Natural Language Processing

Large Language Models

Language Modeling

Recap: What is a language model?

- Language models assign a probability to a sequence of words

$$p(\overline{y})$$

- We can decompose this probability using the chain rule

$$p(\overline{y}) = \Pi_{i=1}^{T} p(y_i | y_{0:i-1})$$

- We can autoregressively generate sequences from the language model by sampling from its token-level probability

$$p(y_i | y_{0:i-1})$$

- We can condition on our language distribution on something else

$$p(y_i | y_{0:i-1}; \overline{x})$$

What can we do with language models?

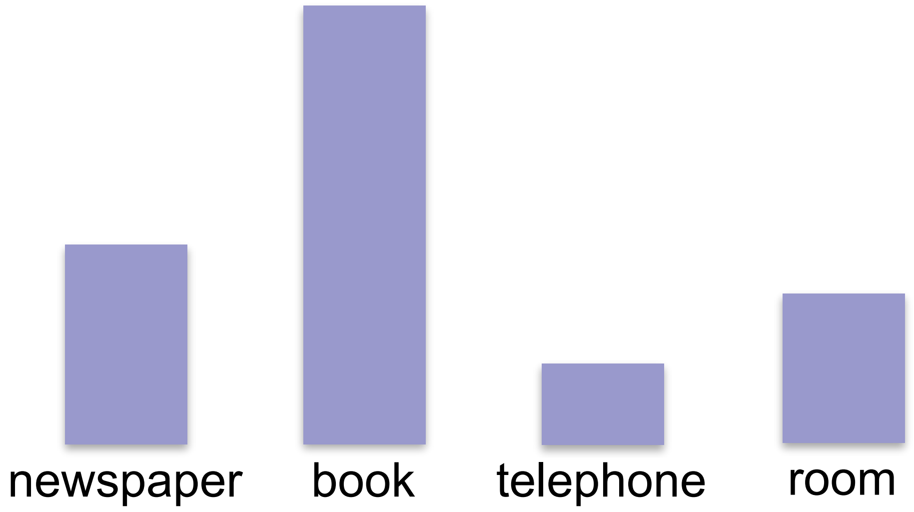
- Computing probabilities of a sequence

- Autoregressive sequence generation

Decoding strategies

- Argmax (greedy decoding)

$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

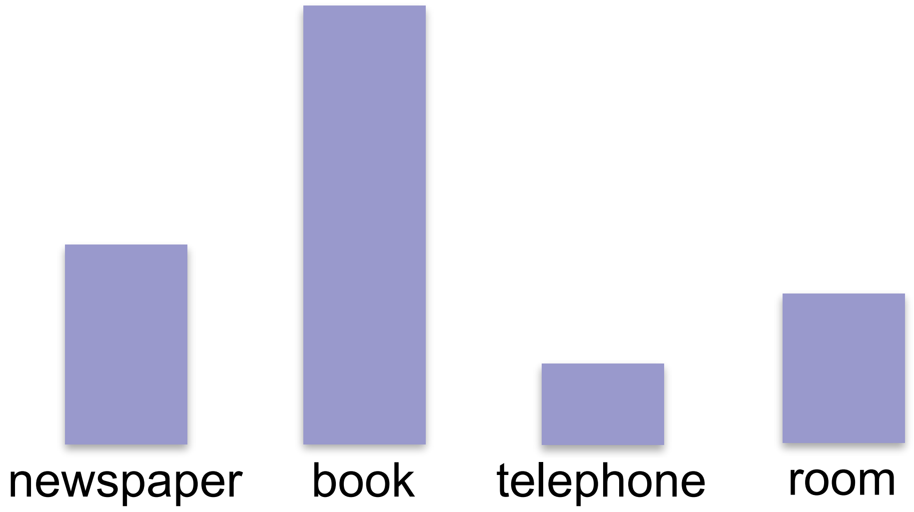


Decoding strategies

- Argmax (greedy decoding)
- Sampling from language model directly

$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

$$y_T \sim p(\cdot \mid y_{0:t-1})$$



Decoding strategies

- Argmax (greedy decoding)

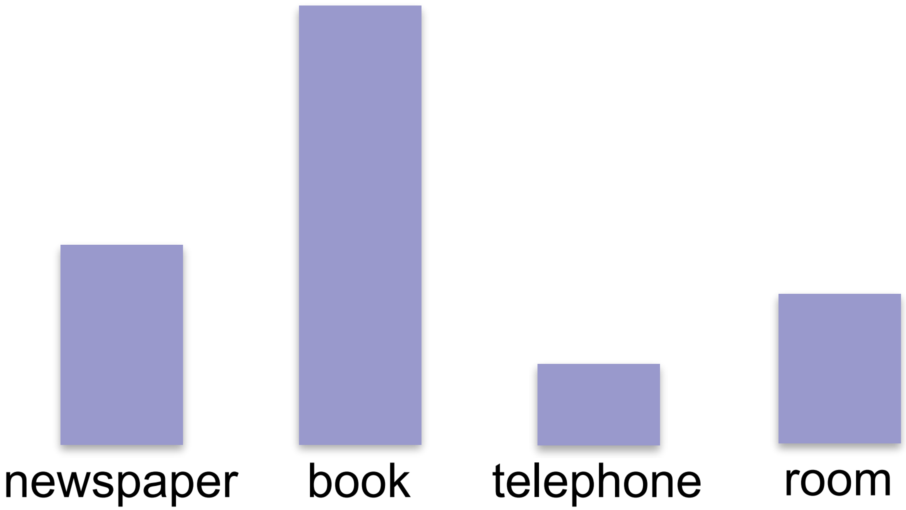
$$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

- Sampling from language model directly

$$y_T \sim p(\cdot \mid y_{0:t-1})$$

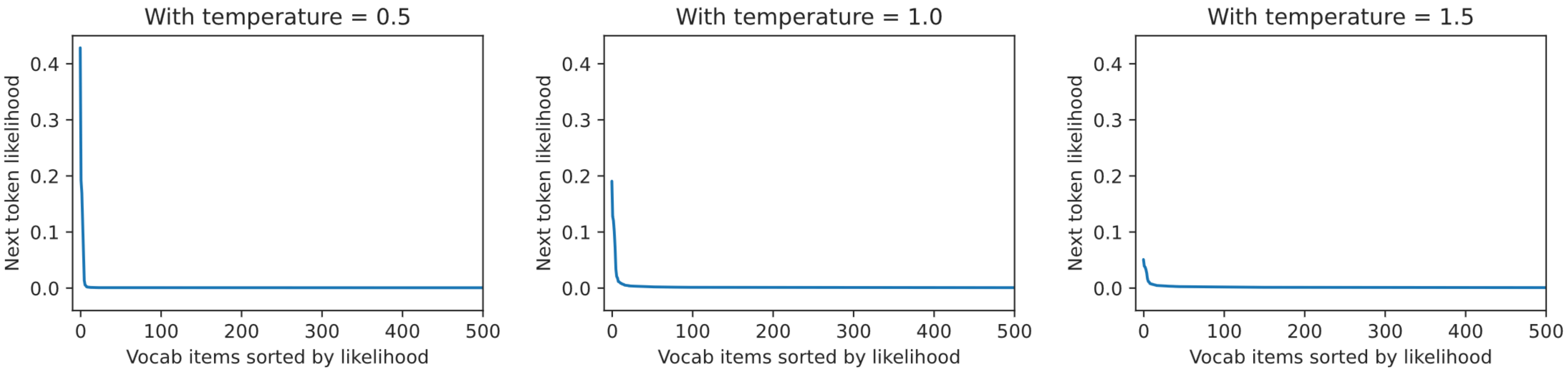
- Adjusting temperature of distribution

$$p'(y_T = y) = \frac{\exp(z_y/T)}{\sum_{y' \in \mathcal{V}} (z_{y'}/T)}$$



Decoding strategies

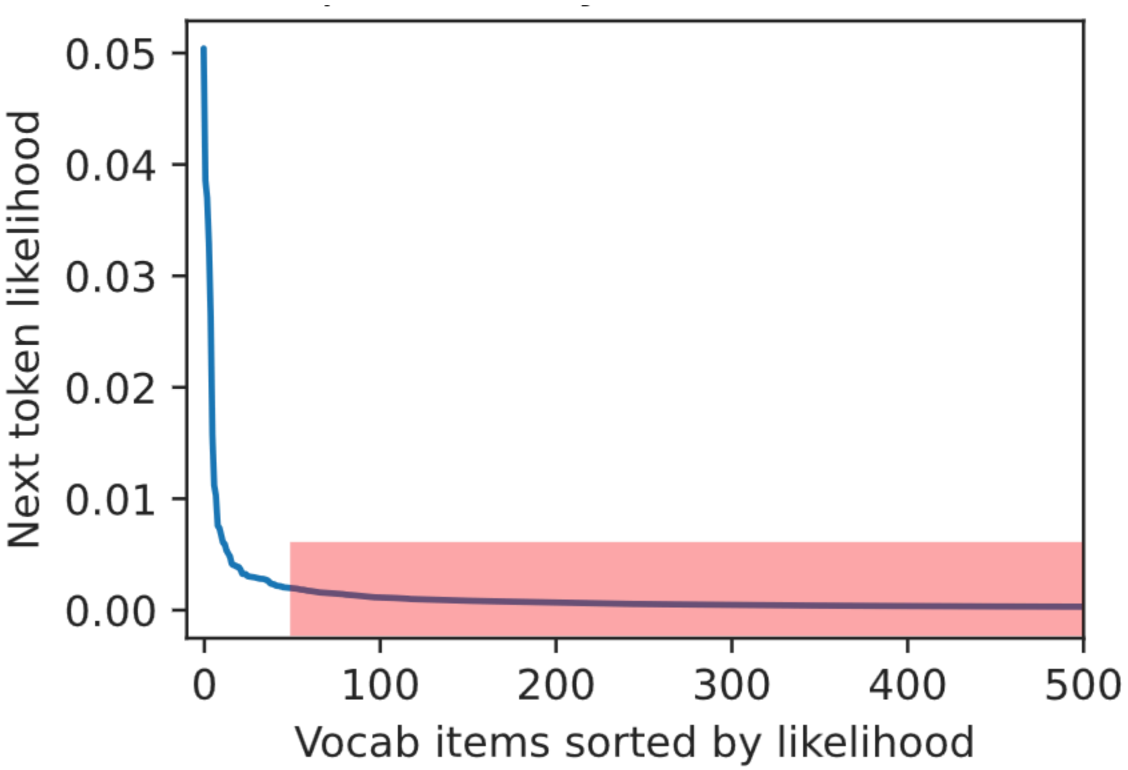
- Argmax (greedy decoding) $$y_T = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$
- Sampling from language model directly $$y_T \sim p(\cdot \mid y_{0:t-1})$$
- Adjusting temperature of distribution $$p'(y_T = y) = \frac{\exp(z_y/T)}{\sum_{y' \in \mathcal{V}} (z_{y'}/T)}$$



Slide from Daphne Ippolito / Chenyan Xiong, CMU LLMs course http://cmu-llms.org/

Decoding strategies

- Top-k sampling: reassign probability mass from all but the top *k* tokens to the top *k* tokens



Slide from Daphne Ippolito / Chenyan Xiong, CMU LLMs course http://cmu-llms.org/

Decoding strategies

- Nucleus sampling: reassign probability mass to the most probable tokens whose cumulative probability is at least *p*

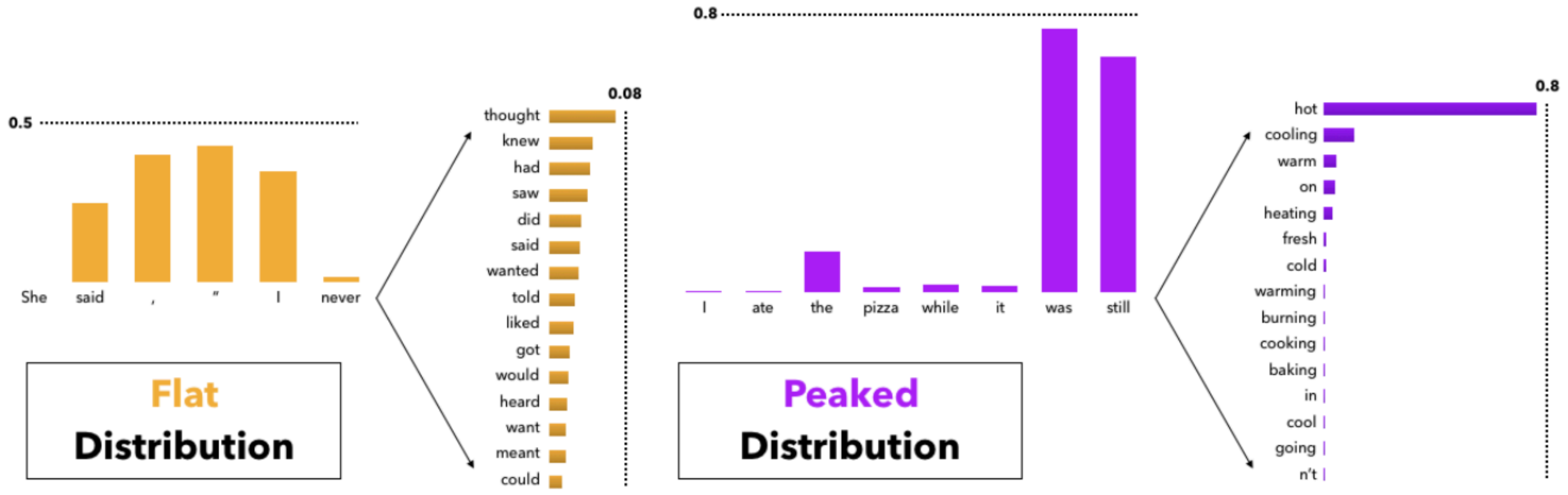


Figure 5: The probability mass assigned to partial human sentences. Flat distributions lead to many moderately probable tokens, while peaked distributions concentrate most probability mass into just a few tokens. The presence of flat distributions makes the use of a small k in top-k sampling problematic, while the presence of peaked distributions makes large k's problematic.

Holtzman et al. 2020, ICLR

Beam search

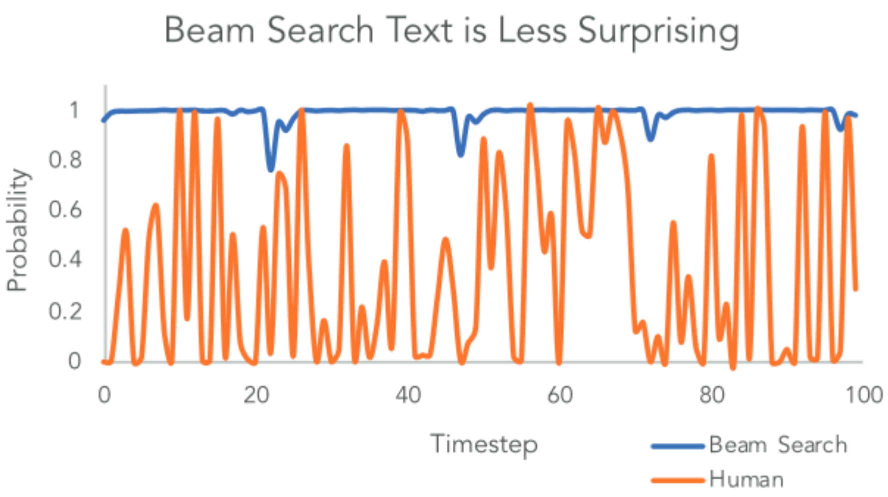
- It's intractable to find the *most probable sequence* according to a language model
- Greedy search doesn't yield the most probably sequence
- Instead: beam search
  - Approximate the search by keeping around candidate continuations
  - At the end, choose the highest probability sequence in the beam

$$\overline{y}^* = \arg\max_{\overline{y} \in \mathcal{V}^*} p(\overline{y})$$

$$y_t = \arg\max_{y \in \mathcal{V}} p(y \mid y_{0:t-1})$$

Beam search

- But do we even want to find the highest-probability sequence according to a LM?
- Human language is noisy and surprising
- Optimizing for LM probability leads to repetitive and uninteresting text



Beam Search	**Human**
...to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and...	...which grant increased life span and three years warranty. The Antec HCG series consists of five models with capacities spanning from 400W to 900W. Here we should note that we have already tested the HCG-620 in a previous review and were quite satisfied With its performance. In today's review we will rigorously test the Antec HCG-520, which as its model number implies, has 520W capacity and contrary to Antec's strong beliefs in multi-rail PSUs is equipped...

Holtzman et al. 2020, ICLR

Beam search

- But do we even want to find the highest-probability sequence according to a LM?
- Human language is noisy and surprising
- Optimizing for LM probability leads to repetitive and uninteresting text

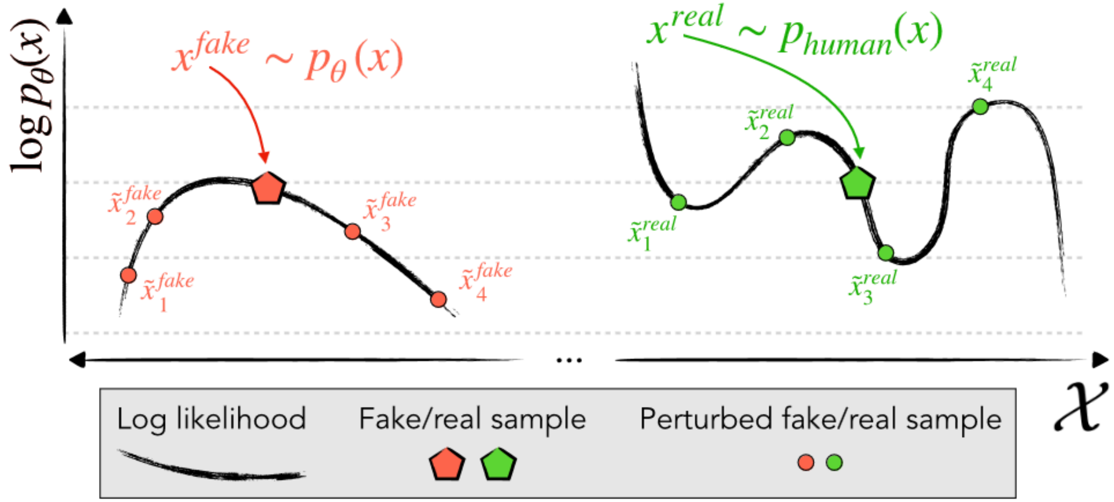


Figure 2. We identify and exploit the tendency of machine-generated passages $x \sim p_\theta(\cdot)$ **(left)** to lie in negative curvature regions of $\log p(x)$, where nearby samples have lower model log probability on average. In contrast, human-written text $x \sim p_{real}(\cdot)$ **(right)** tends not to occupy regions with clear negative log probability curvature; nearby samples may have higher or lower log probability.

