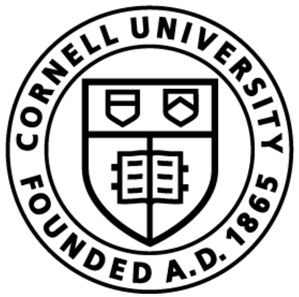
Lecture 8: Transformers cont.



Cornell Bowers CIS
Computer Science

Tanya Goyal

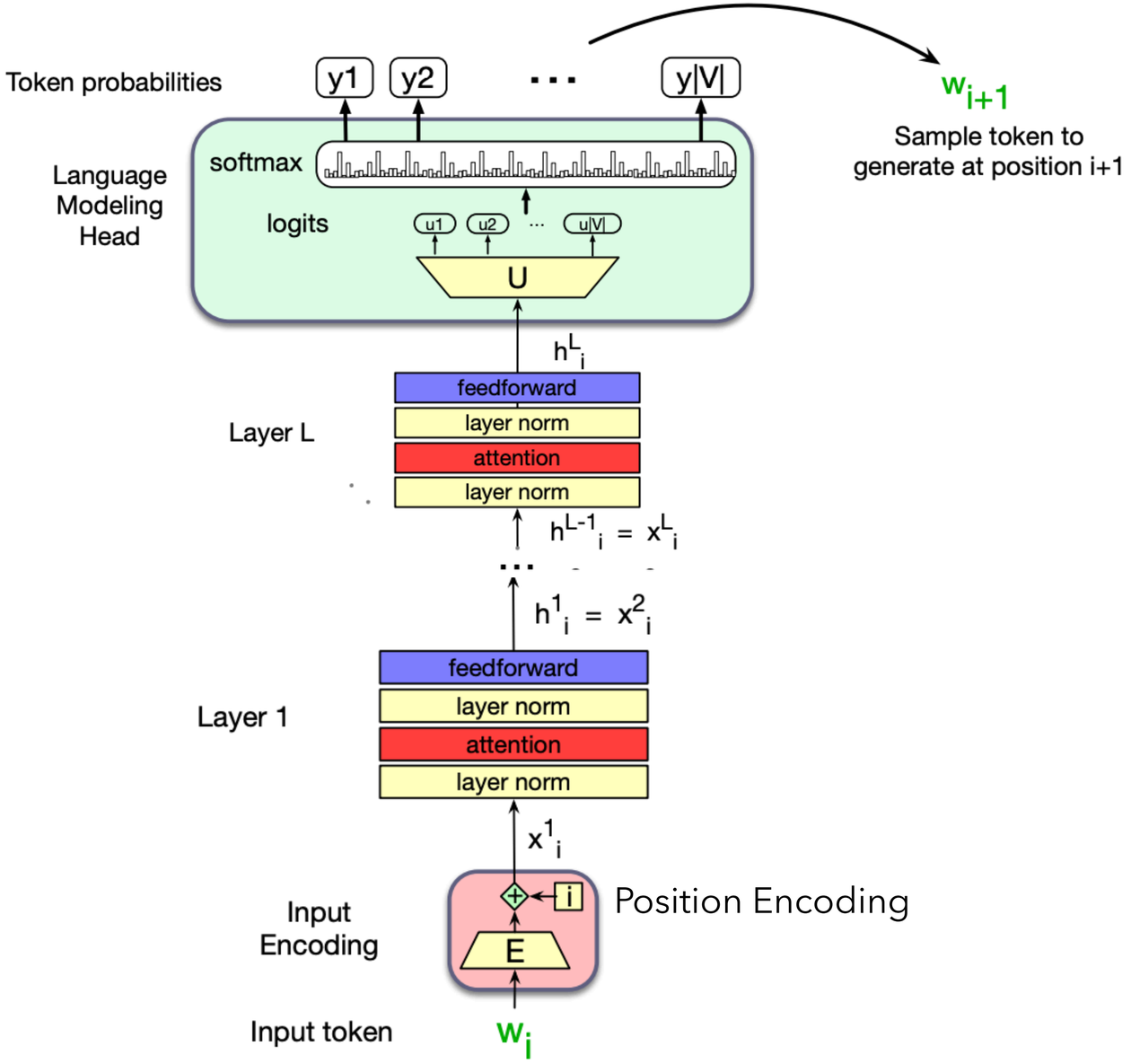
CS 4740 (and crosslists): Introduction to Natural Language Processing

Upcoming deadlines + Today

- **Grade Releases**
  - Midterm grades released. Please file regrades by **March 23, 11.59 p.m.**
  - HW3 will be released tonight/tomorrow morning.

- **Today**
  - Recap: Transformers Attention
  - Transformers

Transformer Architecture (Decoder-only)



- We will build up to this!!
- Main components of a transformer model
  - **(Multi-head) Attention**
  - Feed forward
  - Layer Norm
  - Position Encoding

Recap: Attention in Transformer models

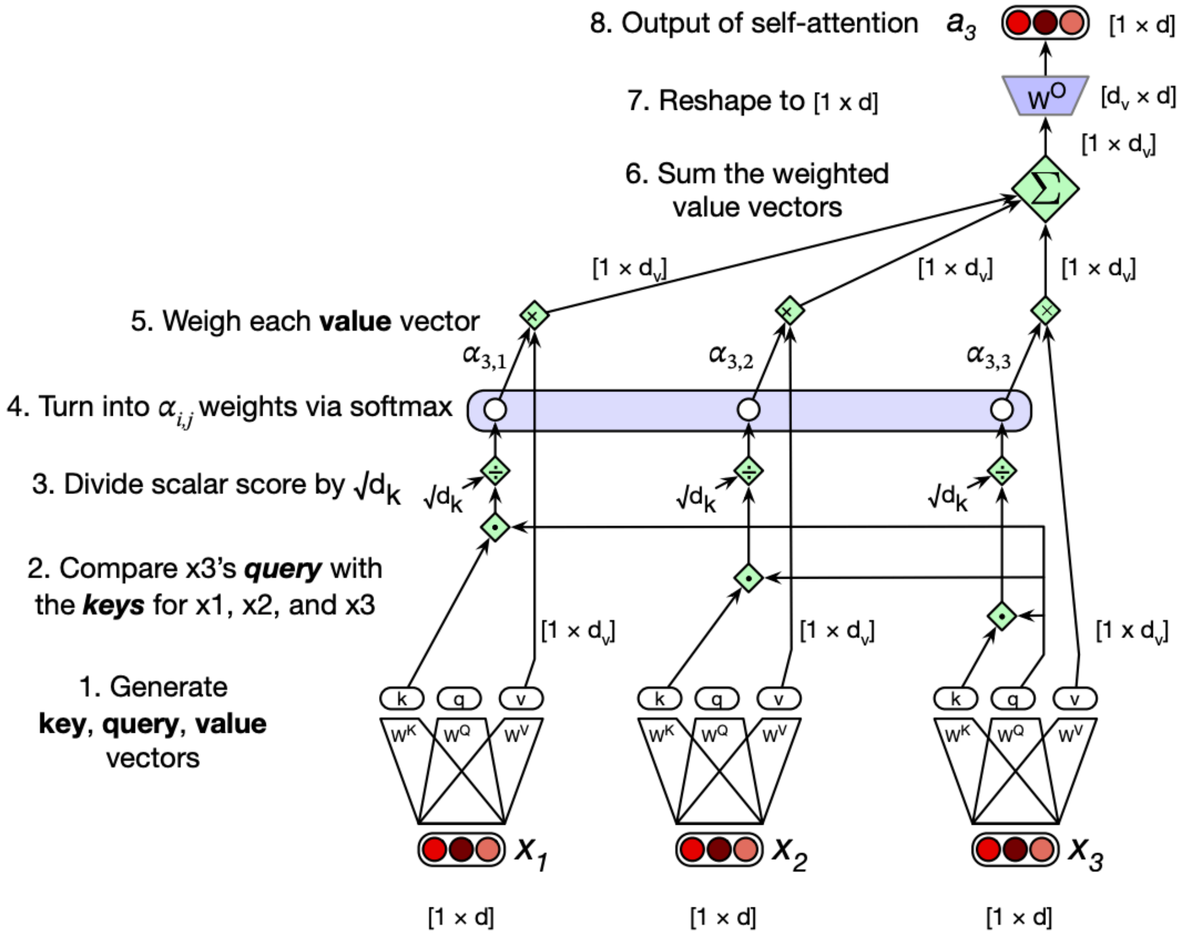


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Recap: Attention in Transformer models

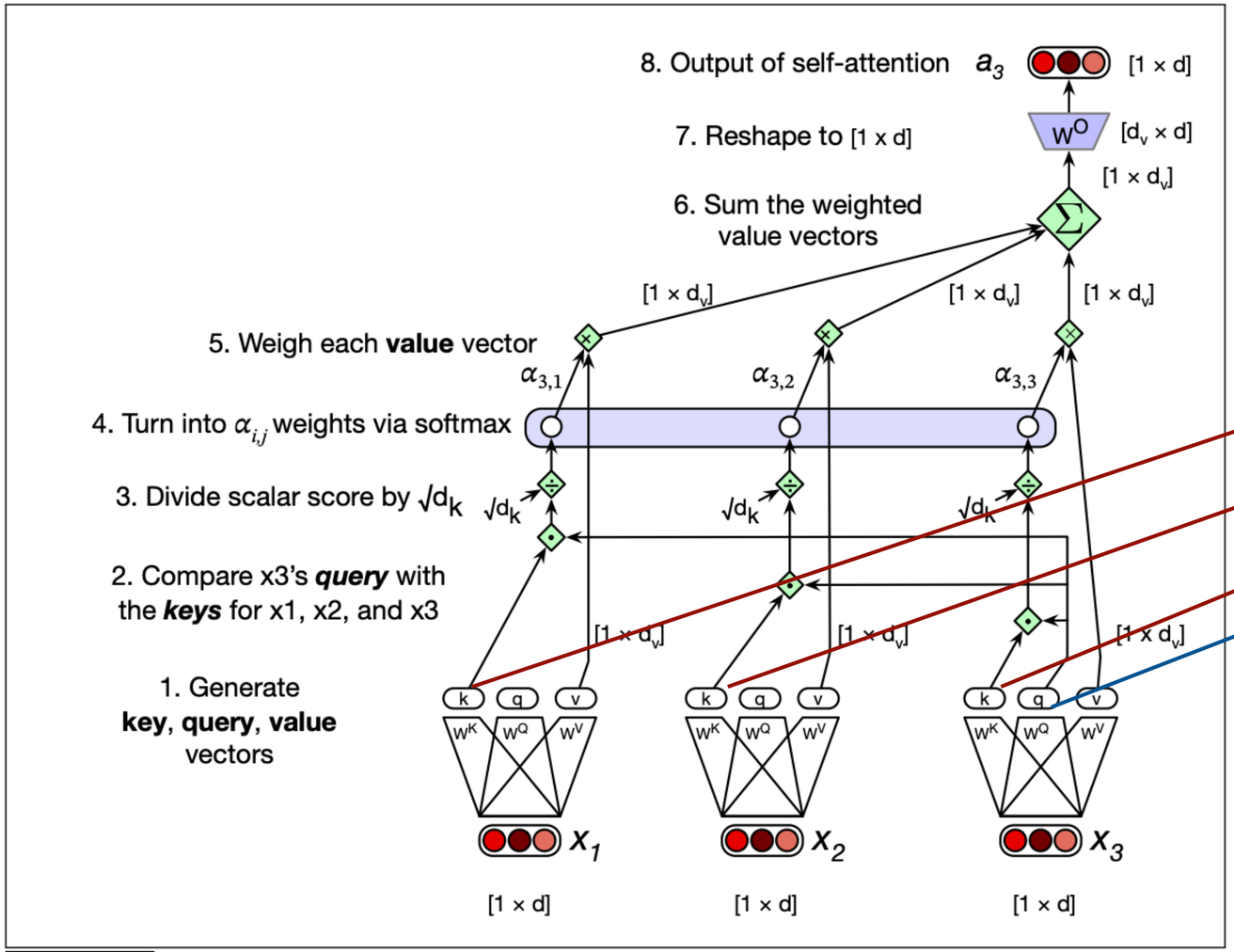
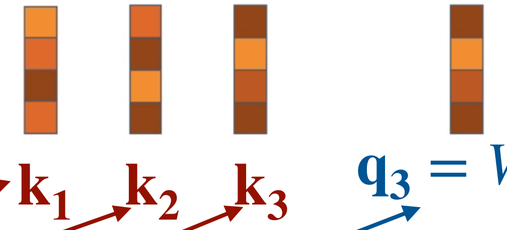


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Computation at time step 3, ie. $\mathbf{a}_3$

Step1: prepare inputs



$\mathbf{k}_1$ $\mathbf{k}_2$ $\mathbf{k}_3$ $\mathbf{q}_3 = W^Q\mathbf{x}_3$

Recap: Attention in Transformer models

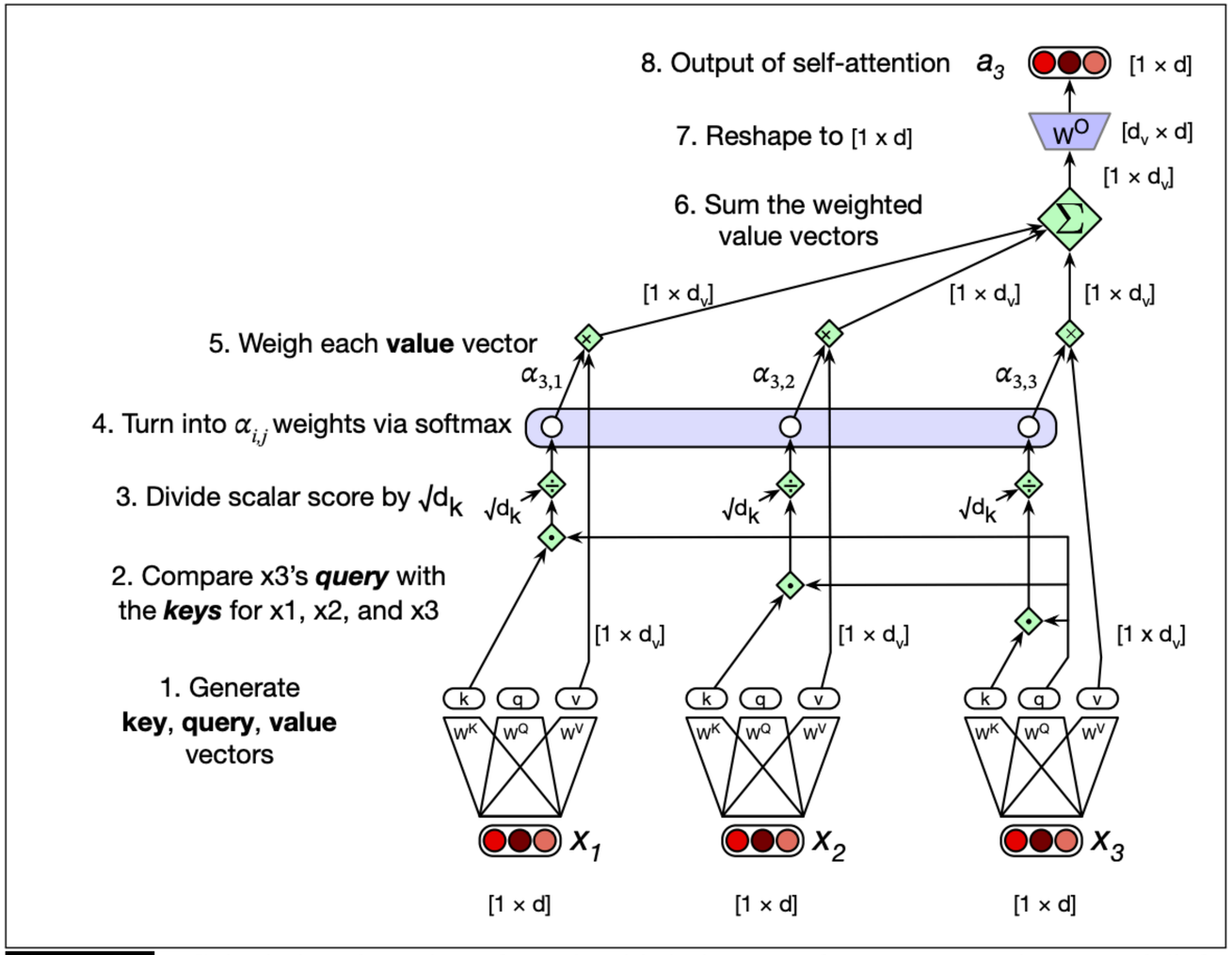
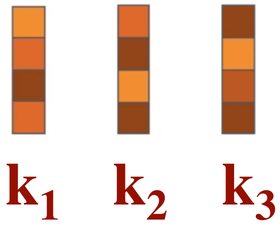


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Computation at time step 3, ie. $\mathbf{a}_3$

Step1: prepare inputs



$\mathbf{k}_1$ $\mathbf{k}_2$ $\mathbf{k}_3$ $\mathbf{q}_3 = W^Q\mathbf{x}_3$

Step2: compute scores

9	9	13

$$\frac{\mathbf{q}_3 \cdot \mathbf{k}_1}{\sqrt{d_k}}$$

d_k is dim of query, key vectors

Step3: softmax scores (Attention weights)

