CSCI 544: Applied Natural Language Processing

Transformer-I

Xuezhe Ma (Max)

USC University of Southern California

Course Organization

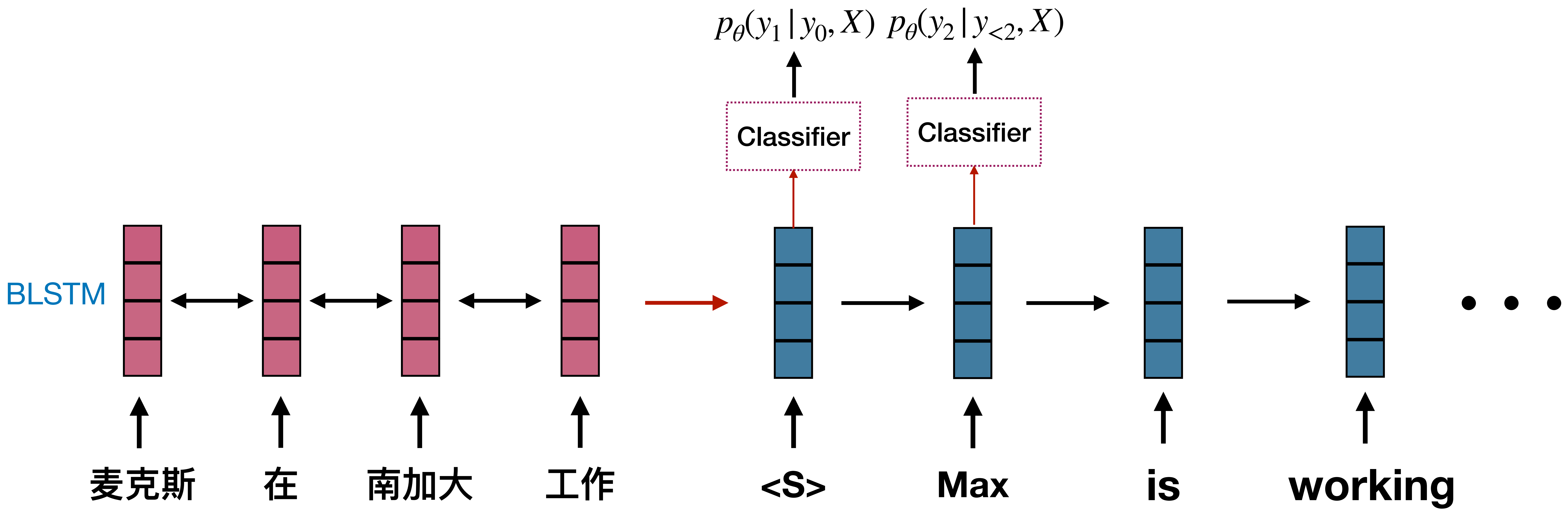
NLP Tasks

Models

	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

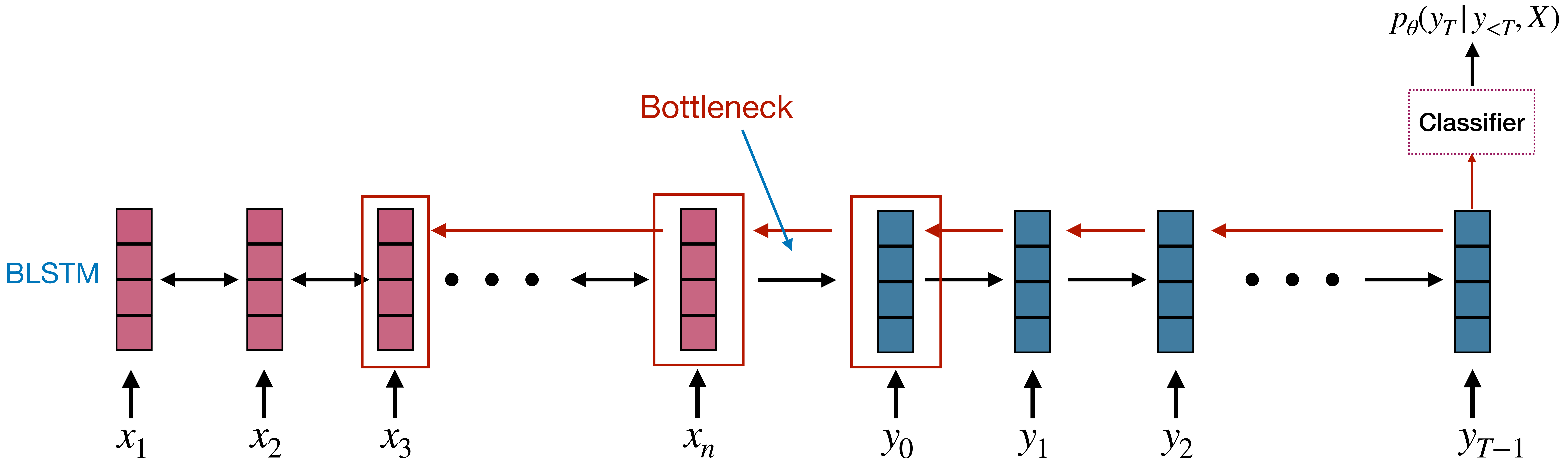
Recap: Encoder-Decoder Architecture

- **Two Components:**
  - Encoder: Convert input sequence into a sequence of vectors
  - Decoder: Convert decoding into a sequence in the output space

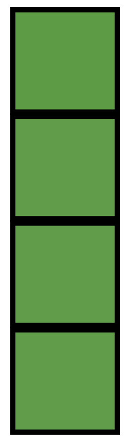


Revisit: Motivation of Attention Mechanism

- A single encoding vector needs to capture **all the information** about source sentence
- Longer sequences can lead to **vanishing gradients**



Recap: Attention Mechanism



$$= \text{attn}([c_1, c_2, \ldots, c_n], h_t) = \sum_{j=1}^{n} a_j^t c_j \in \mathbb{R}^d$$