DetectGPT, Mitchell et al. 2023, ICML

Recap: Feedfoward Networks

- Tokenize
- Embed
- Concatenate
- Linear layer
- Softmax
- Fixed window?
- Word averaging?

output distribution

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

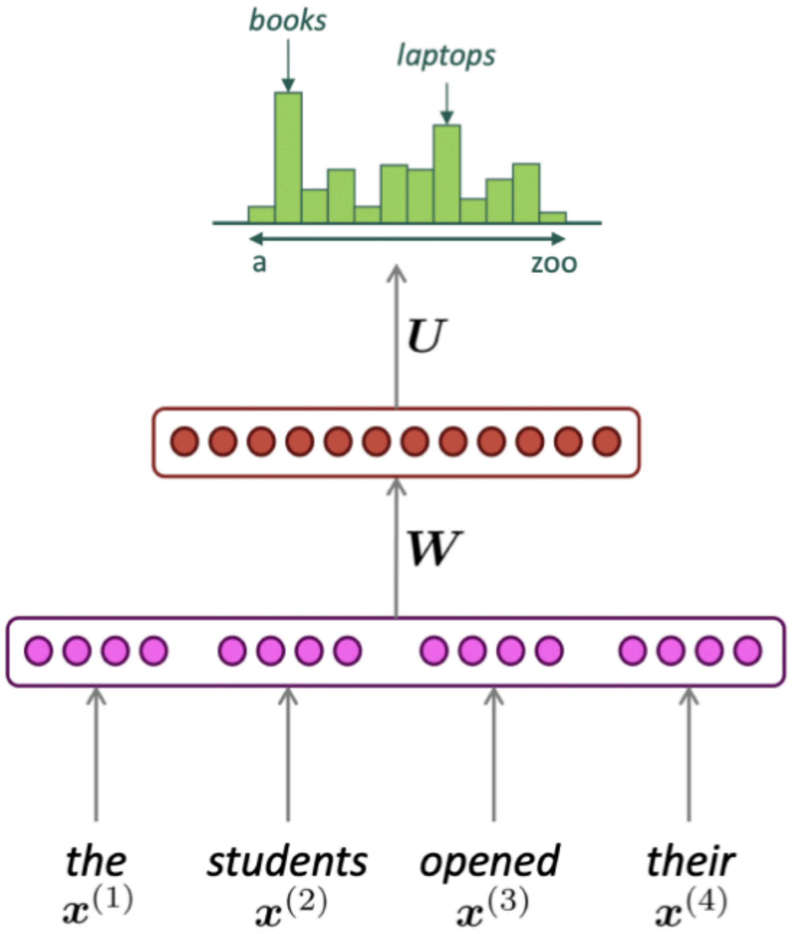
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



Slide from Stanford CS224

Recap: Recurrence

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

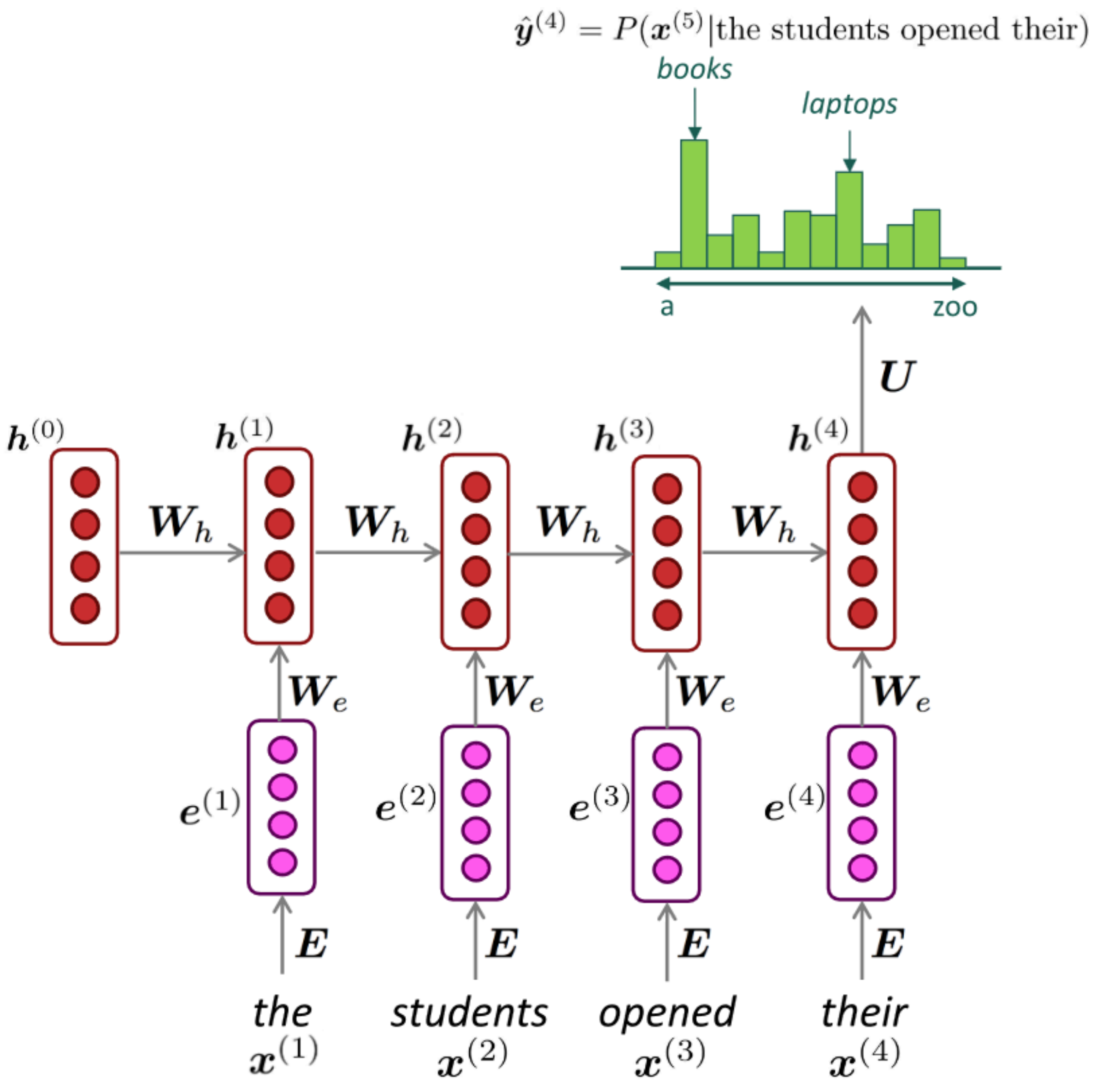
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

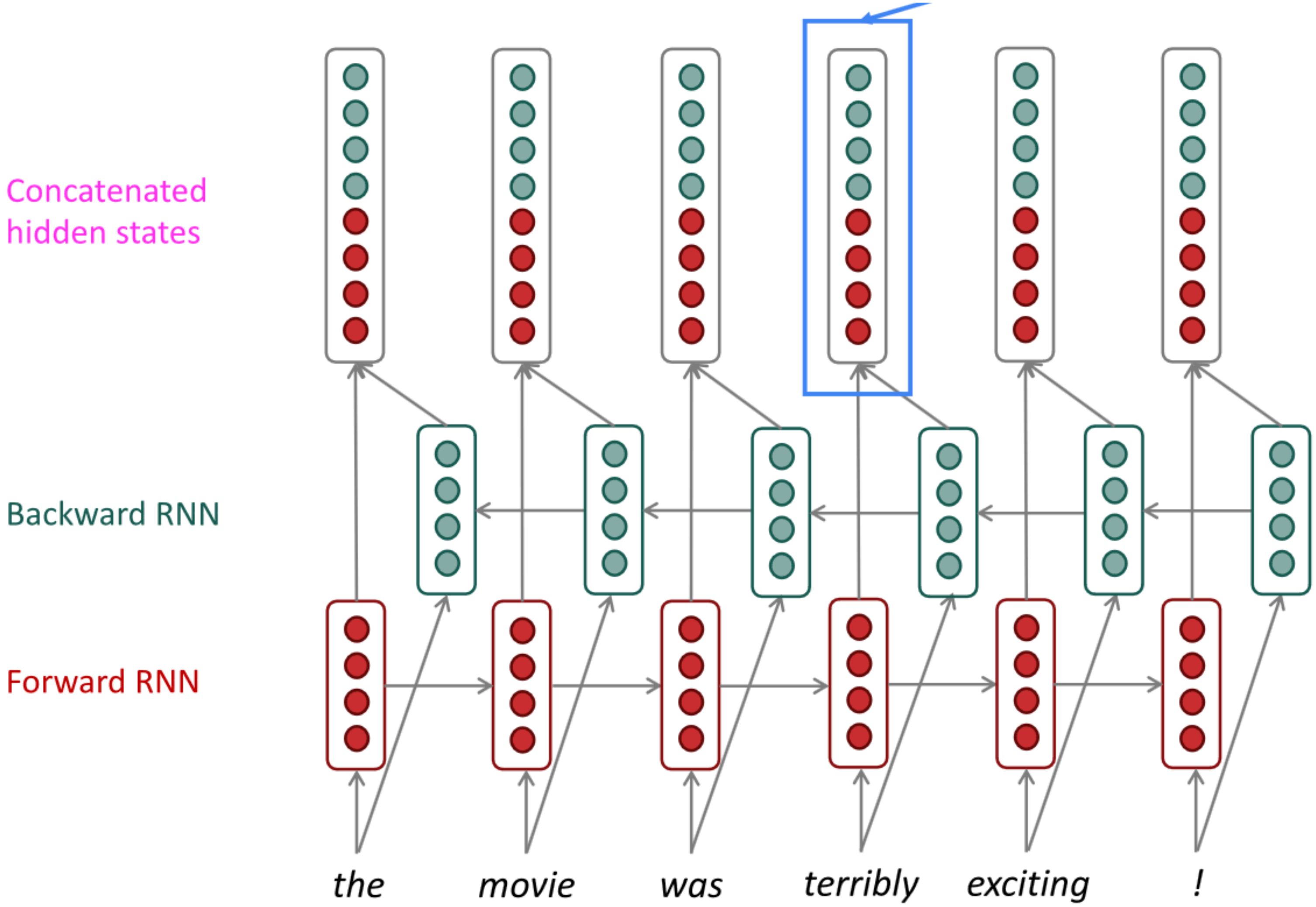
words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



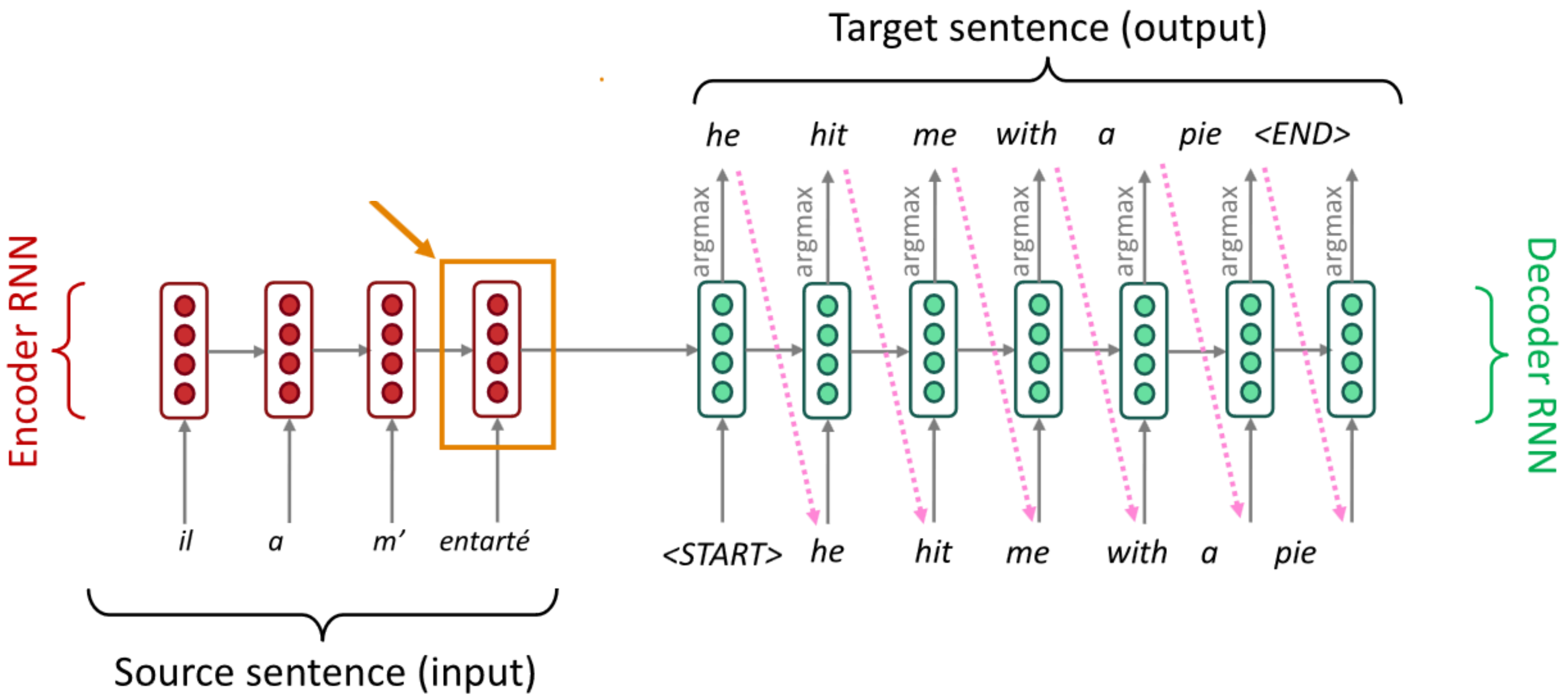
Slide from Stanford CS224

Recap: Recurrence



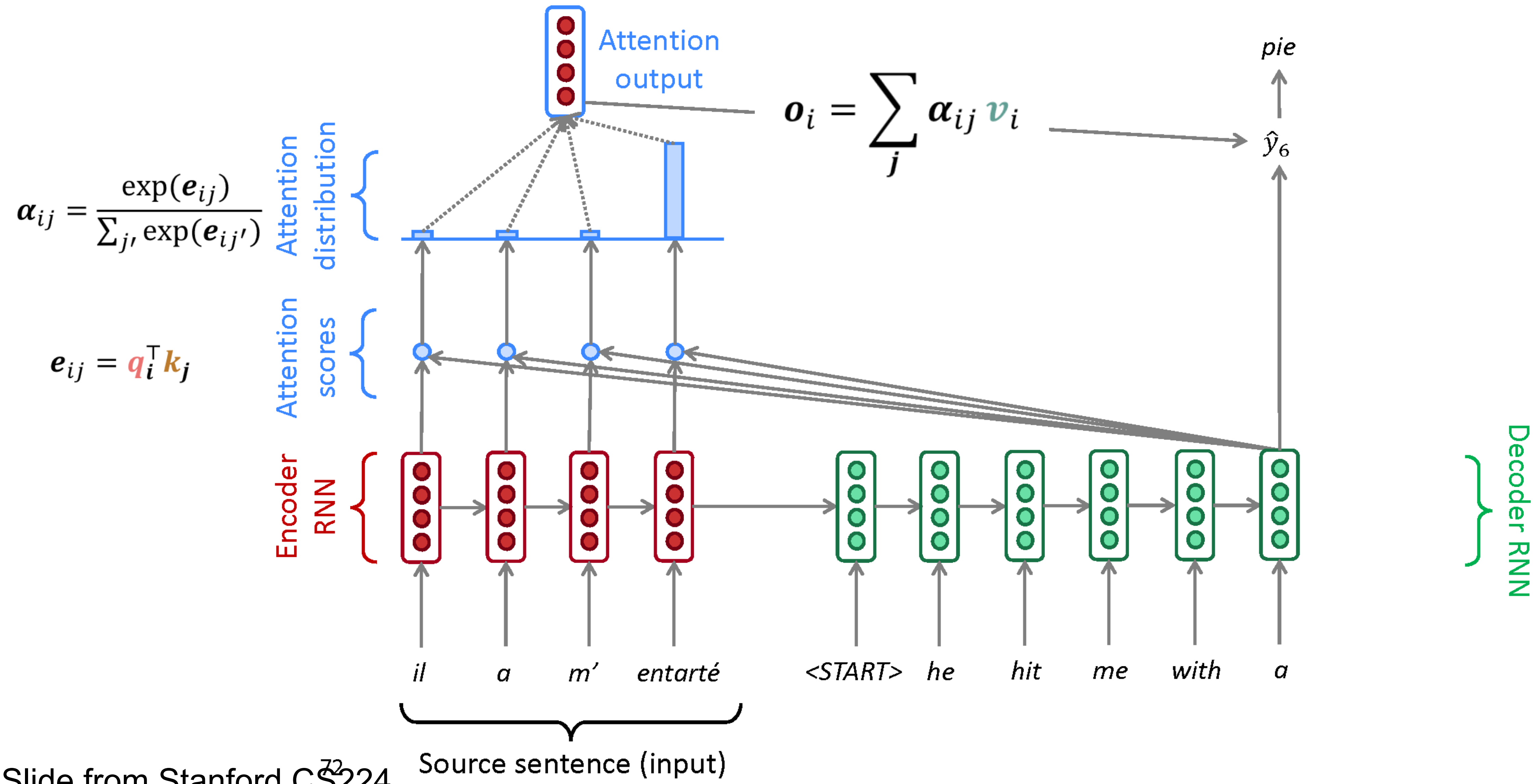
Slide from Stanford CS224

Recap: Recurrence



Slide from Stanford CS224

Recap: Attention



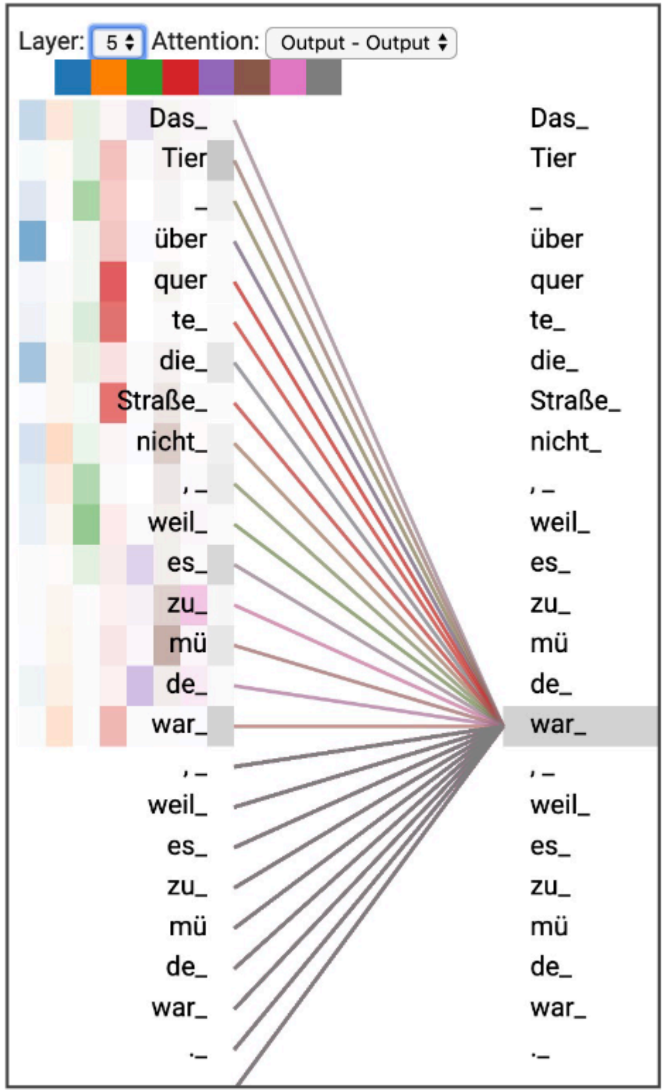
Slide from Stanford CS224

Recap: Attention

- Generic dot-product attention:

$$e_{ij} = q_i^\top k_j \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})} \qquad o_i = \sum_j \alpha_{ij} v_i$$

