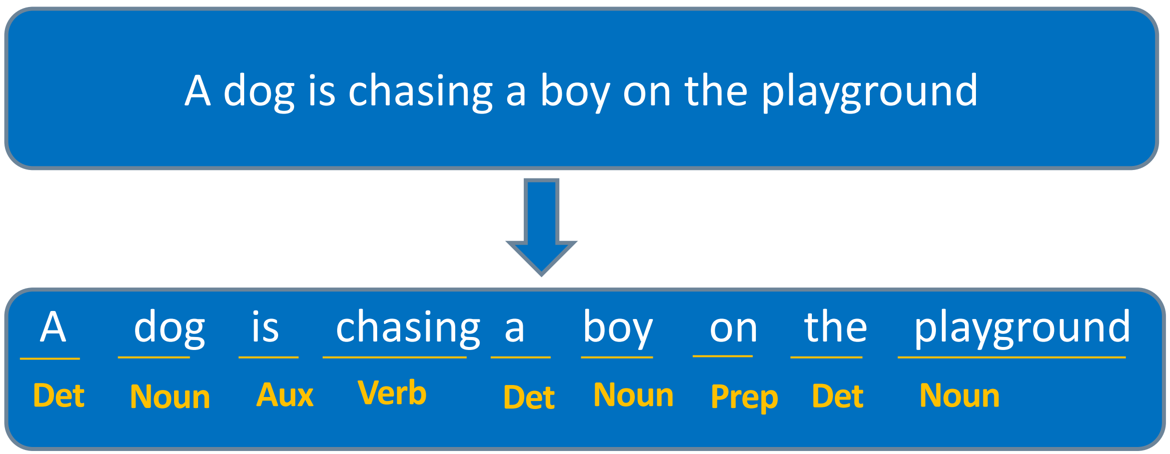
Transformers and Large Language Models

Phạm Quang Nhật Minh
Aimesoft JSC
minhpham0902@gmail.com

The Encoder-Decoder Model: Motivation

- Recall: Sequence labeling models

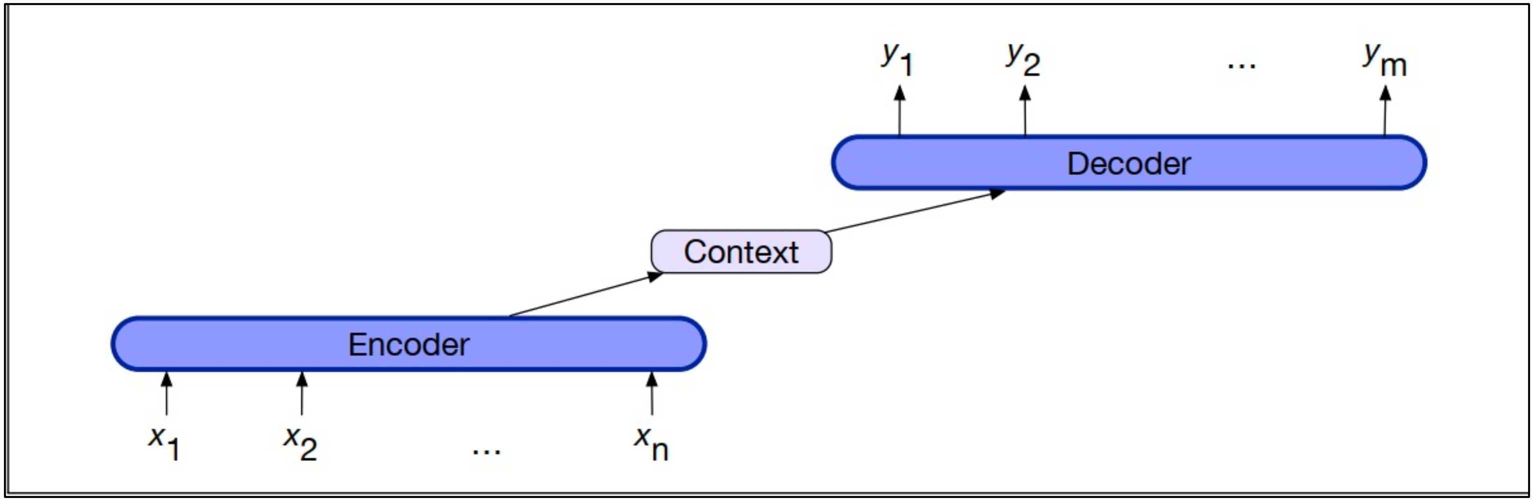


- How we can handle the task where the input sequence and the output sequence have different length?

Some text-to-text tasks

- Machine Translation
- Text summarization
- Title generation

The Encoder-Decoder Model

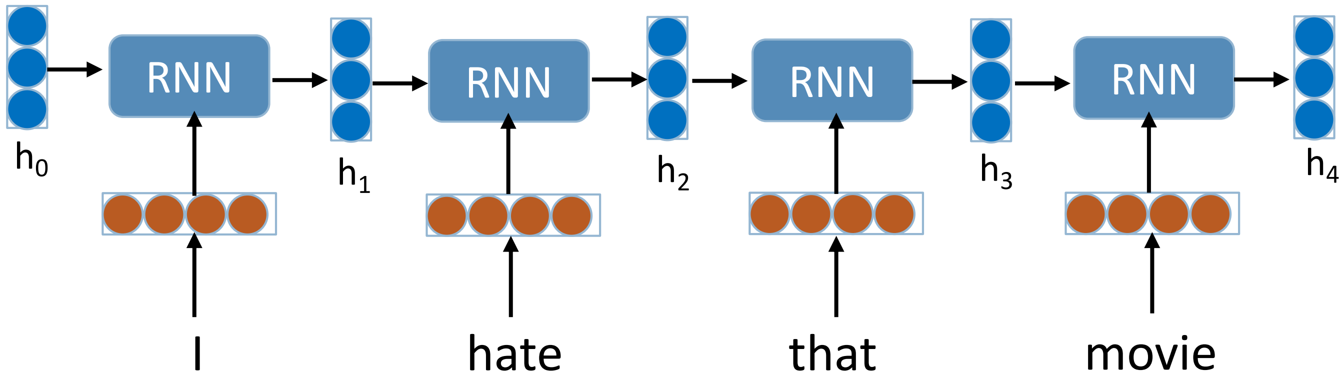


Components of the Encoder-Decoder Model:

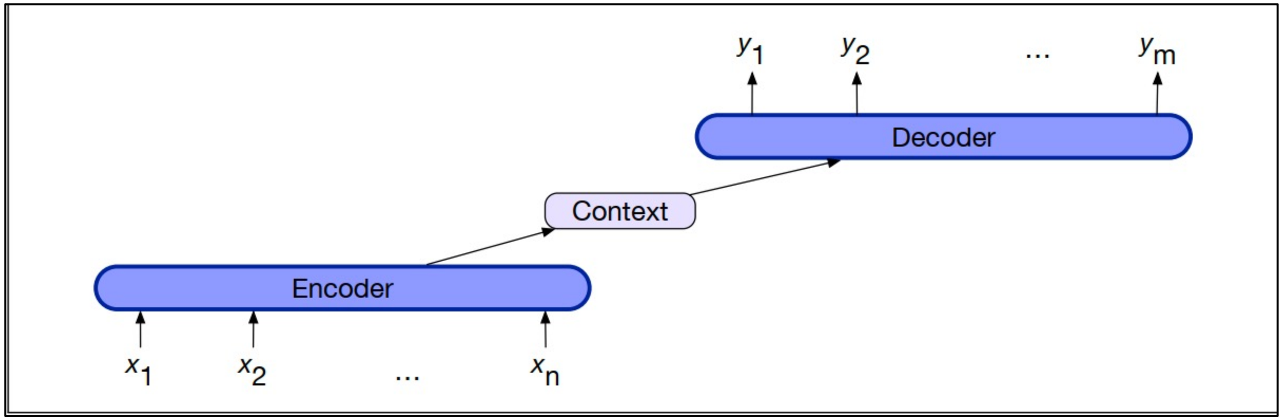
- An encoder
- A context vector
- A decoder

Encoder

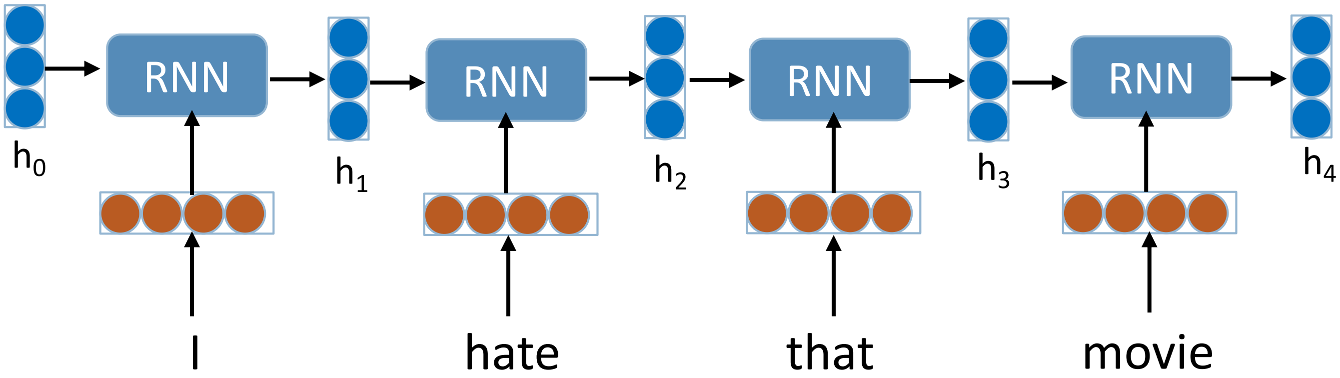
- Given an input sequence, the encoder generates a sequence of hiven vectors (contextualized representations)