0.02	0.02	0.96

Recap: Attention in Transformer models

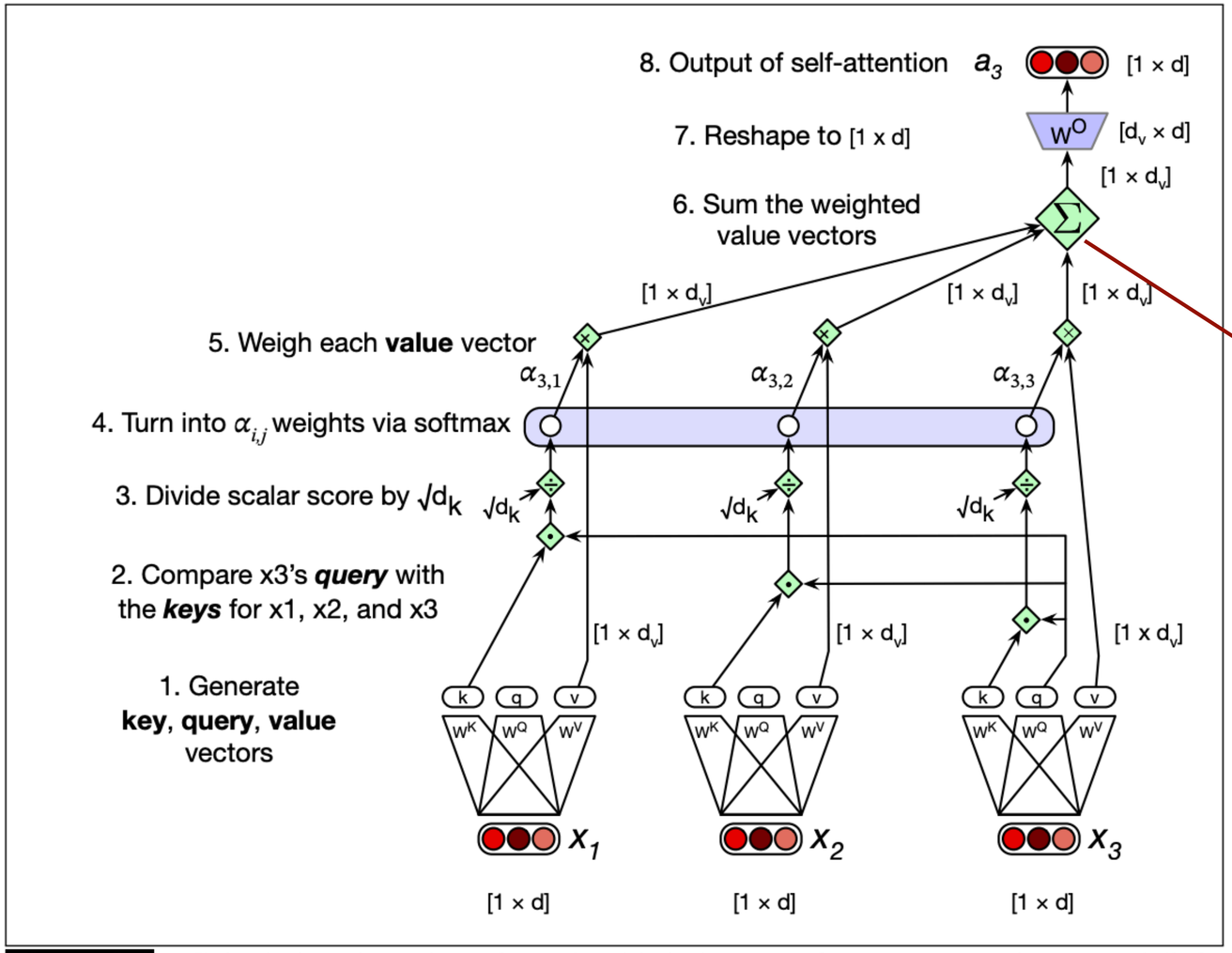
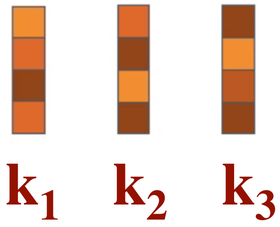


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Computation at time step 3, ie. $\mathbf{a}_3$

Step1: prepare inputs



$\mathbf{k}_1$ $\mathbf{k}_2$ $\mathbf{k}_3$ $\mathbf{q}_3 = W^Q\mathbf{x}_3$

Step2: compute scores

9	9	13

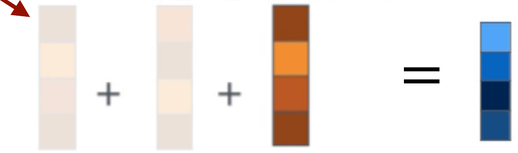
$$\frac{\mathbf{q}_3 \cdot \mathbf{k}_1}{\sqrt{d_k}}$$

Step3: softmax scores (Attention weights)

0.02	0.02	0.96

Step4: multiply each vector by softmax scores

$\alpha_{31}\mathbf{v}_1$ $\alpha_{32}\mathbf{v}_2$ $\alpha_{33}\mathbf{v}_3$



Recap: Attention in Transformer models

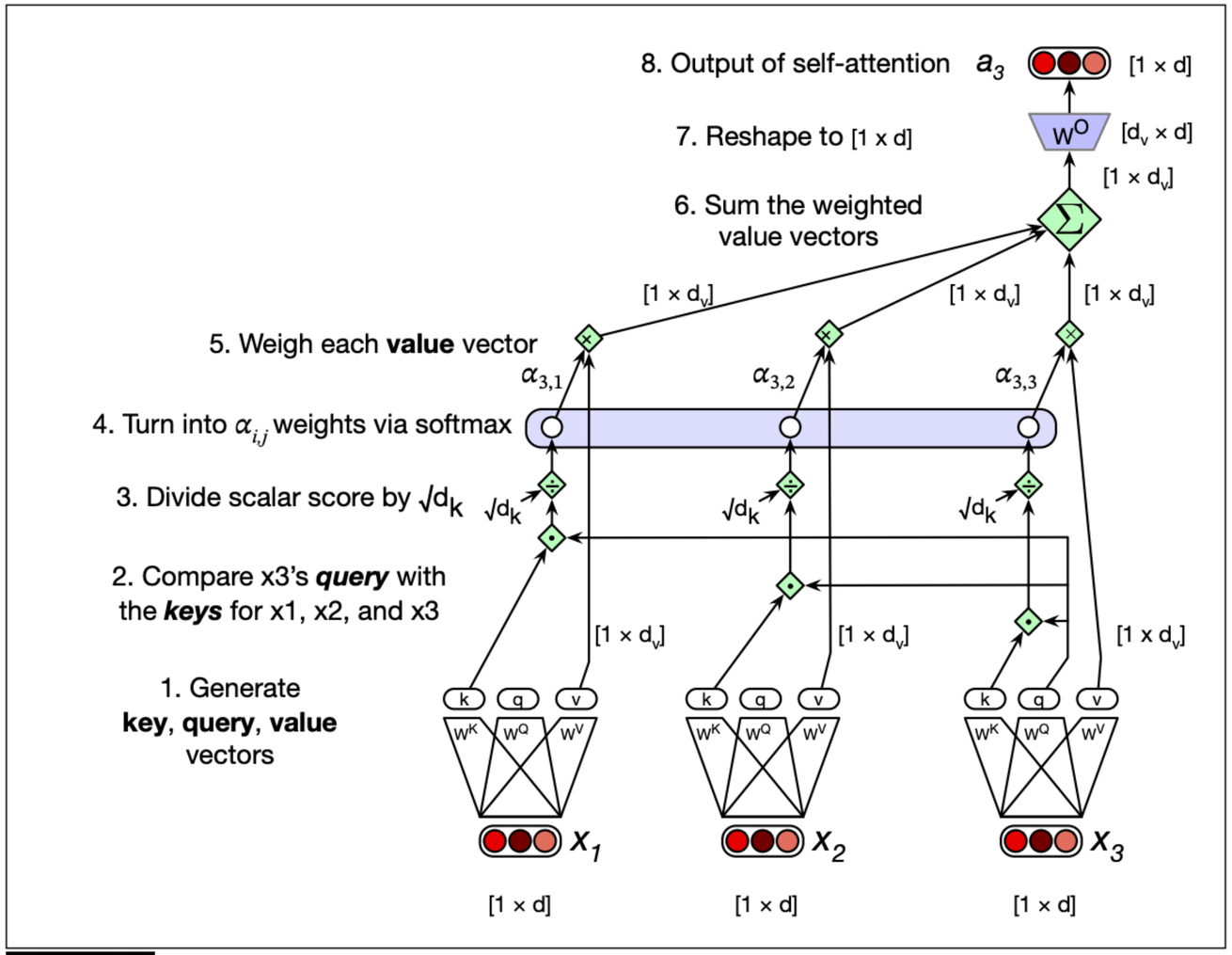
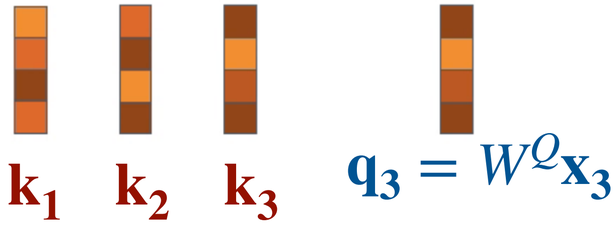


Figure 9.4 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Computation at time step 3, ie. $\mathbf{a}_3$

Step1: prepare inputs



Step2: compute scores

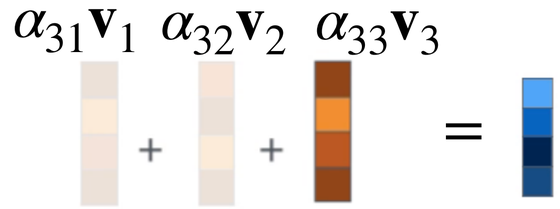
9	9	13

$$\frac{\mathbf{q}_3 \cdot \mathbf{k}_1}{\sqrt{d_k}}$$

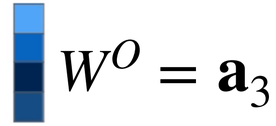
Step3: softmax scores (Attention weights)

0.02	0.02	0.96

Step4: multiply each vector by softmax scores



Step5: sum up the weighted vectors **and project**



Recal: Multi-headed Attention

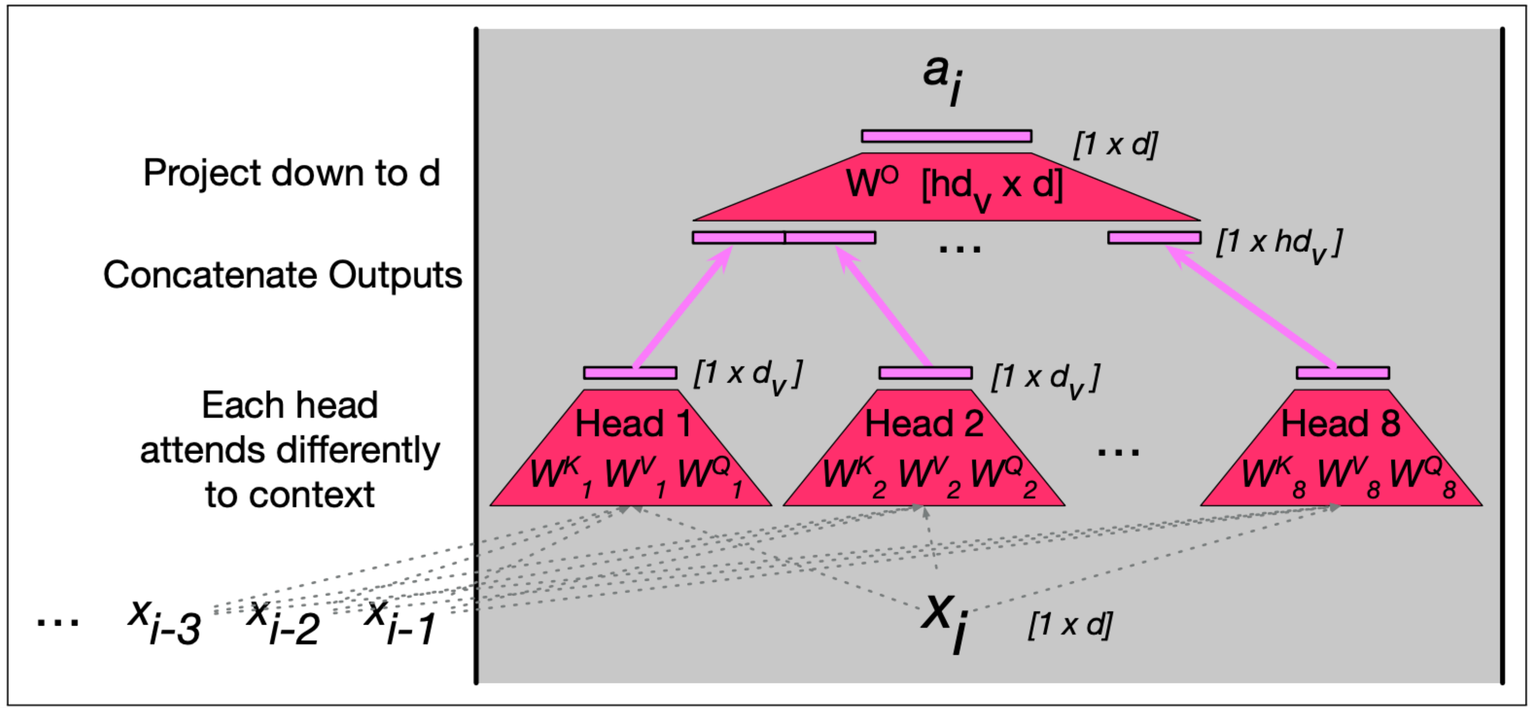


Figure 9.5 The multi-head attention computation for input $\mathbf{x}_i$, producing output $\mathbf{a}_i$. A multi-head attention layer has A heads, each with its own key, query and value weight matrices. The outputs from each of the heads are concatenated and then projected down to d, thus producing an output of the same size as the input.

- Multiple heads –> multiple ``independent'' projections (keys, queries, values) for each input.
- Each head has different W^Q, W^K, W^V matrices

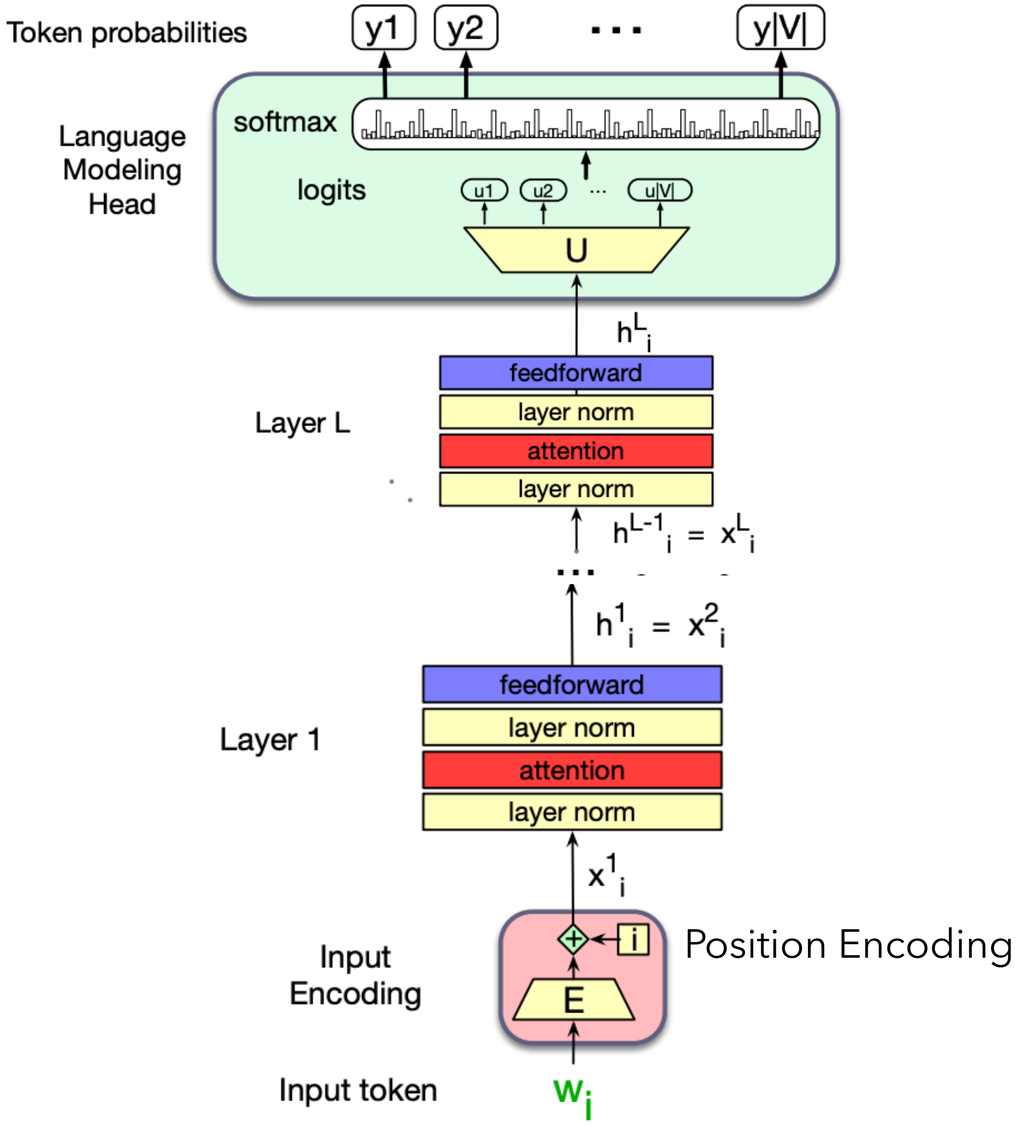
- Different heads can potentially capture different phenomenon.

