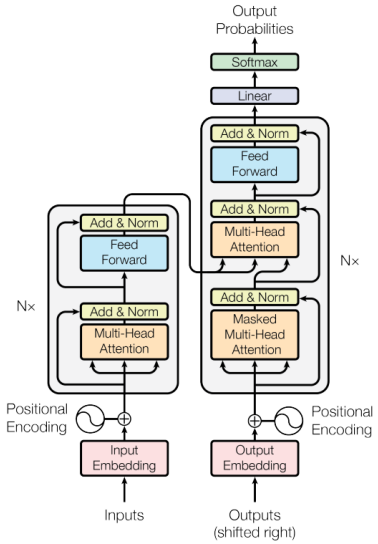
Transformer

The Decoder

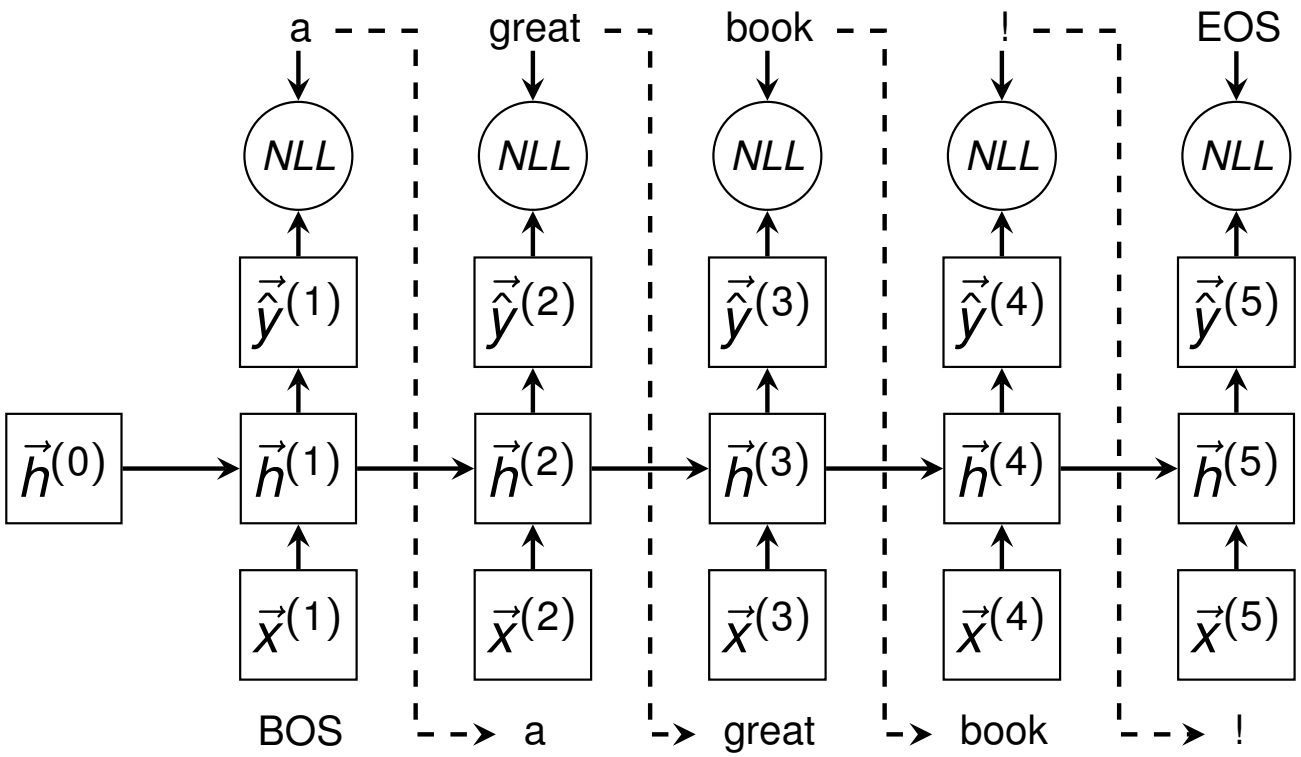


Learning goals

- Understand Masked Self-Attention and the role of causality in decoding
- Understand the connection between the encoder and the decoder