Context vector



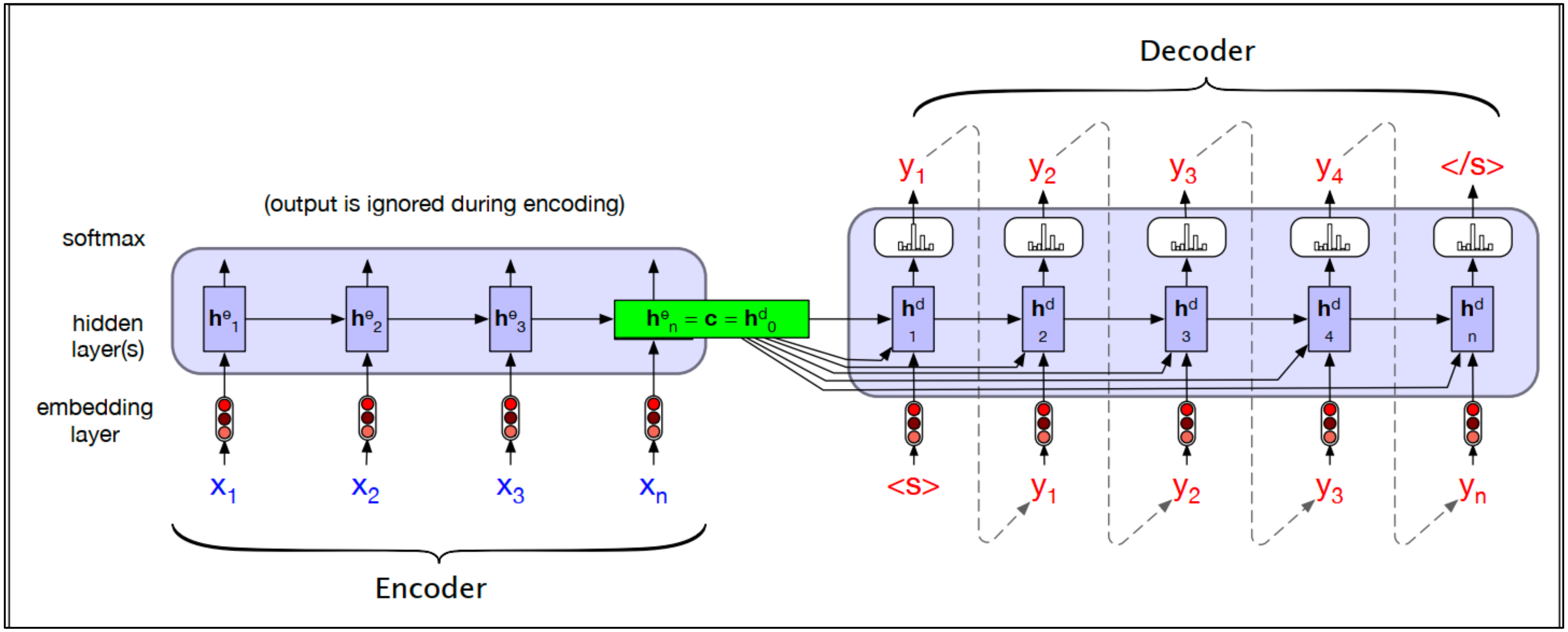
Context vector c is a function of h_1^n



Decoder

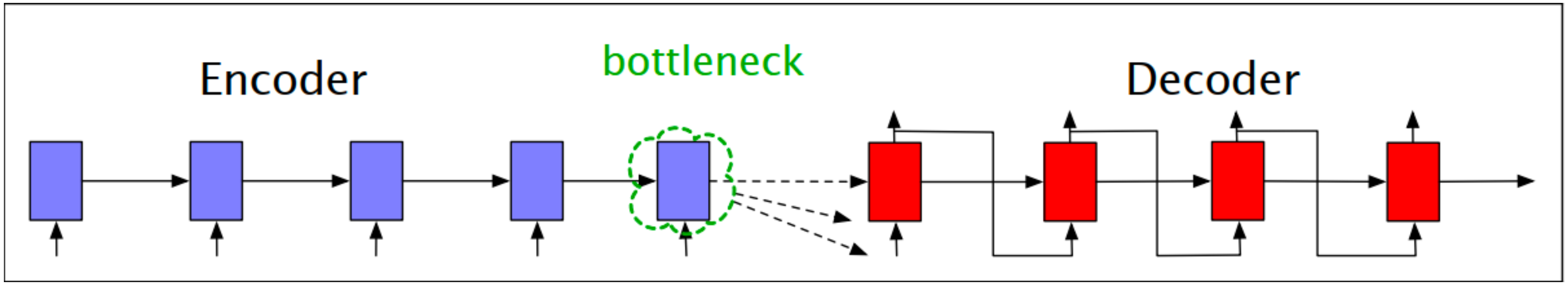
- A decoder accepts context vector c as input and generates an arbitrary length sequence of hidden states h_1^m

How the decoder generates output



Attention Mechanism

- Attention mechanism is to solve the bottleneck problem in vanila encoder-decoder models
  - The last hidden state in the encoder is used as the context vector *c*
  - Information at the beginning of the sequence is not well represented



Attention Mechanism

- Idea: create a fixed-length context vector by taking a weighted sum of all encoder hidden states
  - The weights focus more on a particular part of the source text that is relevant for the token the decoder is currently producing

$$c_i = \sum_j \alpha_{ij} h_j^i$$

How to caculate attention weights α_{ij}?

Dot production attention

- Measure how similar the decoder hidden state to the encoder hidden state

$$\text{score}\left(h_{i-1}^{d}, h_{j}^{e}\right) = h_{i-1}^{d} \cdot h_{j}^{e}$$