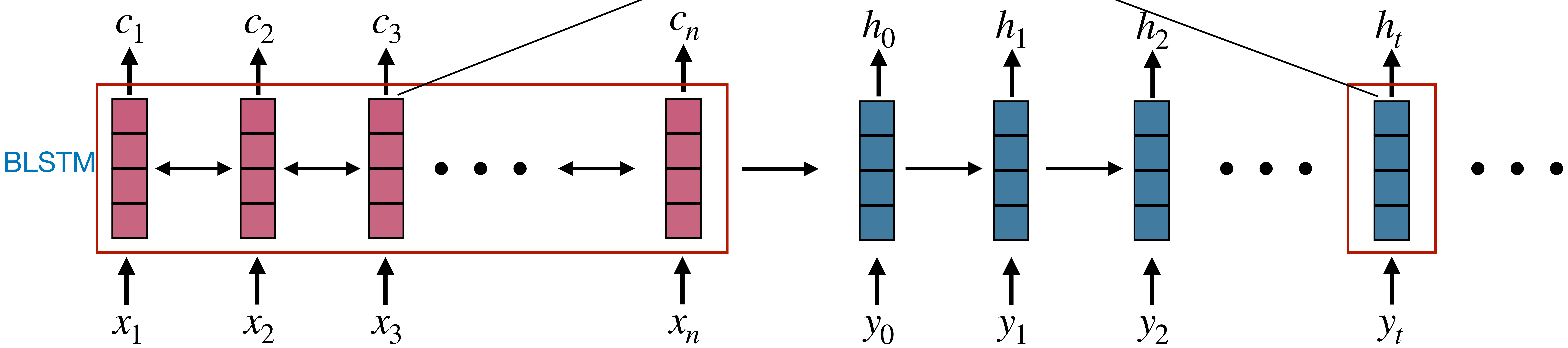
Attention output

$$a^t = \text{softmax}(e^t) \in \mathbb{R}^n$$

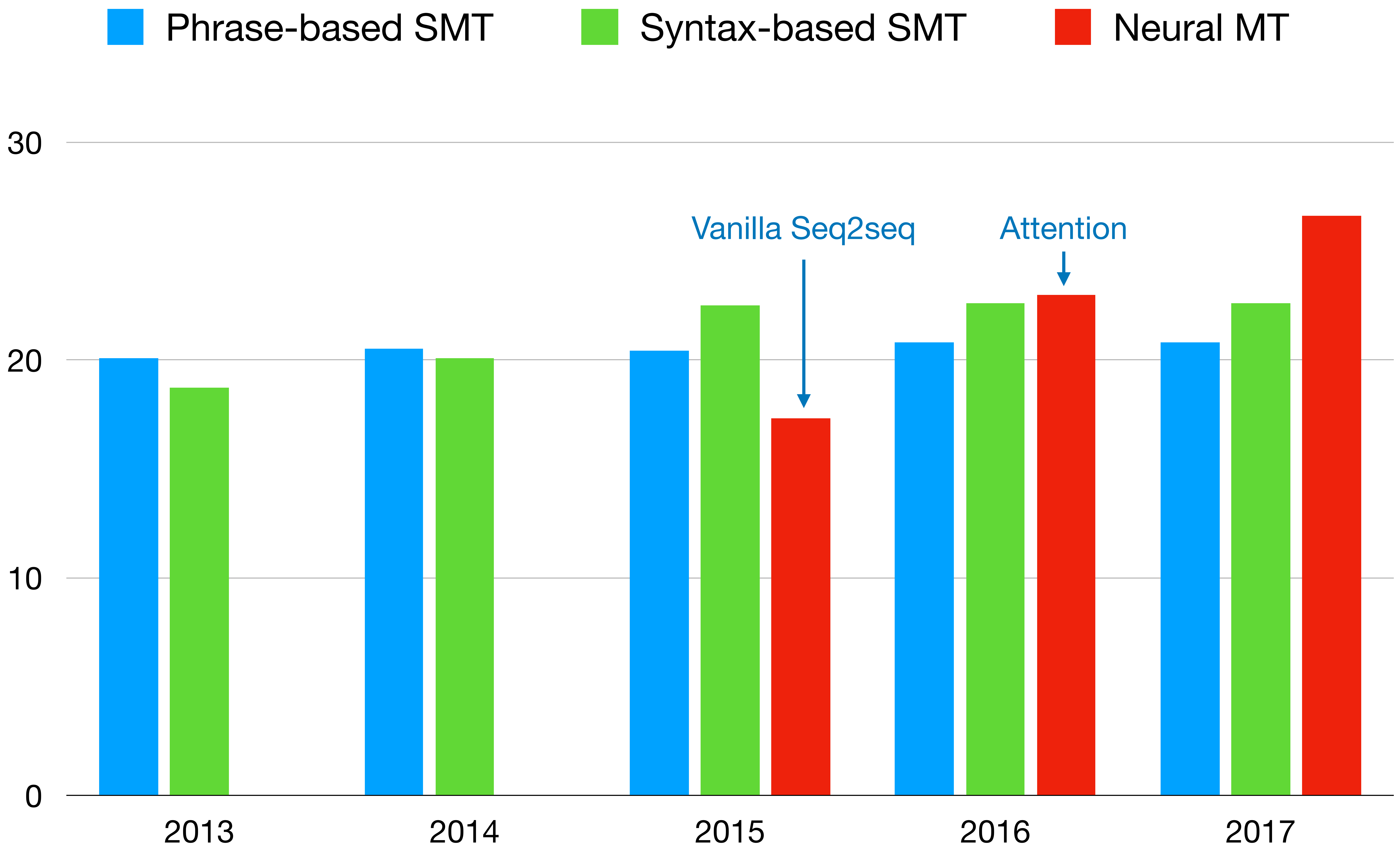
Attention distribution

$$e_j^t = \text{sim}(c_j, h_t),\ \forall j \in \{1, \ldots, n\}$$

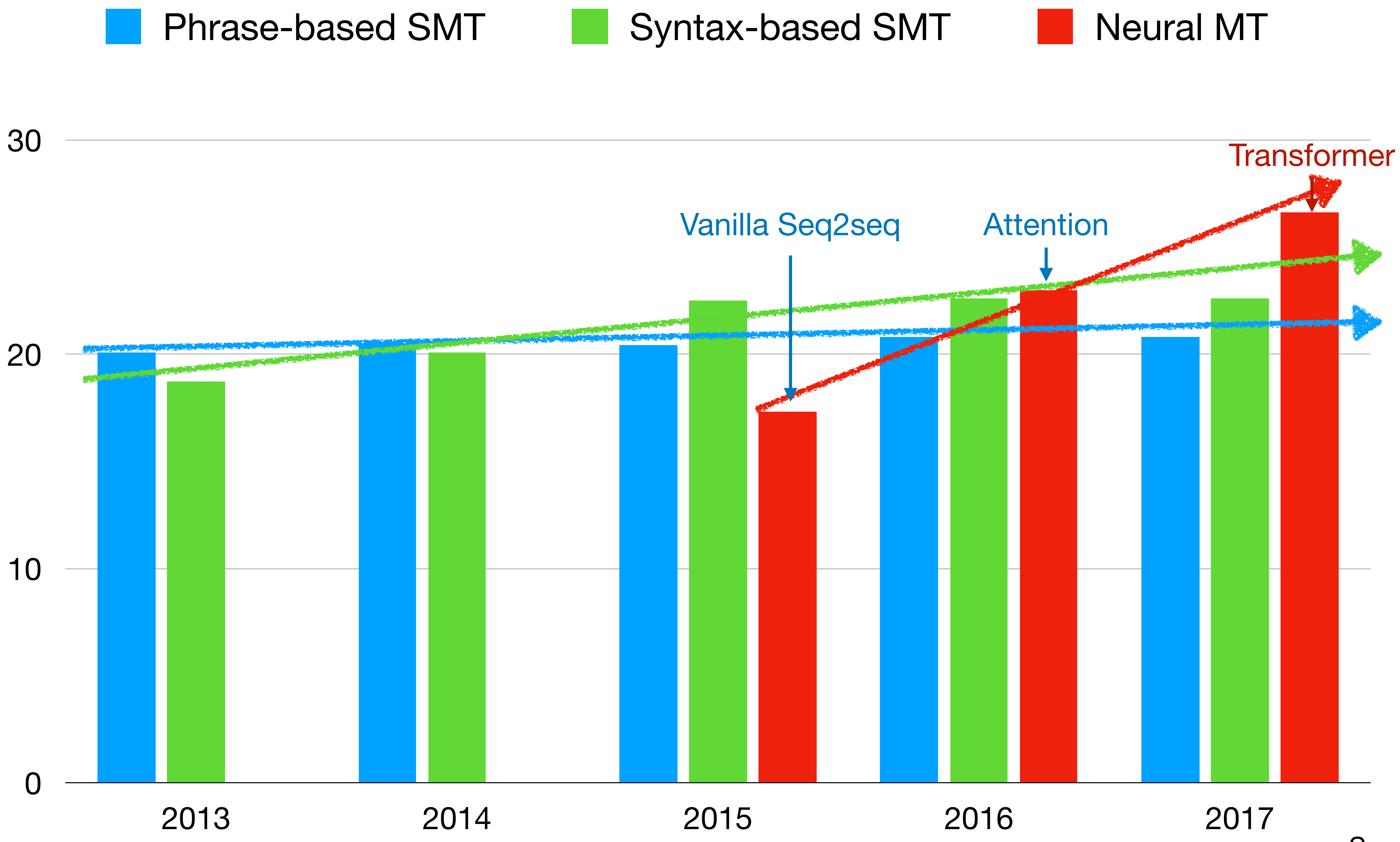
Attention scores



Recap: MT Progress



MT Progress



Source: Rico Sennrich

Transformer

This Lecture

- Do we really need RNNs to model the arbitrary context?
- Maybe attention is all you need!