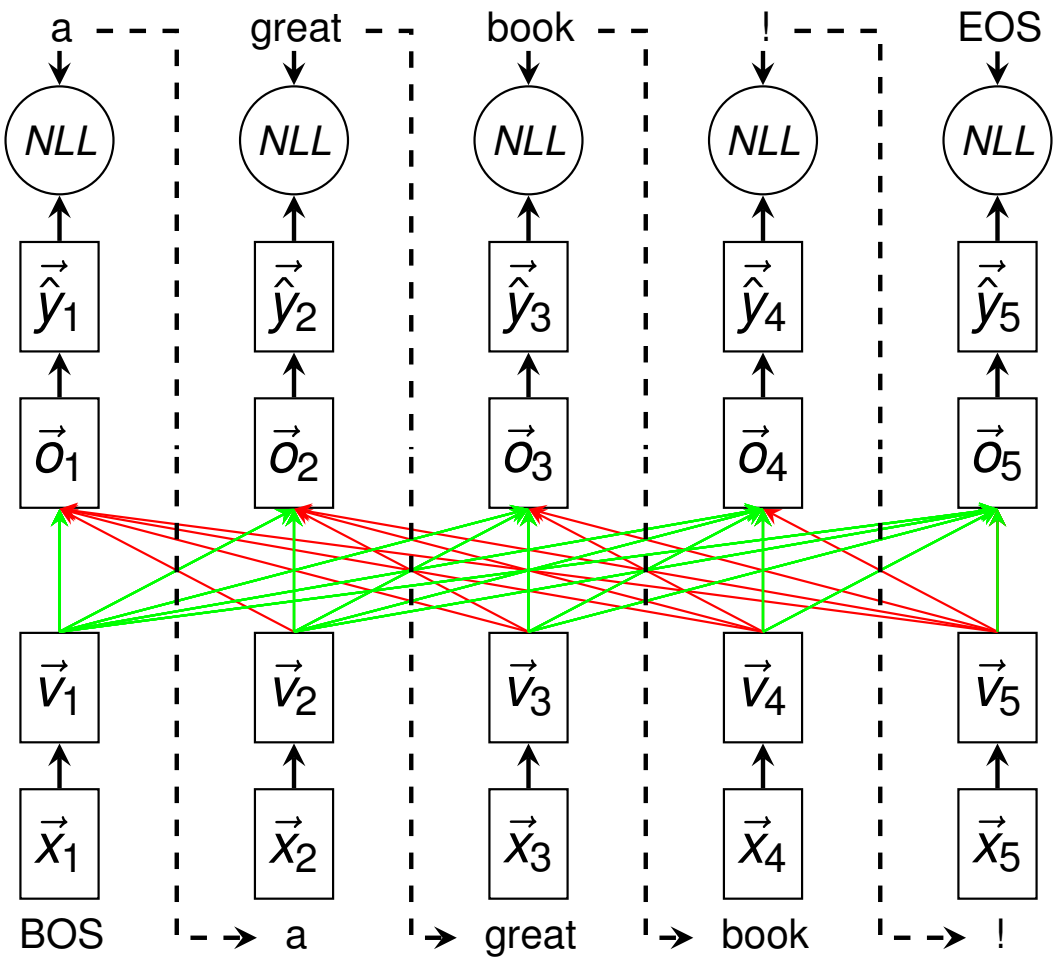
RNNS FOR AUTOREGRESSIVE LM & DECODING

- In autoregressive language modeling, or in the decoder of a sequence-to-sequence model, the task is to always predict the next word
- In an RNN, a given state $\vec{h}^{(j)}$ depends on past inputs $x^{(1)} \ldots x^{(j)}$
- Thus, the RNN is unable to "cheat":