Attention Visualization

BERTViz: https://github.com/jessevig/bertviz

[Caveat] BERT is an encoder, all tokens attend to all other tokens (no causal mask)

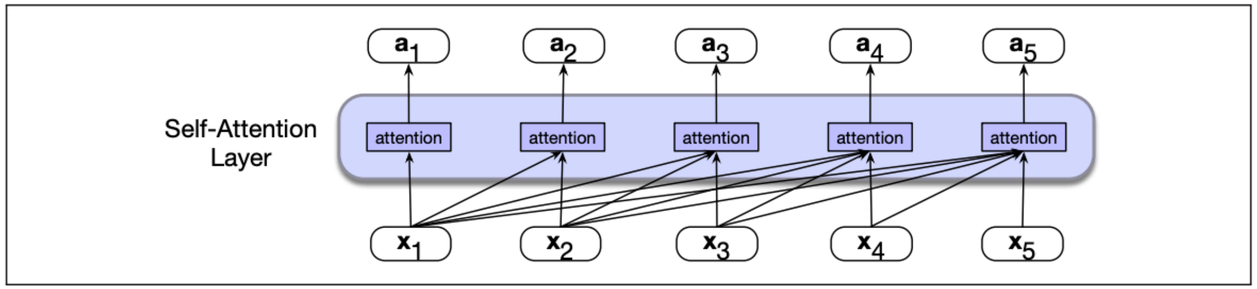
Let's go back to our transformer arch



- Last Lecture:
  - Multi-head self-attention

- This Lecture:
  - Position Embeddings
  - Residual connections
  - Layer Norm
  - Feedforward layer
  - Putting it all together
  - Encoder Decoder

Zooming out

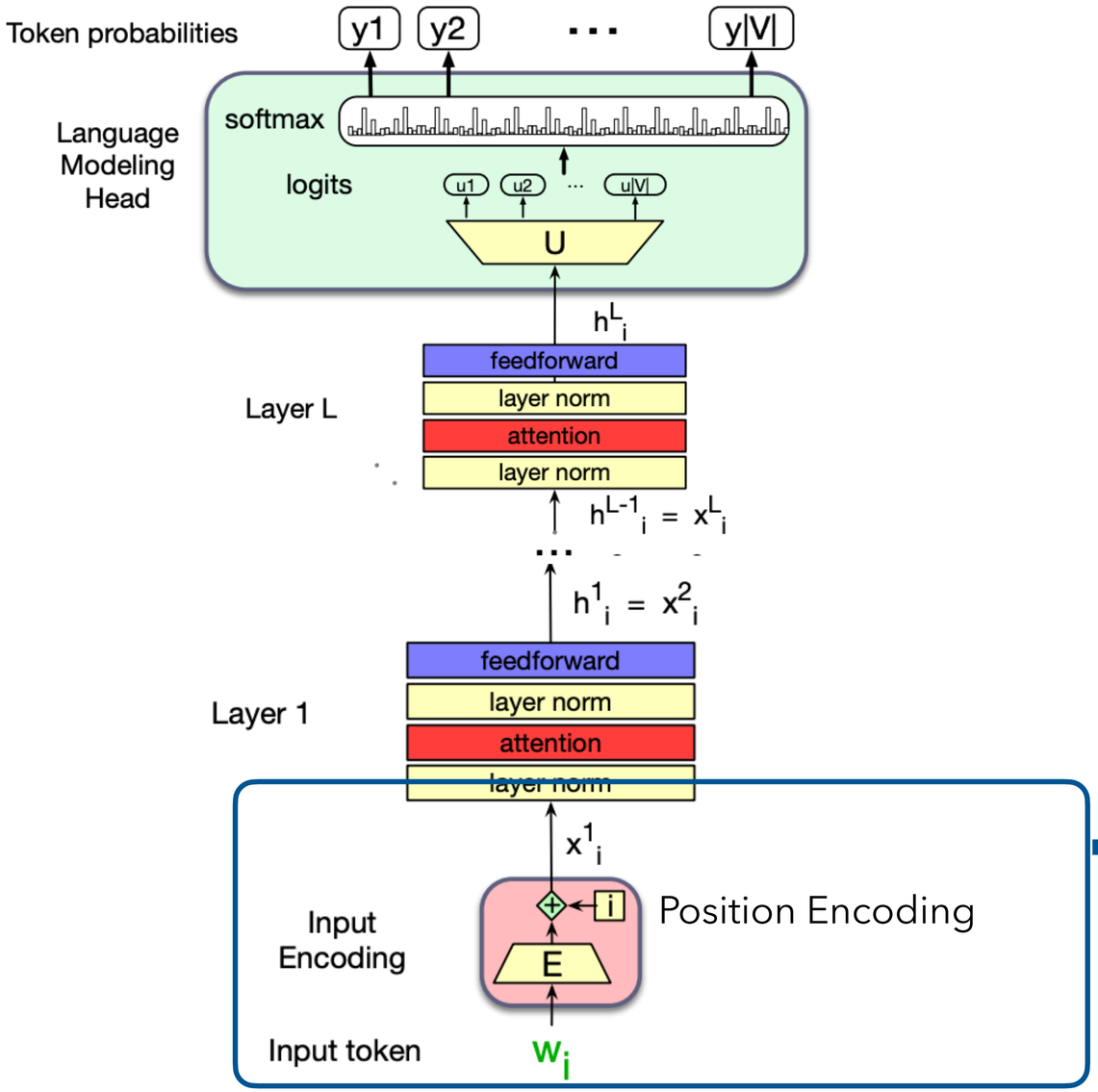


- Self-attention layer transformed the input $\mathbf{x_i}$ to output $\mathbf{a_i}$
- Word order information is lost!

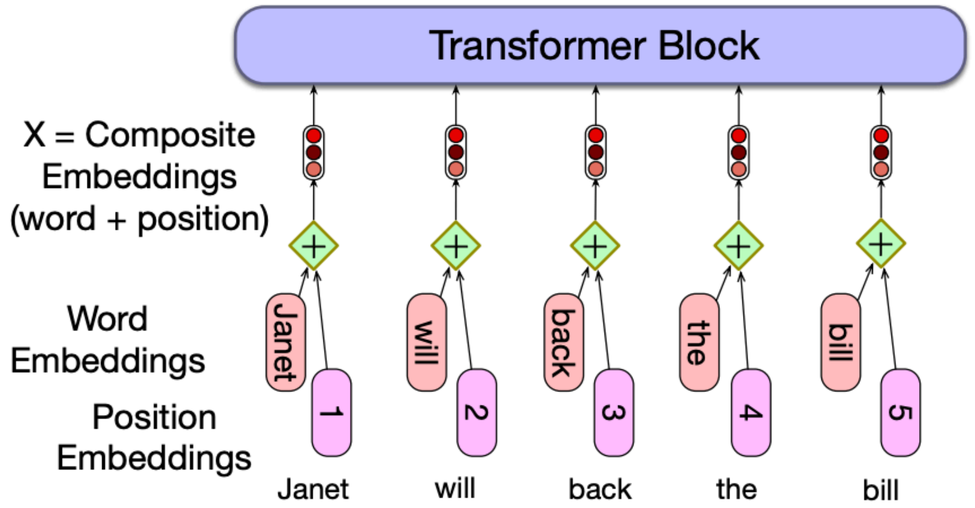
An old dog and a young boy ….

- *boy* attends to both *old* and *young*. We want *young* to have a higher influence on *boy*'s hidden representation than *old*. Attention does not ensure this.

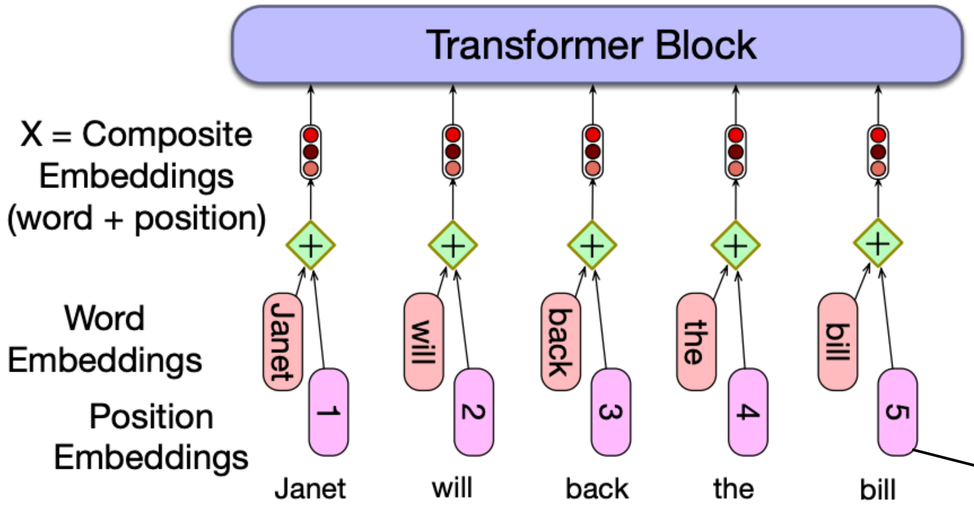
Position Embeddings



- Solution: Element-wise add a “position" embedding to the word embedding to produce a new embedding of the same dimension.



Position Embeddings



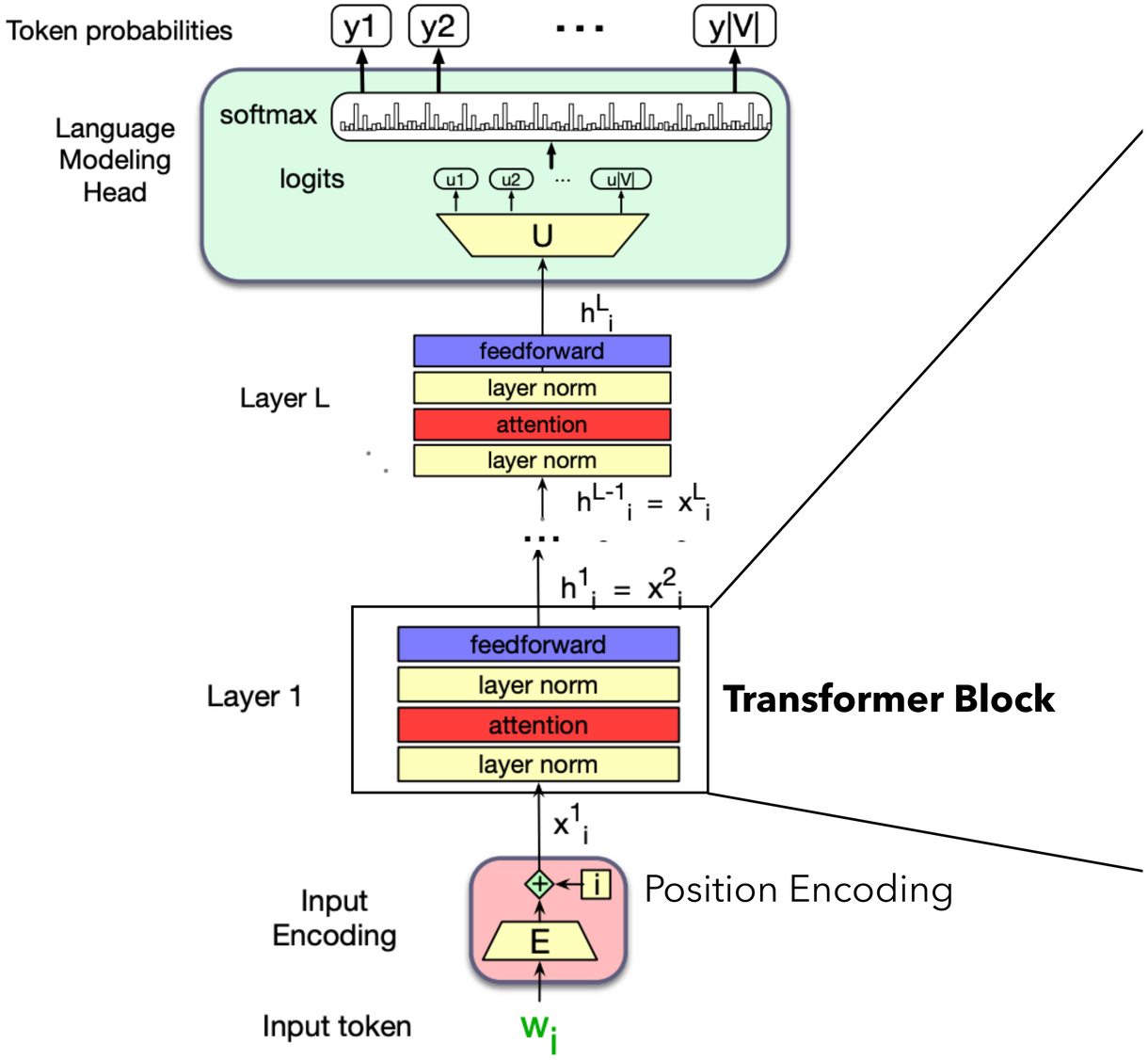
- Solution: Element-wise a "position" embedding to the word embedding to produce a new embedding of the same dimension.

Vectors of real numbers with the same size as word embeddings.

How do we get these positional embeddings?

- Assume all sequences will have length between 0 to N (say 512). Randomly initialize embeddings for each position.
- These will get trained with other transformer parameters.

Let's go back to our transformer arch



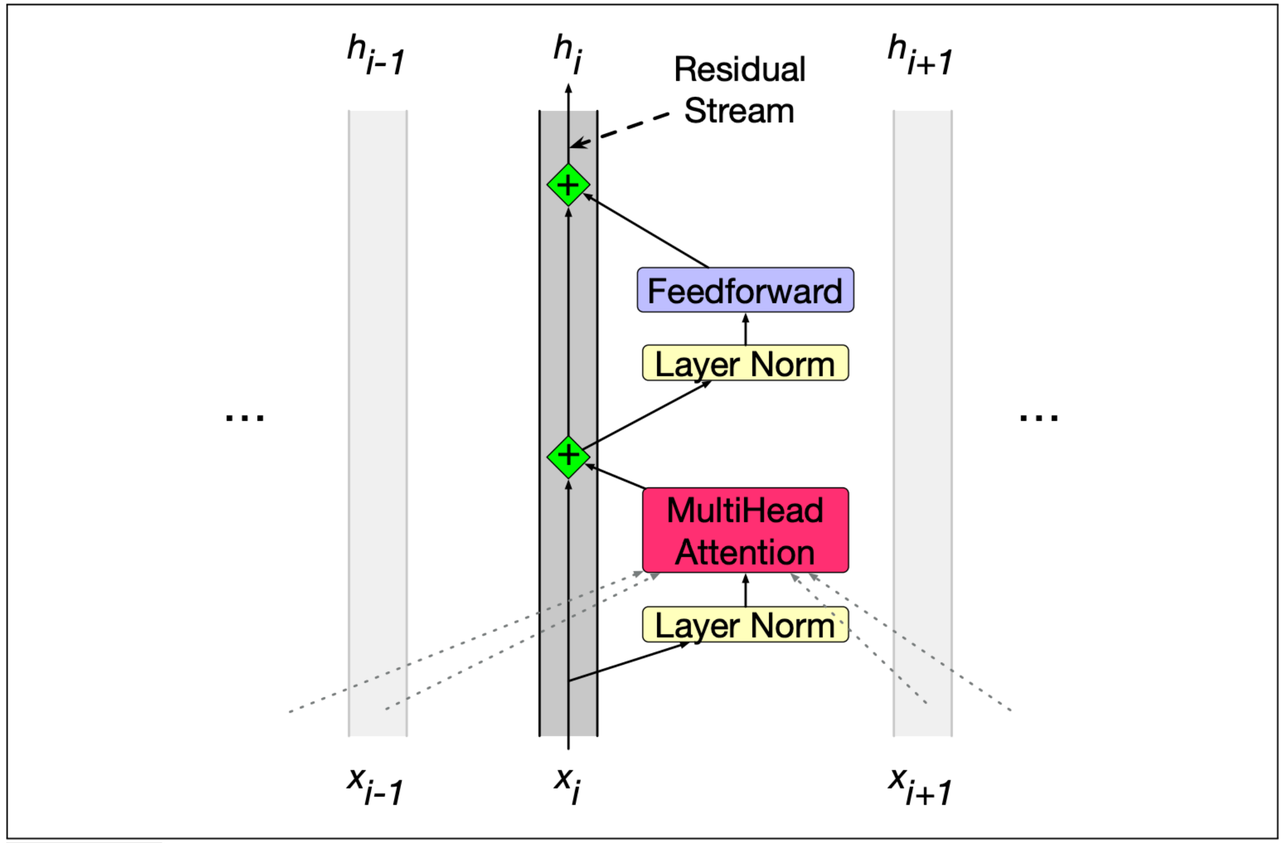


Figure 9.6 **Residual stream view of Transformer Block**

Residual Stream view

- Input x_i (d-dim vector)
  - Layer Norm + Multi-head Attention perform some computation, produce another d-dim vector.
  - Add to the residual stream $x_i \rightarrow x'_i$
- x'_i (d-dim vector)
  - Layer Norm + Feedforward produces another d-dim vector
  - Add to the residual stream $x'_i \rightarrow h_i$
- Output h_i