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhi illia.polosukhin@g

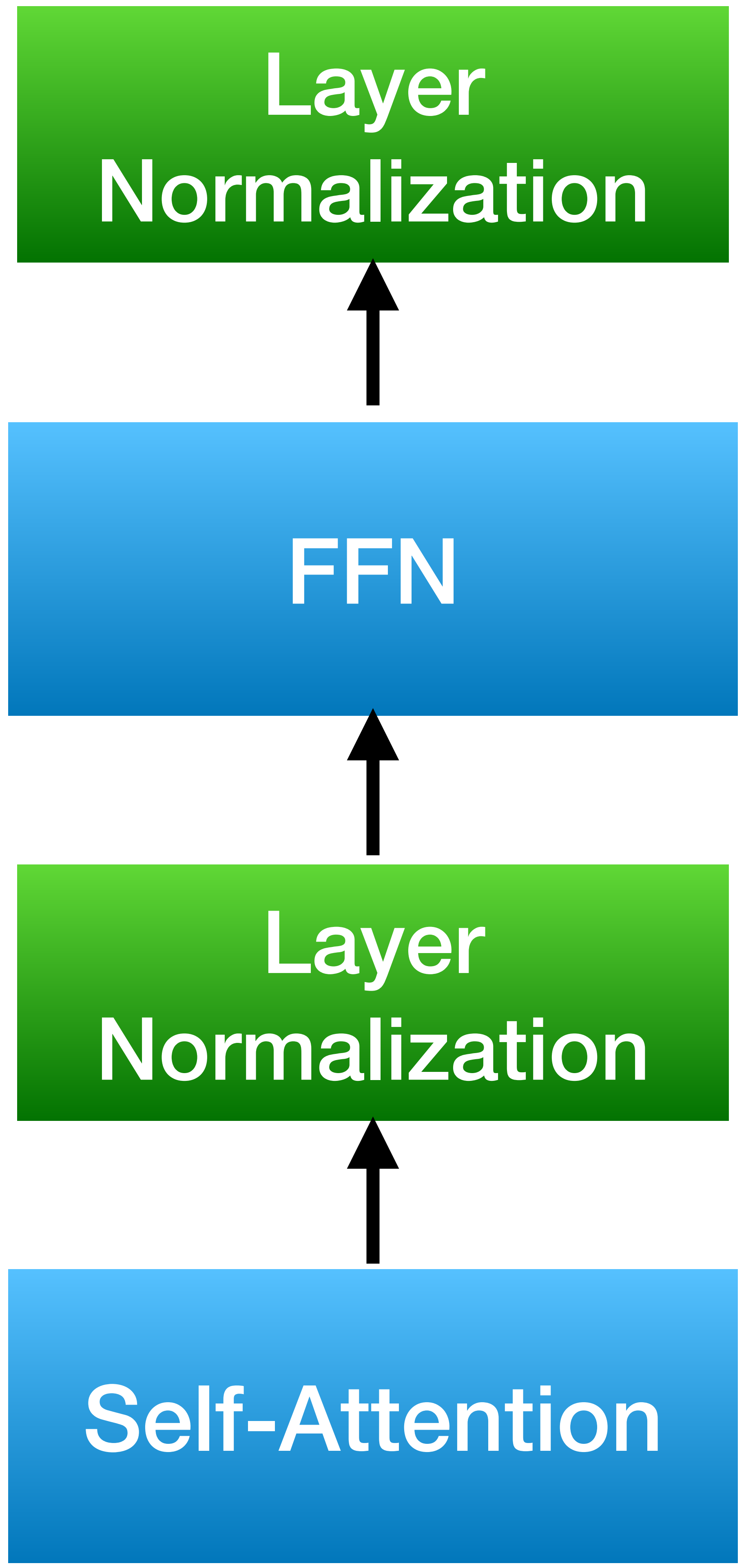
[CITATION] **Attention is all you need**

A Vaswani - Advances in Neural Information Processing Systems, 2017

Save Cite Cited by 153082 Related articles

Transformers

Transformer Encoder Block

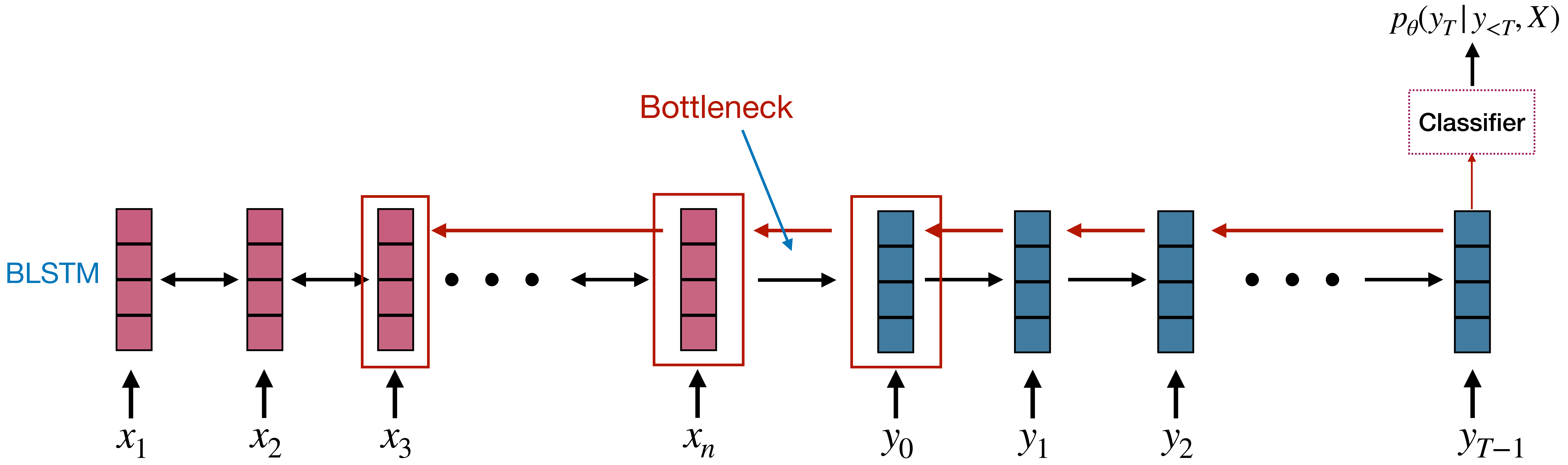


- **Three Key Components**
  - (Masked) Multi-head Self-Attention
  - Layer Normalization
  - Position-wise Feed-Forward Network

Self-Attention

Revisit: Motivation of Attention Mechanism

- A single encoding vector needs to capture **all the information** about source sentence
- Longer sequences can lead to **vanishing gradients**



Issues with RNN!