- Normalize scores with a softmax

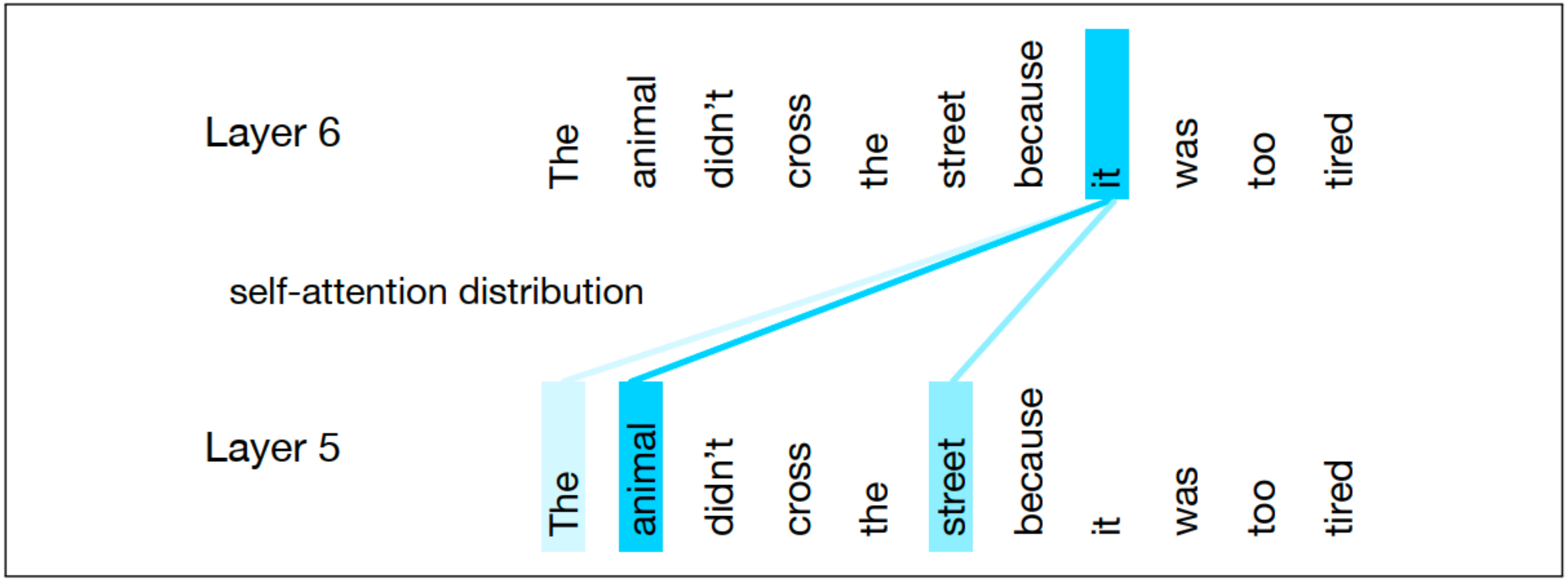
$$\begin{aligned}
\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e})\ \forall j \in e) \\
&= \frac{\exp(\text{score}(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e})}{\sum_{k} \exp(\text{score}(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{k}^{e}))}
\end{aligned}$$

Transformers: Intution

- Intuition: “across a series of layers, we build up richer and richer contextualized representations of the meanings of input words or tokens”
  - At each layer of a transformer, to compute the representation of a word *i* we combine information from the representation of i at the previous layer with information from the representations of the neighboring words
- We need a mechanism to:
  - Weight representations of the different words from the context at the prior level
  - Combine them to compute the representation of this layer

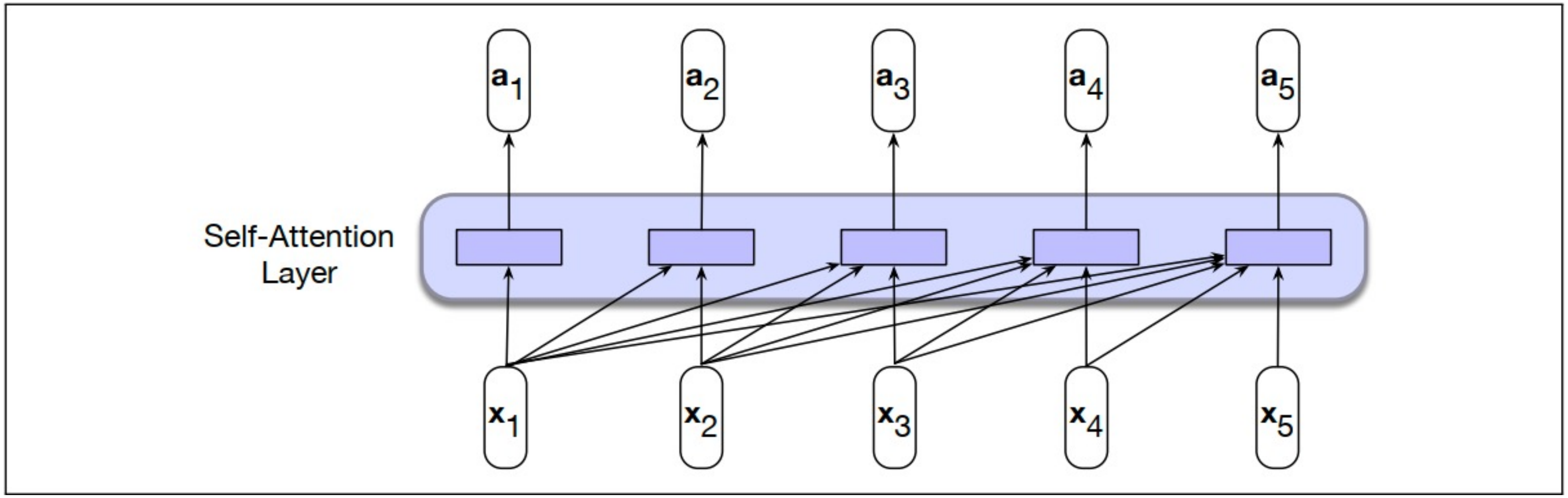
Self-Attention Mechanism

- Self-attention
  - Look the context
  - Integrate the representations from words in that context from layer k-1 to build the



Causal or backward-looking self-attention

- Two types of self-attention
  - Backward-looking self-attention (e.g., GPT)
  - Bidirectional self-attention (e.g., BERT)



Self-Attention in details (1)