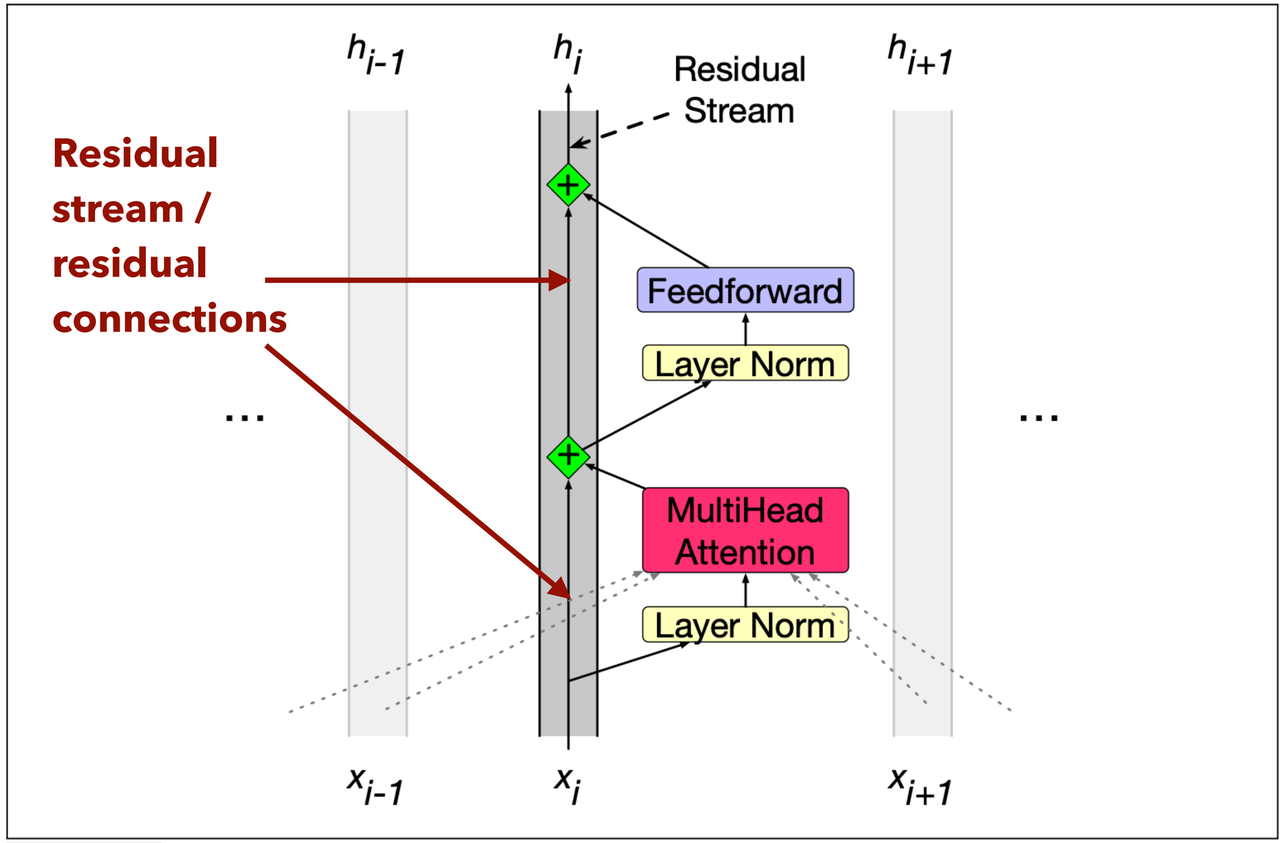


Figure 9.6 **Residual stream view of Transformer Block**

Feedforward layers

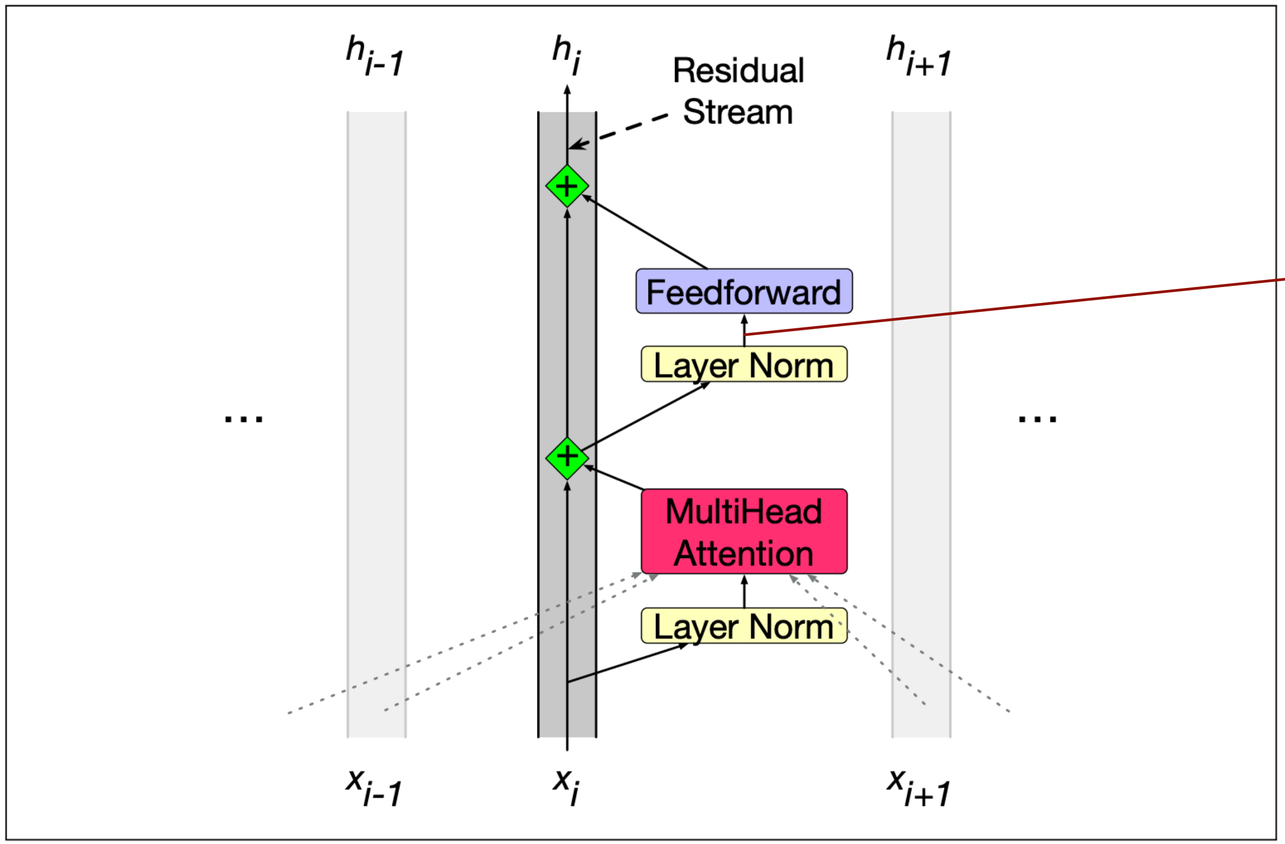


Figure 9.6 **Residual stream view of Transformer Block**

- Fully-connected 2-layer network

$$\text{FFNN}(x_i'') = \text{RELU}(x_i''\mathbf{W_1} + b_1)\mathbf{W_2} + b_2$$

Layer Norm(alization)

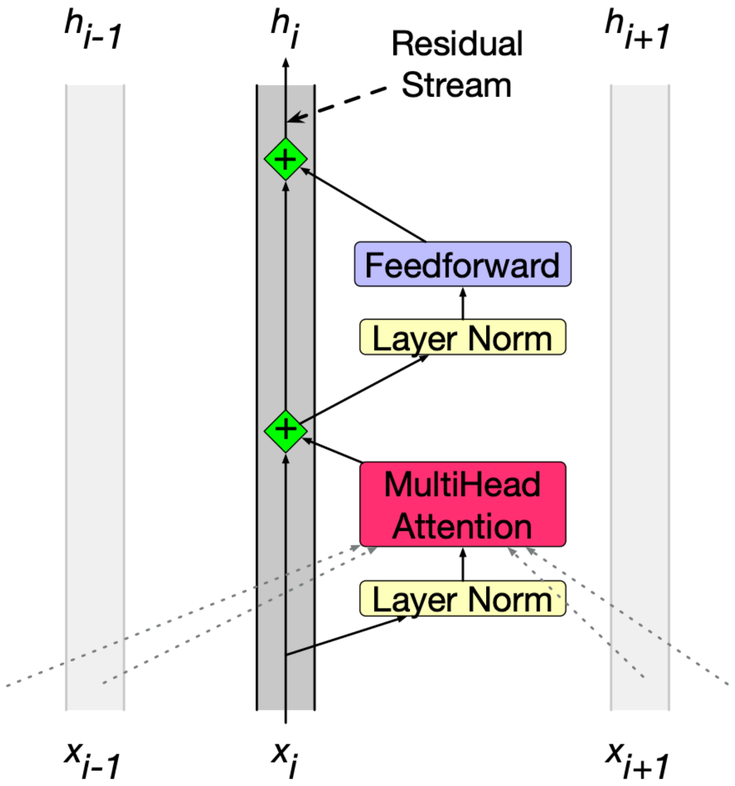
- *Normalize* the input vector.
- **Not** applied to the entire transformer layer, applied to single token vectors in isolation.
- Given input a of dimension d, $\mu = \frac{1}{d}\sum_{i=1}^{d} a_i \quad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(a_i - \mu)^2}$

$$\hat{a} = \frac{a - \mu}{\sigma}$$

$$\text{LayerNorm}(a) = \gamma\frac{(a - \mu)}{\sigma} + \beta \qquad \gamma, \beta \text{ are learnable parameters}$$

How many additional parameters does layer norm add?

Transformer Block: Putting it all together



...

Input x_i at time step i

$$t_i^1 = \text{LayerNorm}(x_i)$$

$$t_i^2 = \text{MultiHead-Attention}(t_i^1, [t_1^1, t_2^1, \ldots t_N^1])$$

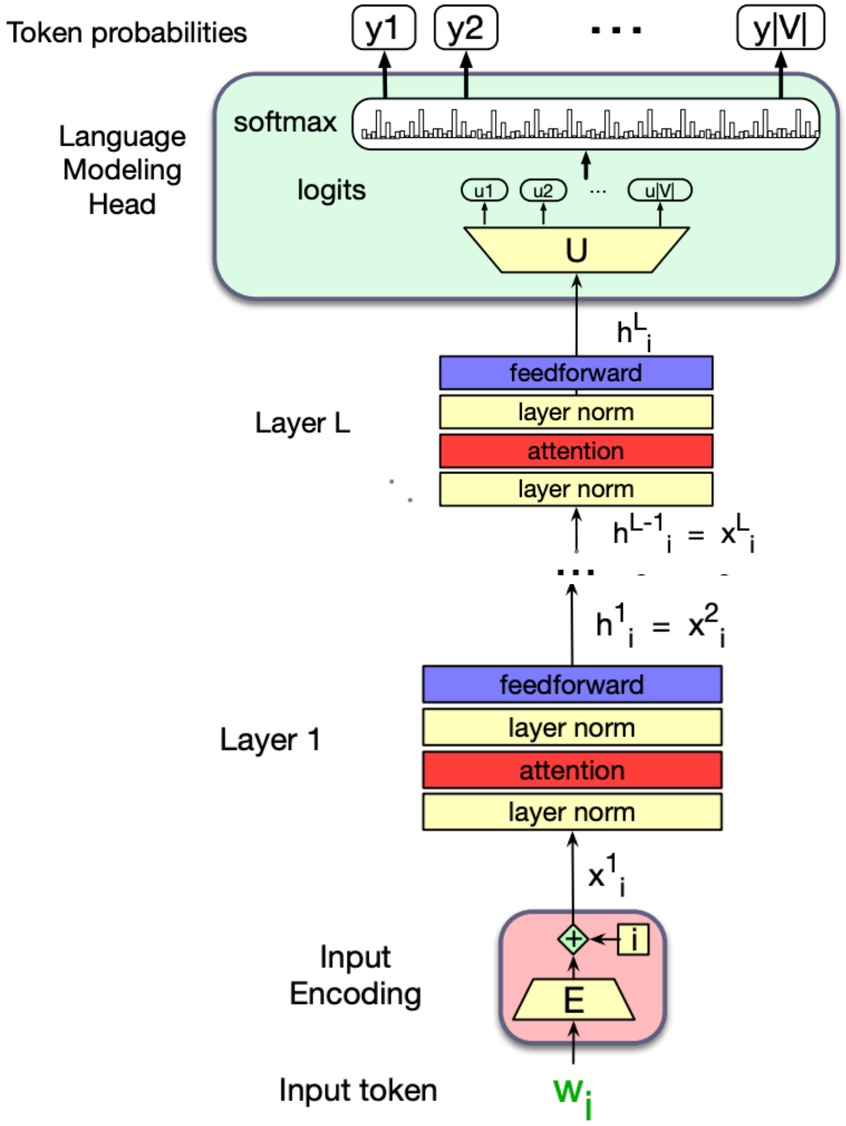
$$t_i^3 = t_i^2 + x_i$$

$$t_i^4 = \text{LayerNorm}(t_i^3)$$

$$t_i^5 = \text{FFNN}(t_i^4)$$

$$h_i = t_i^5 + t_i^3$$

Zooming out



Same transformer blocks repeated N times.

- GPT2 was a family of 4 different models with the same architecture.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Large Language Models

- Decoder-only transformer models allows us to model the task of language modeling. Why should we care about language modeling?
- Many practical tasks in NLP can be cast as next token prediction.

Sentiment Analysis:

The sentiment of the sentence “I like Jackie Chan" is:

P(positivelThe sentiment of the sentence “I like Jackie Chan" is:)
P(negativelThe sentiment of the sentence “I like Jackie Chan" is:)

Large Language Models

- Decoder-only transformer models allows us to model the task of language modeling. Why should we care about language modeling?
- Many practical tasks in NLP can be cast as next token prediction.

Question Answering:

Q: Who wrote the book “The Origin of Species"? A:

P(w | Q: Who wrote the book “The Origin of Species"? A:)

Large Language Models

- Decoder-only transformer models allows us to model the task of language modeling. Why should we care about language modeling?
- Many practical tasks in NLP can be cast as next token prediction.

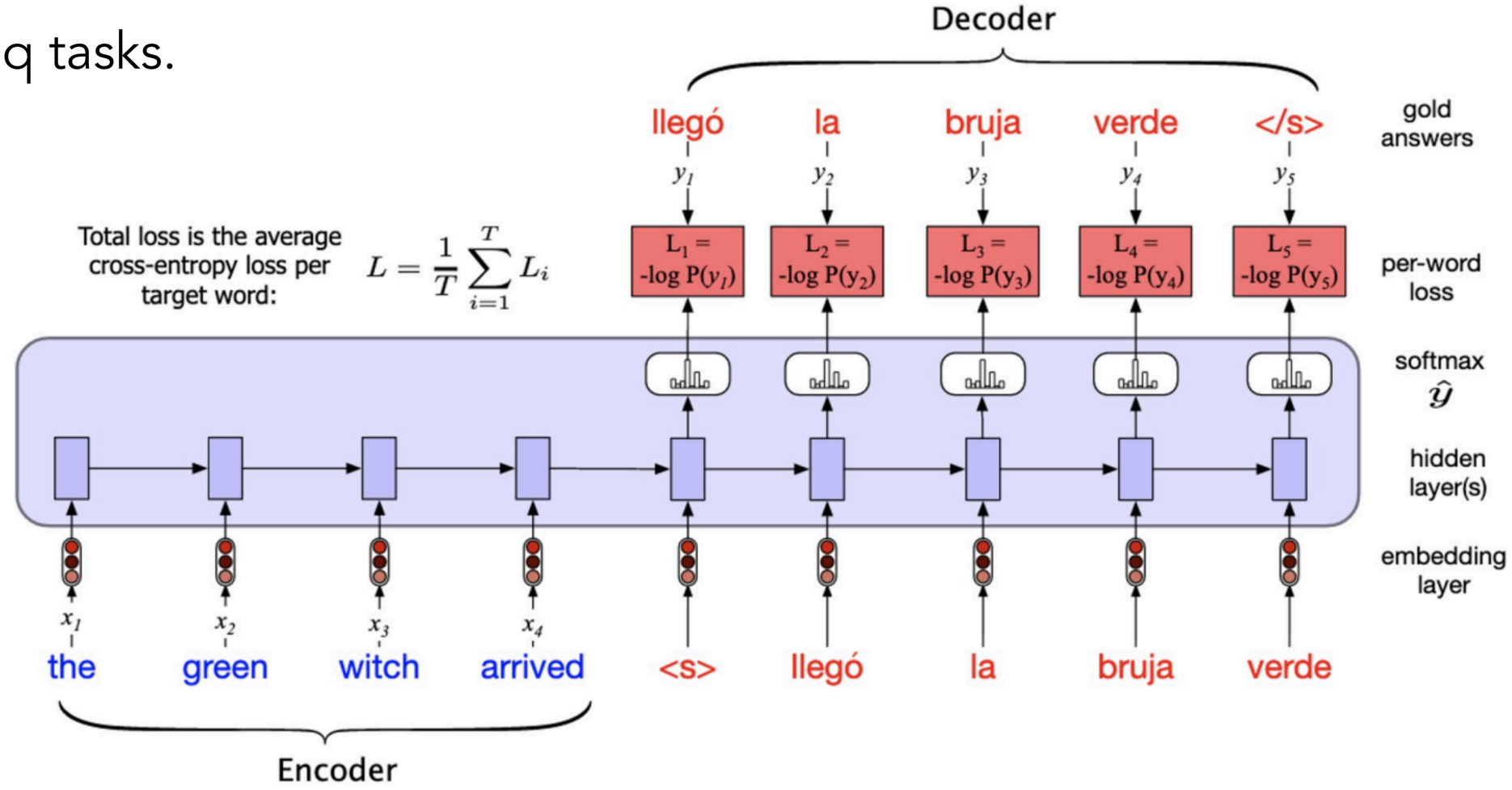
Language models need to be very powerful perform well at all these tasks!

- **Very** deep network
- Train on a lot of data

    - E.g. GPT-3 model (released in 2020) trained on 300B tokens, LLaMA-3 model trained on 15T tokens.

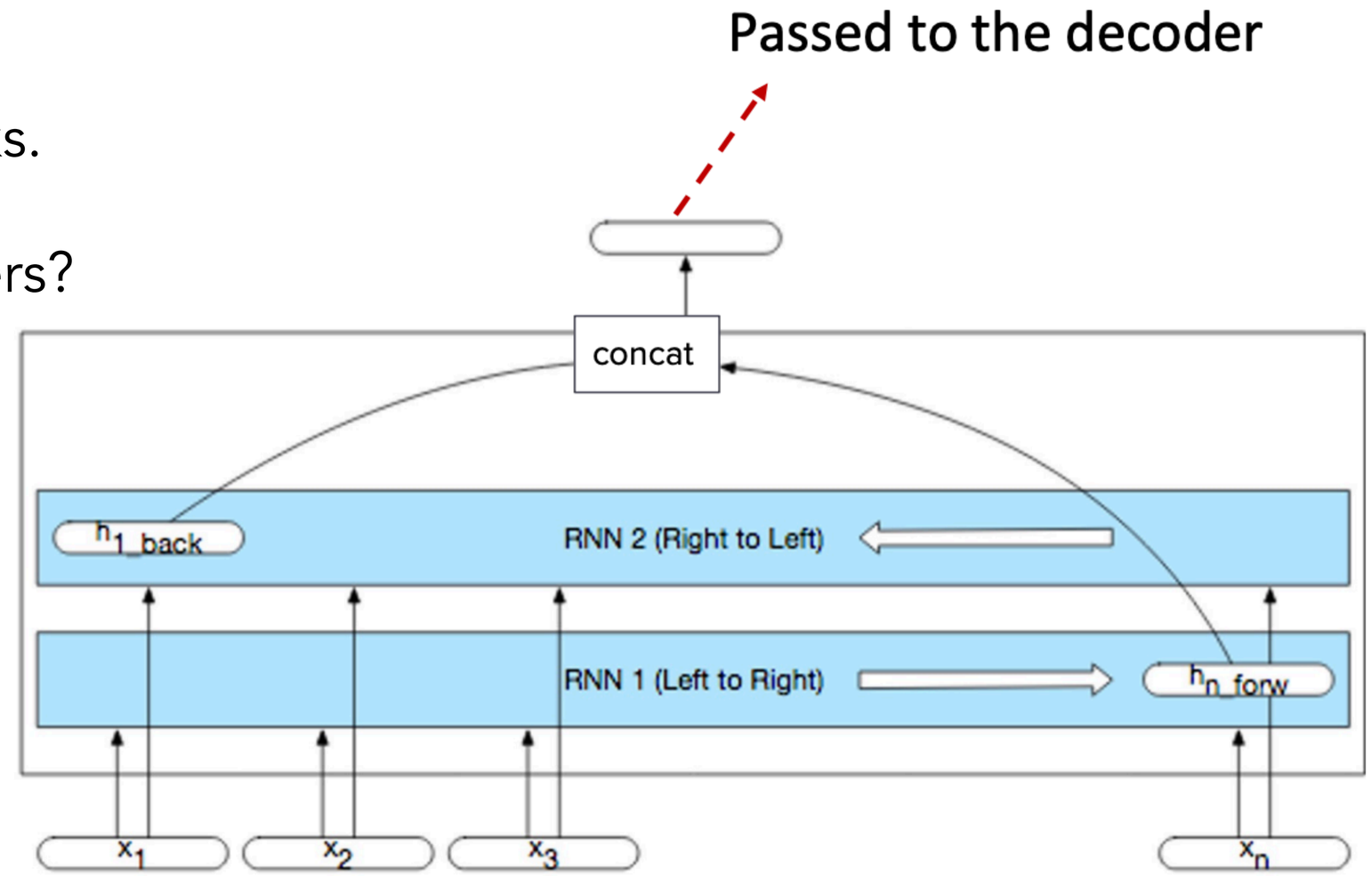
Encoder-Decoder Architecture

- Recall RNNs.
- Useful for seq2seq tasks.