Recap: Attention Mechanism

$$\blacksquare = \mathrm{attn}([c_1, c_2, \ldots, c_n], h_t) = \sum_{j=1}^{n} a_j^t c_j \in \mathbb{R}^d$$ Attention output

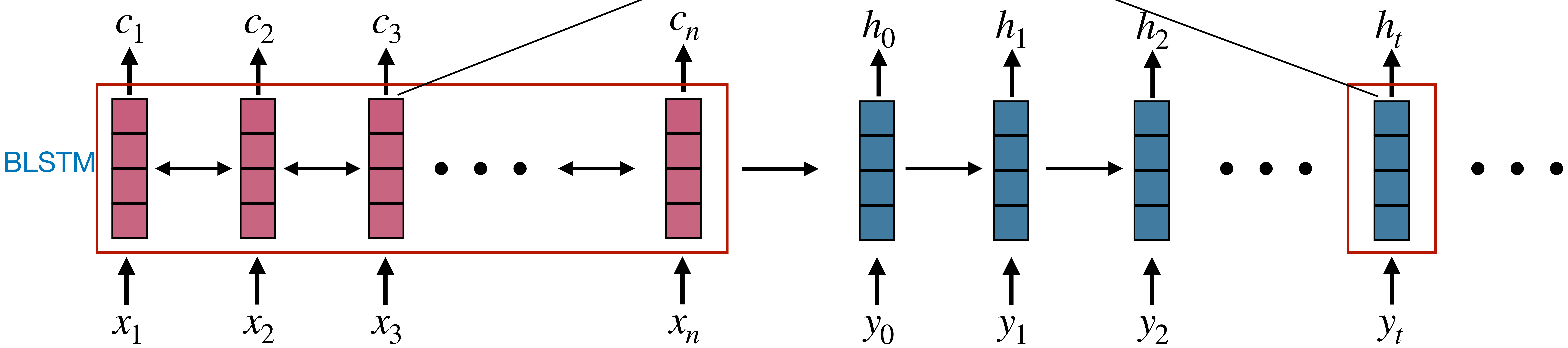
$$a^t = \mathrm{softmax}(e^t) \in \mathbb{R}^n$$ Attention distribution

$$e_j^t = \mathrm{sim}(c_j, h_t), \forall j \in \{1, \ldots, n\}$$ Attention scores

Why we need LSTM?

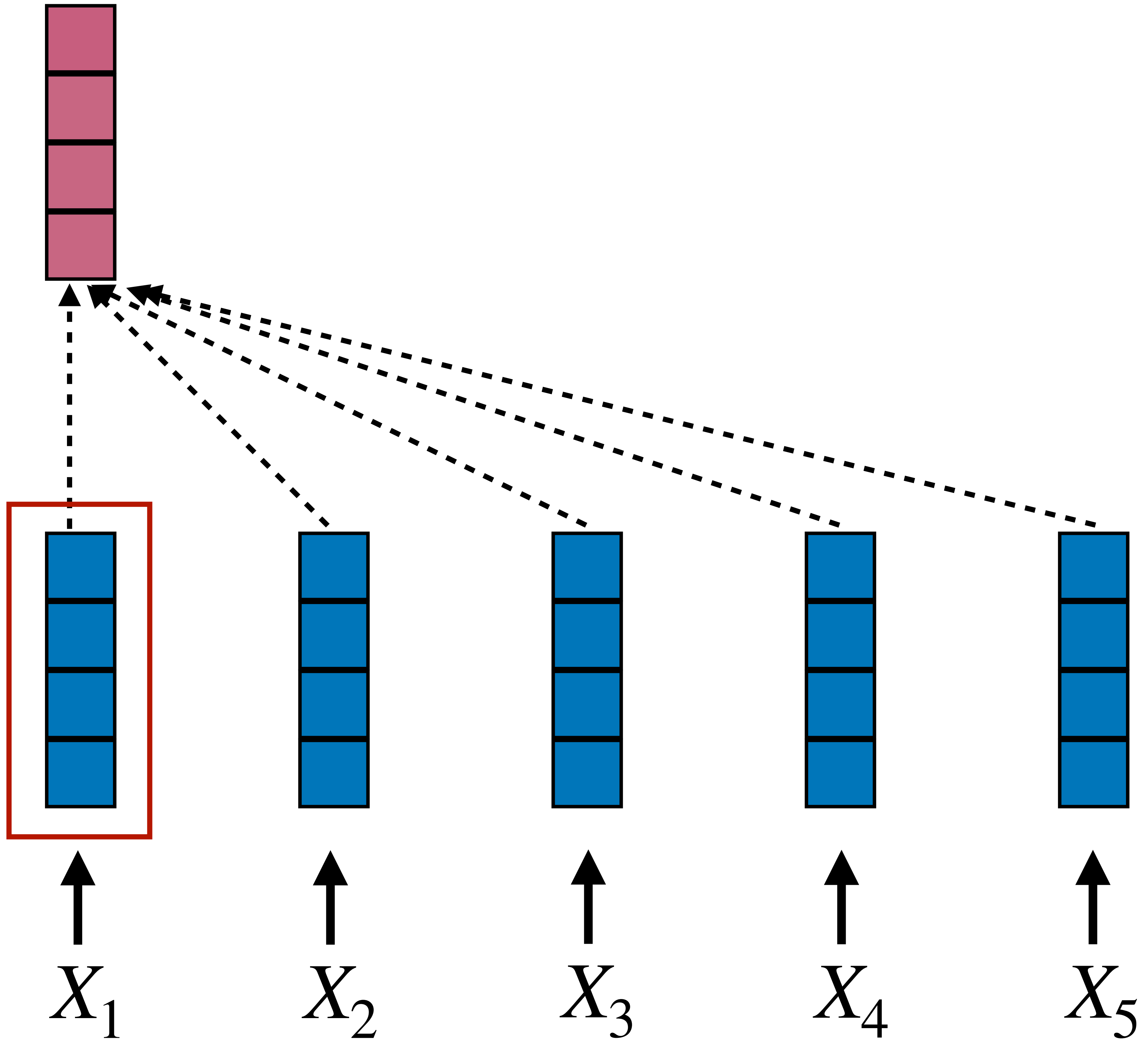
Modeling contextual information in both source and target languages

Contextual information via attention?