- Based on the idea of the attention mechanism, but more sophisticated
- Map a **query** to an **ouput** by comparing the query with **keys**

$$
\begin{aligned}
\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}}; \mathbf{k}_i &= \mathbf{x}_i\mathbf{W}^{\mathbf{K}}; \mathbf{v}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{V}} && (10.11)\\
\textit{Final verson:} \quad \operatorname{score}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} && (10.12)\\
\alpha_{ij} &= \operatorname{softmax}(\operatorname{score}(\mathbf{x}_i, \mathbf{x}_j)) \ \forall j \leq i && (10.13)\\
\mathbf{a}_i &= \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j && (10.14)
\end{aligned}
$$

Self-Attention in details (2)

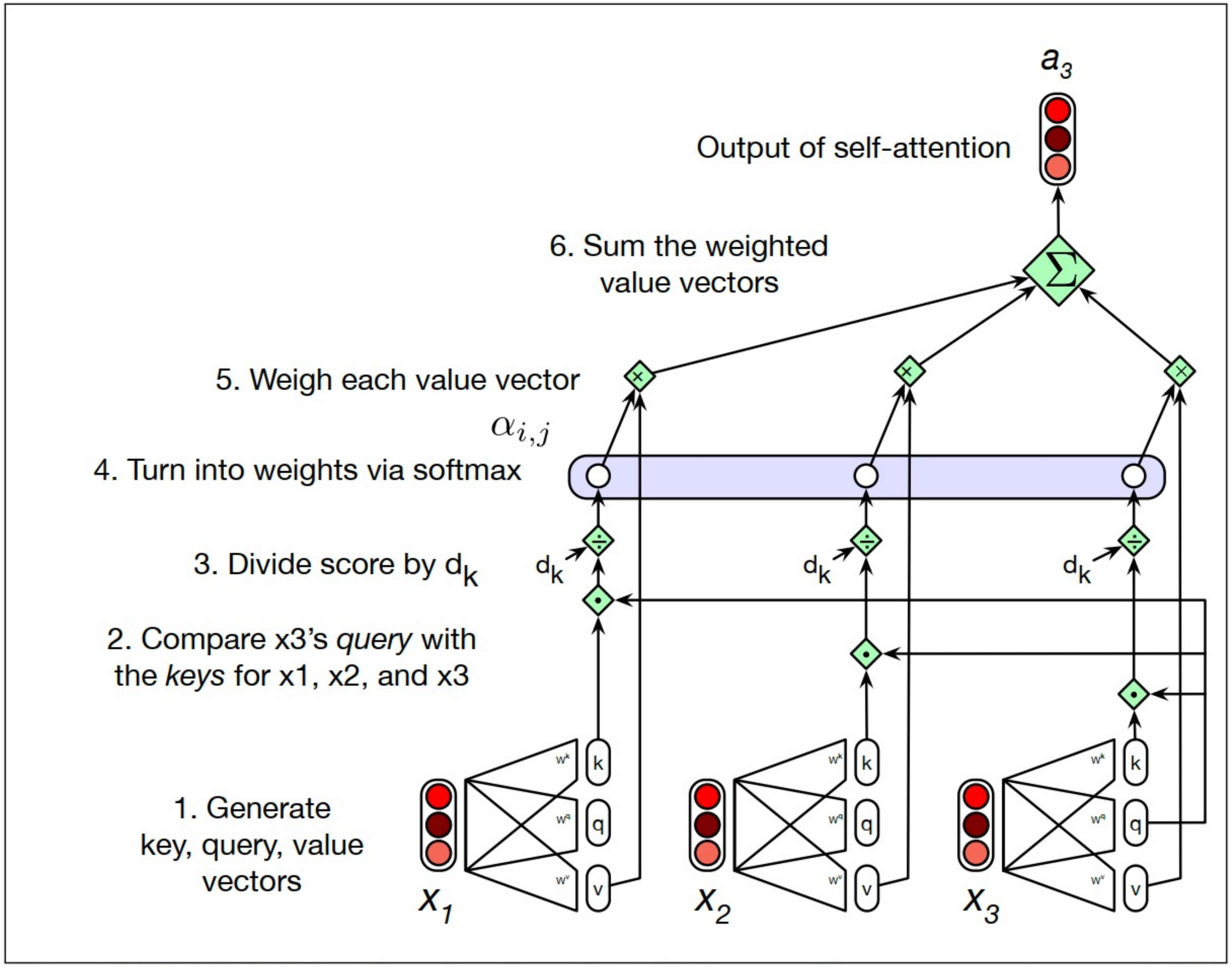