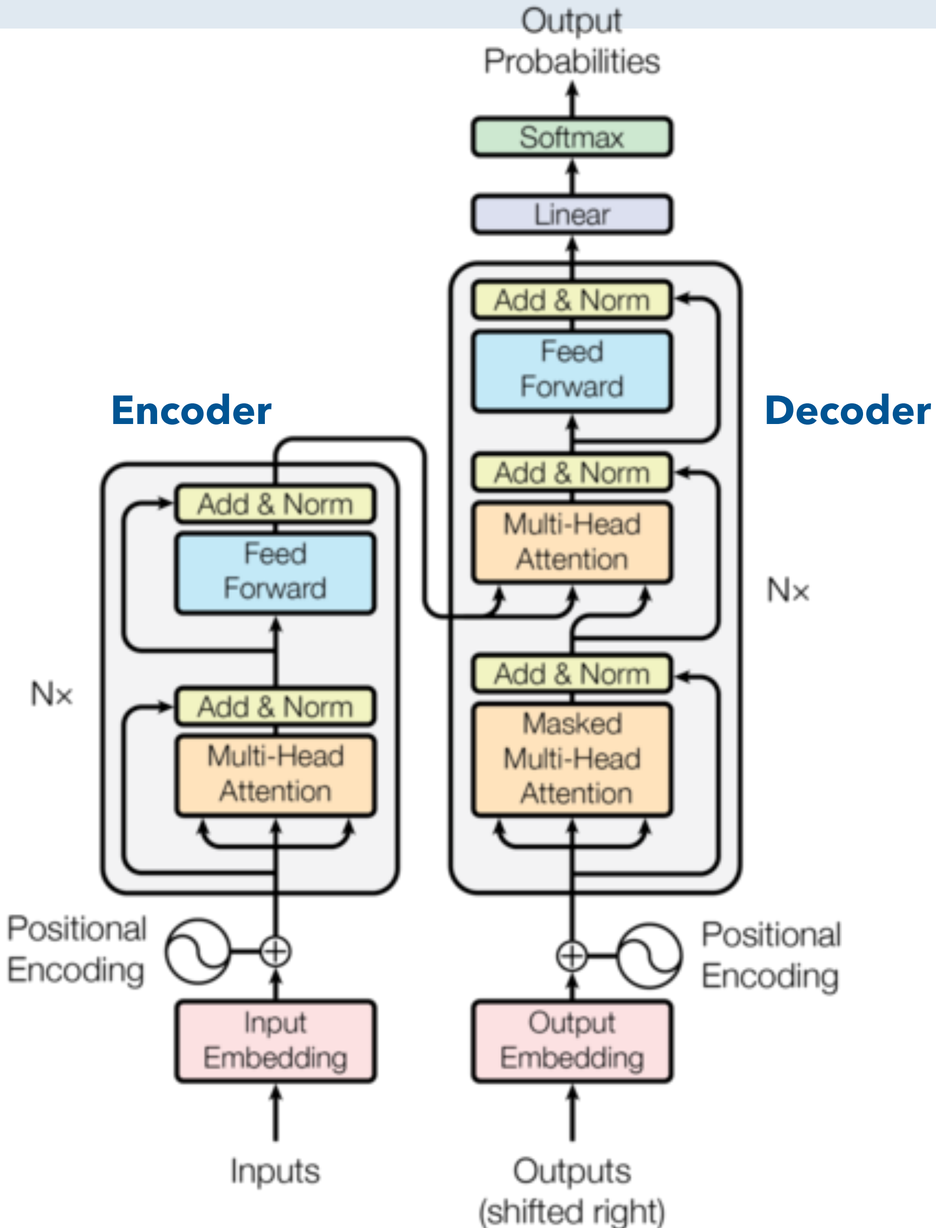


Encoder-Decoder Architecture

- Recall RNNs.
- Useful for seq2seq tasks.
- What about transformers?

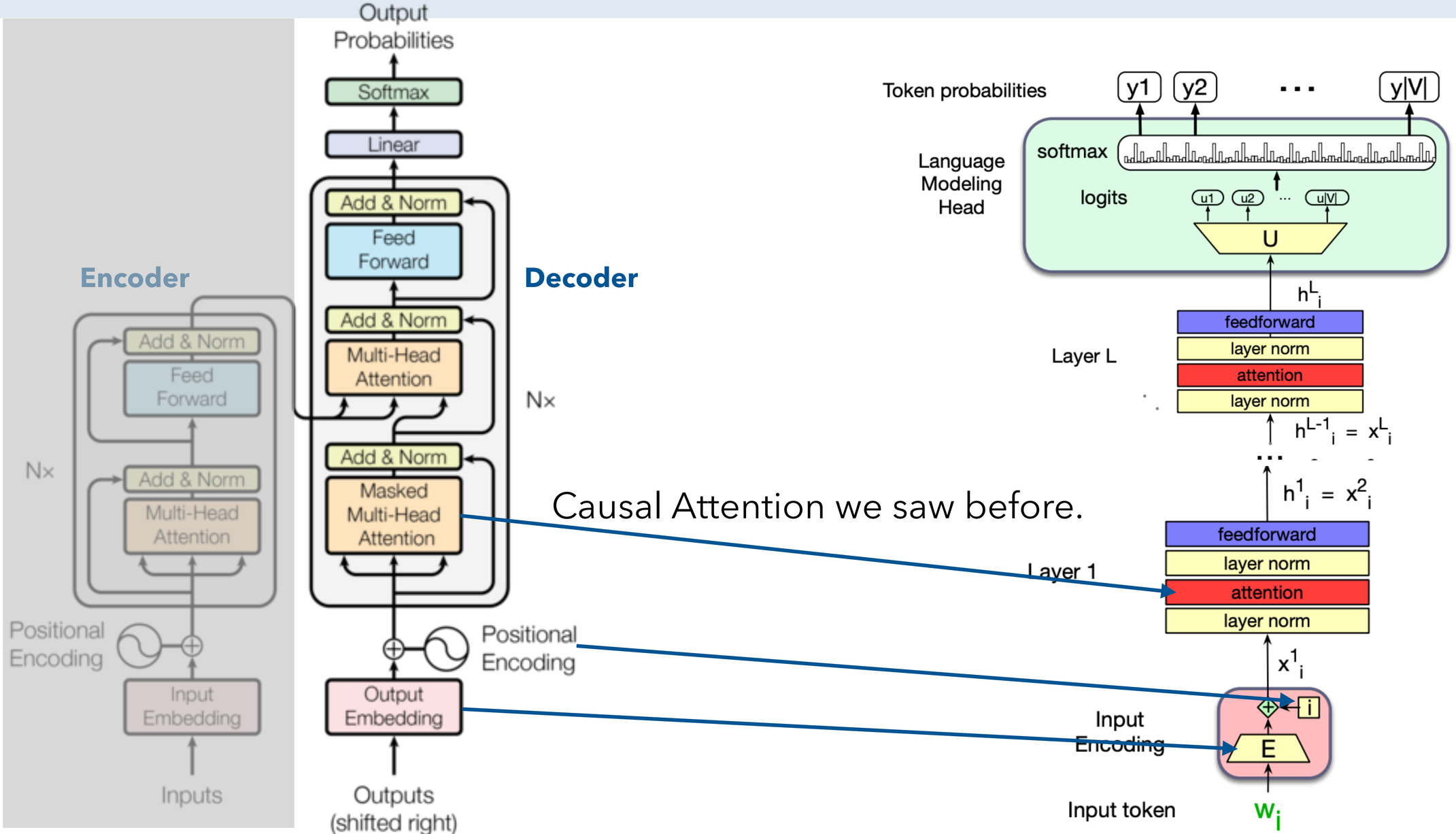


Encoder-Decoder Architecture

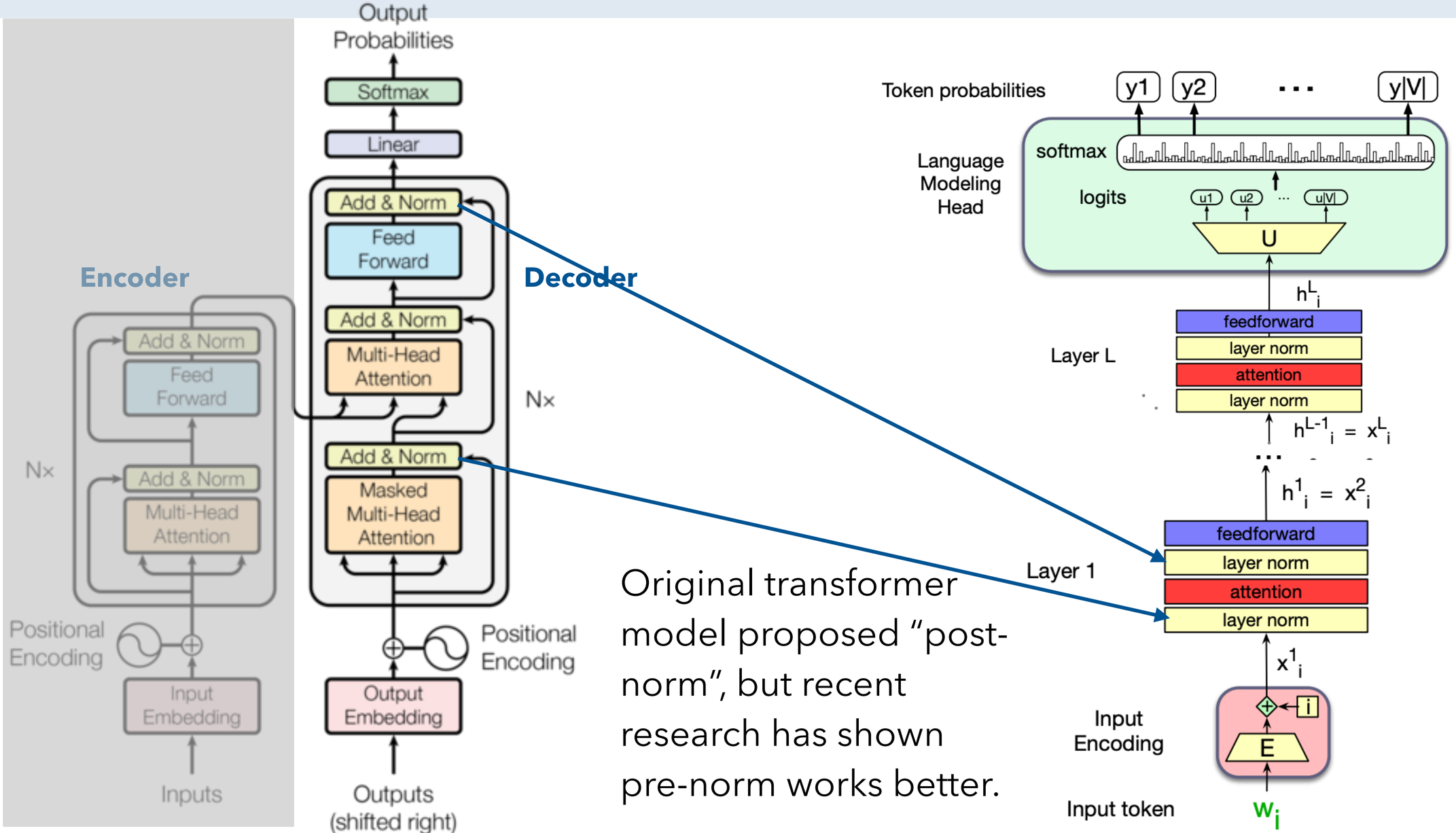


- The actual figure from “Attention is all you need”, Vaswani et al, 2017 paper.
- … and the figure you will see everywhere on the internet

Encoder-Decoder Architecture

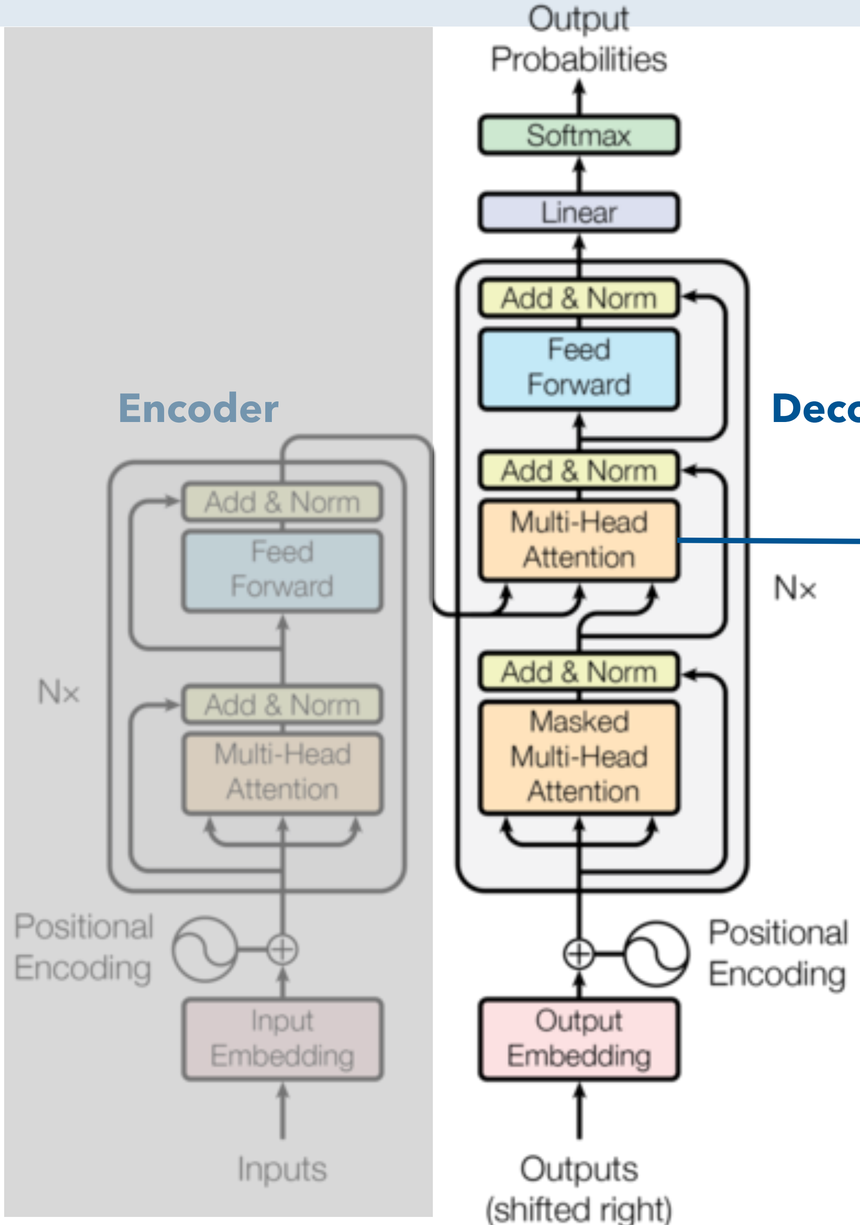


Encoder-Decoder Architecture

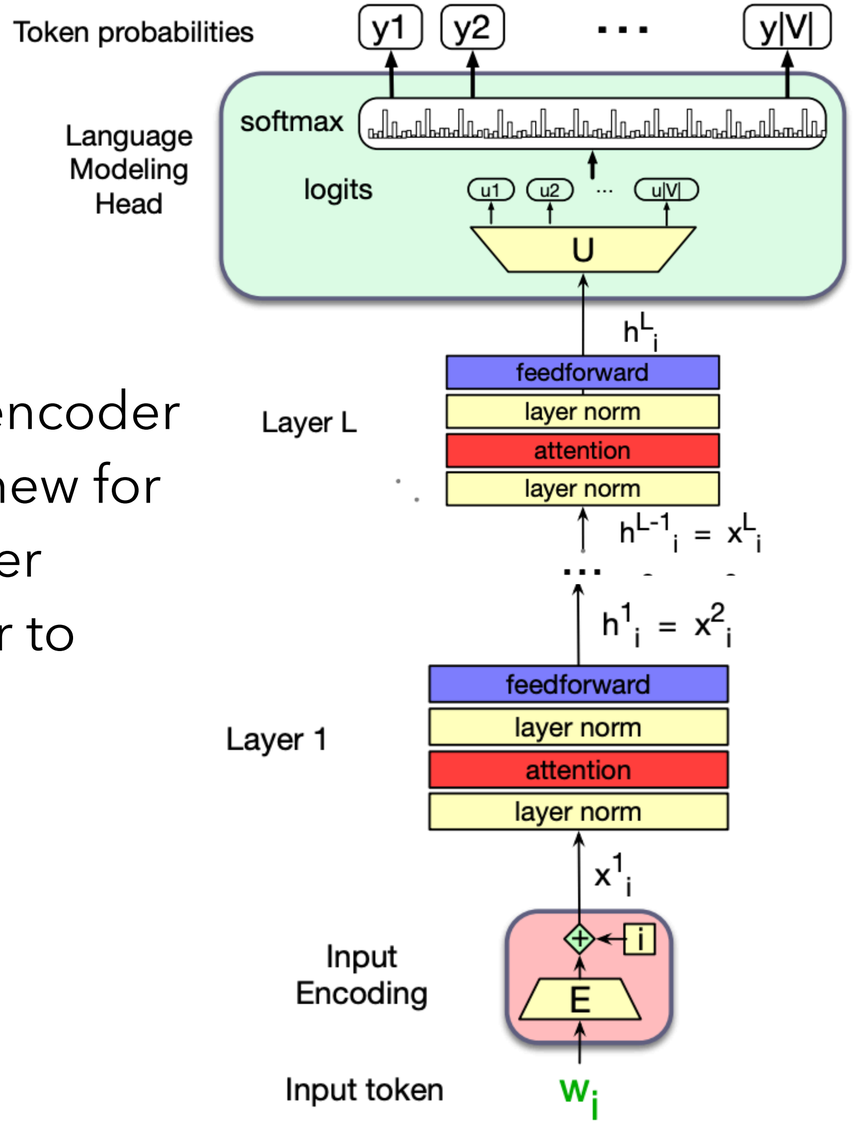


Original transformer model proposed "post-norm", but recent research has shown pre-norm works better.

