p_\theta(x_T | x_{1:T-1}) p_\theta(x_{T-1} | x_{1:T-2}) p_\theta(x_1, \dots, x_{T-2})$$

$$= \dots = \prod_{i=1}^{T} p_\theta(x_i | x_{1:i-1})$$

Then the parameters θ for the LM can be learned by minimizing the NLL over the observed sequences

$$\mathcal{L}(\theta) = \frac{1}{T}\sum_{i=1}^{T} \log p_\theta(x_i | x_{1:i-1})$$

The GPT-1 model used a decoder-transformer to model $p_\theta(x_i | x_{1:i-1})$. The cross-attention subblock is dropped.