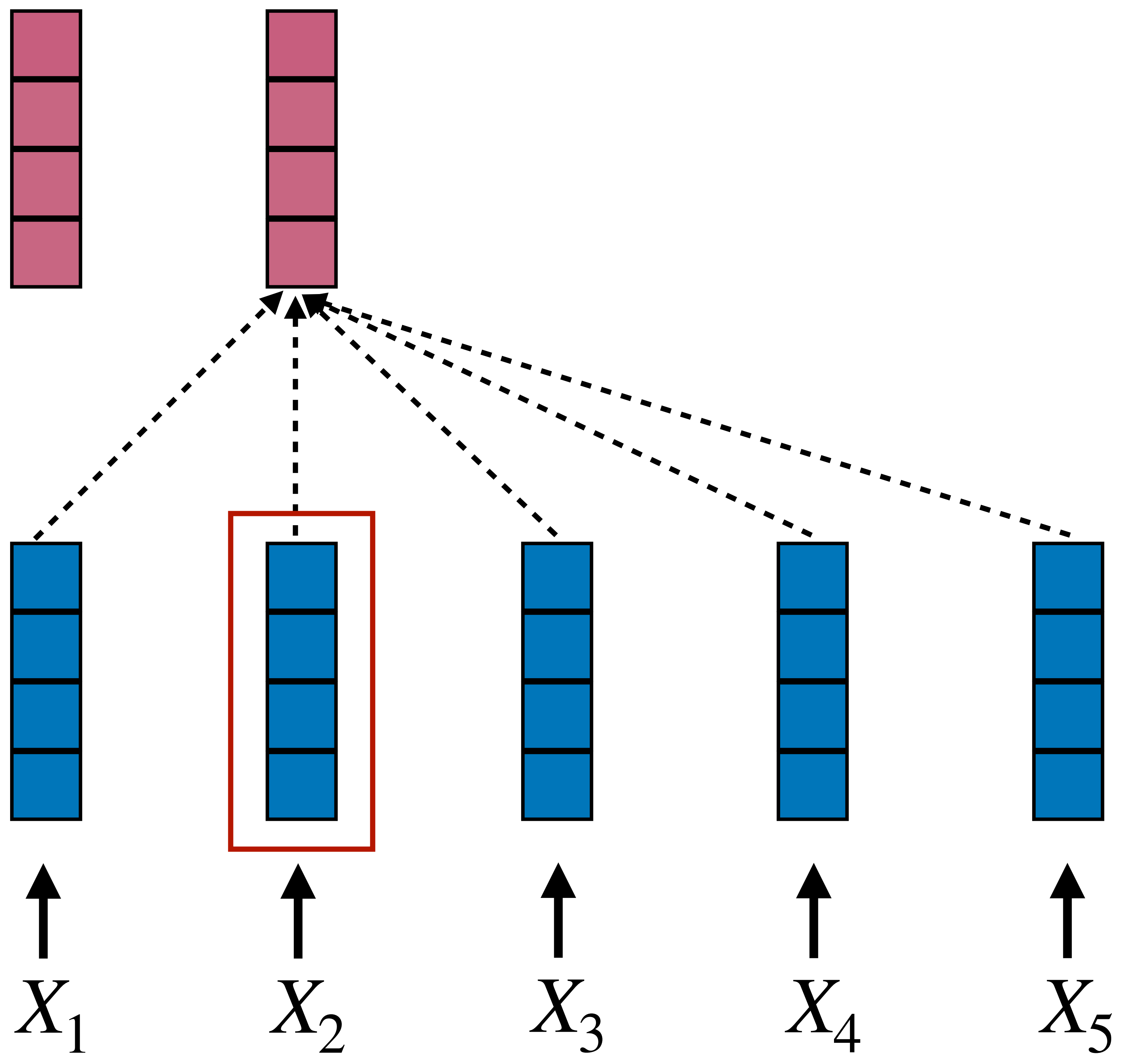
Self-Attention

- **Self-attention: attention within on single sequence**
  - Contexts and queries are drawn from the same source
- **Contextual information via self-attention**



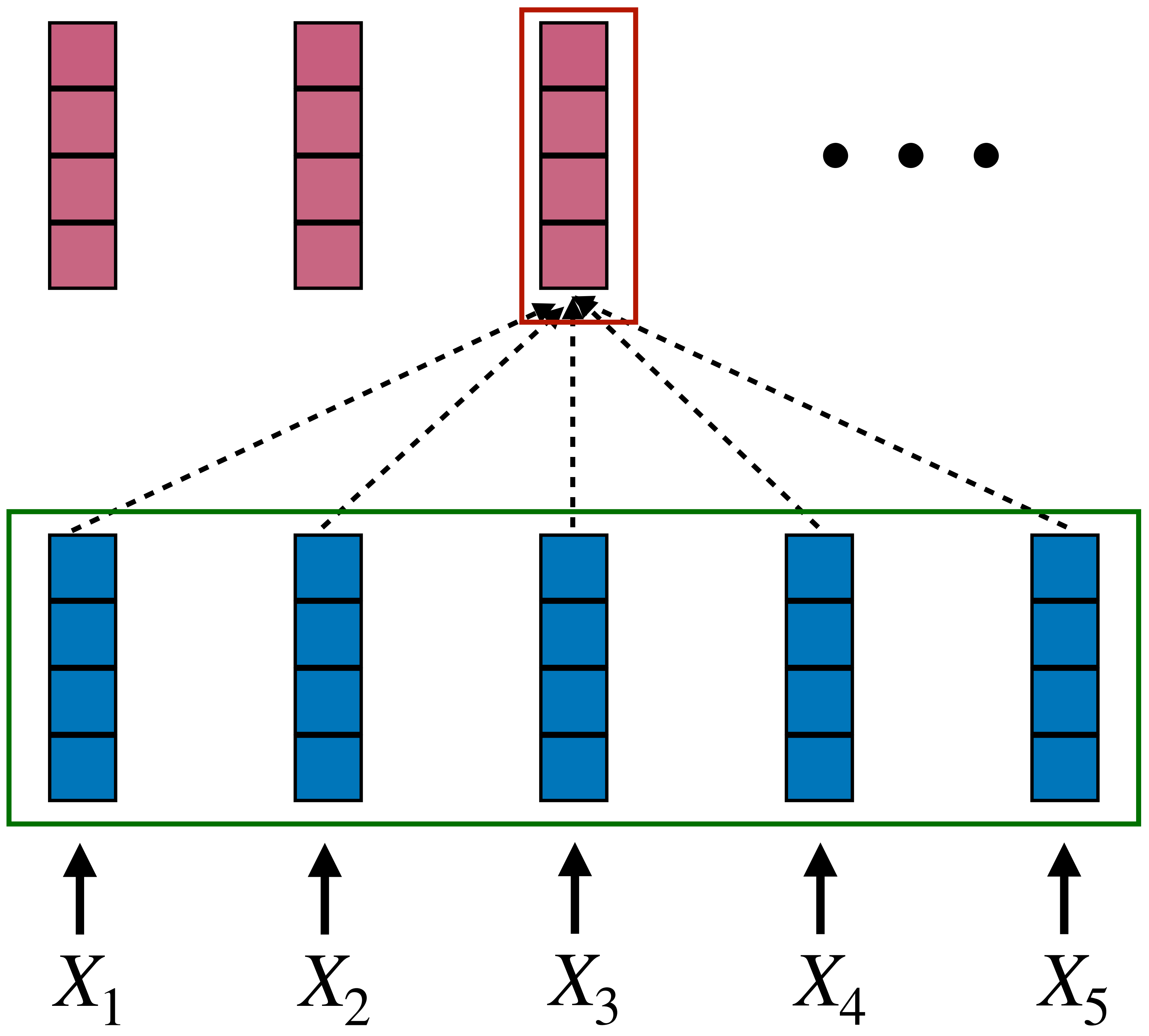
Self-Attention

- **Self-attention: attention within on single sequence**
  - Contexts and queries are drawn from the same source
- **Contextual information via self-attention**



Self-Attention

- **Self-attention: attention within on single sequence**
  - Contexts and queries are drawn from the same source
- **Contextual information via self-attention**