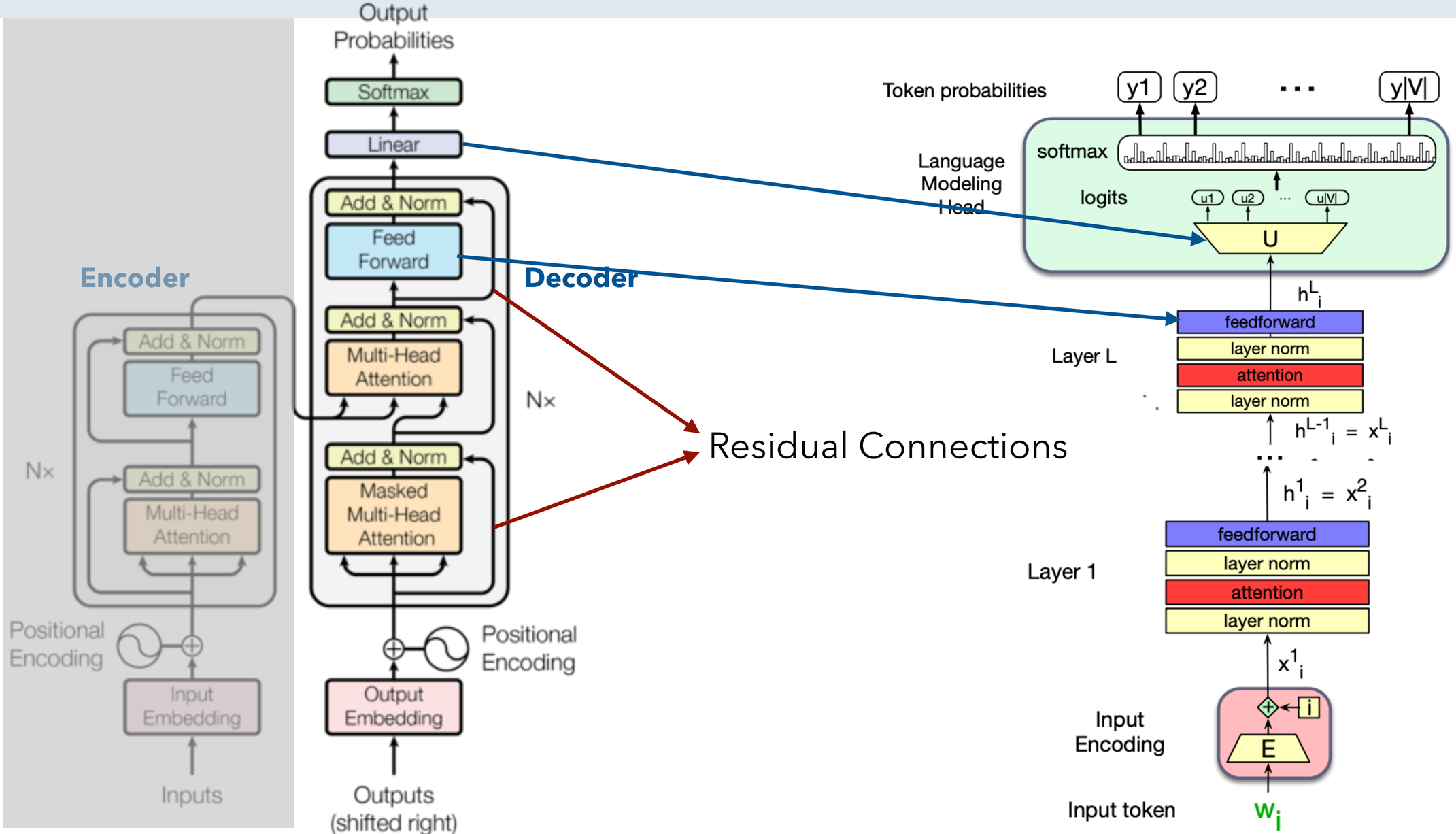
Encoder-Decoder Architecture



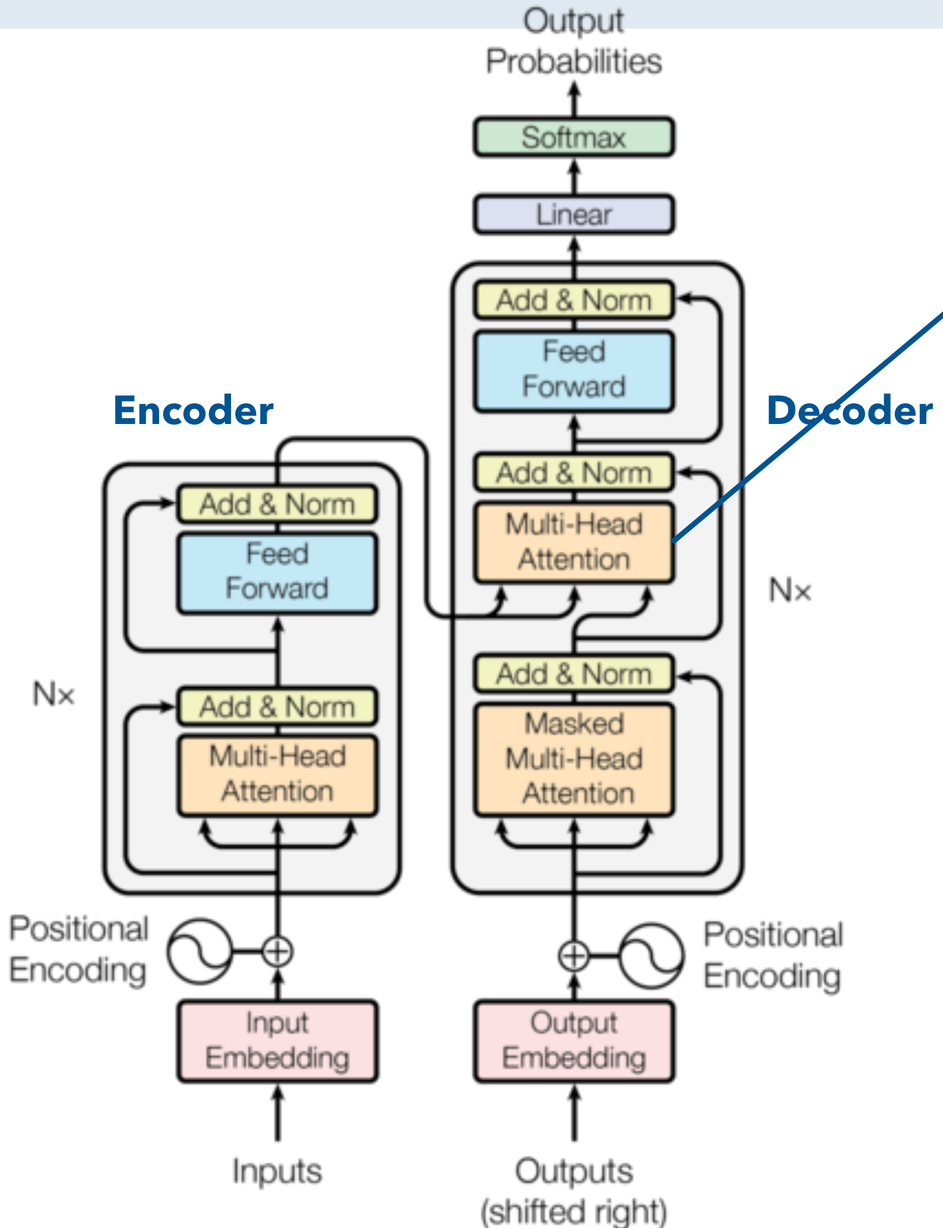
Attention over encoder outputs (this is new for encoder-decoder models!), similar to RNNs



Encoder-Decoder Architecture

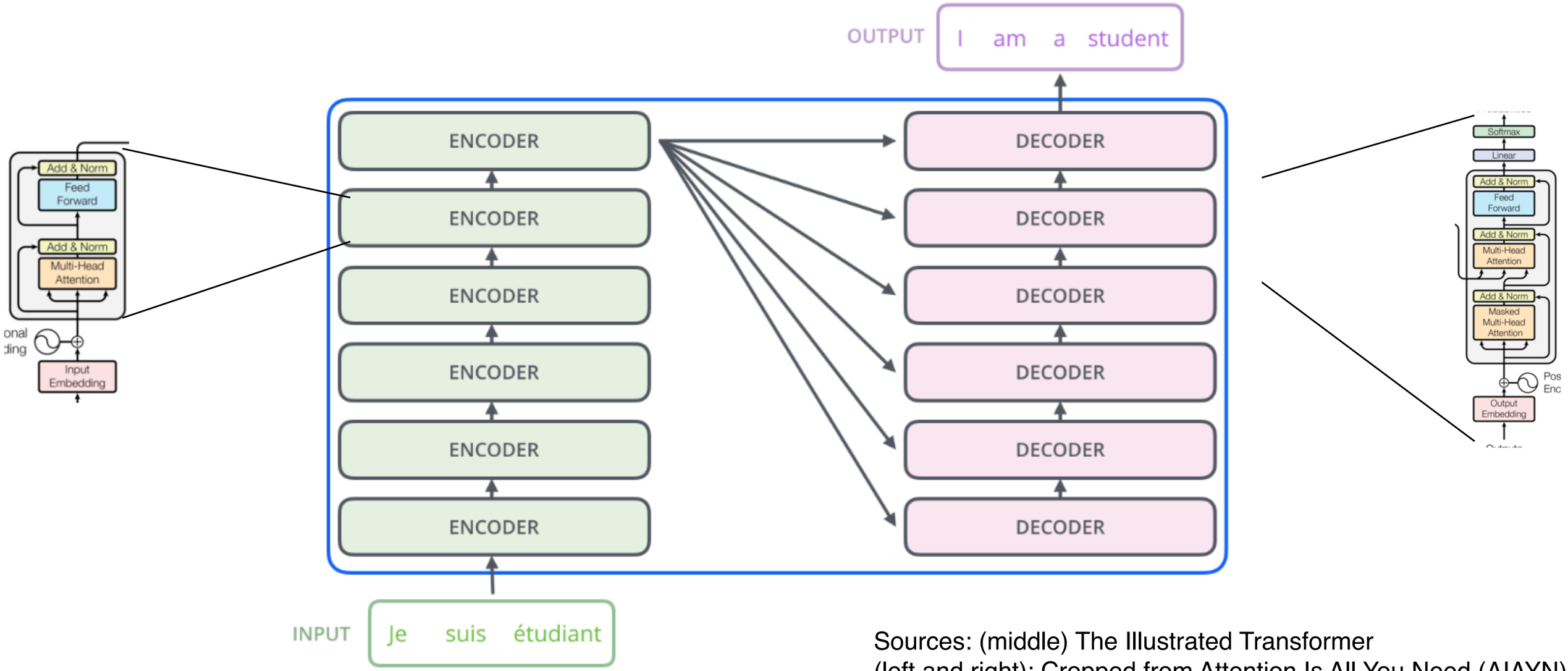


Encoder-Decoder Architecture



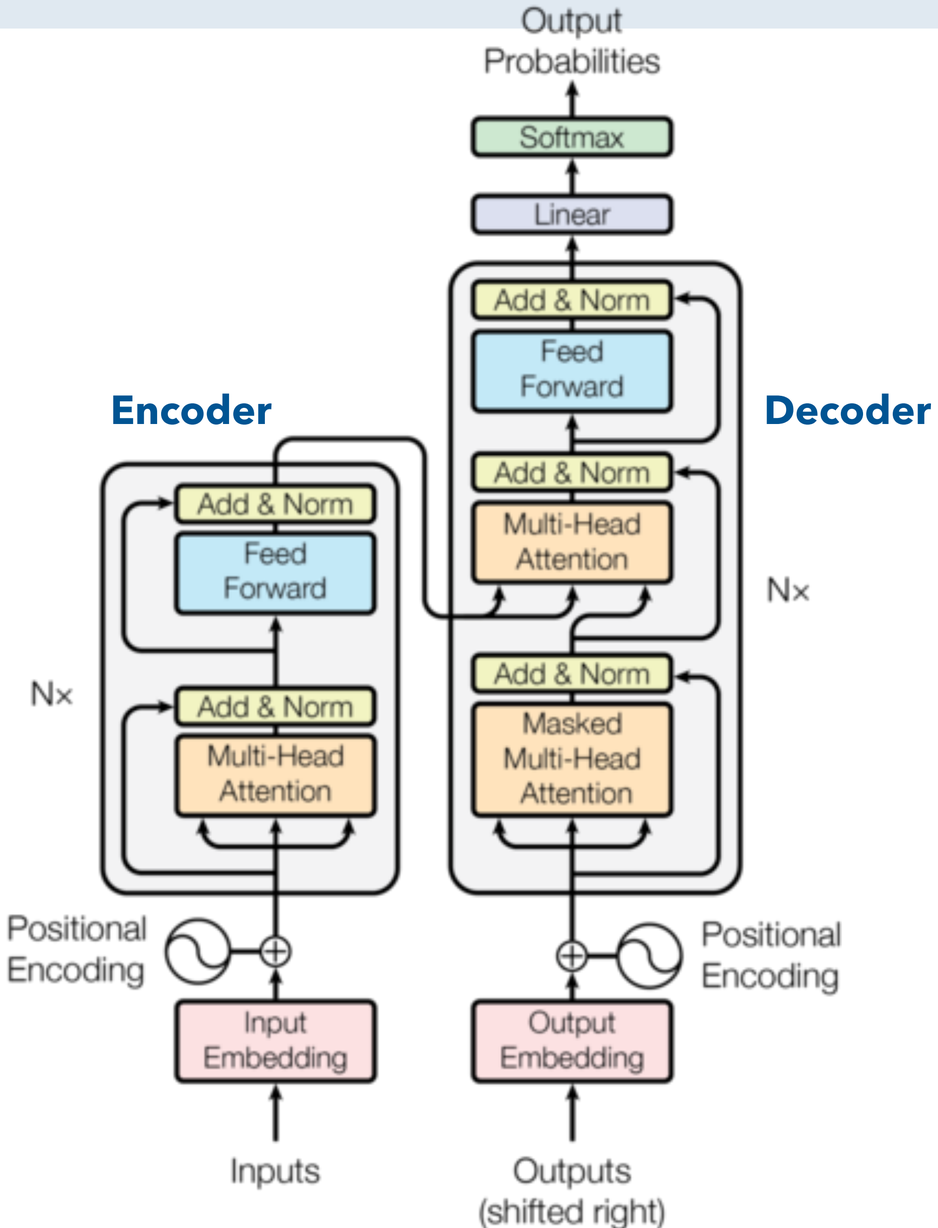
- Multi-head attention from decoder states at each layer to **output of the last layer** of the encoder.

Encoder-Decoder Architecture

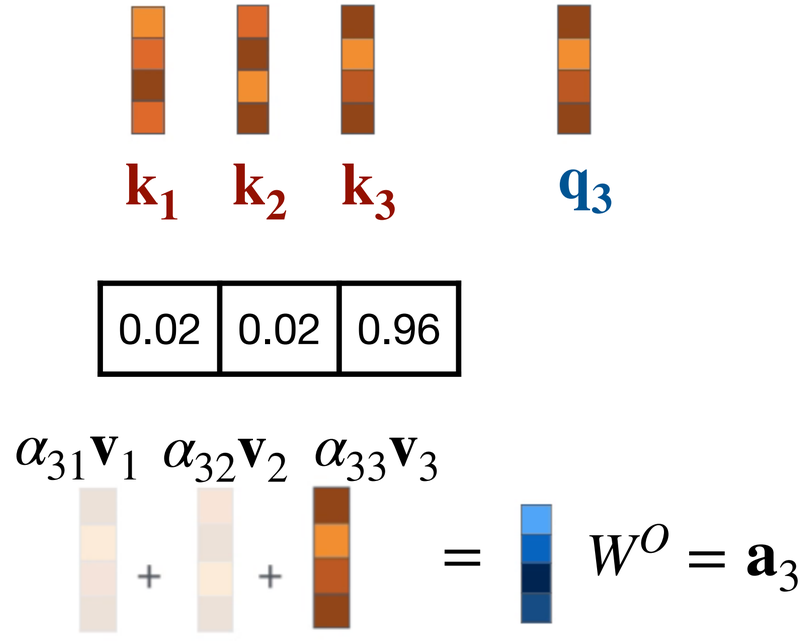


Sources: (middle) The Illustrated Transformer
(left and right): Cropped from Attention Is All You Need (AIAYN)

Encoder-Decoder Architecture



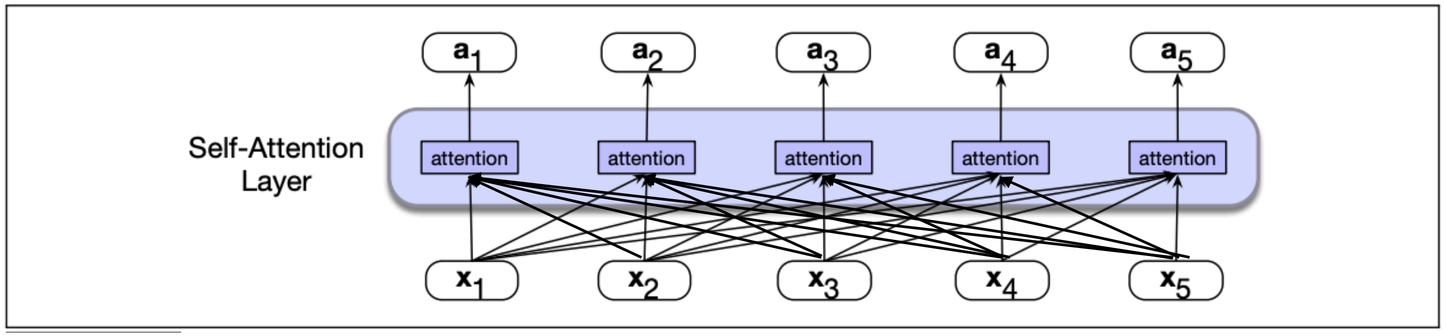
- Recall:

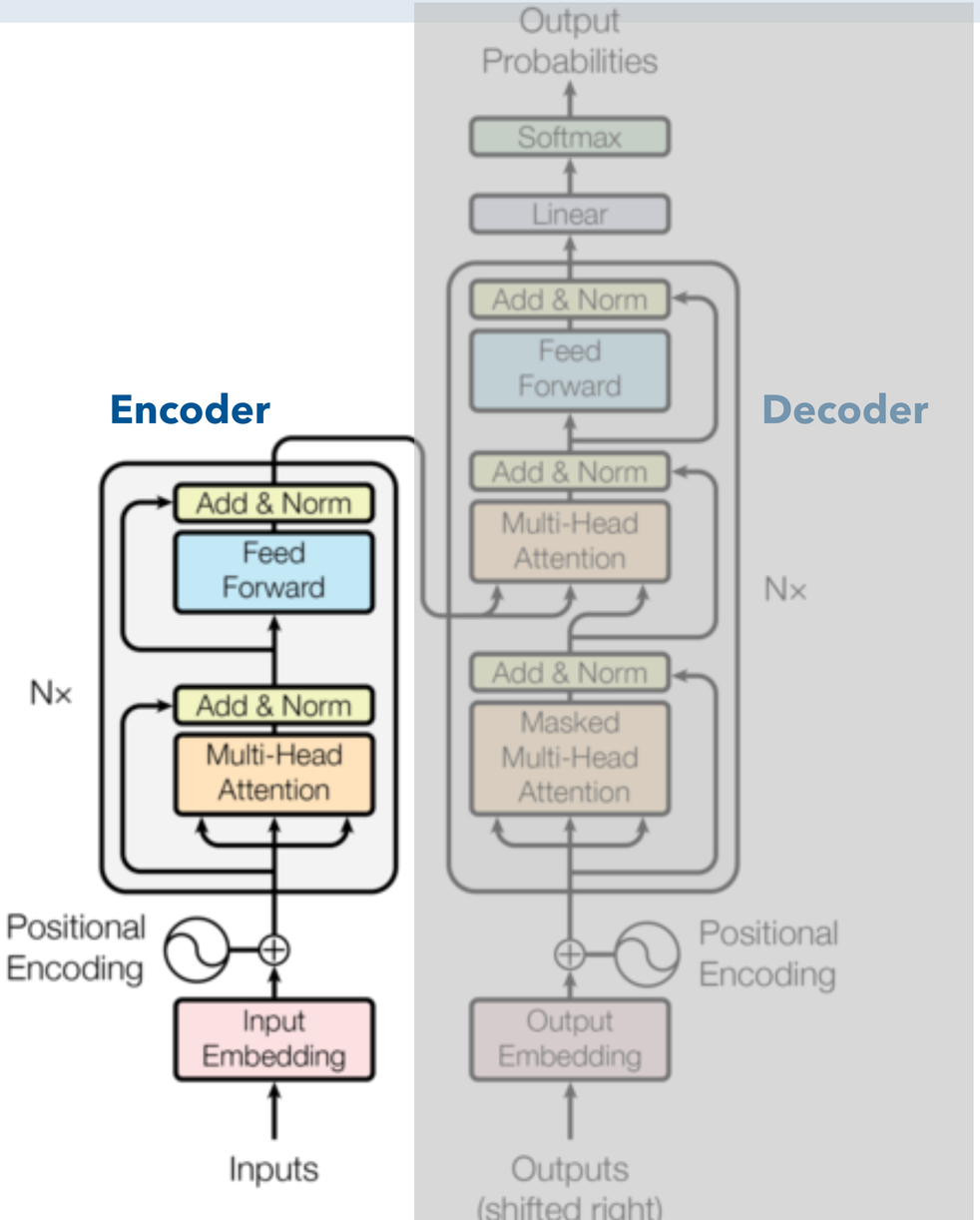


- What is the source of keys, queries, values in attention from decoder to encoder?