SELF-ATTENTION FOR AR LM & DECODING

- With attention, all $\vec{o}_j$ depend on all $\vec{v}_{j'}$ (and by extension, all $\vec{x}_{j'}$).
- This means that the model can easily cheat by looking at future words (red connections)



MASKED SELF-ATTENTION

- So when we use self-attention for language modeling or in a sequence-to-sequence decoder, we have to prevent $\vec{o}_j$ from attending to any $\vec{v}_{j'}$ where $j' > j$.
- **Question:** How can we do that?
- Remember:

$$\vec{o}_j = \sum_{j'=1}^{J} \alpha_{j,j'} \vec{v}_{j'}$$

$$\alpha_{j,j'} = \frac{\exp(e_{j,j'})}{\sum_{j''=1}^{J} \exp(e_{j,j''})}$$

  - By hardcoding $e_{j,j'} = -\infty$ when $j' > j$ (in practice, "∞" is just a large constant)
  - That way, $\exp(e_{j,j'}) = \alpha_{j,j'} = 0$, so $\vec{v}_{j'}$ has no impact on $\vec{o}_j$

PARALLELIZED MASKED SELF-ATTENTION

- Step 1: Calculate $\vec{E}$ like we usually would
- Step 1B:

$$\vec{E}^{\text{masked}} = \vec{E} \odot \vec{M} + \infty\vec{M} - \infty; \qquad m_{j,j'} = \begin{cases} 1 & \text{if } j' \leq j \\ 0 & \text{otherwise} \end{cases}$$

- Example:

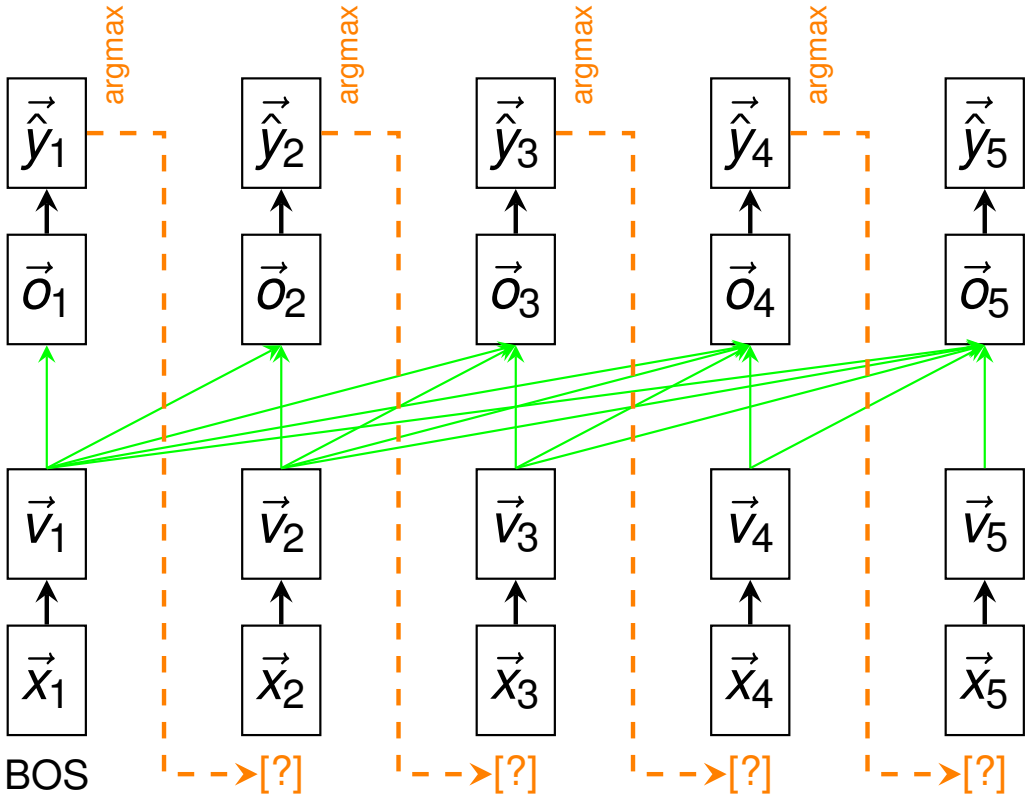
$$\vec{E} = \begin{bmatrix} e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix}; \qquad \vec{M} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\vec{E}^{\text{masked}} = \begin{bmatrix} e_{1,1} & -\infty & -\infty \\ e_{2,1} & e_{2,2} & -\infty \\ e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix}; \qquad \vec{A}^{\text{masked}} = \begin{bmatrix} 1 & 0 & 0 \\ \alpha_{2,1} & \alpha_{2,2} & 0 \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{bmatrix}$$