Figure 10.3 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

Multihead Attention

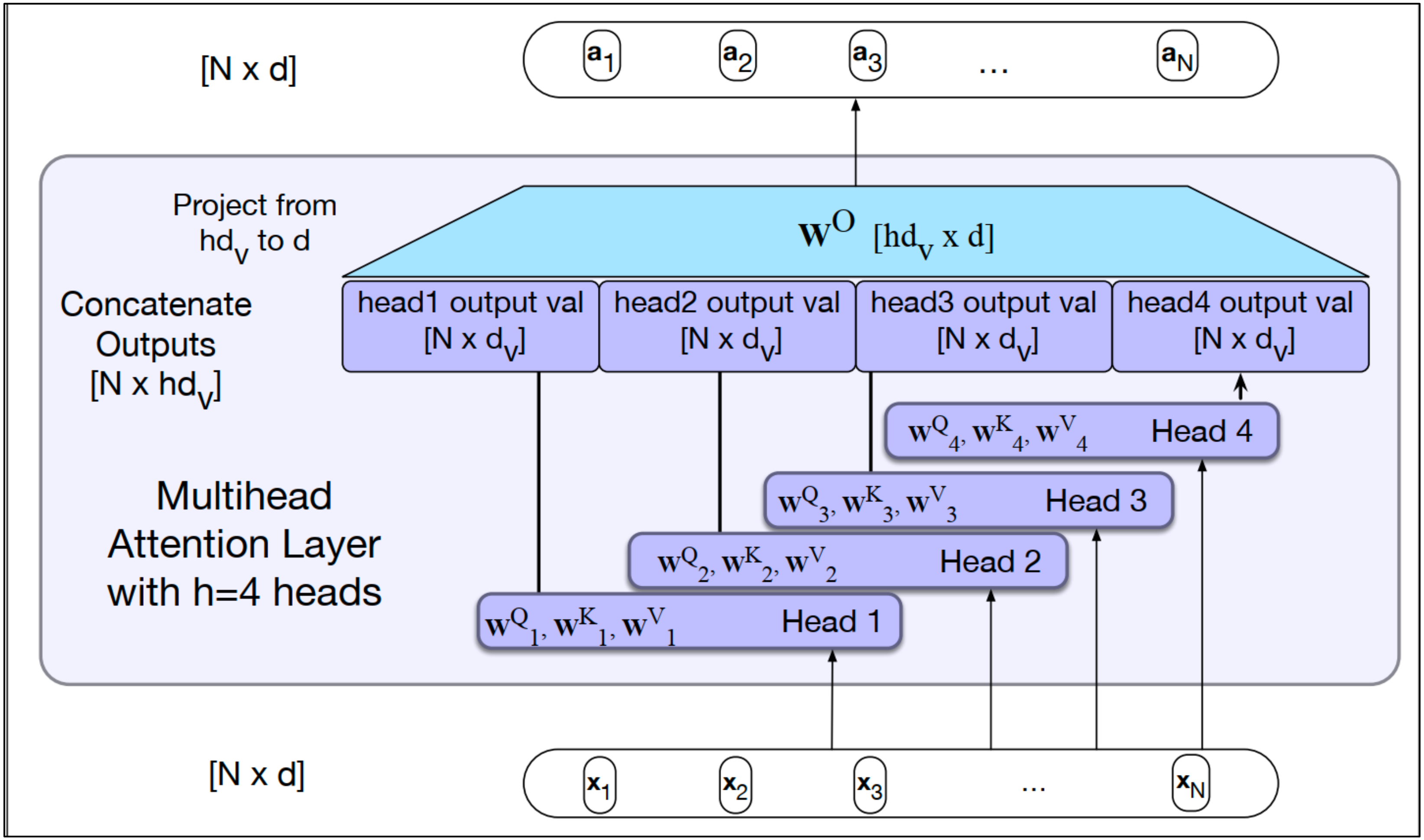
- Idea: use multi-heads to capture relationships between token in different ways: syntactic, semantic, discourse relationships
- Each head i is provided with its own sets of key, query, value matrices: $\Sigma_i^K, \Sigma_i^Q, \Sigma_i^V$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q \; ; \; \mathbf{K} = \mathbf{X}\mathbf{W}_i^K \; ; \; \mathbf{V} = \mathbf{X}\mathbf{W}_i^V \qquad (10.17)$$

$$\mathbf{head}_i = \mathrm{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \qquad (10.18)$$

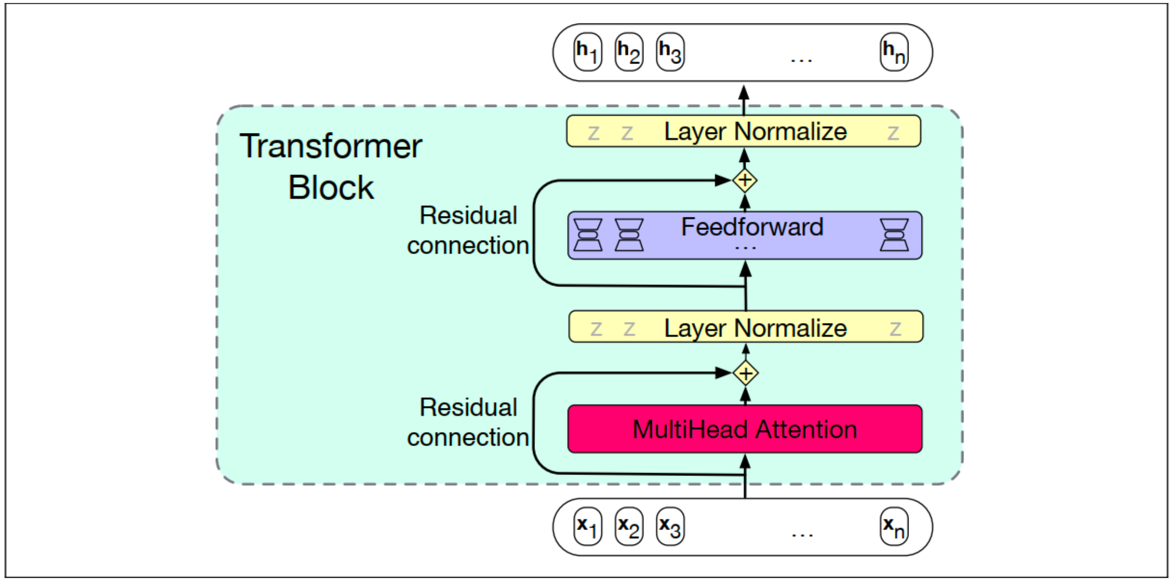
$$\mathbf{A} = \mathrm{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O \qquad (10.19)$$