Encoder-Decoder Architecture

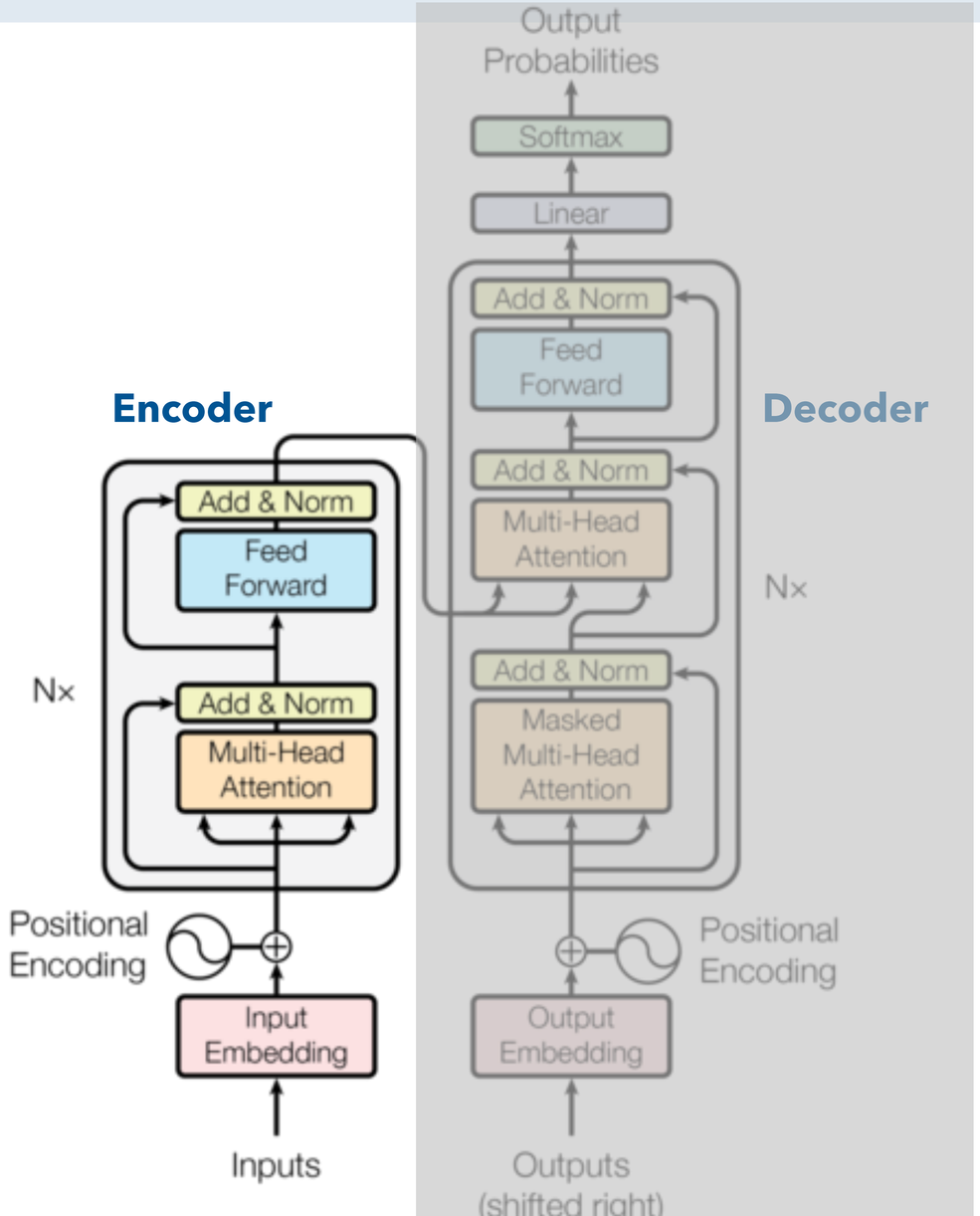
- Encoder architecture is similar to decoder.
- Only difference: This attention is **not** causal.
  - All tokens attend to all other tokens.





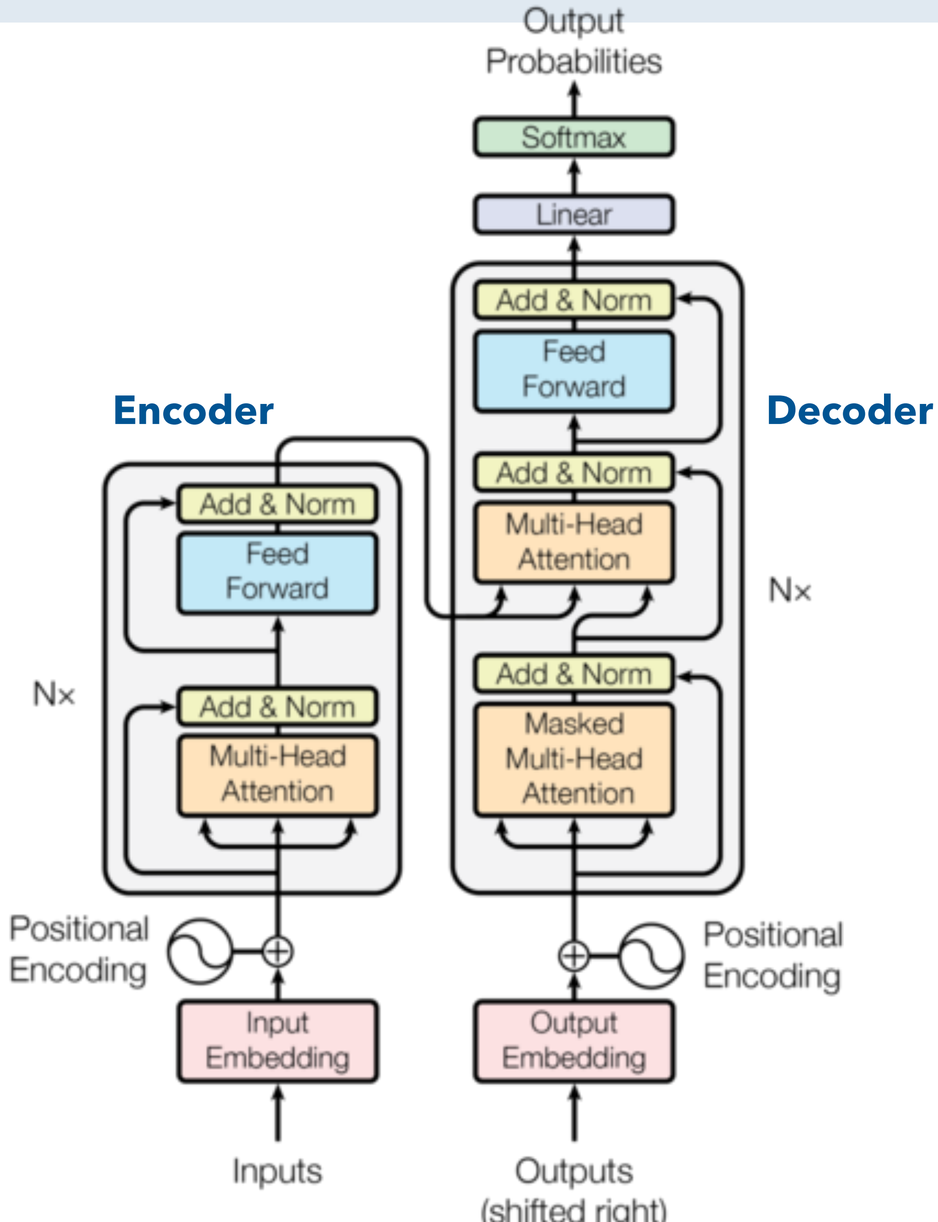
Encoder-Decoder Architecture

- Encoder architecture is similar to decoder.
- Only difference: This attention is **not** causal.
  - All tokens attend to all other tokens.
- At each time step, the encoder output h_t can be viewed as a "contextual" representation of the input word w_t.



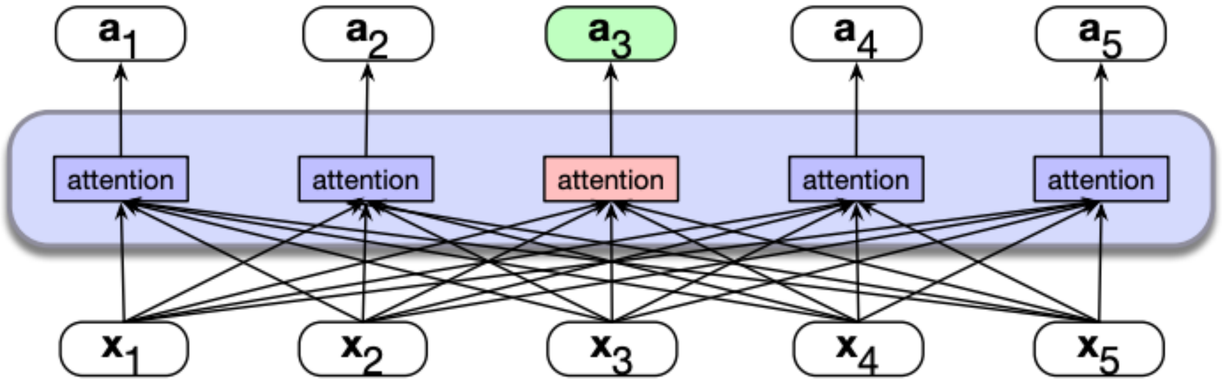
Encoder-Decoder Architecture

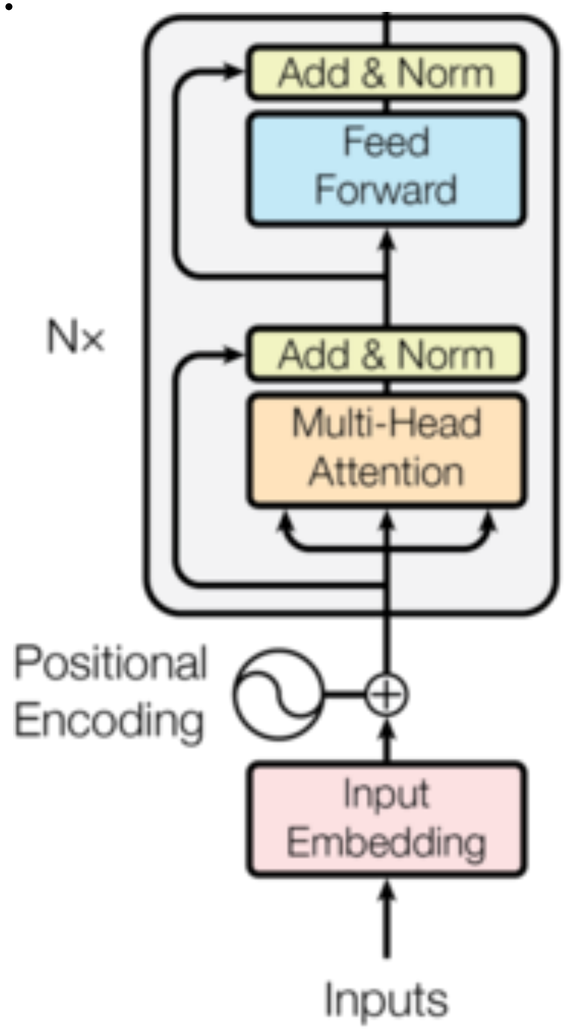
- Training
  - Next-token prediction at the decoder output.



Encoder-only architectures

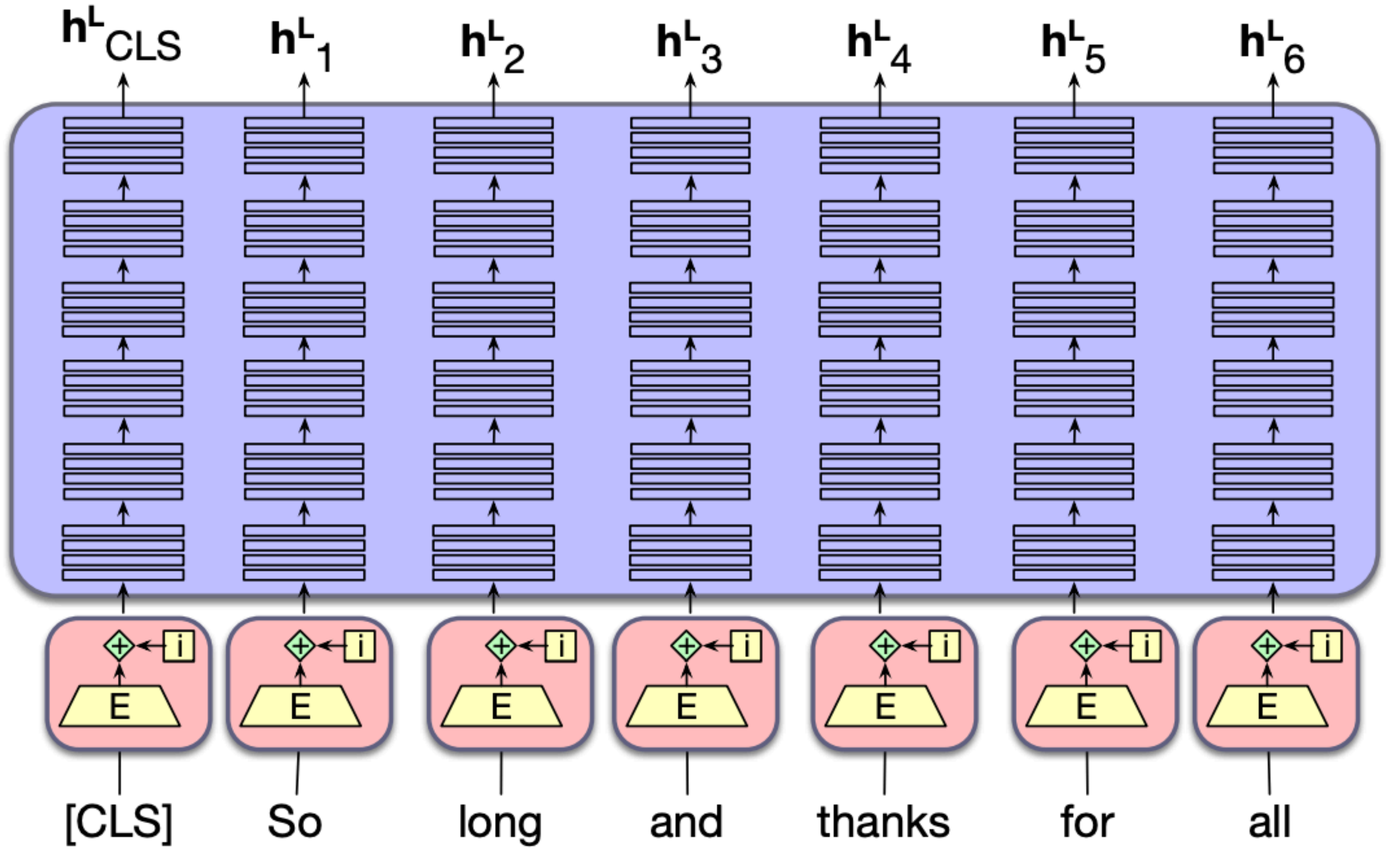
- Same architecture components as the decoder.
- **Difference:** the attention is bidirectional, not causal





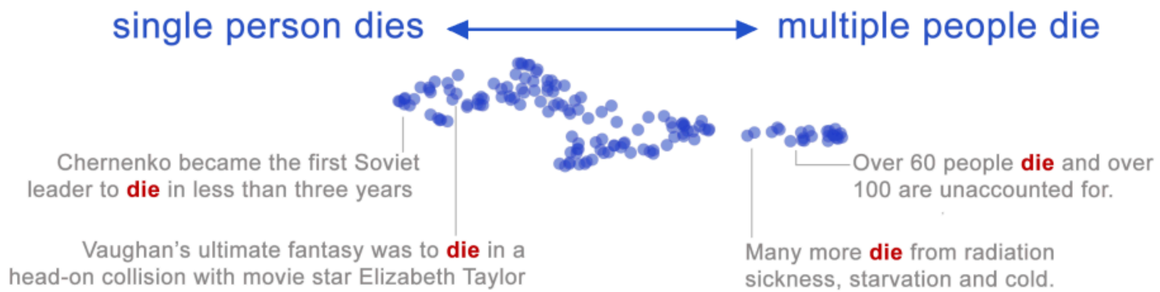
Encoder-only Architecture

- What is this useful for?
  - Gives contextual representations for each input word.