- Self-attention: queries, keys, and values are all different transformations of the same item-level representation of some sequence:

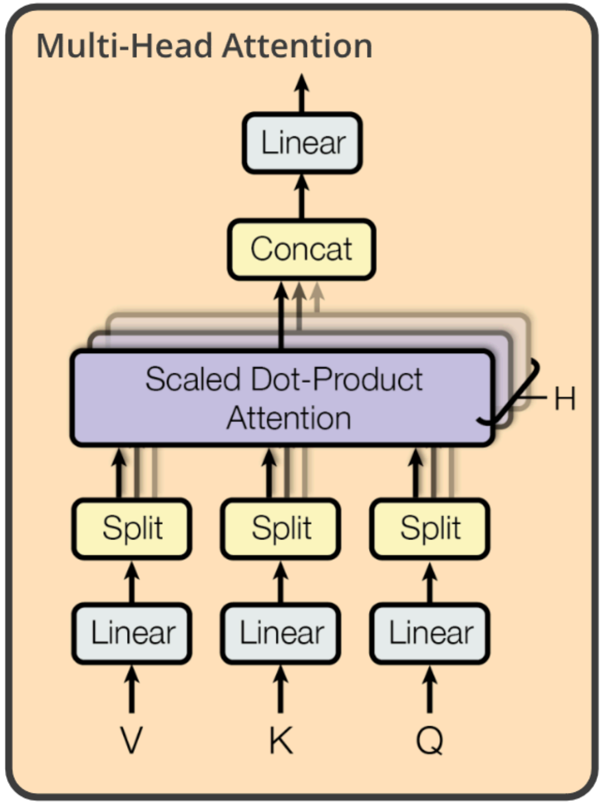


$q_i = Qx_i$ (queries) $\quad k_i = Kx_i$ (keys) $\quad v_i = Vx_i$ (values)

Slide from Stanford CS224 and CMU LLMs course

Multi-Head Attention

$q_i = Qx_i$ (queries) $k_i = Kx_i$ (keys) $v_i = Vx_i$ (values)

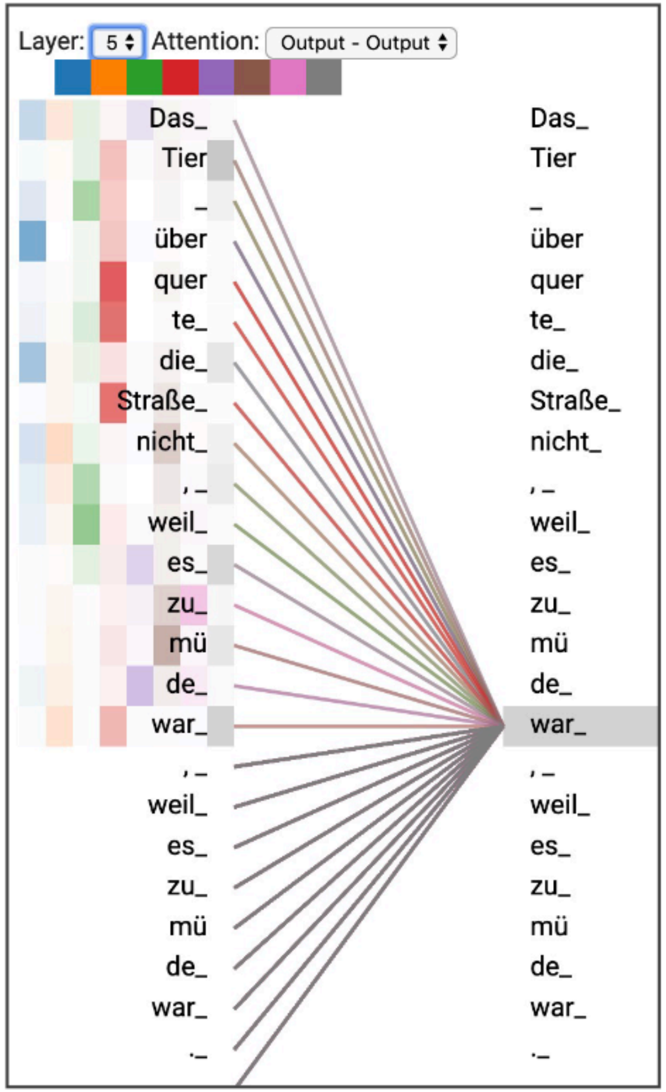


$$\text{head}_1 = \text{Attention}(\mathbf{QW}_1^Q, \mathbf{KW}_1^K, \mathbf{VW}_1^V)$$
$$\vdots$$
$$\text{head}_H = \text{Attention}(\mathbf{QW}_H^Q, \mathbf{KW}_H^K, \mathbf{VW}_H^V)$$

$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)$$

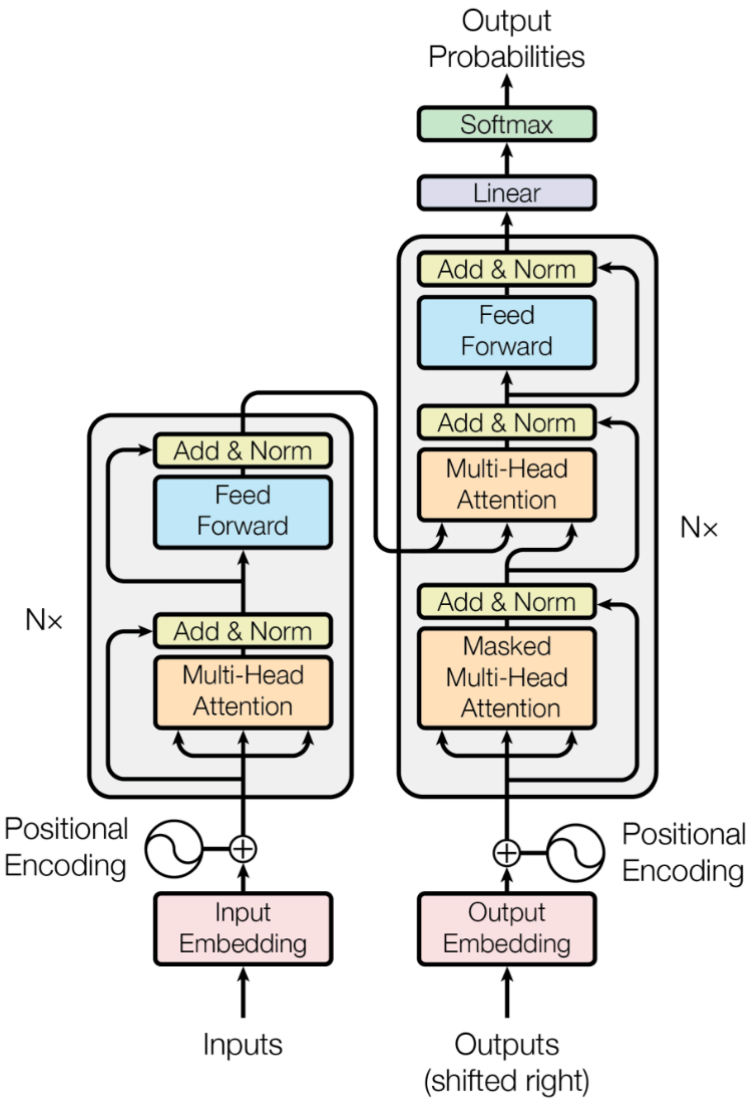
Inputs and outputs of each layer are the same dimensions:

$$\mathbf{Q} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{K} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\mathbf{V} \in \mathbb{R}^{T \times d_{\text{model}}}$$
$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{T \times d_{\text{model}}}$$



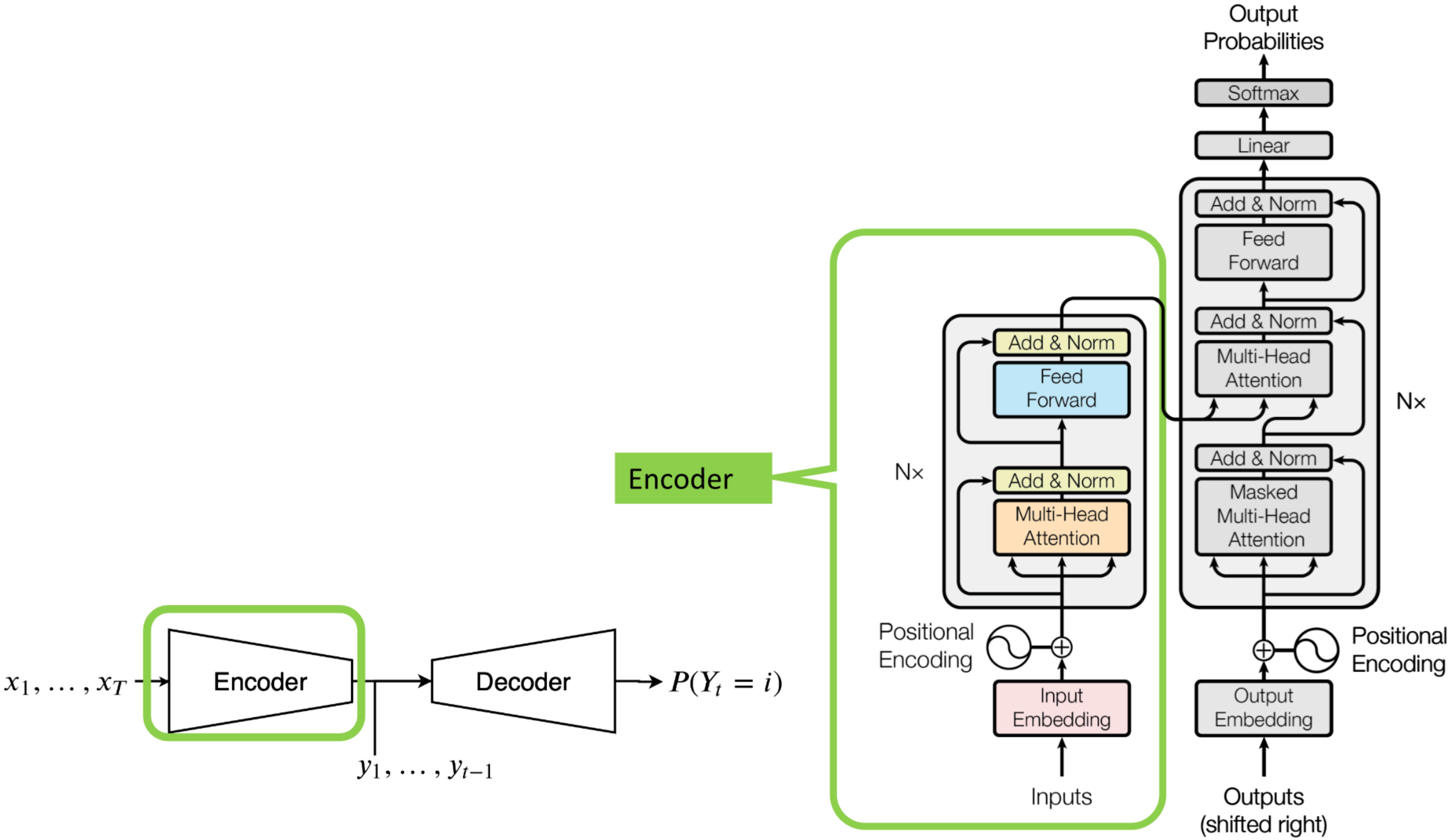
Slide from Stanford CS224 and CMU LLMs course

Transformer



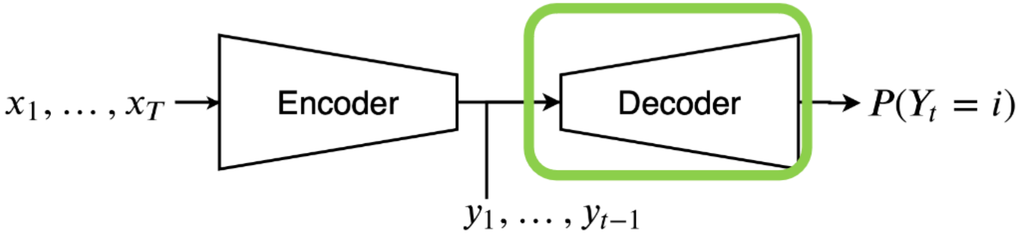
Vaswani et al. 2017, slide from CMU LLMs course

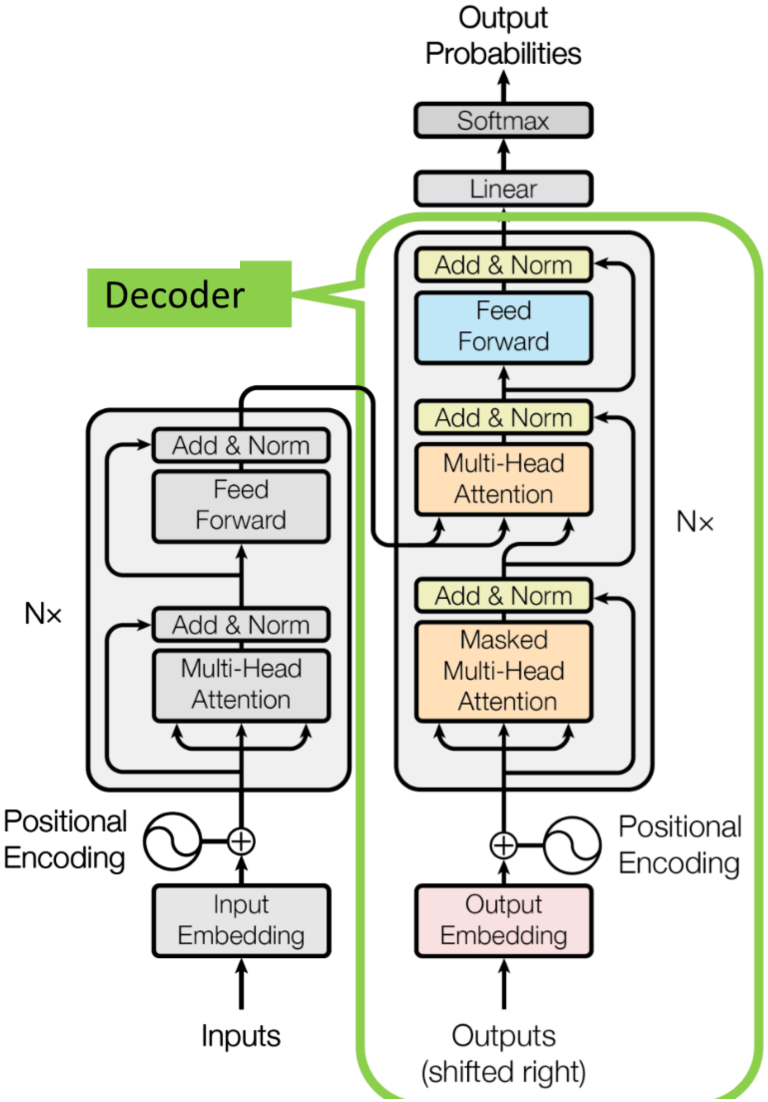
Encoder



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

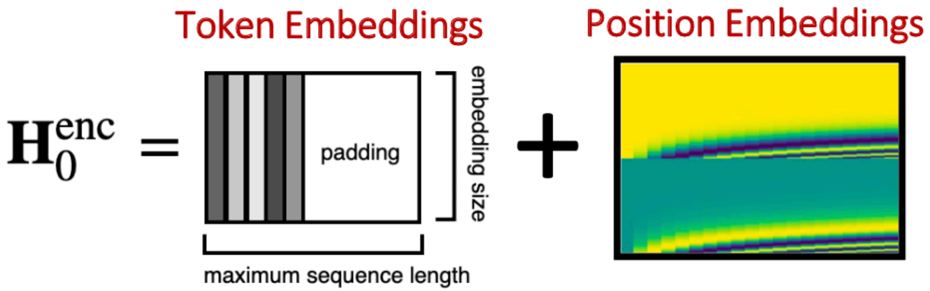




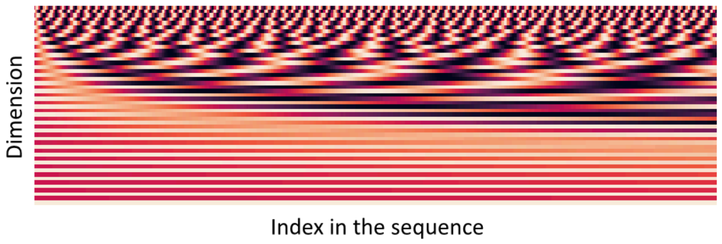
Vaswani et al. 2017, slide from CMU LLMs course

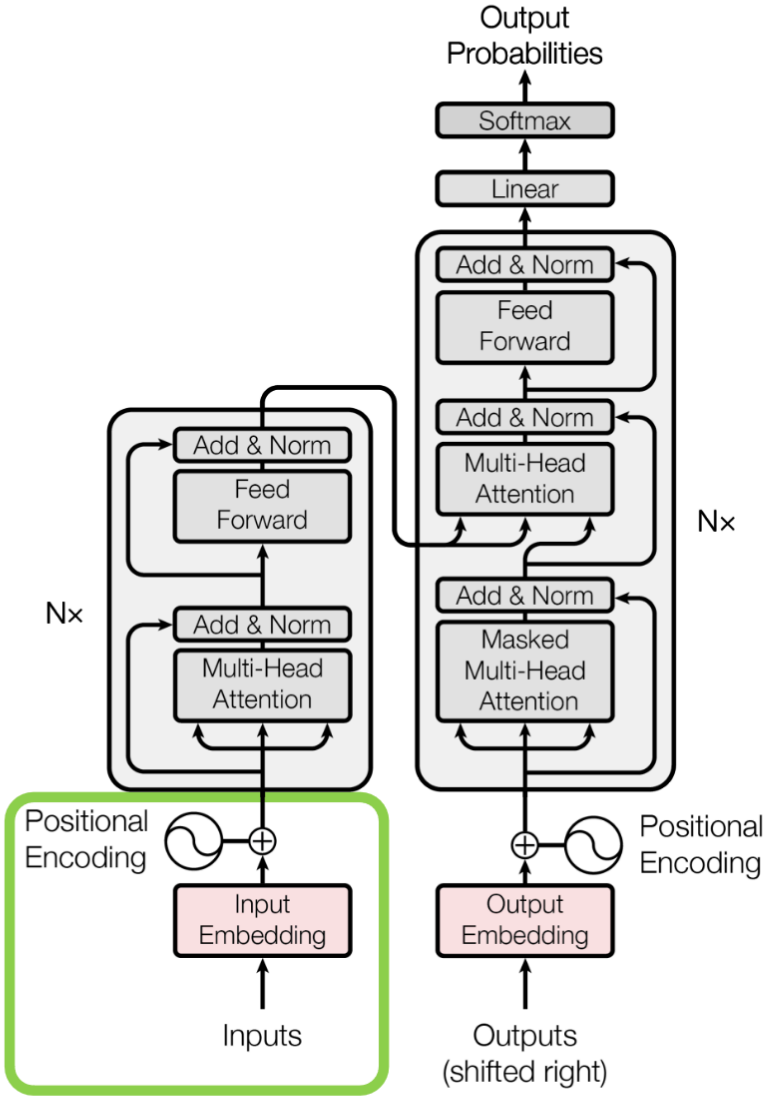
Encoder Input

The input into the encoder looks like:



$$\boldsymbol{p}_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$

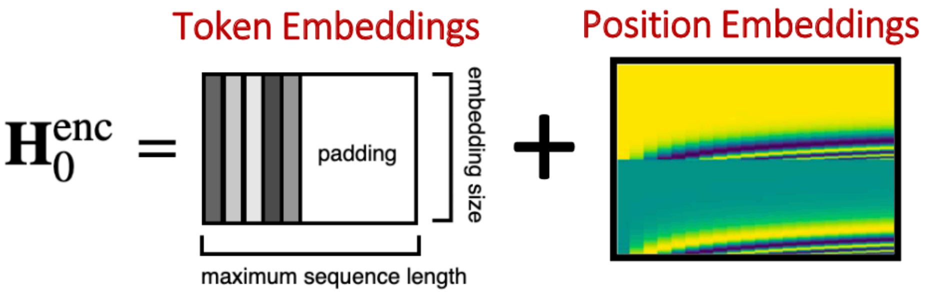




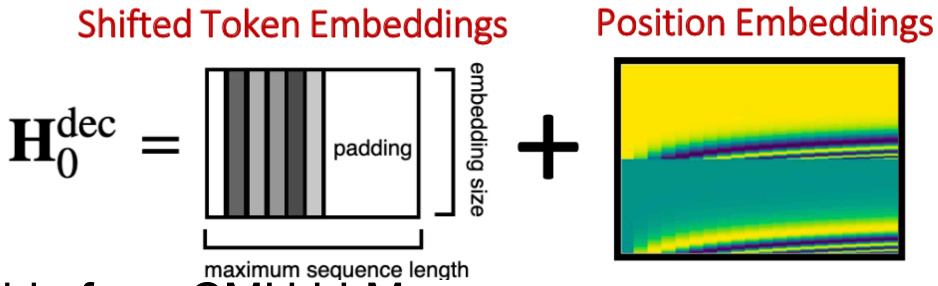
Vaswani et al. 2017, slide from CMU LLMs course and Stanford CS 224