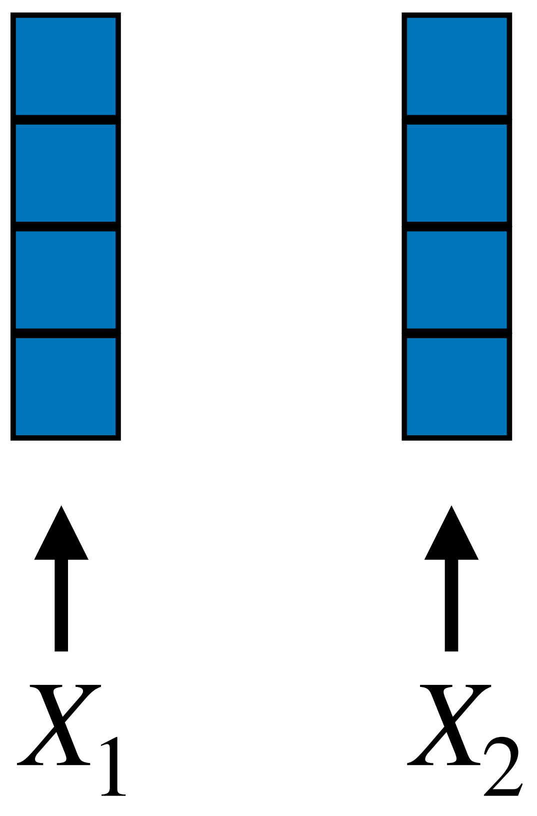


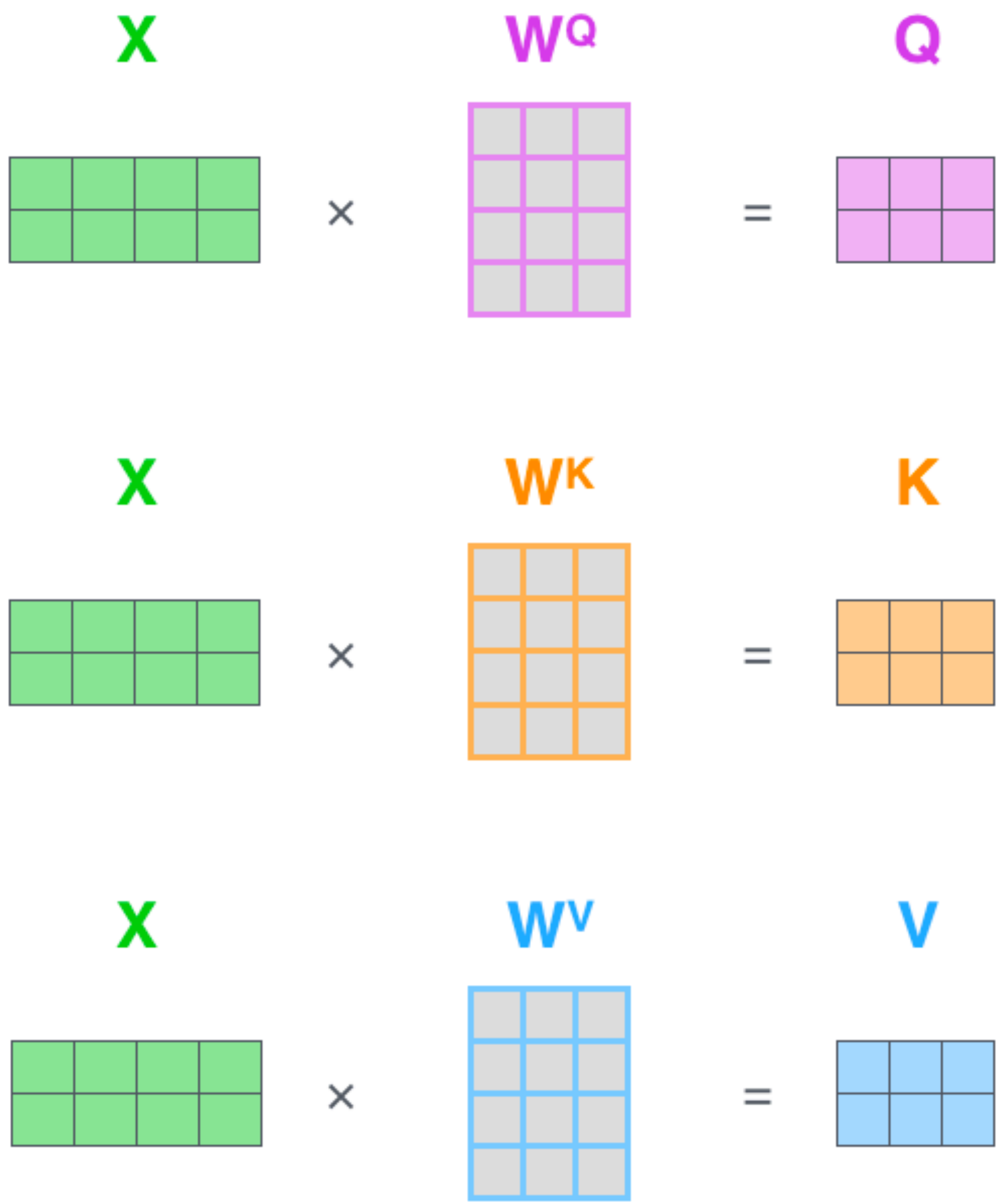
- **Capturing long-distance dependencies**
- **No gradient vanishing**
- **Parallel computation!**

Self-attention in equations

- A sequence of input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$
- First, construct a set of queries, keys and values:

$$q_i = W_Q x_i,\ k_i = W_K x_i,\ v_i = W_V x_i$$





Self-attention in equations

- A sequence of input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$
- First, construct a set of queries, keys and values:

$$q_i = W_Q x_i,\ k_i = W_K x_i,\ v_i = W_V x_i$$

- Second, for each q_i, compute attention scores and attention distributions:

$$a_{i,j} = \mathrm{softmax}(\frac{q_i^T k_j}{\sqrt{d}})$$

aka. "scaled dot product"

- Finally, compute the weighted sum:

$$y_i = \sum_{j=1}^{n} a_{i,j} v_j$$

Why *Scaled* Dot Product?

- **Softmax is sensitive to scale**

If $[x_1, x_2] = [0.1, 0.5]$, $\alpha = 10$

$$\text{softmax}([x_1, x_2]) = \left[\frac{e^{x_1}}{e^{x_1} + e^{x_2}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2}}\right]$$

$[0.4013, 0.5987]$

$$\text{softmax}([\alpha x_1, \alpha x_2]) = \left[\frac{e^{\alpha x_1}}{e^{\alpha x_1} + e^{\alpha x_2}}, \frac{e^{\alpha x_2}}{e^{\alpha x_1} + e^{\alpha x_2}}\right]$$

$[0.0180, 0.9820]$

Self-attention in equations

- A sequence of input vectors $x_1, \ldots, x_n \in \mathbb{R}^d$
- First, construct a set of queries, keys and values:

$$q_i = W_Q x_i,\ k_i = W_K x_i,\ v_i = W_V x_i$$

- Second, for each q_i, compute attention scores and attention distributions:

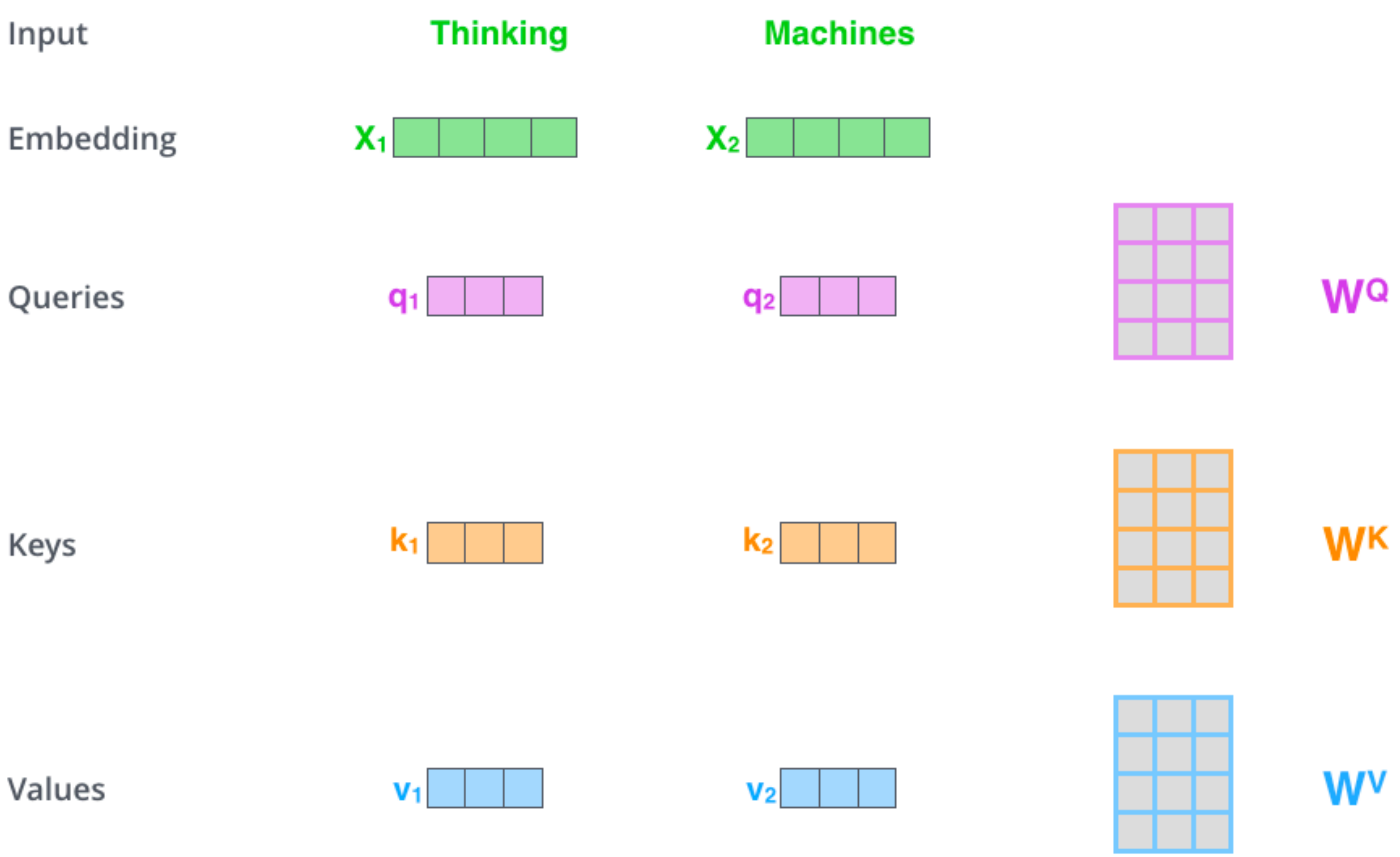
$$a_{i,j} = \mathrm{softmax}\left(\frac{q_i^T k_j}{\sqrt{d}}\right)$$

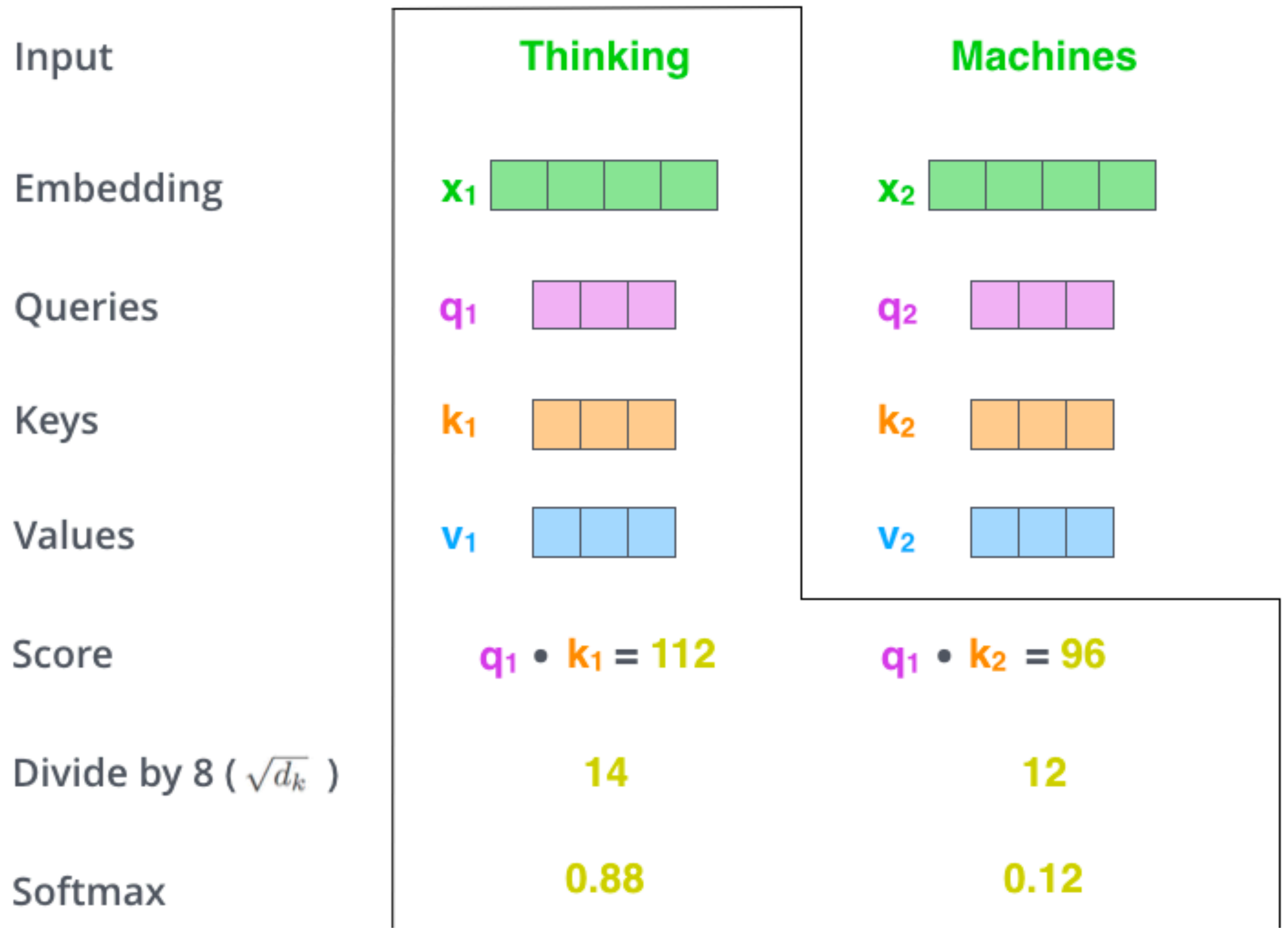
aka. “scaled dot product”