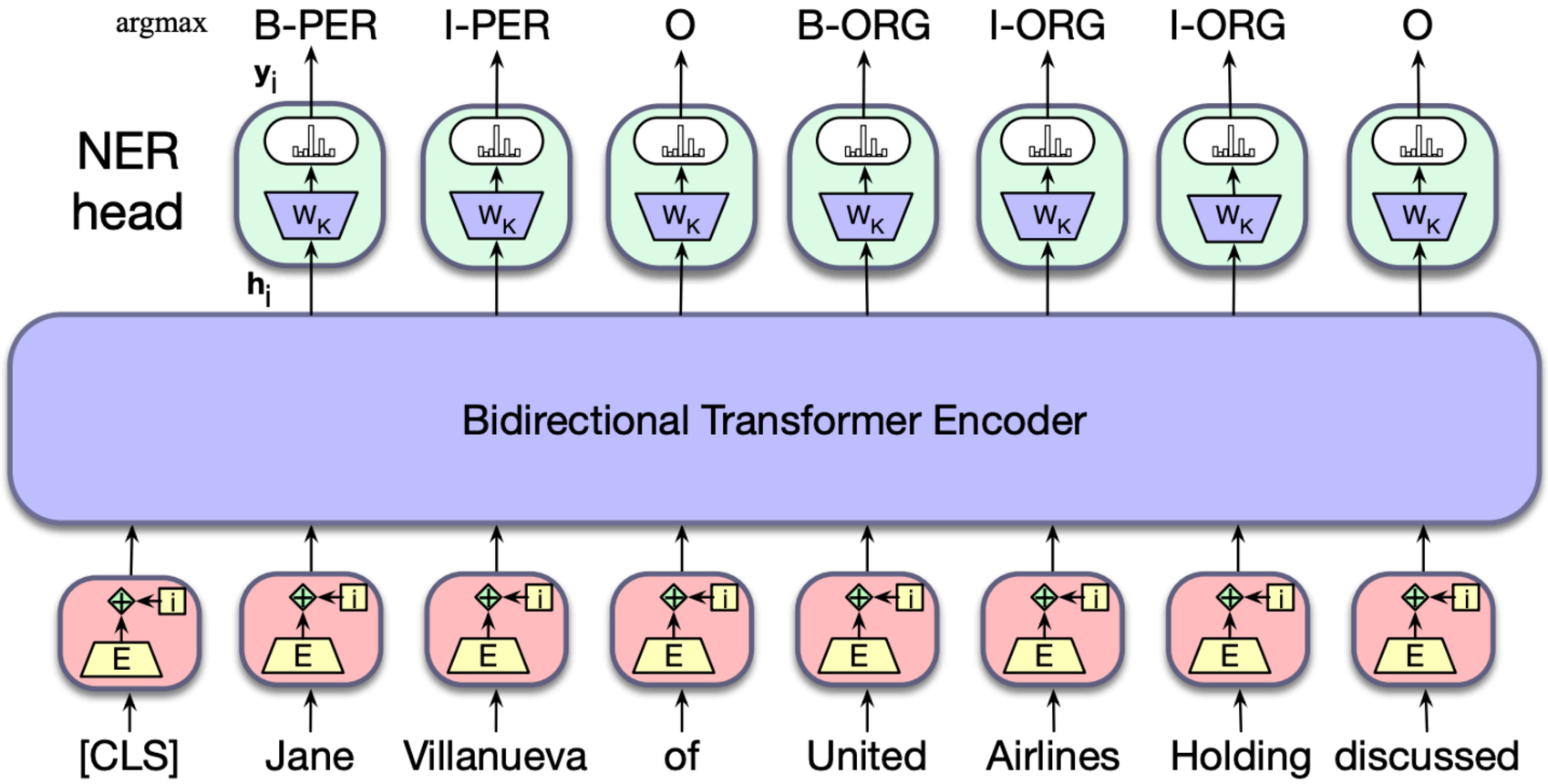
Encoder-only Architecture

- Word sense disambiguation
  - A sense (or word sense) is a discrete representation of one aspect of the meaning of a word



Encoder-only Architecture

- Classification?
  - Sequence Labeling

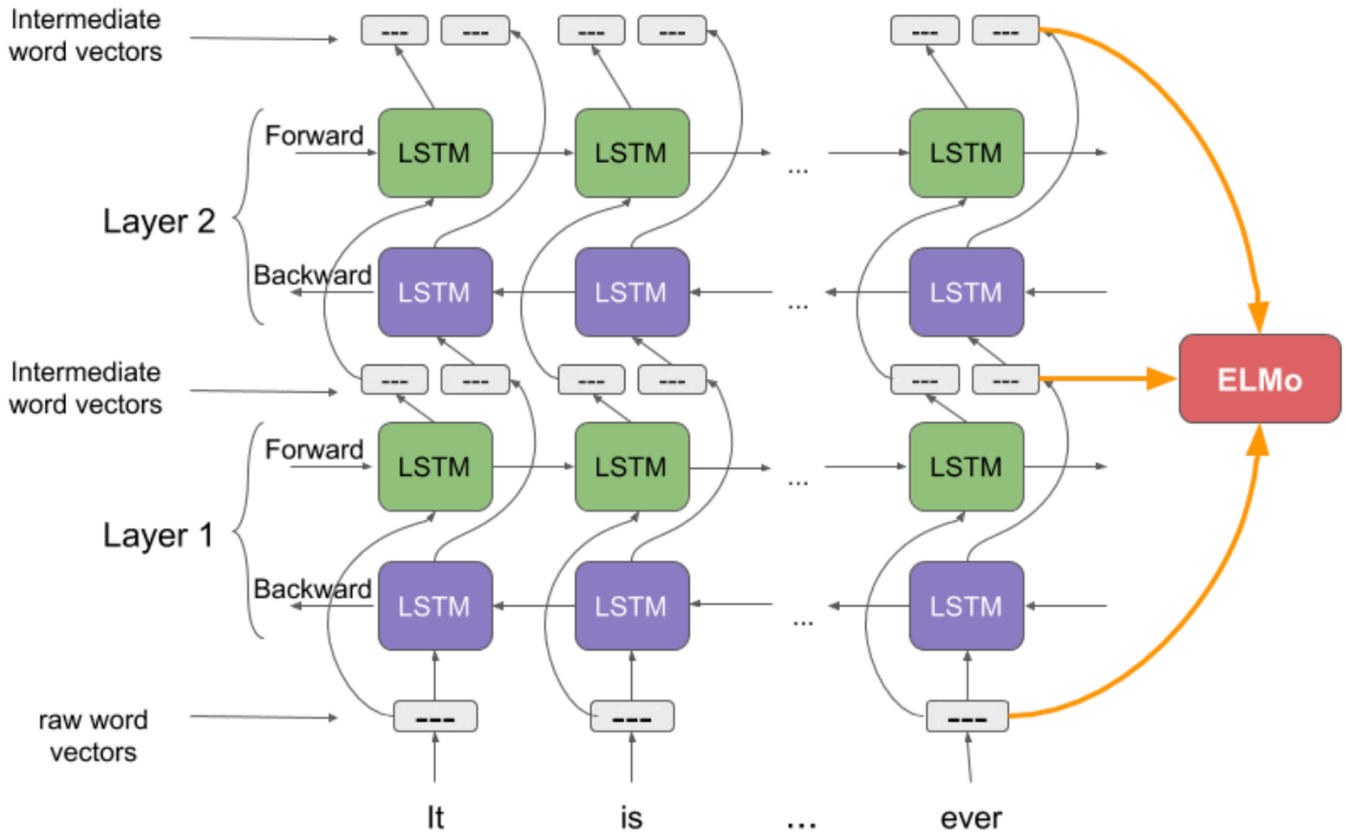


Why are these more powerful than bidirectional RNNs?

- Predecessor: ELMo (Peters et al., 2018)

Uses bi-LSTM-based encoder-decoder.

Combines hidden vectors from different layers.



Why are these more powerful than bidirectional RNNs?

- BiLSTMs are not quite the same as full self-attention:

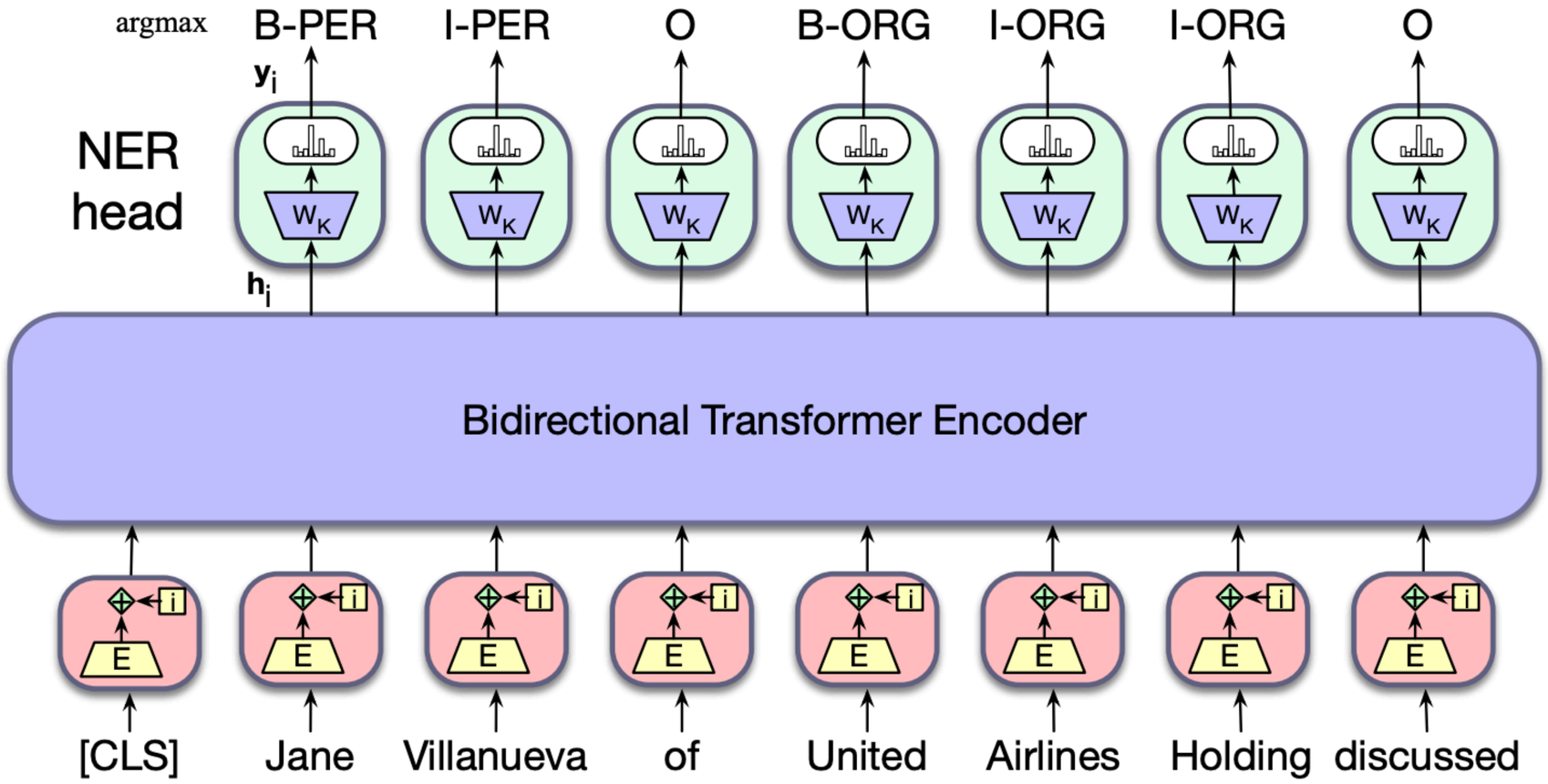
Try to predict the italicized word from just the left or just the right context.
The celebrity , Michael ***Jordan*** , was a player in the NBA .

A left-to-right model could also answer “Jackson” and be sensible but wrong. (Reference to singer/songwriter Michael Jackson)

A right-to-left model could also answer “Curry” and be sensible but wrong. (Reference to NBA player Steph Curry)

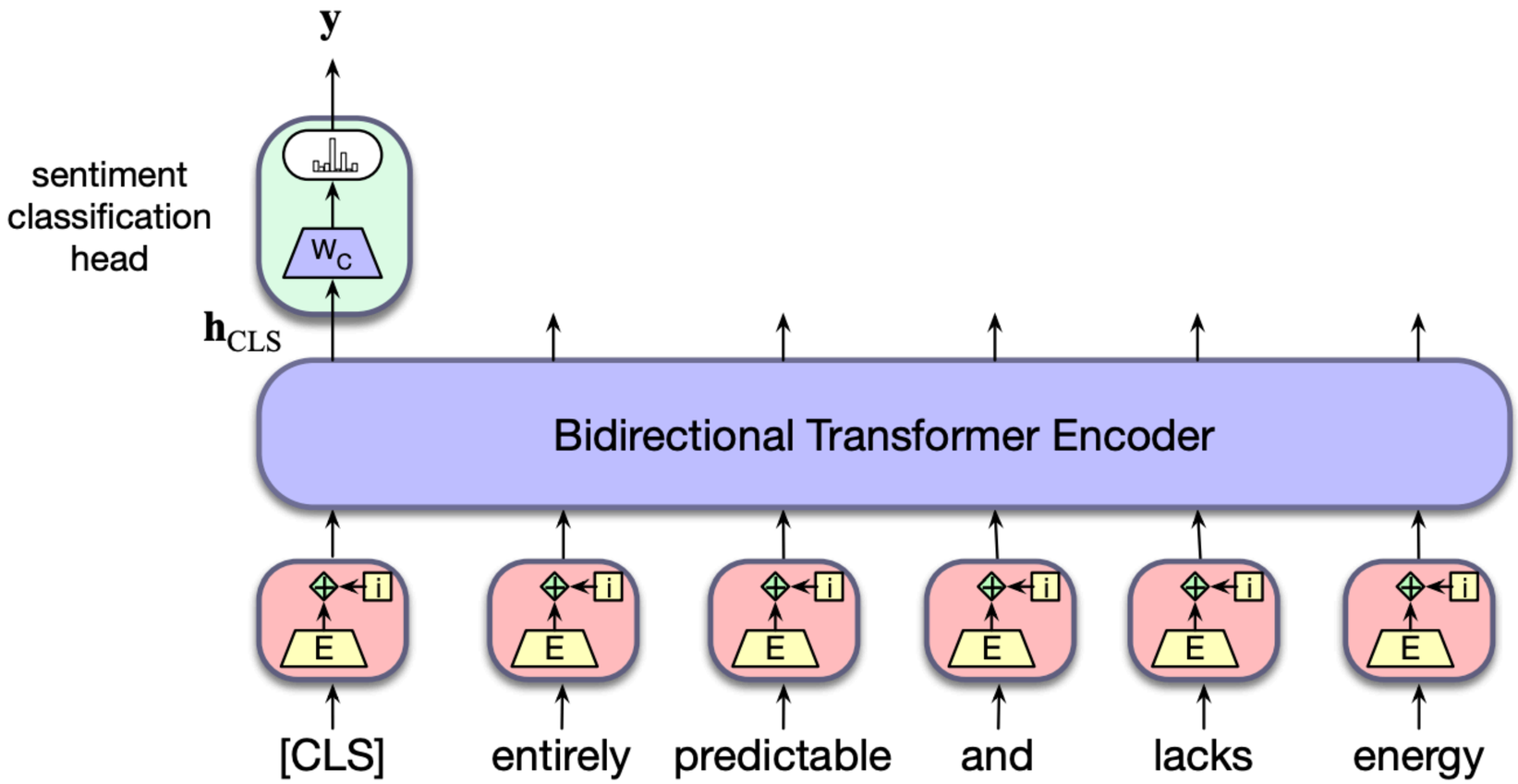
Encoder-only Architecture

- Classification?
  - Sequence Tagging



Encoder-only Architecture

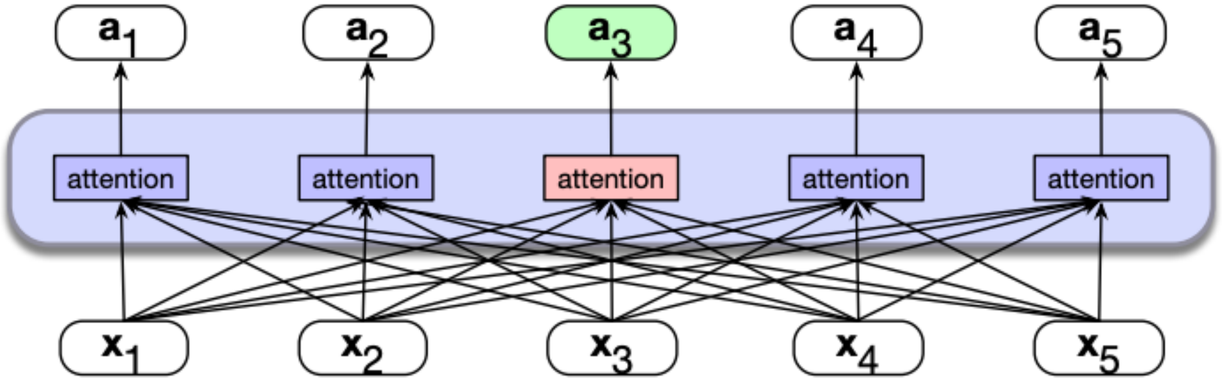
- Classification?
  - Sequence-level classification

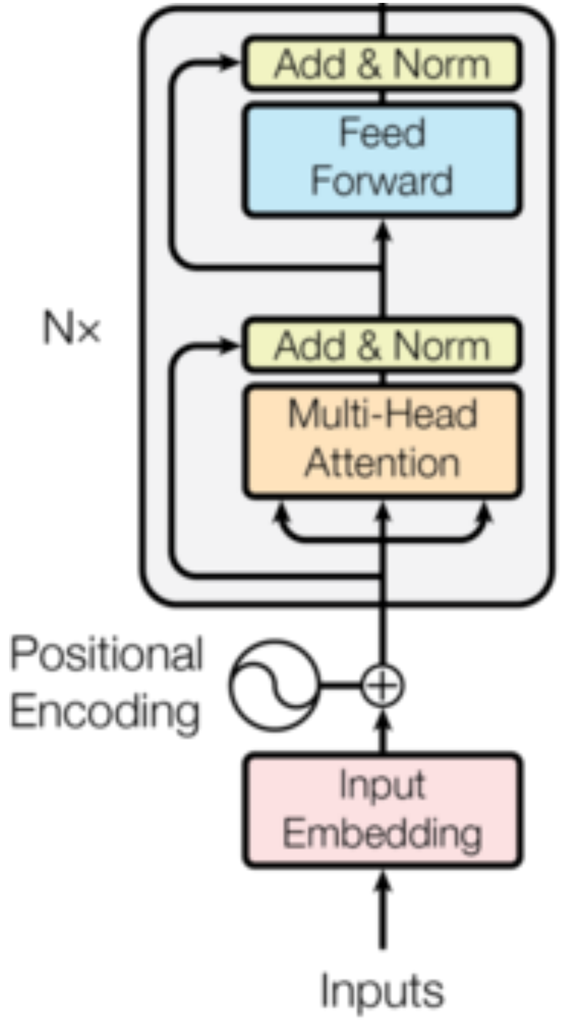


Encoder-only architectures

- Can we use the same next-token prediction task to train encoder models?

No! The desired output is part of the input!





Slide Acknowledgements

- Earlier versions of this course offerings including materials from Claire Cardie, Marten van Schijndel, Lillian Lee.