Multihead Attention



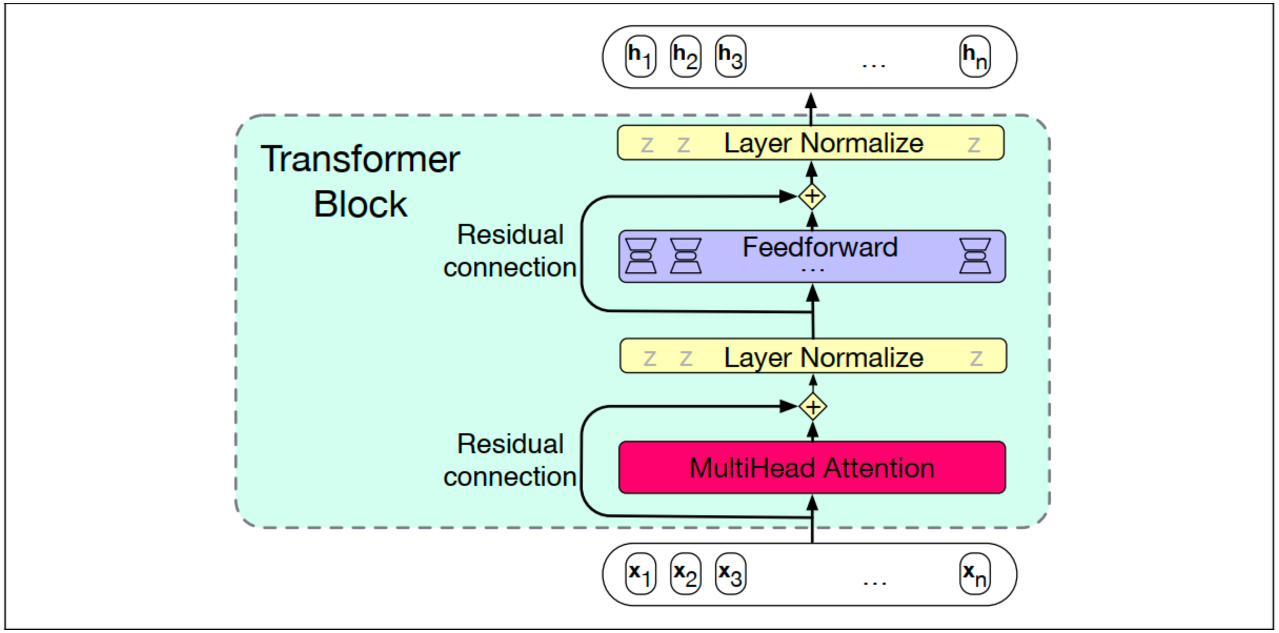
Transformer Block

- A Transformer block includes
  - A multihead self-attention layer
  - A feedforward layer
  - Residual connections
  - Normalizing Layer (Layer Norm)



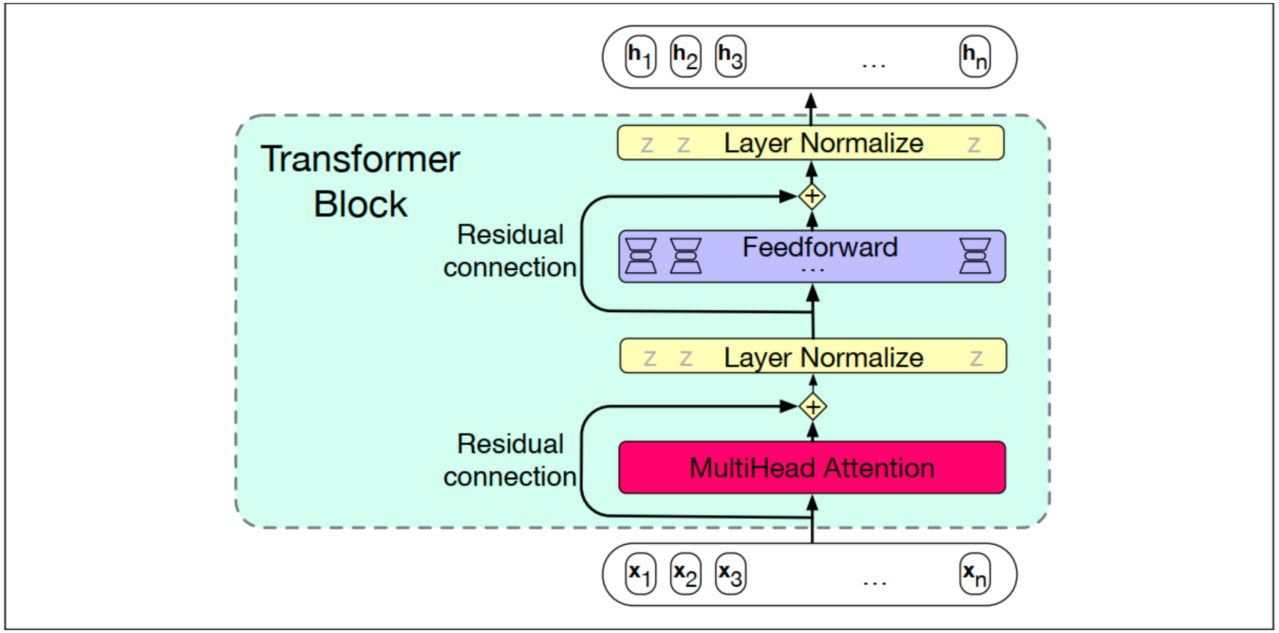
Feedforward layer

- Contains *N* position-wise network, one for each position



Residual Connections

- Adding a layer's input to its output vector before passing it forward



Layer Norm

- Normalize by the mean μ and standard deviation σ

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$$

$$\hat{\mathbf{x}} = \frac{(\mathbf{x} - \mu)}{\sigma}$$

$$\text{LayerNorm} = \gamma\,\hat{\mathbf{x}} + \beta$$

LLM with Transformers

- Sentiment analysis
  - The sentiment of the sentence "I like Jackie Chan" is:
  - Compare two probabilities calculated by Transformers
    - P(positive|The sentiment of the sentence "I like Jackie Chan" is:)
    - P(negative|The sentiment of the sentence "I like Jackie Chan" is:)
- Question Answering
  - Generate next tokens given the context
    - Q: Who wrote the book ‘‘The Origin of Species"? A:
    - P(w|Q: Who wrote the book "The Origin of Species"? A:)