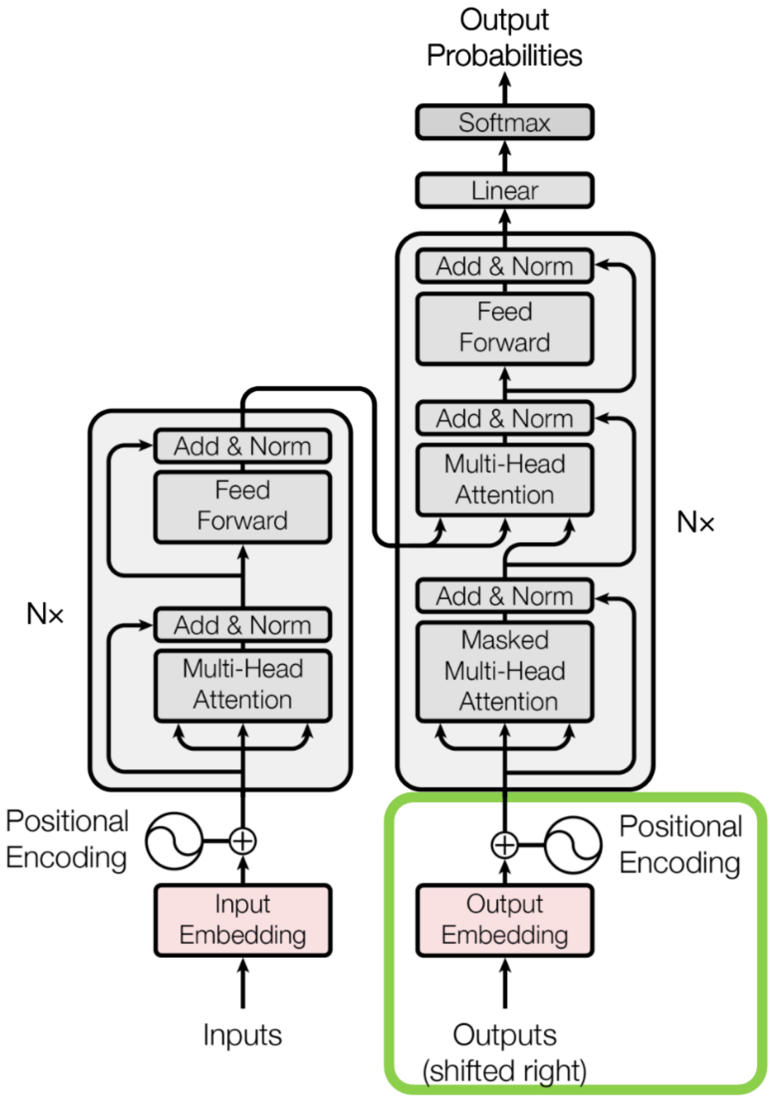
Decoder Input

The input into the encoder looks like:



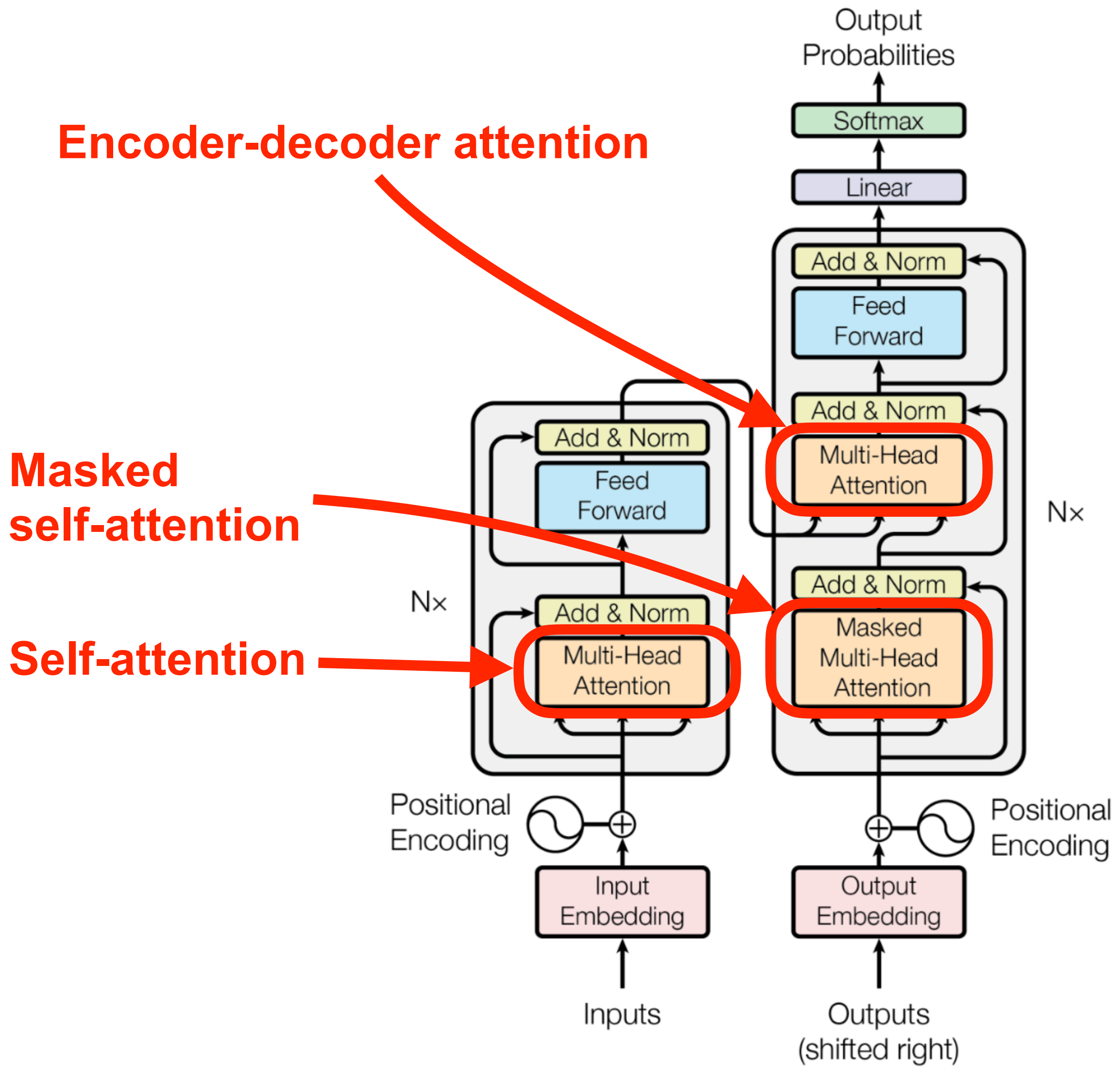
The input to the decoder looks like:





Vaswani et al. 2017, slide from CMU LLMs course

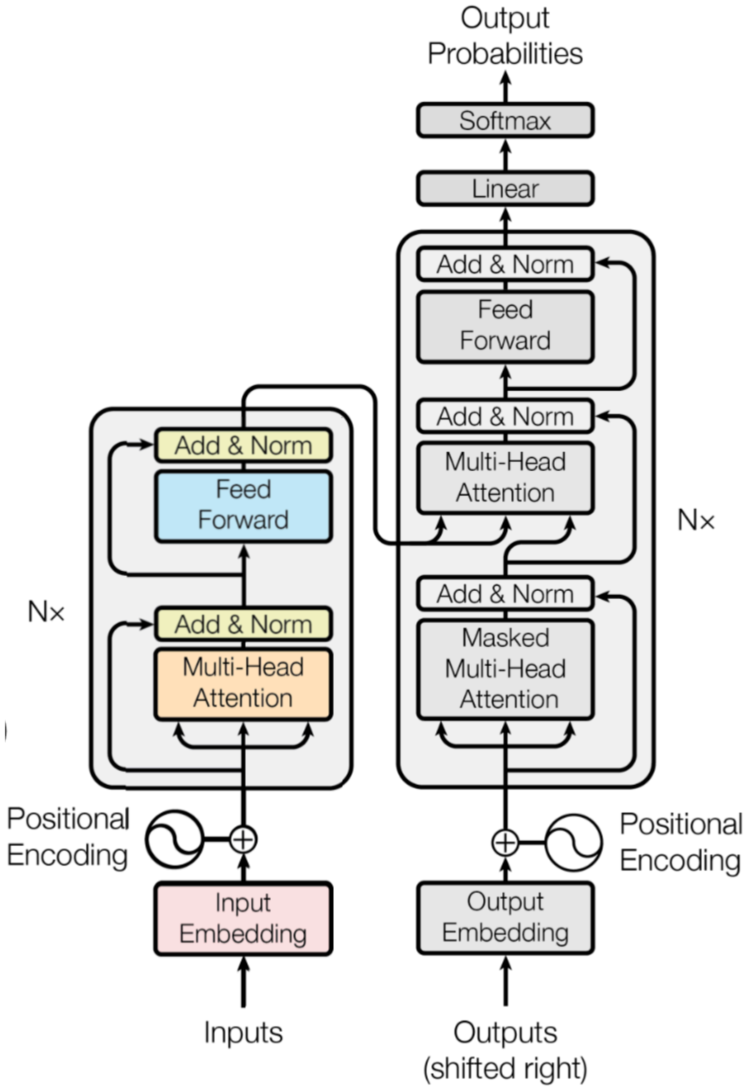
Attention



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$= \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$



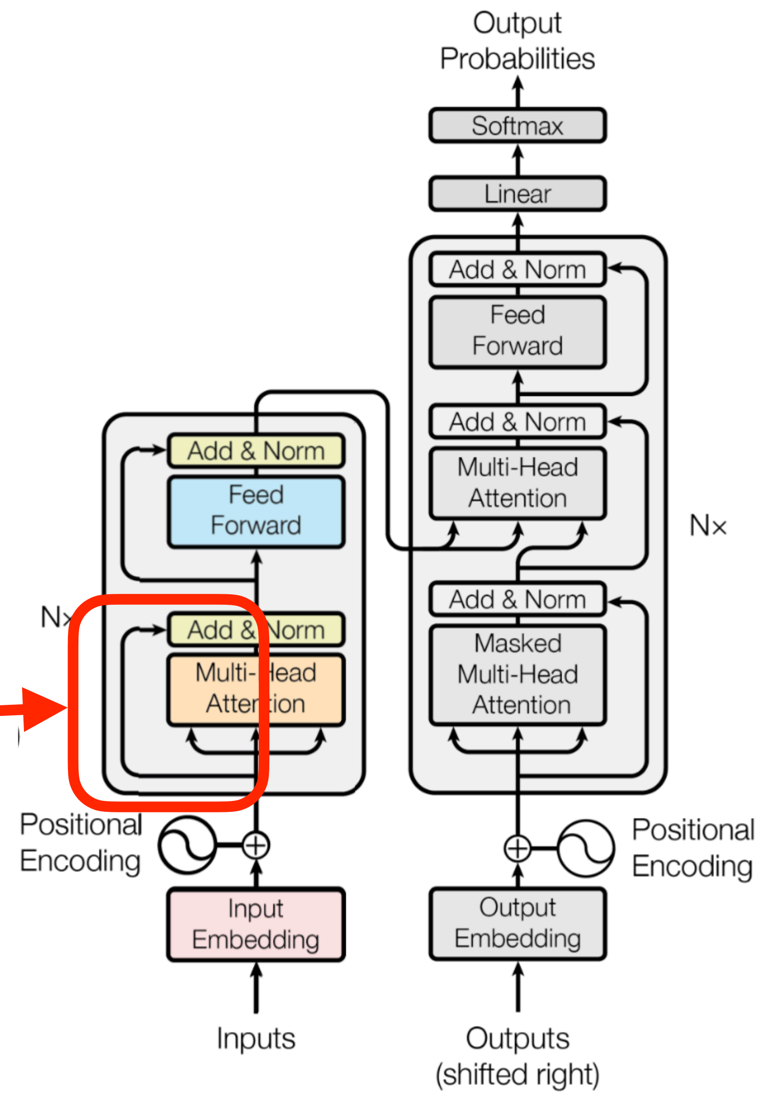
Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$\boxed{\text{Multi-Head Attention}} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\boxed{\text{Add \& Norm}} = \mathrm{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{enc})$$

Residual connection



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$

Add & Norm $= \text{LayerNorm}($ Multi-Head Attention $+ \mathbf{H}_i^{enc})$

Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.

Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.

Let $\sigma = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.

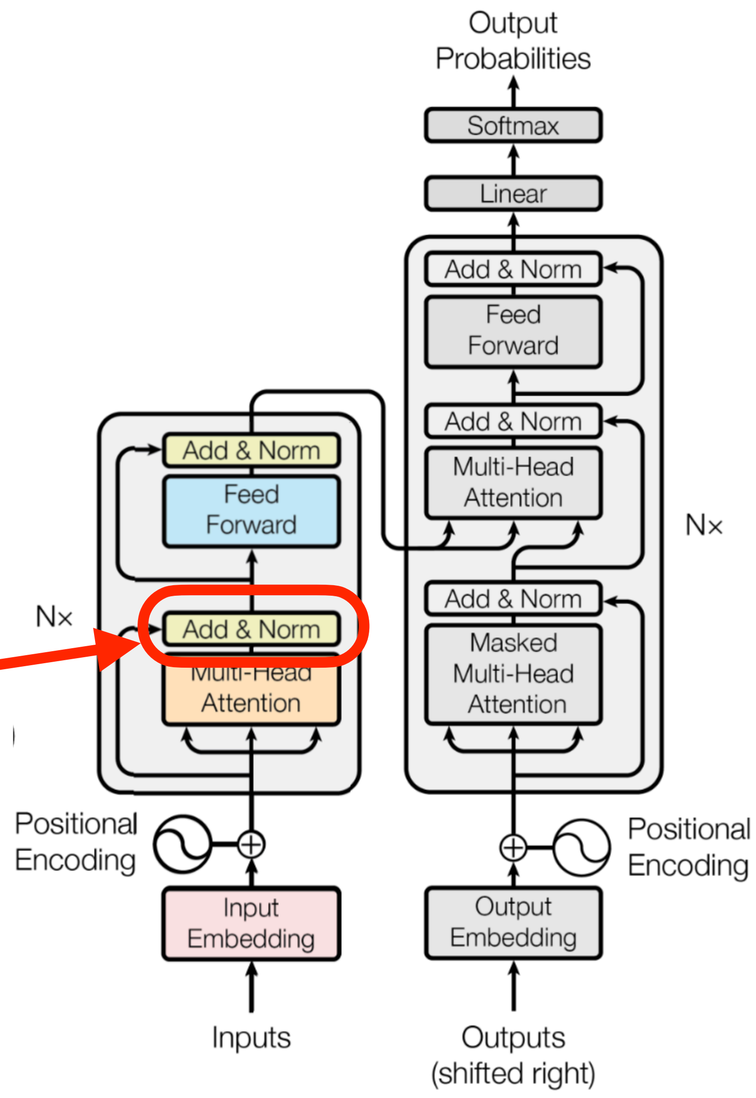
Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)

Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma} + \epsilon} * \gamma + \beta$$

Normalize by scalar mean and variance

Modulate by learned elementwise gain and bias



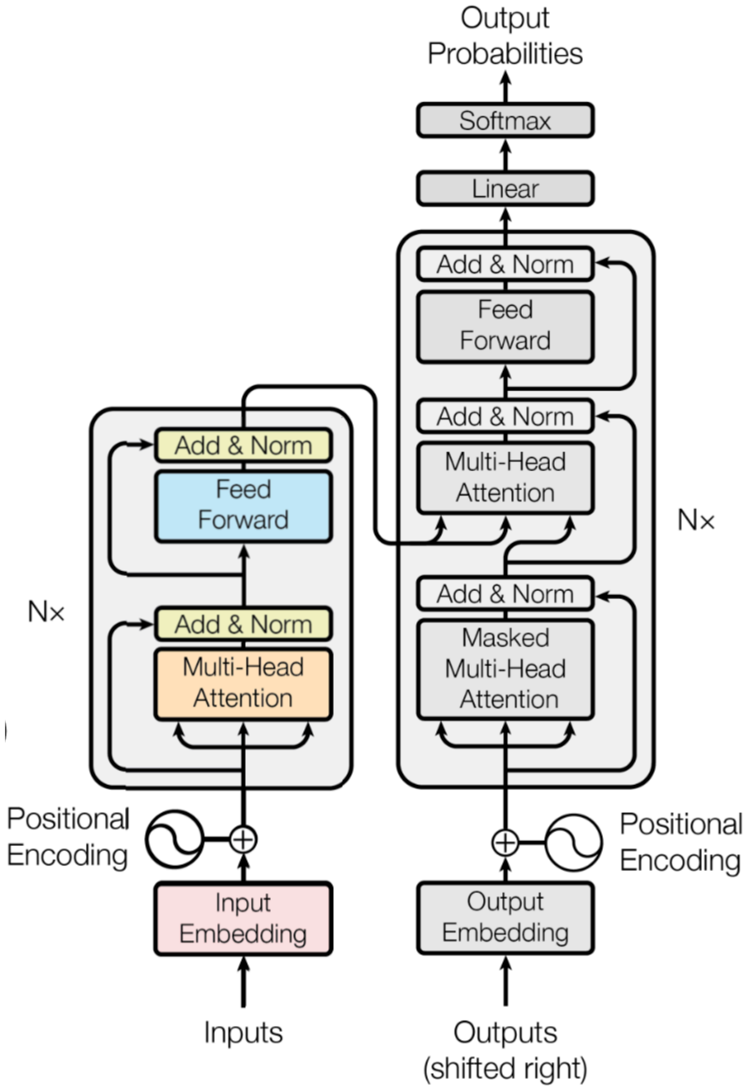
Vaswani et al. 2017, slide from CMU LLMs course