$$\vec{o}_1 = \vec{v}_1; \qquad \vec{o}_2 = \alpha_{2,1}\vec{v}_1 + \alpha_{2,2}\vec{v}_2; \qquad \vec{o}_3 = \alpha_{3,1}\vec{v}_1 + \alpha_{3,2}\vec{v}_2 + \alpha_{3,3}\vec{v}_3$$

AR TRANSFORMER AT INFERENCE TIME

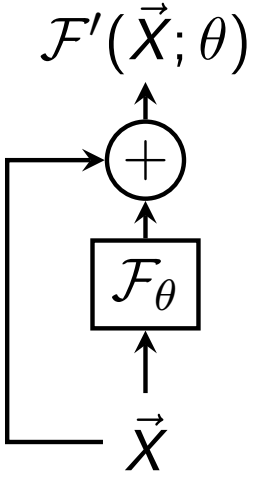
- During training (targets known): Use parallelized masked attention
- At inference time (targets unknown): Decode prediction in a loop
- Slower, but at least we don't have to worry about masking anymore



ADD. SUBTLETIES: RESIDUAL CONNECTIONS

- Let $\mathcal{F}$ be a function with parameters θ
- $\mathcal{F}$ with a residual connection:

$$\mathcal{F}'(\vec{X};\theta) = \mathcal{F}(\vec{X};\theta) + \vec{X}$$



- Benefits: Information retention (we add to $\vec{X}$ but don't replace it)

ADD. SUBTLETIES: LAYER NORMALIZATION

- Let $\theta = \{\vec{\gamma} \in \mathbb{R}^d, \vec{\beta} \in \mathbb{R}^d\}$ be trainable parameters
- Let $\vec{h} \in \mathbb{R}^d$ be an output vector of some layer (e.g., an $\vec{o}_j$ vector from an attention layer)
- Then layer normalization calculates:

$$\vec{\gamma} \odot \frac{\vec{h} - \mu}{\sigma} + \vec{\beta}$$

- where μ, σ are mean and standard deviation over the dimensions of $\vec{h}$:

$$\mu = \frac{1}{d} \sum_{i=1}^{d} h_i; \qquad \sigma = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (h_i - \mu)^2}$$

- Benefits: Allows us to normalize vectors after every layer; helps against exploding activations on the forward pass
- In the Transformer, layer normalization is applied position-wise, i.e., every $\vec{o}_j$ is normalized independently

ENCODER-DECODER ATTENTION

Open question: How do we connect encoder and decoder?

Construction of one decoder block:

1. Masked (Multi-Head) Attention layer (only target sequence)
2. "Ordinary" (Multi-Head) Attention layer
   - Queries from the previous decoder layer
   - Keys, Values from the encoder output
3. Feed-Forward layer (w/ residual connections & layer norm)

$\rightarrow$ Allows the decoder to attend to *all* tokens from the input sequences (cf. Bahdanau et al. (2014) for RNNs)

THE TRANSFORMER ARCHITECTURE

(For simpler problems (e.g., classification, tagging), you would simply use the encoder.)