Training Transformer Language Models

Self-supervision (or self-training)

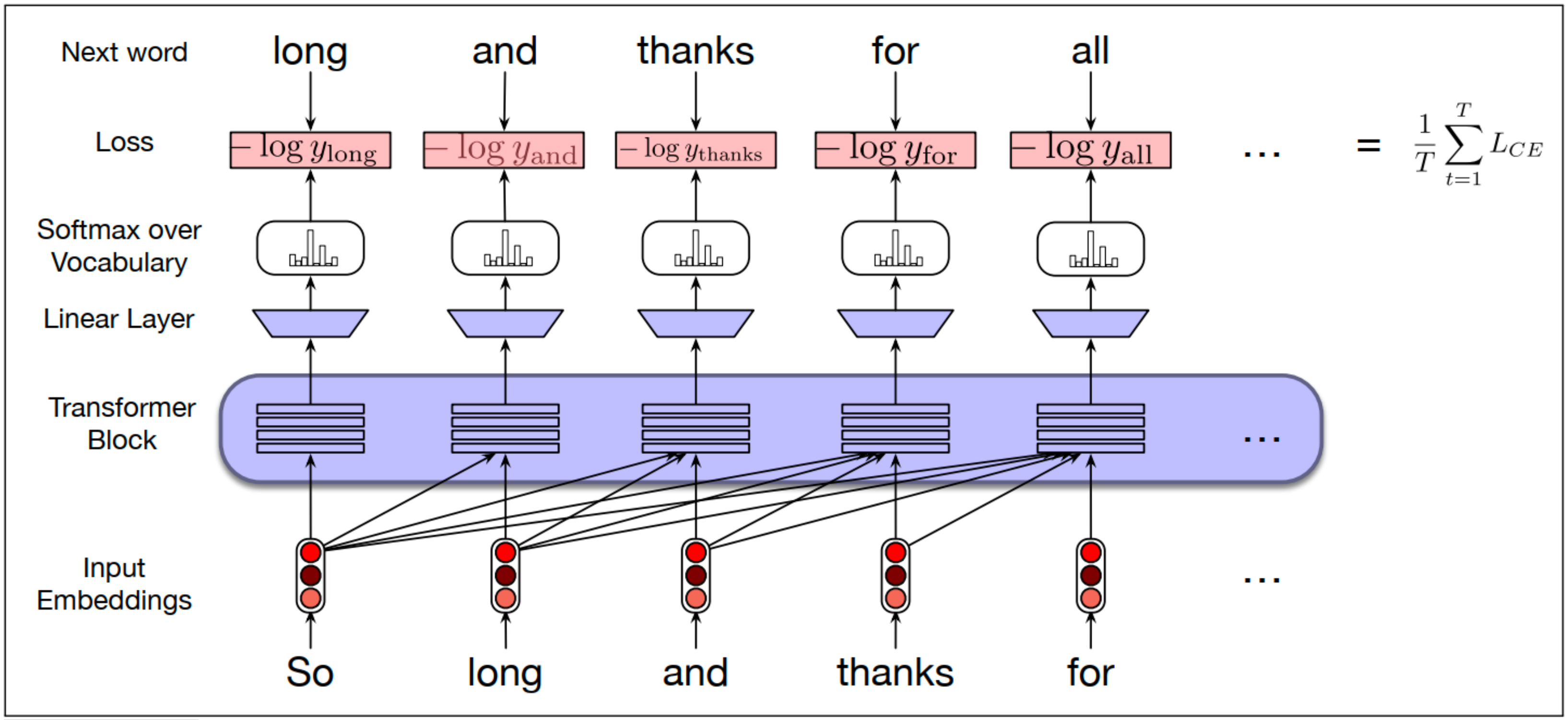


Figure 10.18 Training a transformer as a language model.

$$L_{CE} = -\sum_{w\in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

Generation by Sampling

- Two important factors in generation
  - Quality
  - Diversity
- Some sampling methods
  - Top-k sampling
  - Nucleus or top-p sampling
  - Temperature sampling

Top-k sampling

A simple generalization of greedy decoding.

- Choose in advance a number of words *k*
- For each word in the vocabulary V, use the language model to compute the likelihood of this word given the context $p(w_t|w_{<t})$
- Sort the words by their likelihood, and throw away any word that is not one of the top *k* most probable words
- Renormalize the scores of the *k* words to be a legitimate probability distribution.
- Randomly sample a word from within these remaining *k* most-probable words according to its probability.

Nucleus or top-p sampling

- Keep not top *k* words, but the top *p* percent of the probability mass
- Given a distribution $P(w_t|w_{<t})$, the top-p vocabulary $V^{(p)}$ is the smallest set of words such that

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p.$$

Temperature sampling

- Instead of computing the probability distribution by:

$$y = \mathrm{softmax}(u)$$

we compute the probability distribution by:

$$y = \mathrm{softmax}(u/\tau)$$

Useful properties of sofmax function: tends to push high values toward 1 and low values toward 0