Encoder

$$\text{Multi-Head Attention} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\text{Add \& Norm} = \mathrm{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{enc})$$

$$\text{Feed Forward} = \max(0, \text{Add \& Norm}\ \mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$



Vaswani et al. 2017, slide from CMU LLMs course

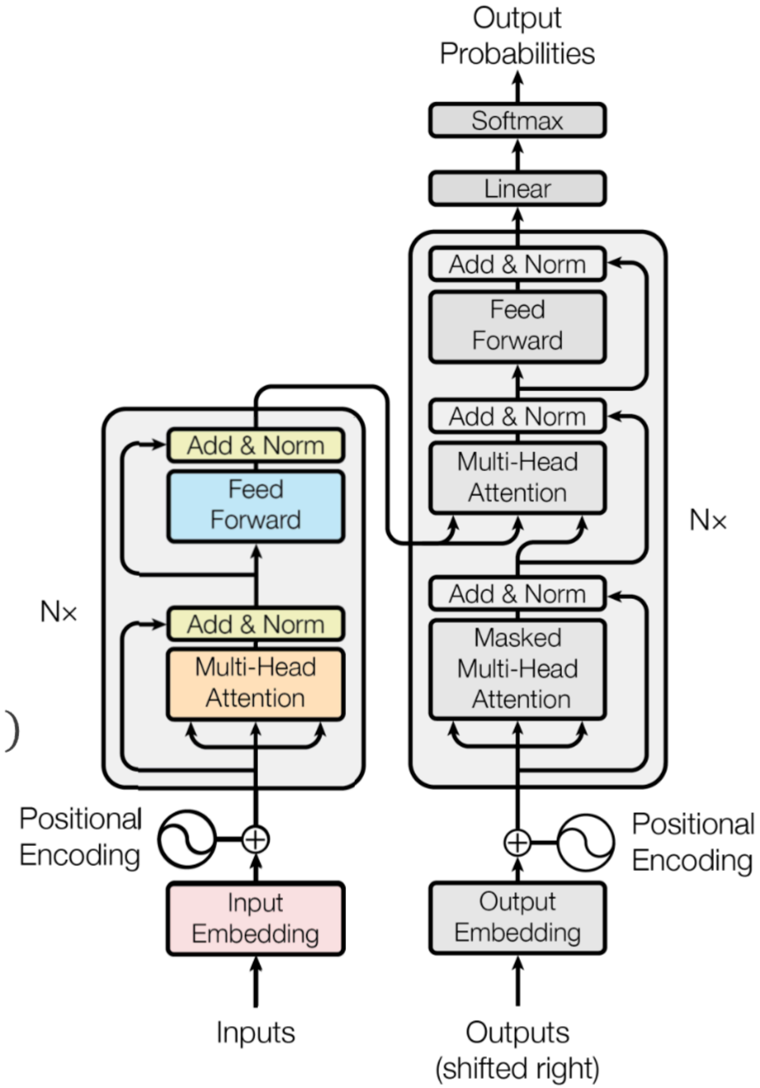
Encoder

$$\text{Multi-Head Attention} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\text{Add \& Norm} = \mathrm{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{enc})$$

$$\text{Feed Forward} = \max(0, \text{Add \& Norm}\, \mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

$$\text{Add \& Norm (2)} = \mathrm{LayerNorm}(\text{Feed Forward} + \text{Add \& Norm})$$



Vaswani et al. 2017, slide from CMU LLMs course

Encoder

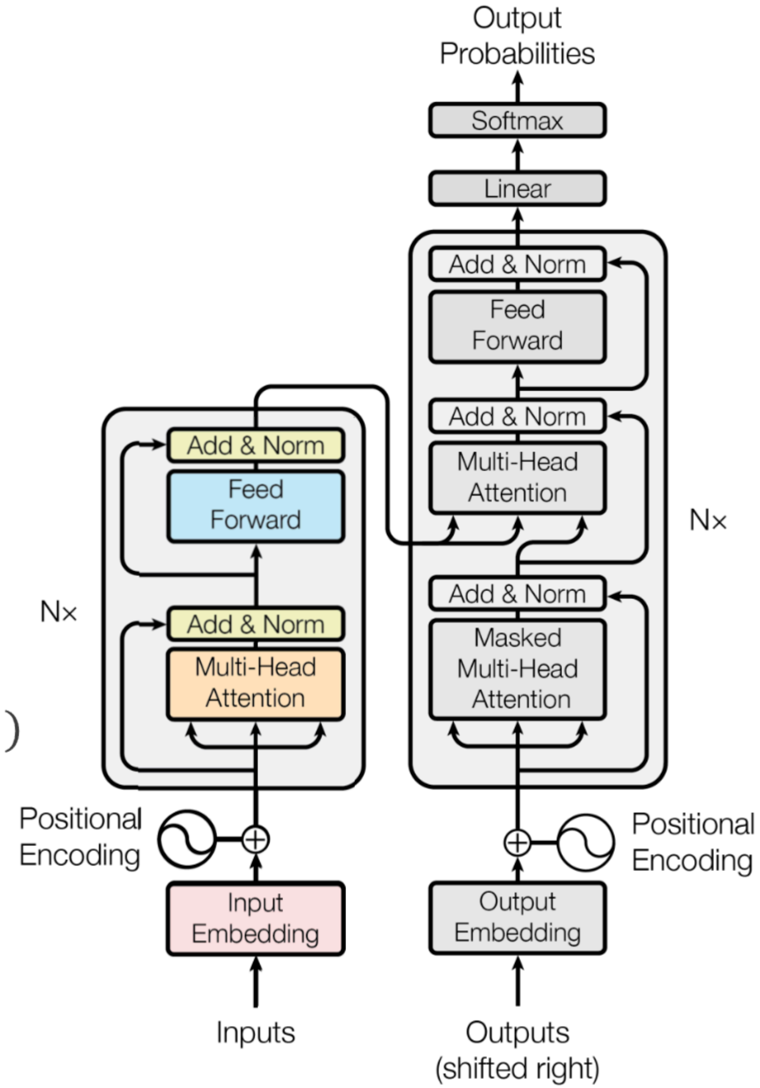
$$\text{Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \mathbf{H}_i^{enc})$$

$$\text{Add \& Norm} = \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{enc})$$

$$\text{Feed Forward} = \max(0, \text{Add \& Norm}\, \mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

$$\text{Add \& Norm (2)} = \text{LayerNorm}(\text{Feed Forward} + \text{Add \& Norm})$$

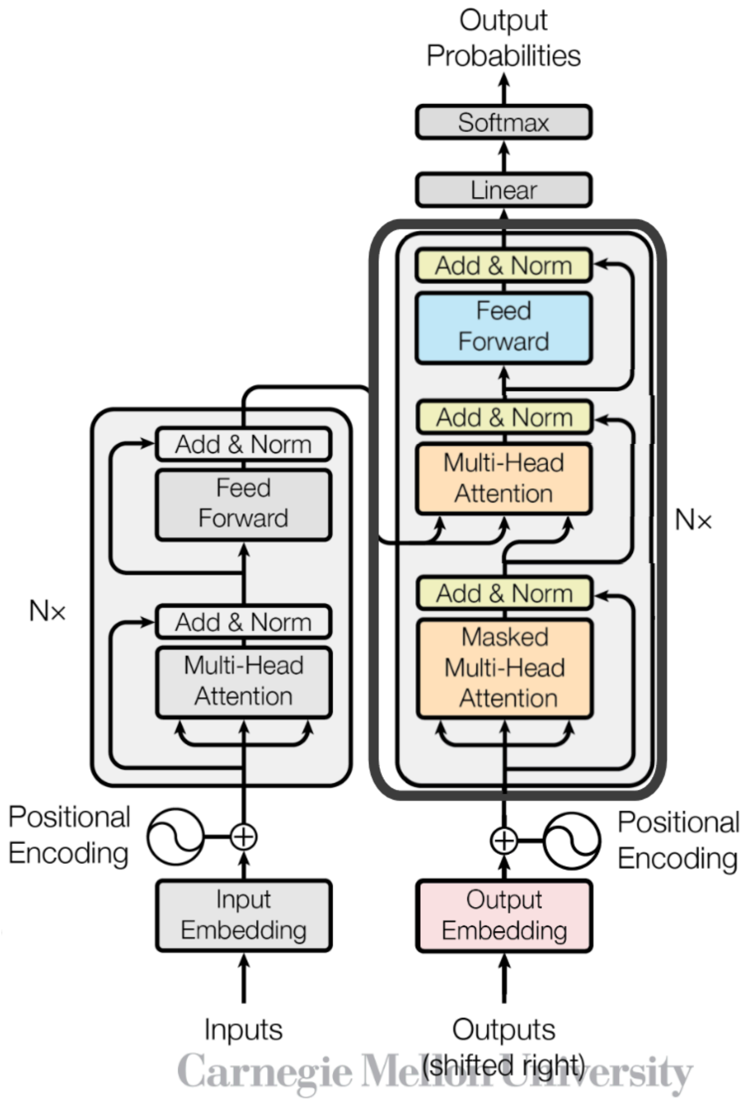
$$\mathbf{H}_{i+1}^{enc} = \text{Add \& Norm(2)}$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

Masked Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

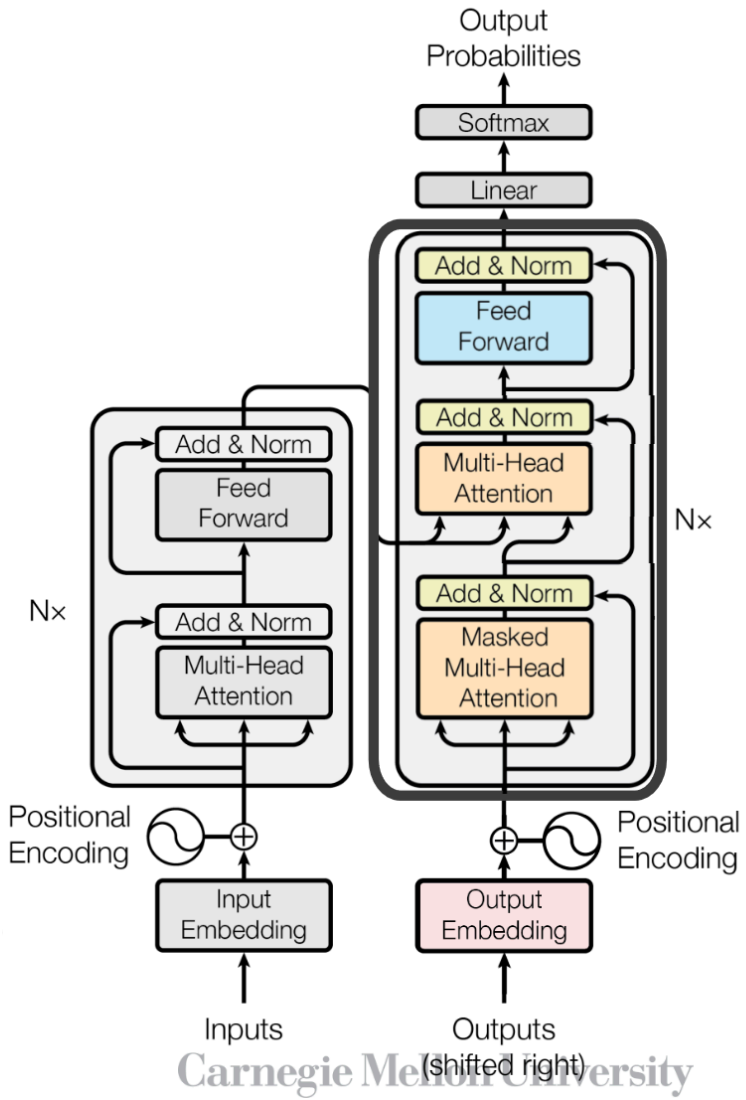


Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\text{Masked Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\text{Add \& Norm} = \text{LayerNorm}(\text{Masked Multi-Head Attention} + \mathbf{H}_i^{dec})$$



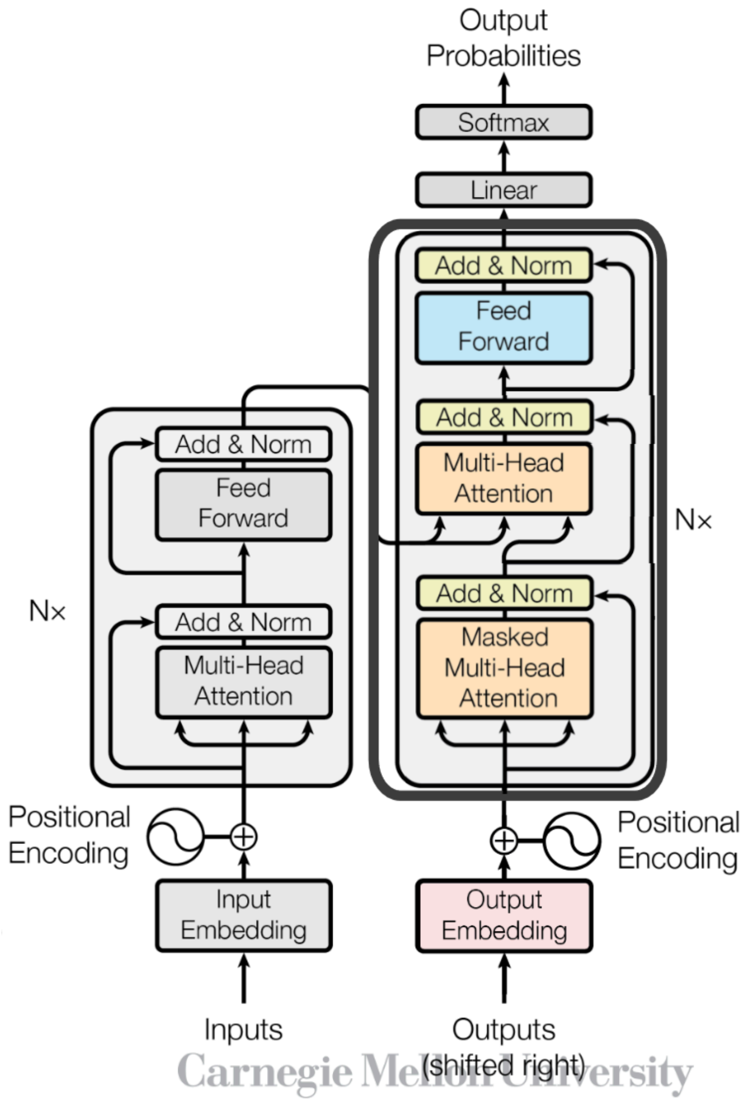
Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\text{Masked Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\text{Add \& Norm} = \text{LayerNorm}(\text{Masked Multi-Head Attention} + \mathbf{H}_i^{dec})$$

$$\text{Enc-Dec Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \text{Add \& Norm})$$



Vaswani et al. 2017, slide from CMU LLMs course

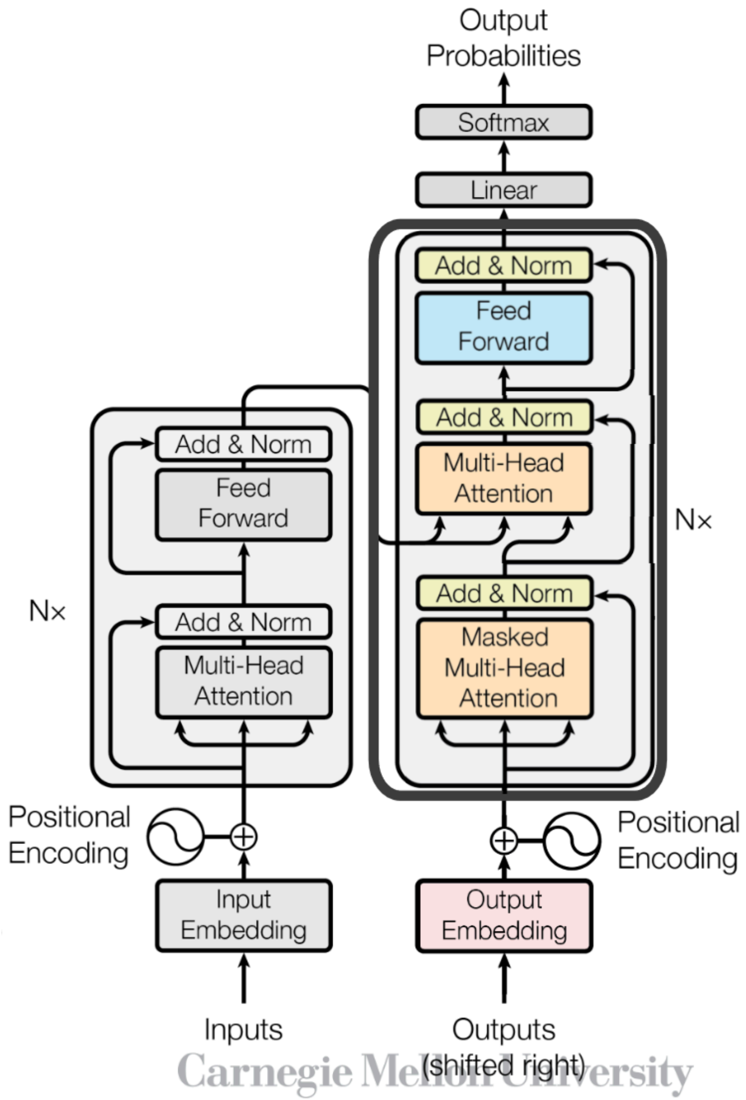
Decoder

Masked Multi-Head Attention $= \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$

Add & Norm $= \mathrm{LayerNorm}($ Masked Multi-Head Attention $+ \mathbf{H}_i^{dec})$

Enc-Dec Multi-Head Attention $= \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc},$ Add & Norm $)$

Add & Norm (2) $= \mathrm{LayerNorm}($ Enc-Dec Multi-Head Attention $+$ Add & Norm $)$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