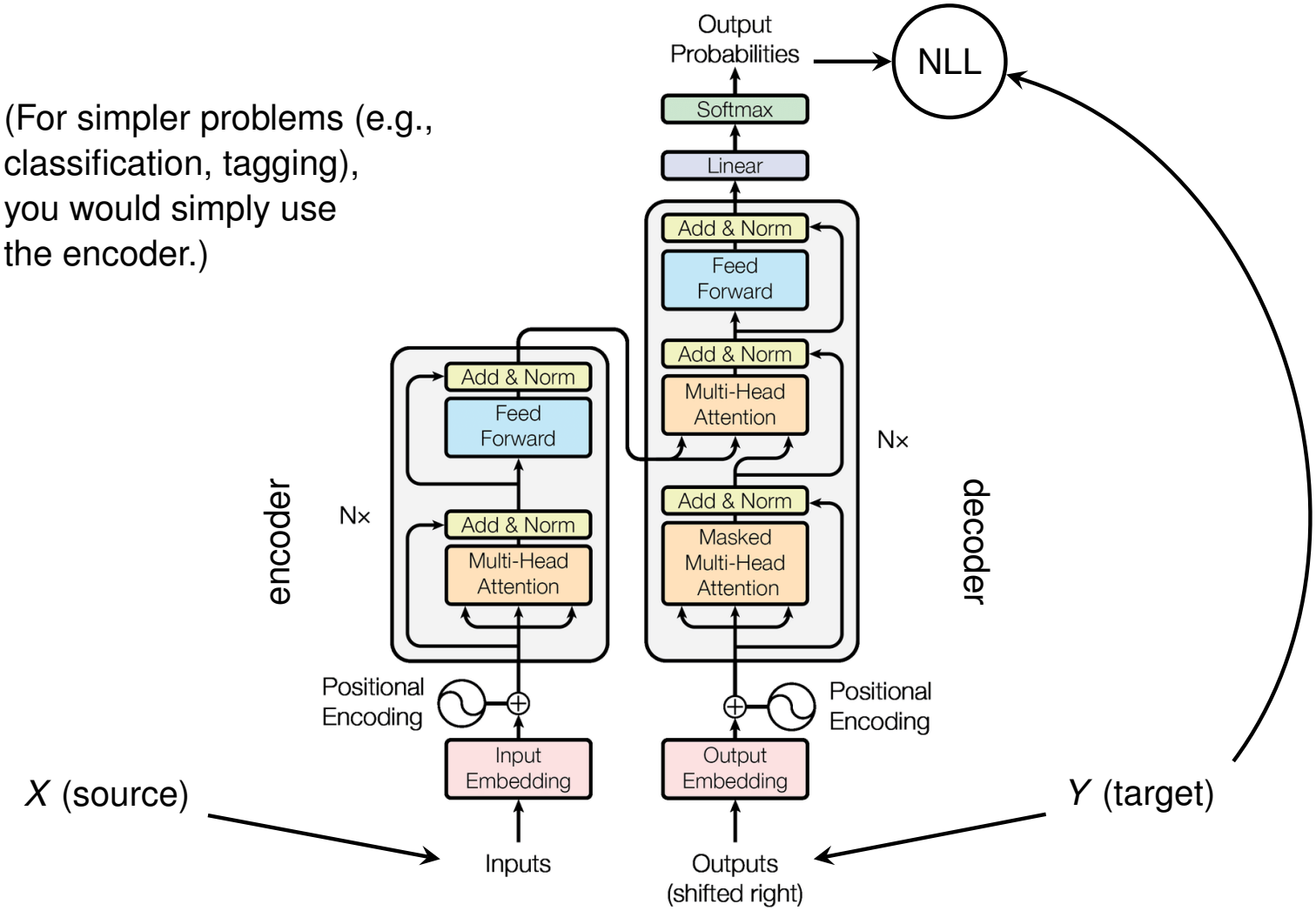


Figure from Vaswani et al. 2017: Attention is all you need