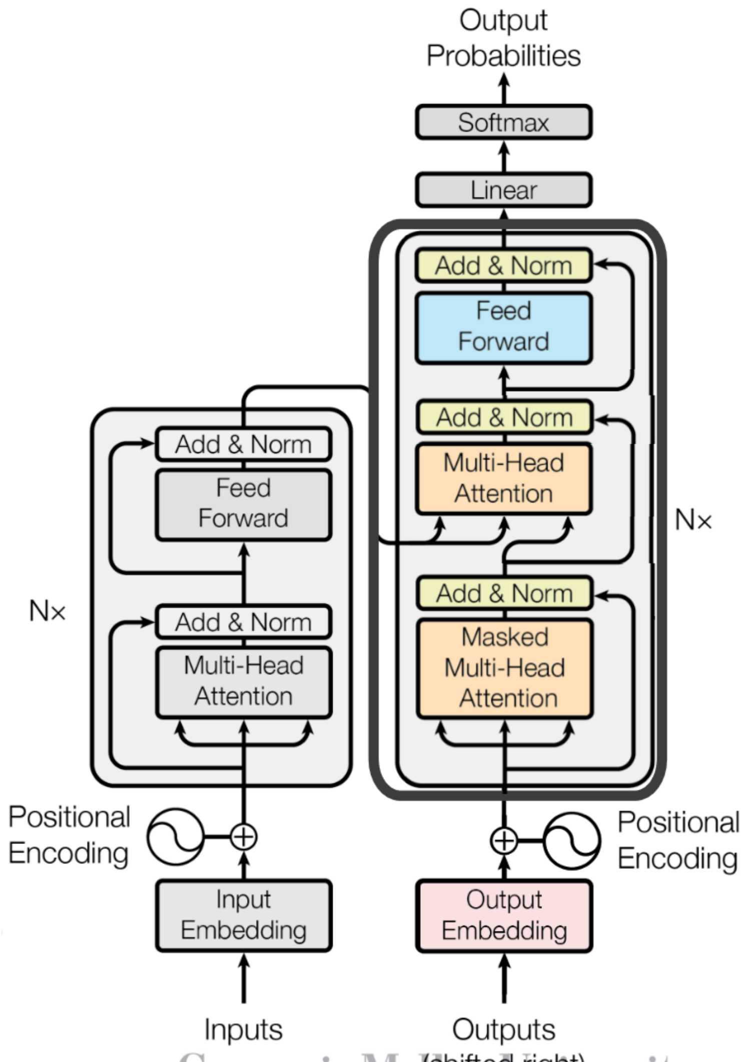
$$\text{Masked Multi-Head Attention} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\text{Add \& Norm} = \mathrm{LayerNorm}(\text{Masked Multi-Head Attention} + \mathbf{H}_i^{dec})$$

$$\text{Enc-Dec Multi-Head Attention} = \mathrm{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \text{Add \& Norm})$$

$$\text{Add \& Norm (2)} = \mathrm{LayerNorm}(\text{Enc-Dec Multi-Head Attention} + \text{Add \& Norm})$$

$$\text{Feed Forward} = \max(0, \text{Add \& Norm (2)}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\boxed{\text{Masked Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

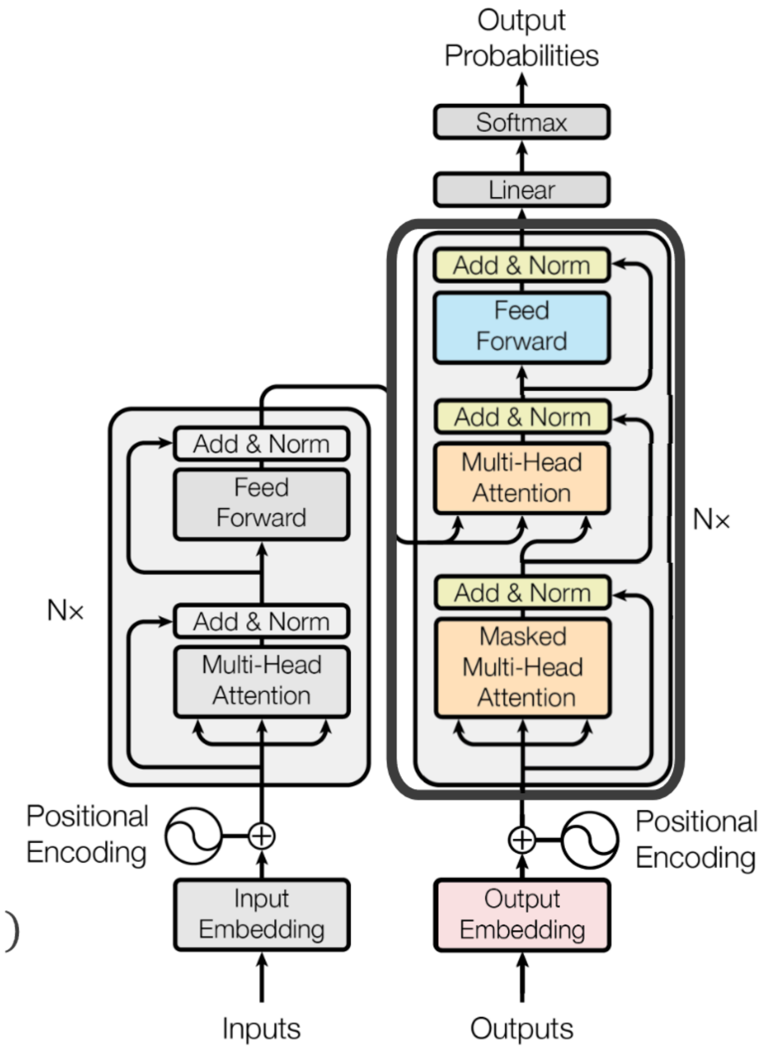
$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Masked Multi-Head Attention}} + \mathbf{H}_i^{dec})$$

$$\boxed{\text{Enc-Dec Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Add \& Norm (2)}} = \text{LayerNorm}(\boxed{\text{Enc-Dec Multi-Head Attention}} + \boxed{\text{Add \& Norm}})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm (2)}}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$$\boxed{\text{Add \& Norm (3)}} = \text{LayerNorm}(\boxed{\text{Feed Forward}} + \boxed{\text{Add \& Norm (2)}})$$



Vaswani et al. 2017, slide from CMU LLMs course

Decoder

$$\text{Masked Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{dec}, \mathbf{H}_i^{dec}, \mathbf{H}_i^{dec})$$

$$\text{Add \& Norm} = \text{LayerNorm}(\text{Masked Multi-Head Attention} + \mathbf{H}_i^{dec})$$

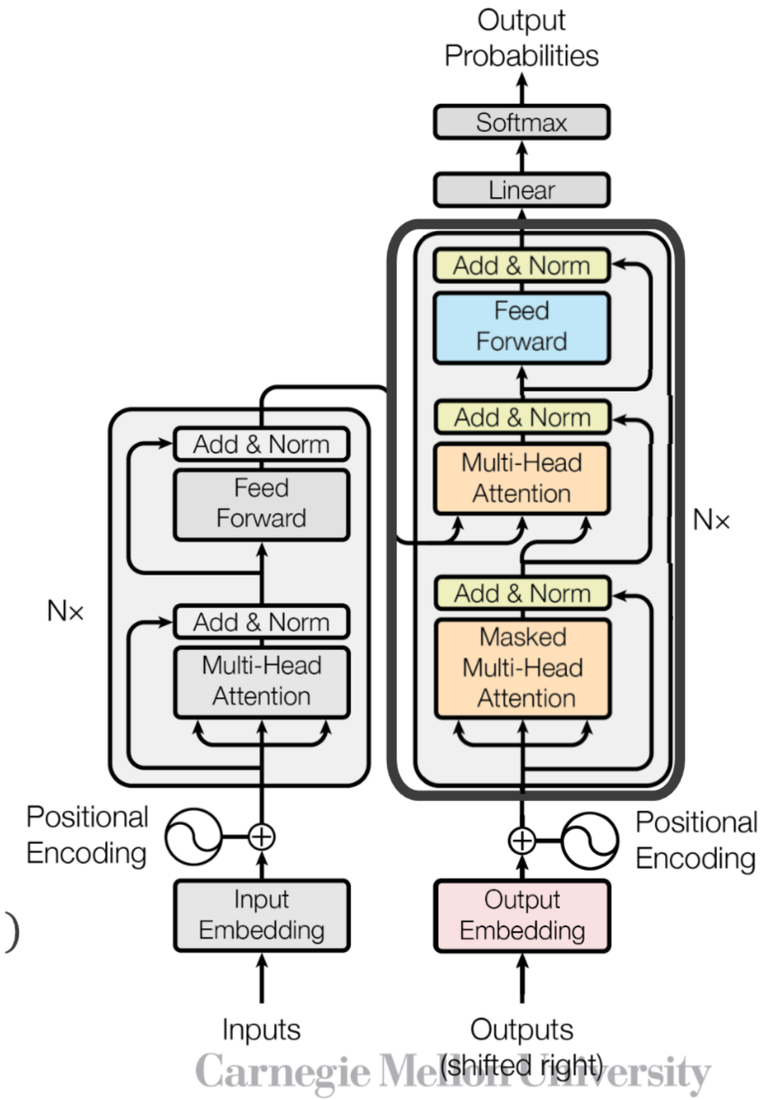
$$\text{Enc-Dec Multi-Head Attention} = \text{MultiHeadAtt}(\mathbf{H}_i^{enc}, \mathbf{H}_i^{enc}, \text{Add \& Norm})$$

$$\text{Add \& Norm (2)} = \text{LayerNorm}(\text{Enc-Dec Multi-Head Attention} + \text{Add \& Norm})$$

$$\text{Feed Forward} = \max(0, \text{Add \& Norm (2)}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2)$$

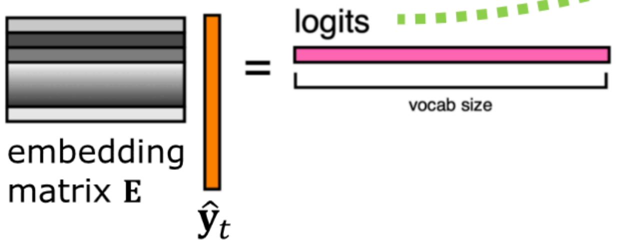
$$\text{Add \& Norm (3)} = \text{LayerNorm}(\text{Feed Forward} + \text{Add \& Norm (2)})$$

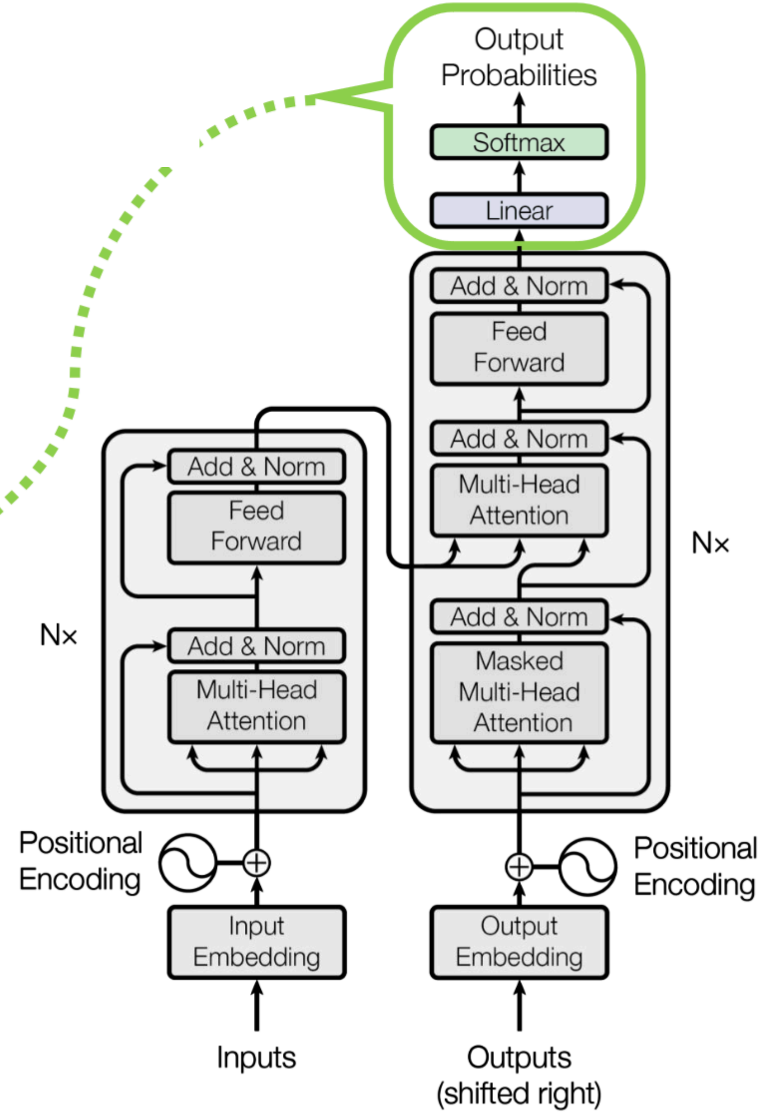
$$\mathbf{H}_{i+1}^{\text{dec}} = \text{Add \& Norm(3)}$$



Vaswani et al. 2017, slide from CMU LLMs course

Output Probabilities



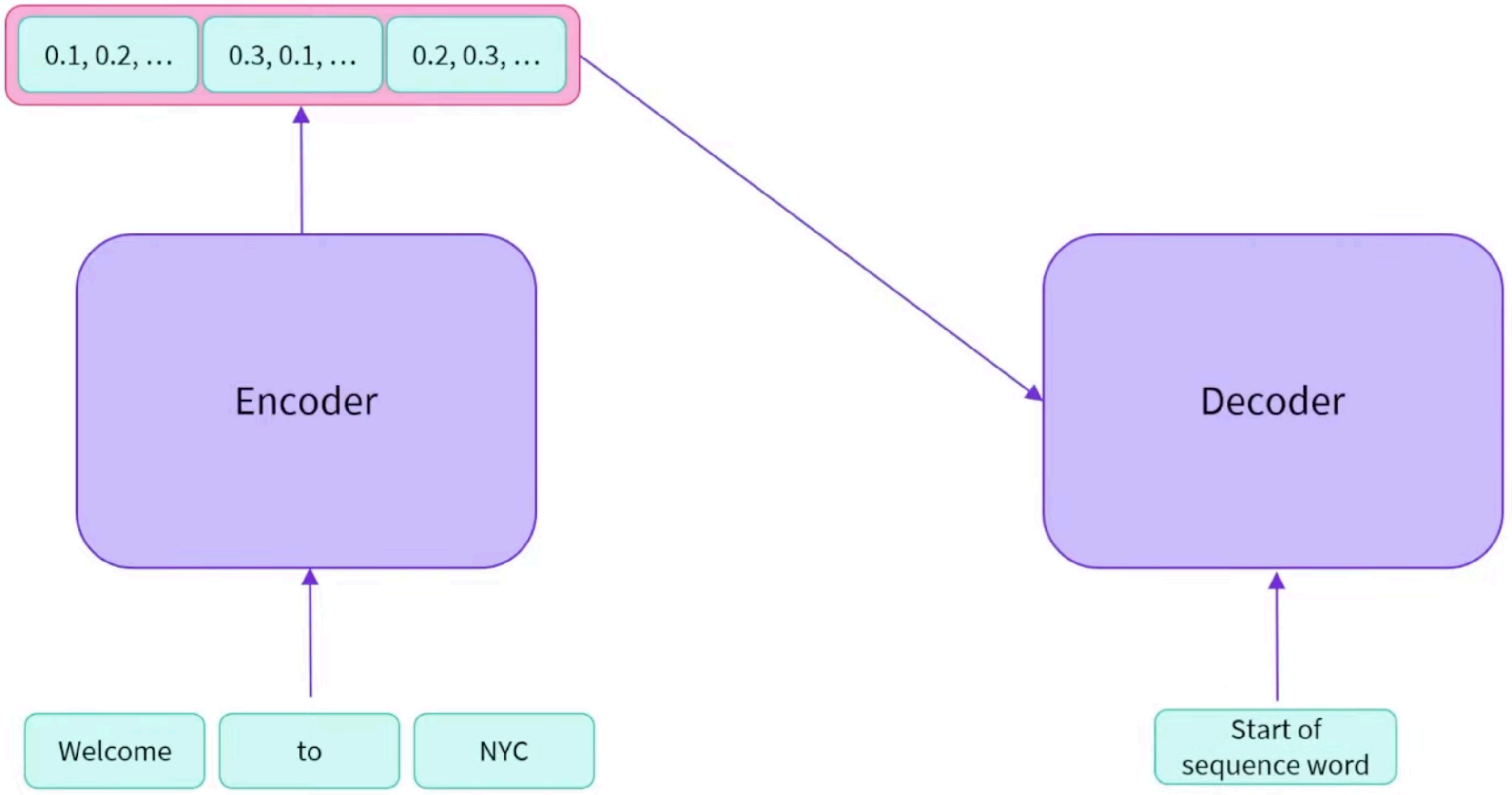


$$P(Y_t = i \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$

Vaswani et al. 2017, slide from CMU LLMs course

Encoder-Decoder Inference

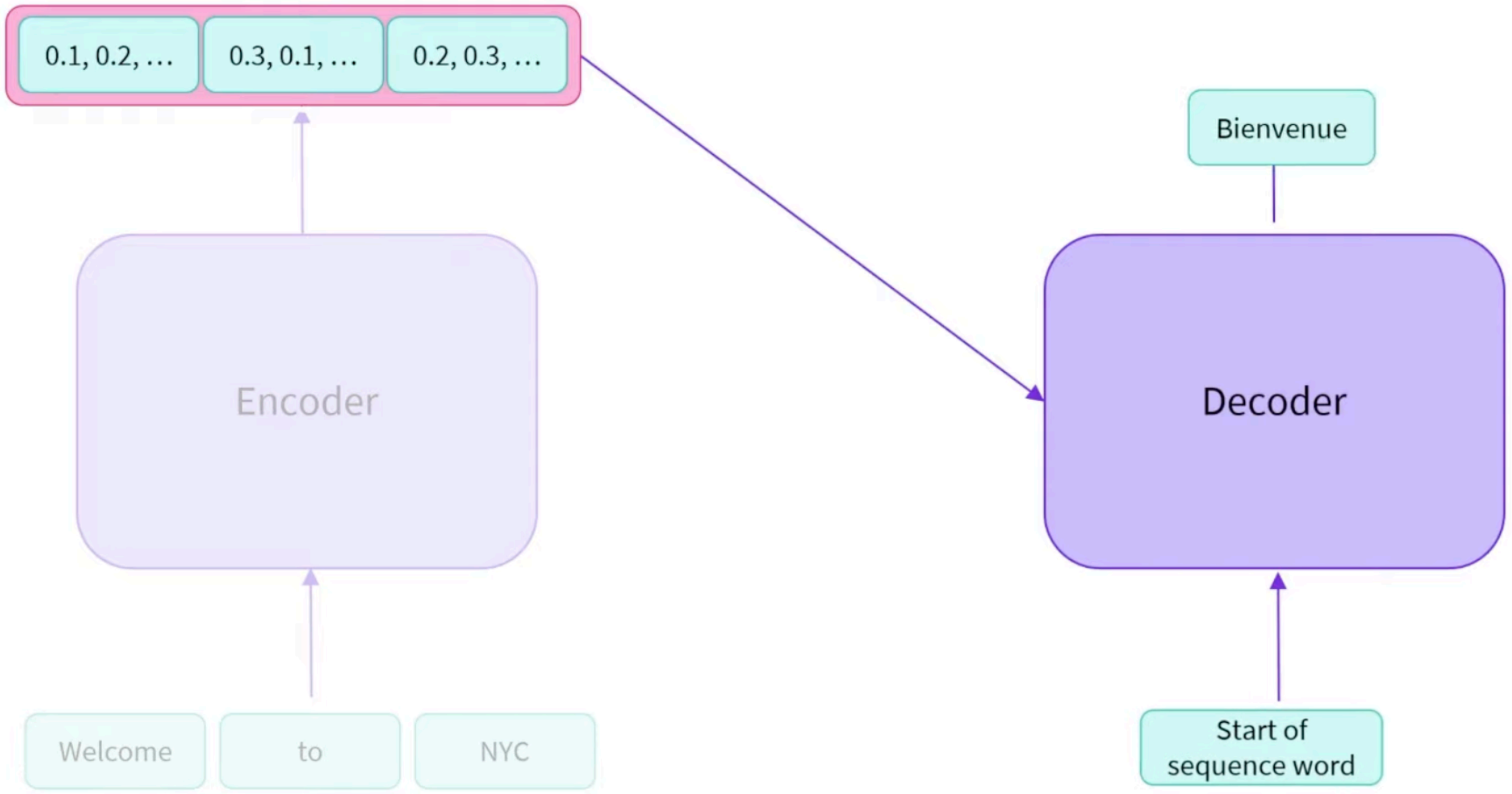
- Encode input sequence



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder-Decoder Inference

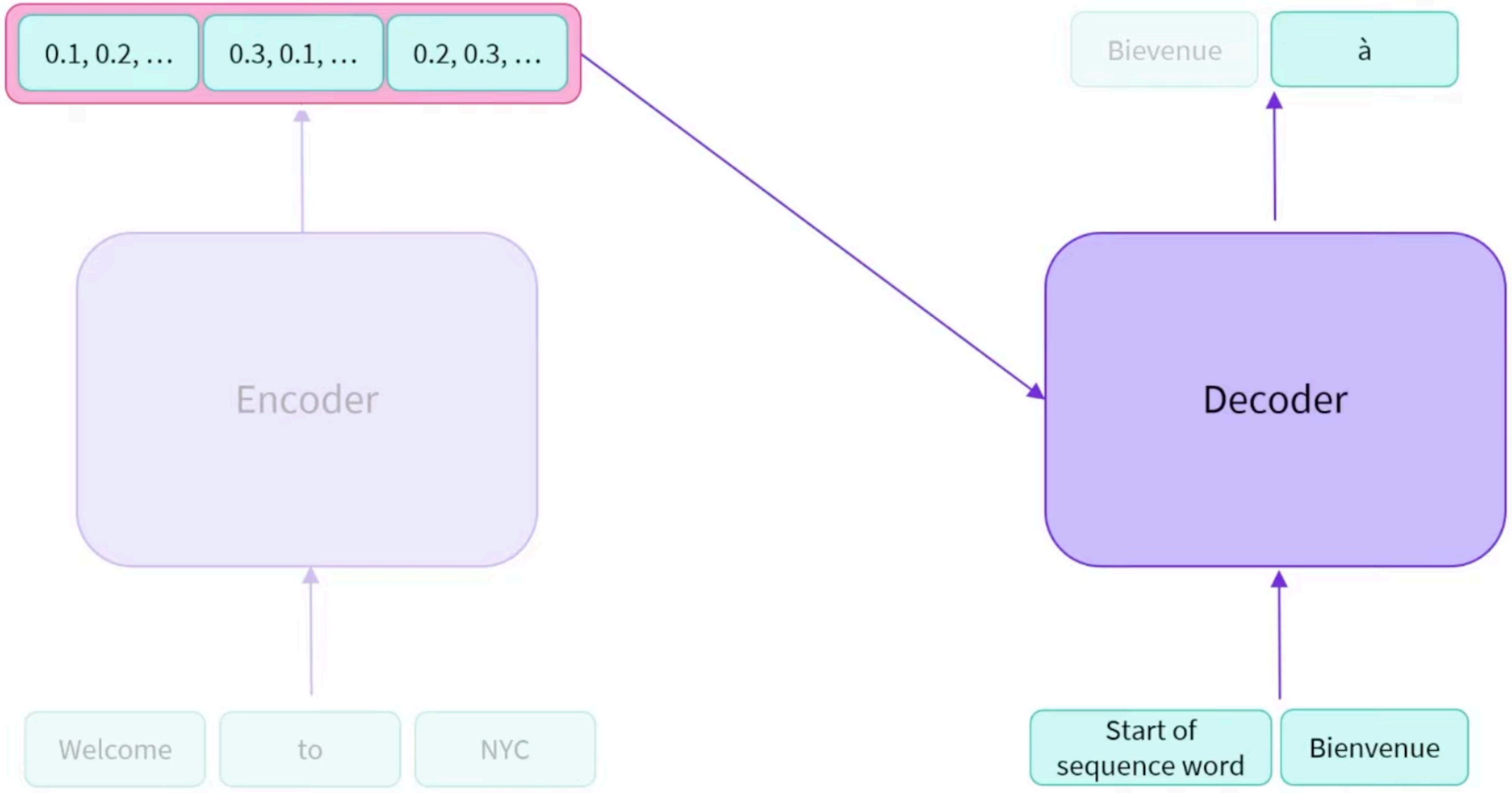
- Encode input sequence
- Attention over input token representations and <start>



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder-Decoder Inference

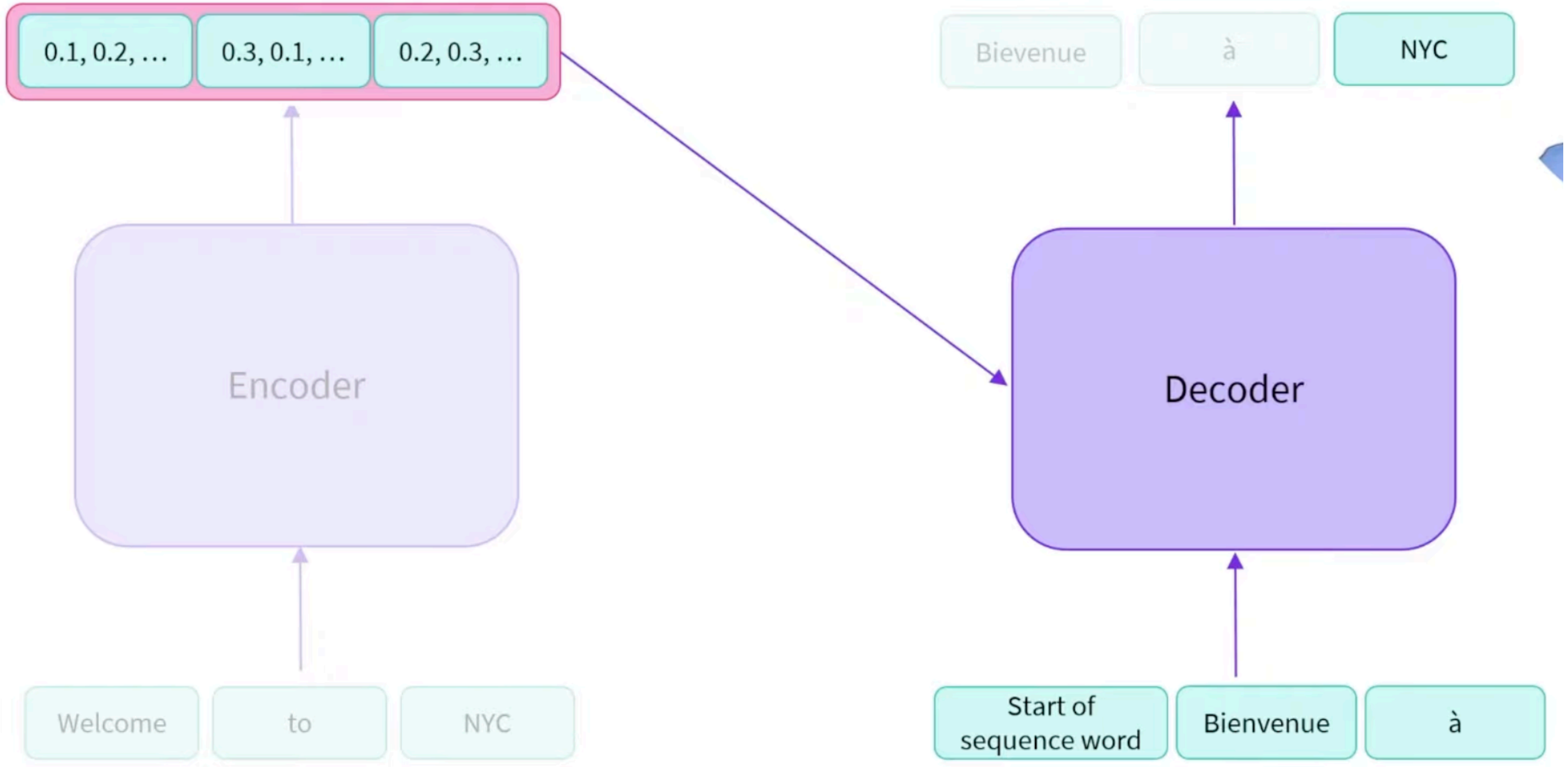
- Encode input sequence
- Attention over input token representations and <start>
- Self-attention



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

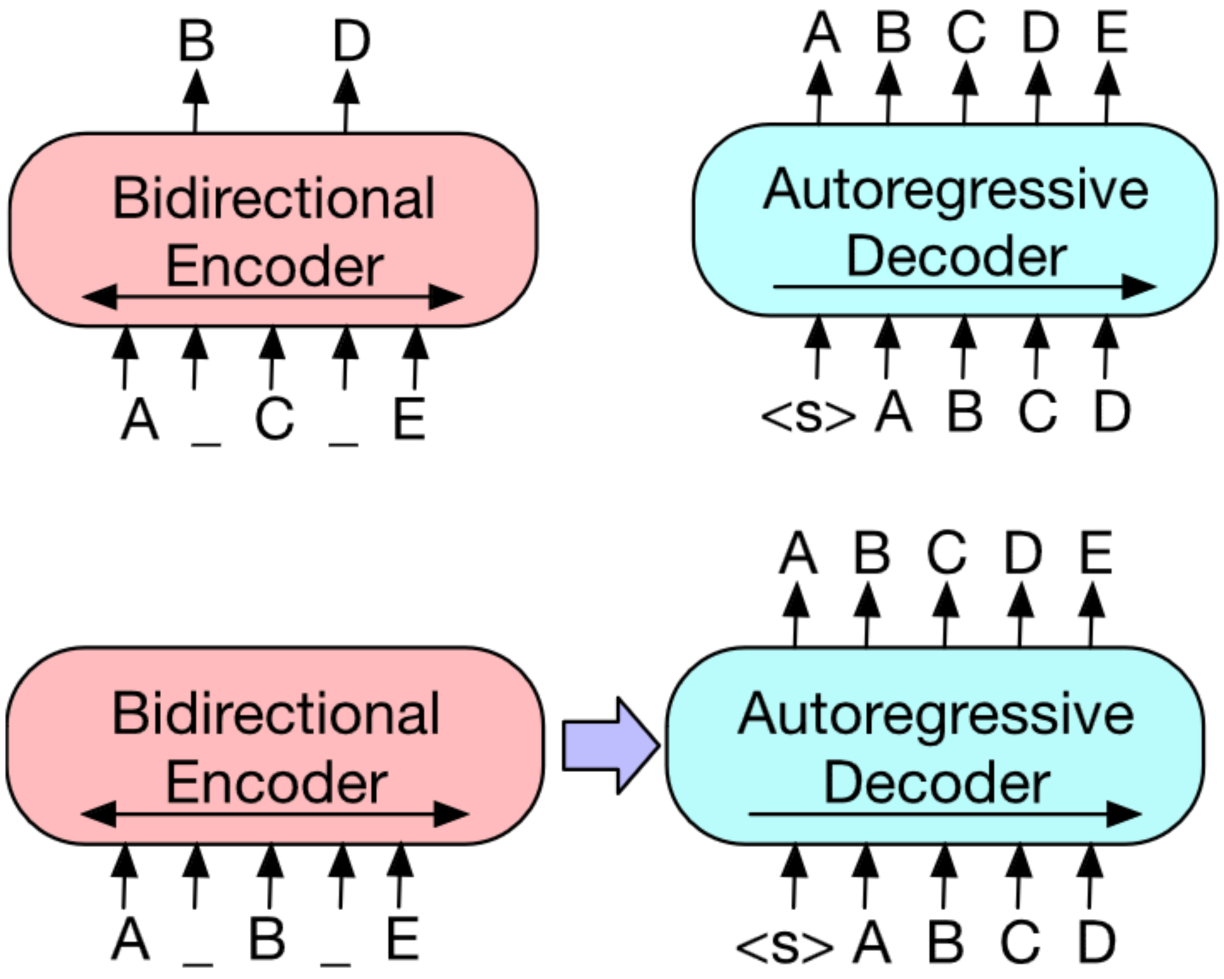
Encoder-Decoder Inference

- Encode input sequence
- Attention over input token representations and <start>
- Self-attention



slide from HuggingFace Transformers course: https://www.youtube.com/watch?v=0_4KEb08xrE&t=204s

Encoder, Decoder, Encoder-Decoder



BART, Lewis et al. 2019

Problems with the Transformer?

- Fixed context lengths "solved" with position embeddings
- Self-attention has quadratic cost $O(n^2d)$

- Plug: Annotated Transformer (Sasha Rush): http://nlp.seas.harvard.edu/annotated-transformer/

Training Language Models

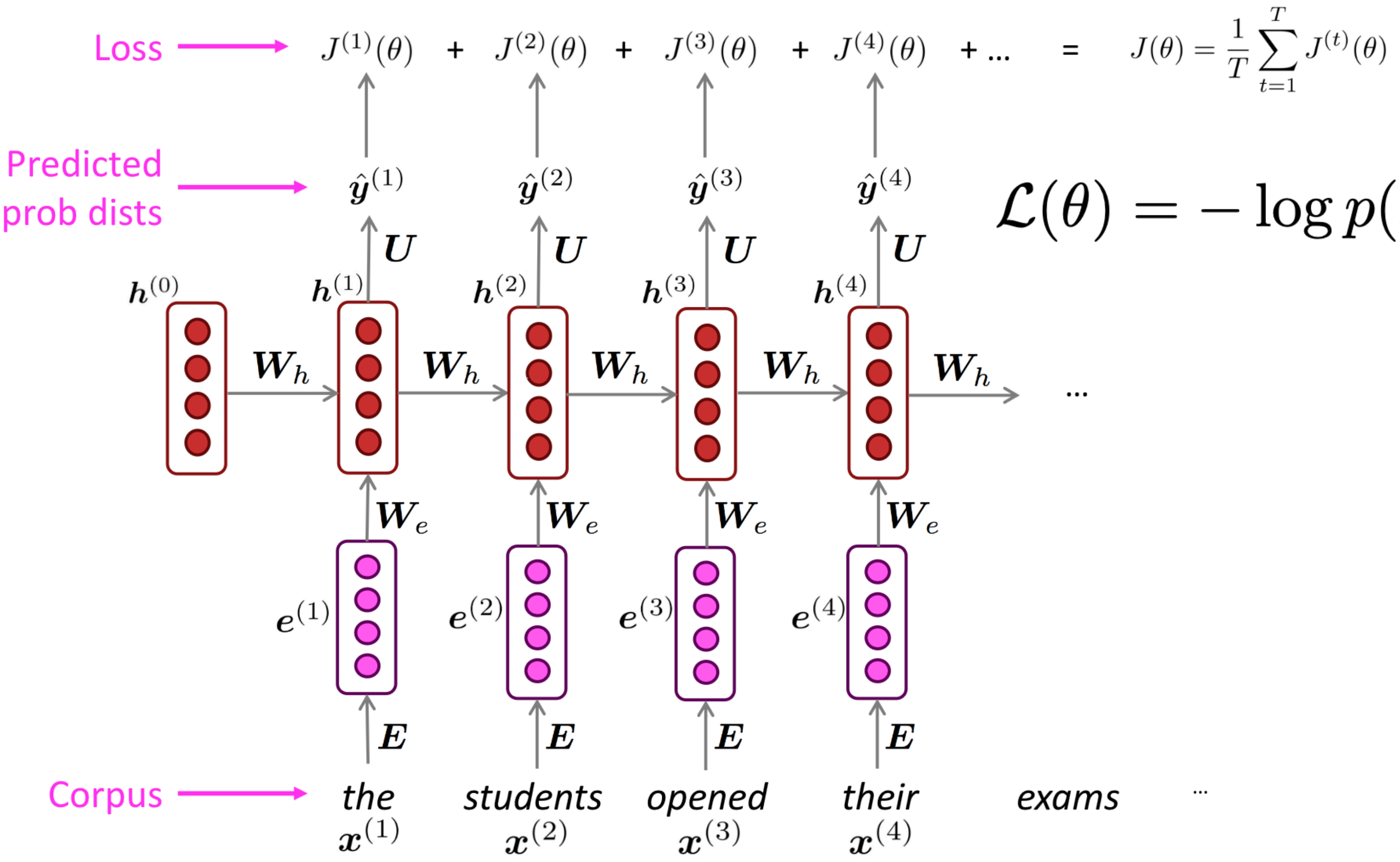
Recap: Language Modeling Objective

- Assume we have training data $\langle x_0 \ldots x_T \rangle$
- Use current LM parameters to compute probability distributions over each token independently, conditioned on the prefix:
  $P(X_i) = p(\cdot \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$
- Loss for step *i* is cross-entropy between true distribution p^* (i.e., one-hot) and predicted distribution:

$$\mathcal{L}(\theta) = -\sum_{x \in \mathcal{V}} p^*(x_i = x \mid \langle x_0 \ldots x_{i-1} \rangle) \log p(x_i = x \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

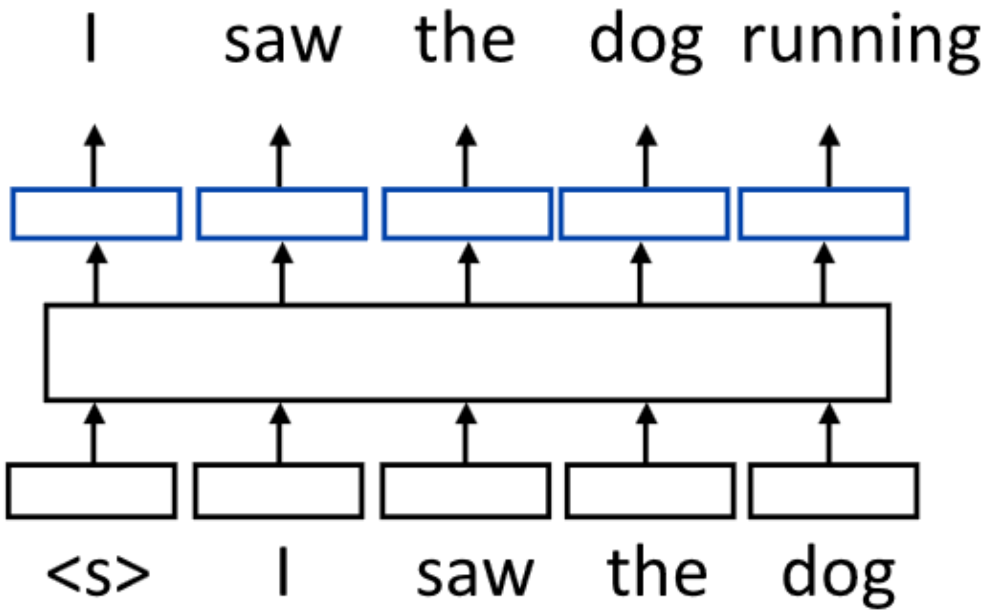
$$\mathcal{L}(\theta) = -\log p(x_i \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

Next token prediction



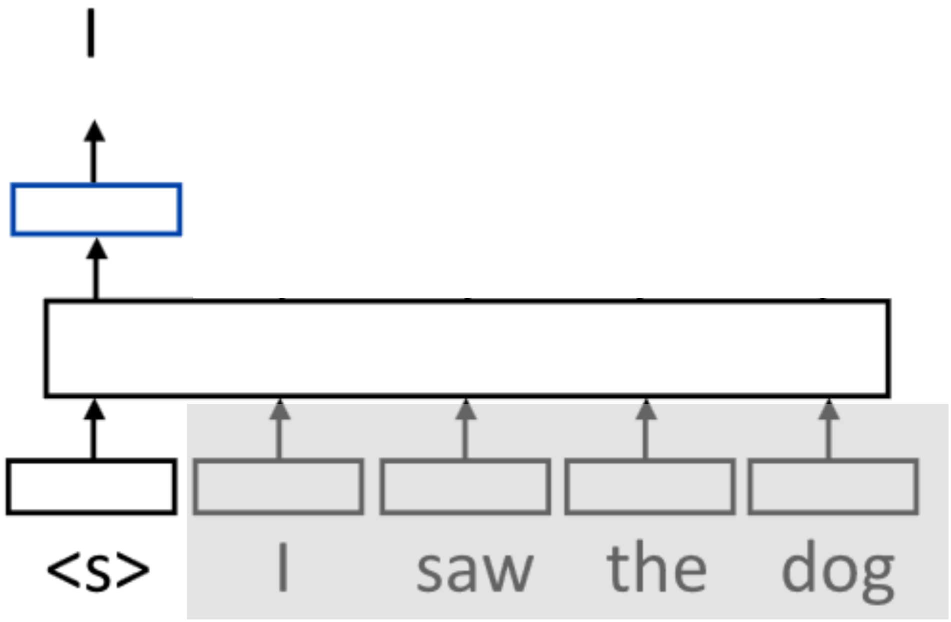
$$\mathcal{L}(\theta) = -\log p(x_i \mid \langle x_0 \ldots x_{i-1} \rangle; \theta)$$

Next token prediction in Transformers



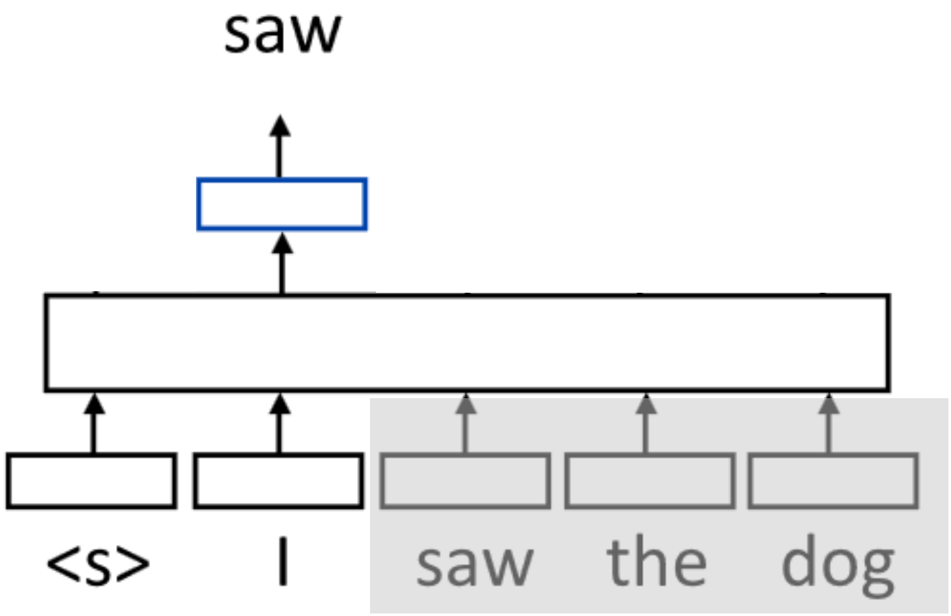
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



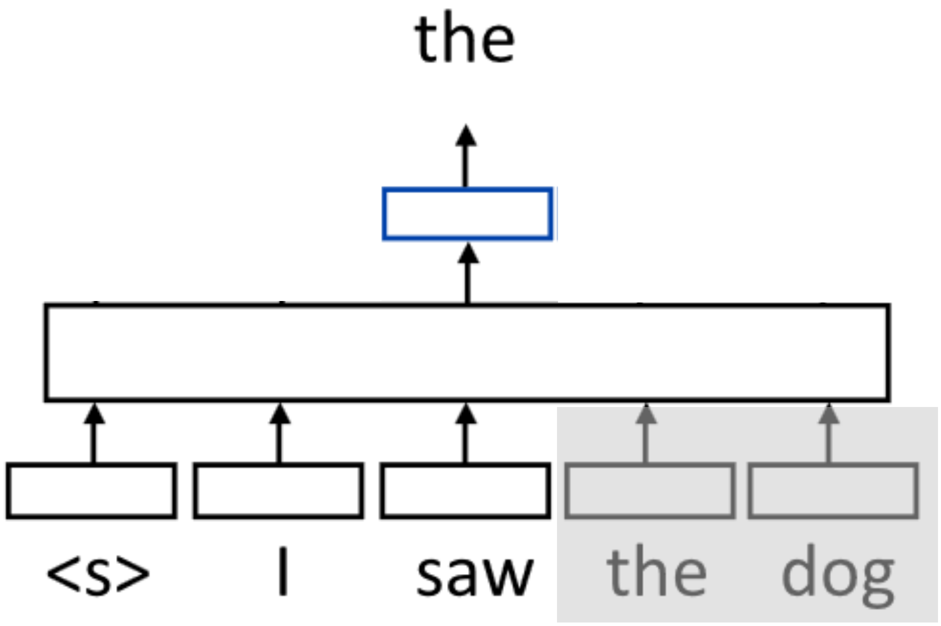
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



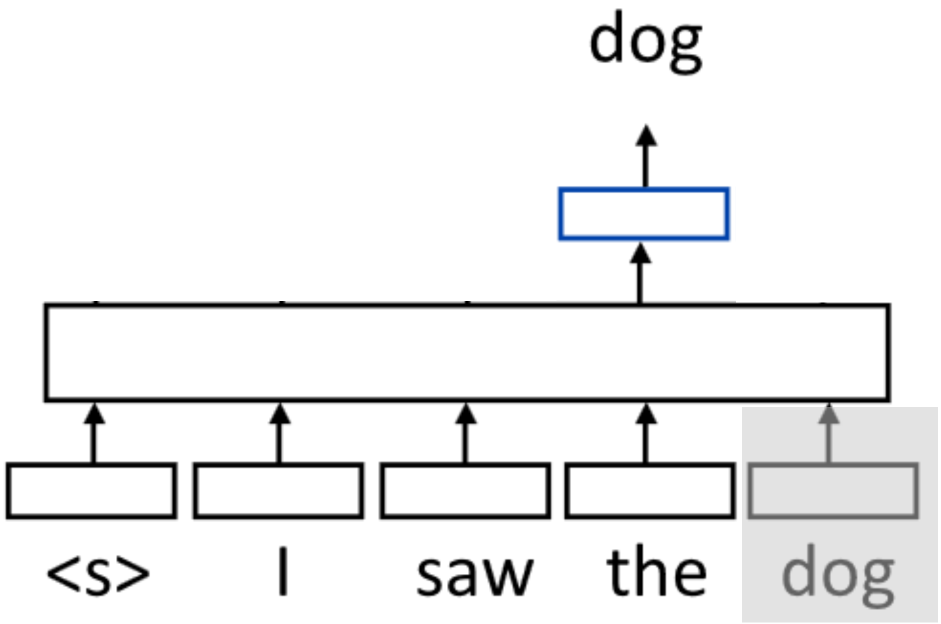
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



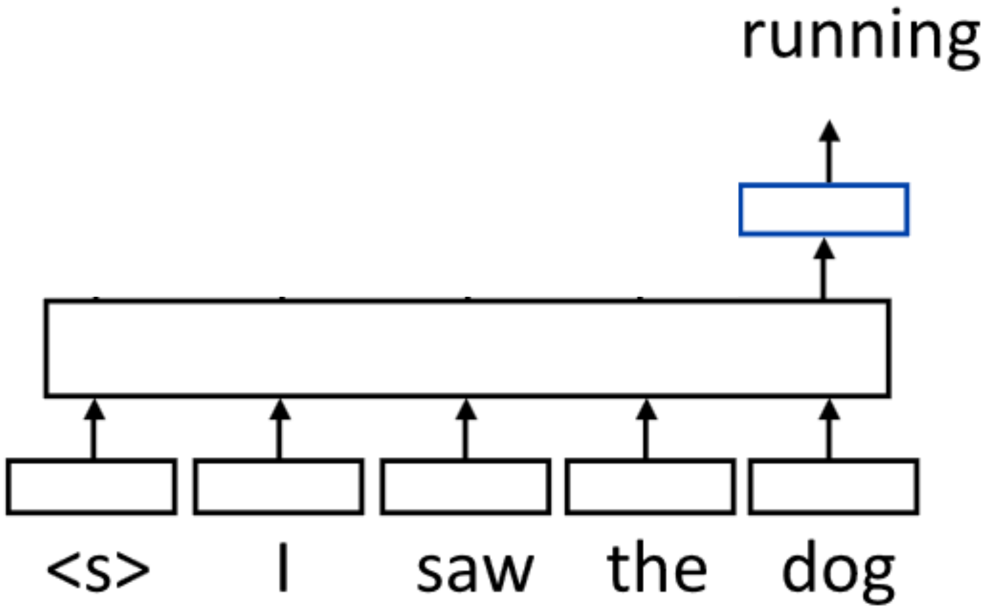
Slide from Greg Durrett, UT Austin CS 388

Next token prediction in Transformers



Slide from Greg Durrett, UT Austin CS 388

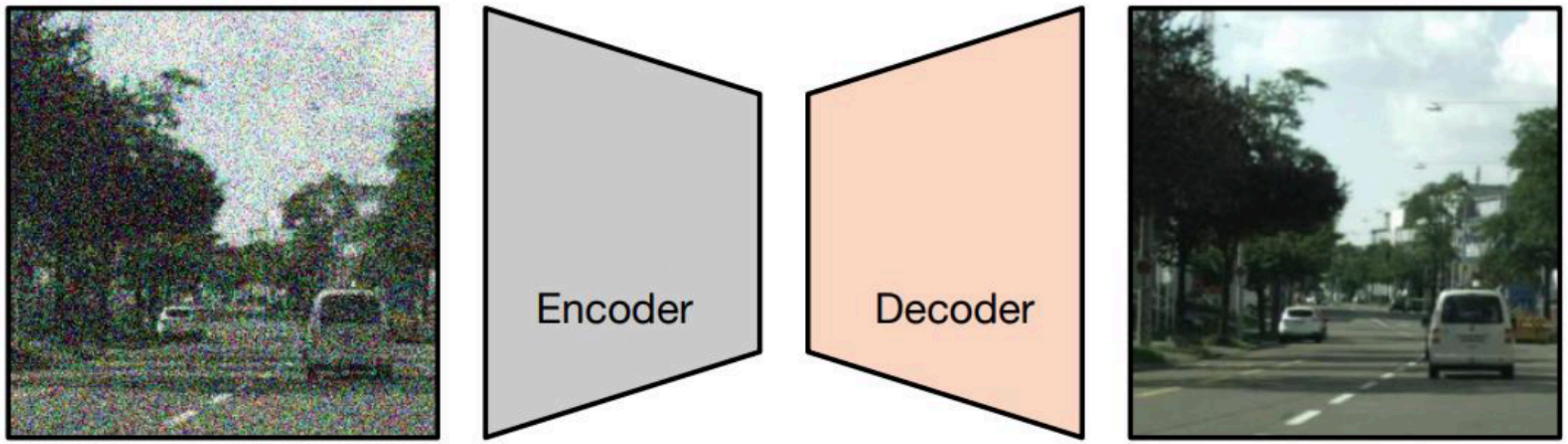
Next token prediction in Transformers



Slide from Greg Durrett, UT Austin CS 388

Denoising Objectives

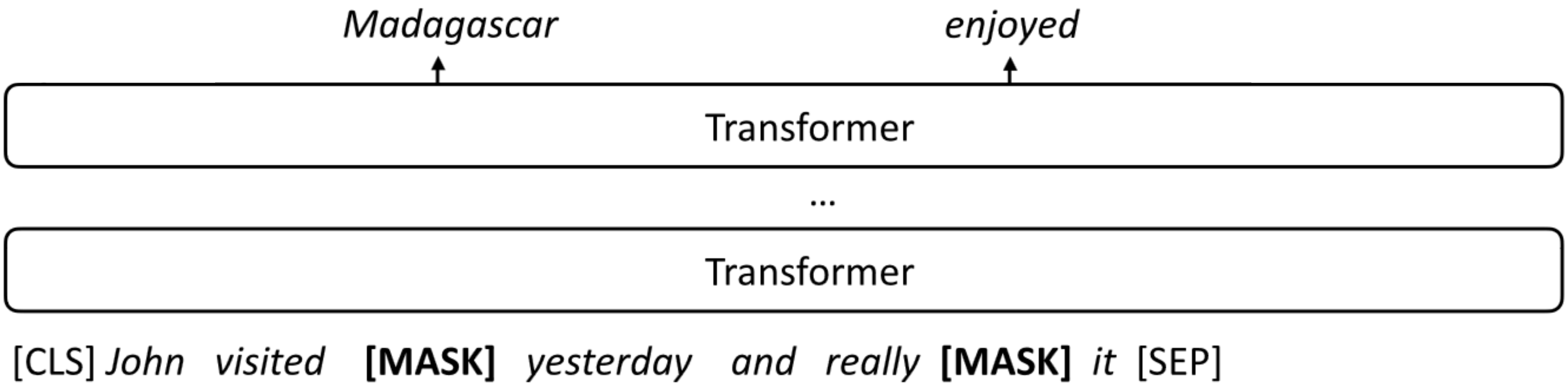
- Our goal: learn a distribution over text sequences
- Our assumption so far: this distribution is only backwards-looking (conditioned on prefix of the sequence)
- What if we remove this assumption?



Brempong et al. 2022, CVPR

Masking / Infilling Objectives

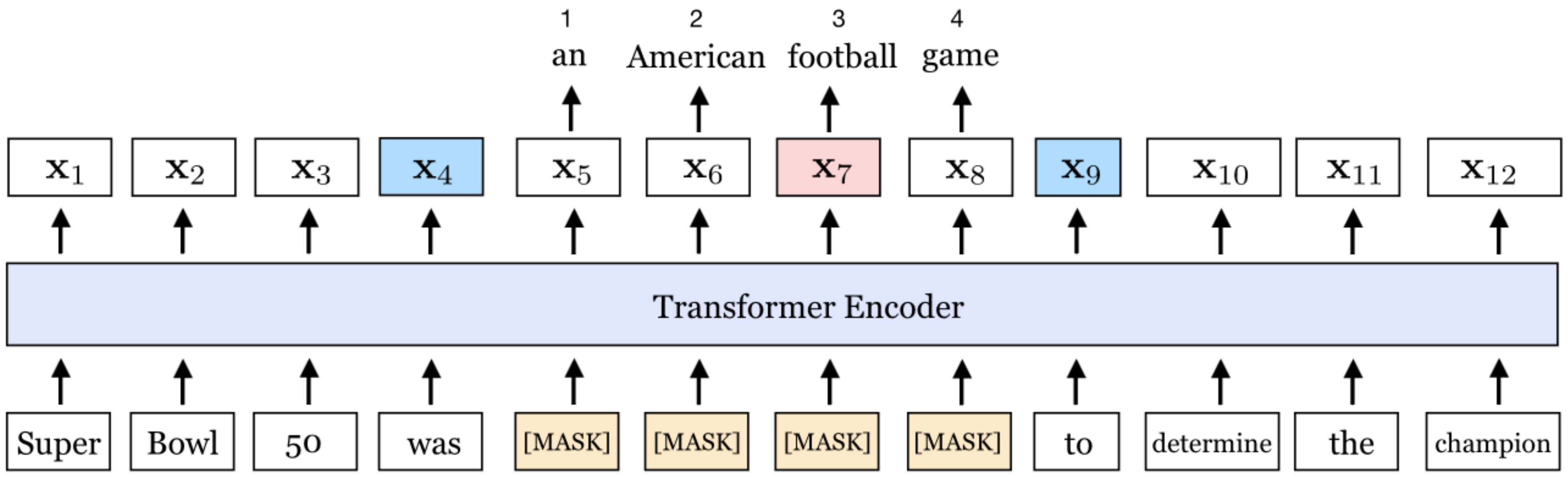
- Randomly mask out ~15% of tokens in the input, and try to predict them from past *and future* context



BERT, Devlin et al. 2019 (slide from UT Austin CS 388)

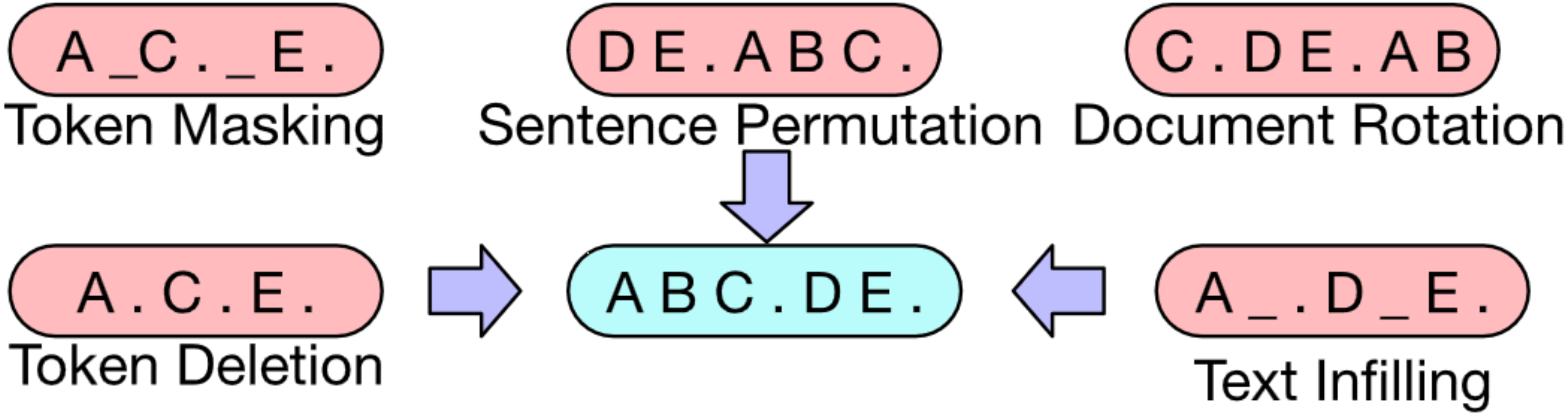
Masking / Infilling Objectives

- Randomly mask out ~15% of tokens in the input, and try to predict them from past *and future* context
- Or mask out spans of text



SpanBERT, Joshi et al. 2020 (TACL)

Auxiliary Objectives



BART, Lewis et al. 2019