- Finally, compute the weighted sum:

$$y_i = \sum_{j=1}^{n} a_{i,j} v_j$$

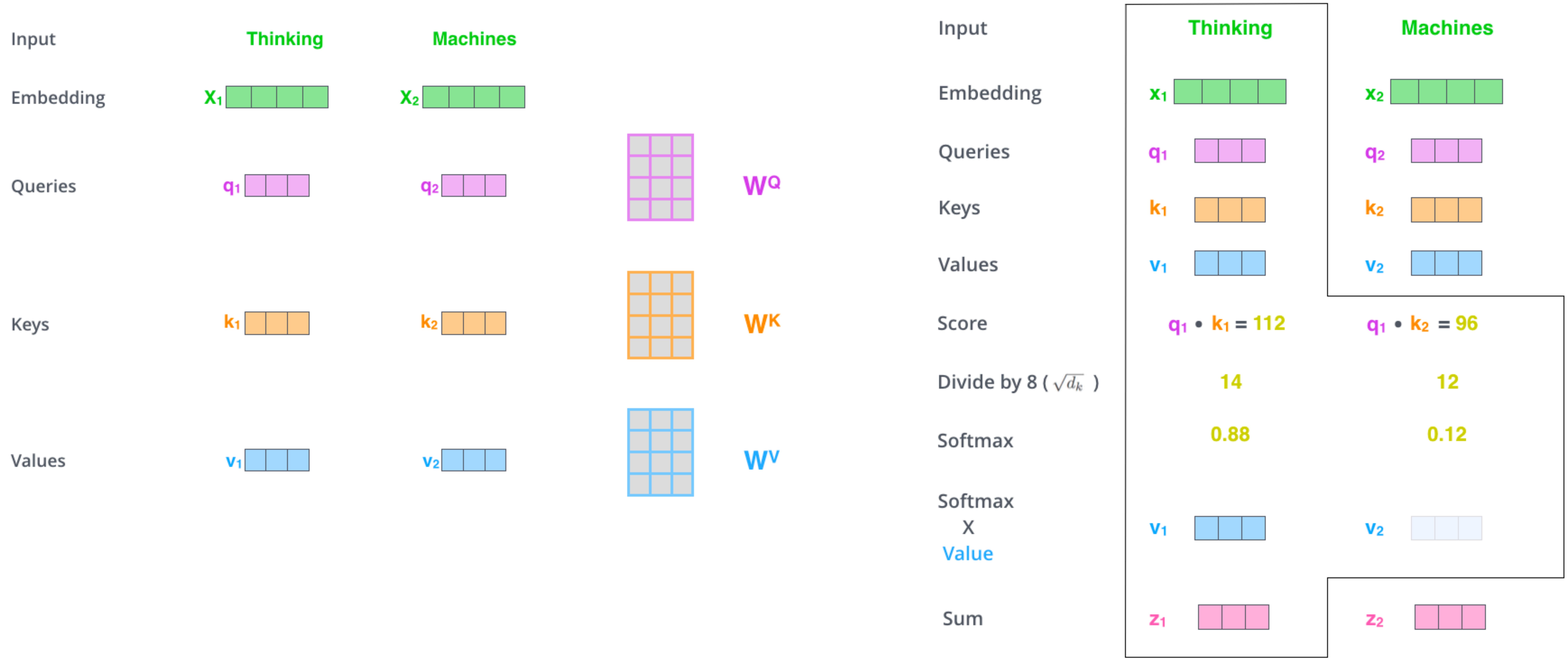
Self-attention: Illustration





http://jalammar.github.io/illustrated-transformer/

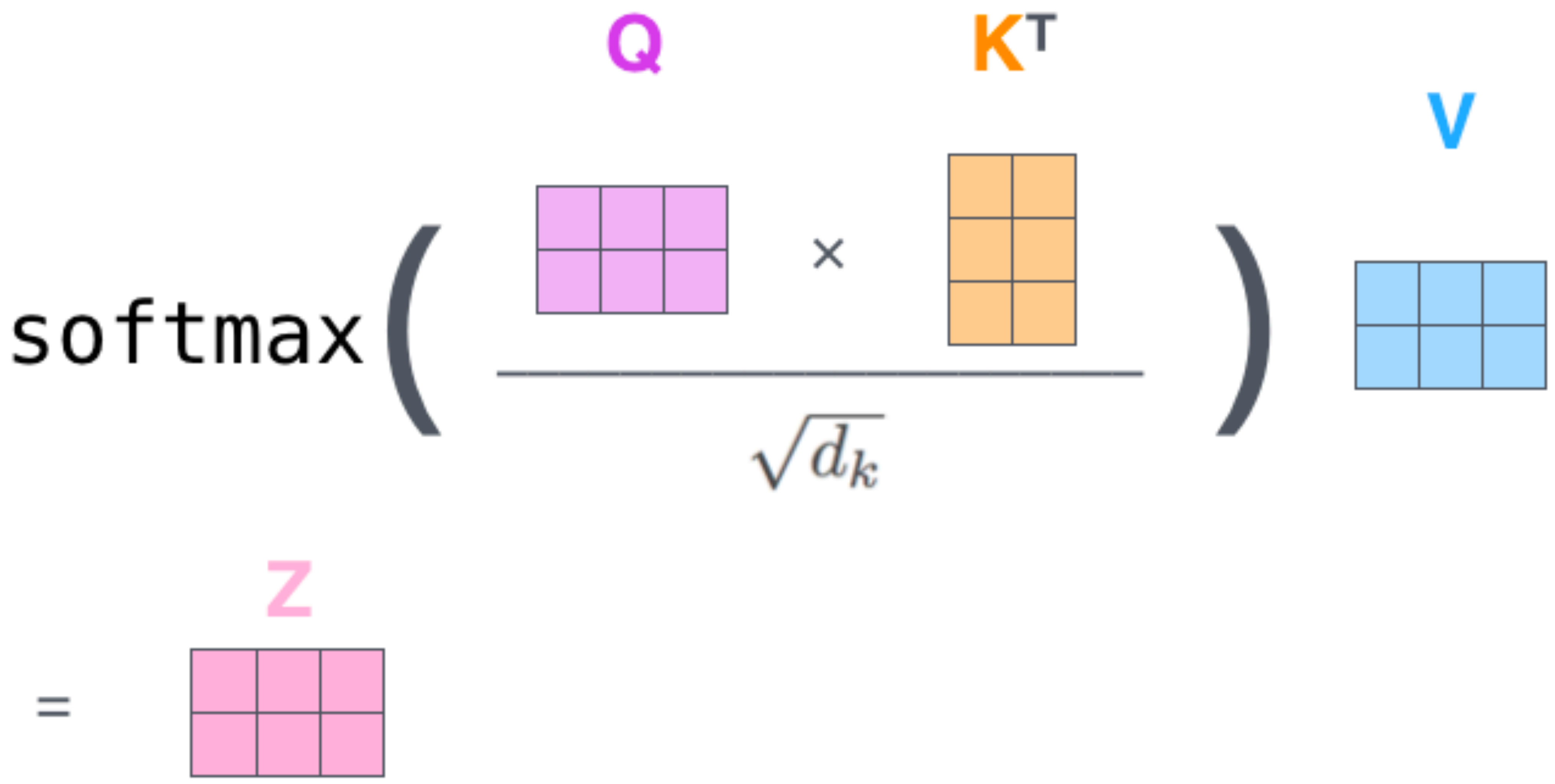
Self-attention: Illustration



http://jalammar.github.io/illustrated-transformer/

Self-attention: matrix notations

$$\boxed{\mathrm{attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}})V}$$



Self-attention: matrix notations

hardmaru
@hardmaru

The most important formula in deep learning after 2018

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d, $X \in \mathbf{R}^{n \times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ to extract feature representations Q, K, and V, referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q, K, V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{1}$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{2}$$

where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

9:08 PM · Feb 9, 2021 · Twitter Web App

580 Retweets **38** Quote Tweets **3,407** Likes

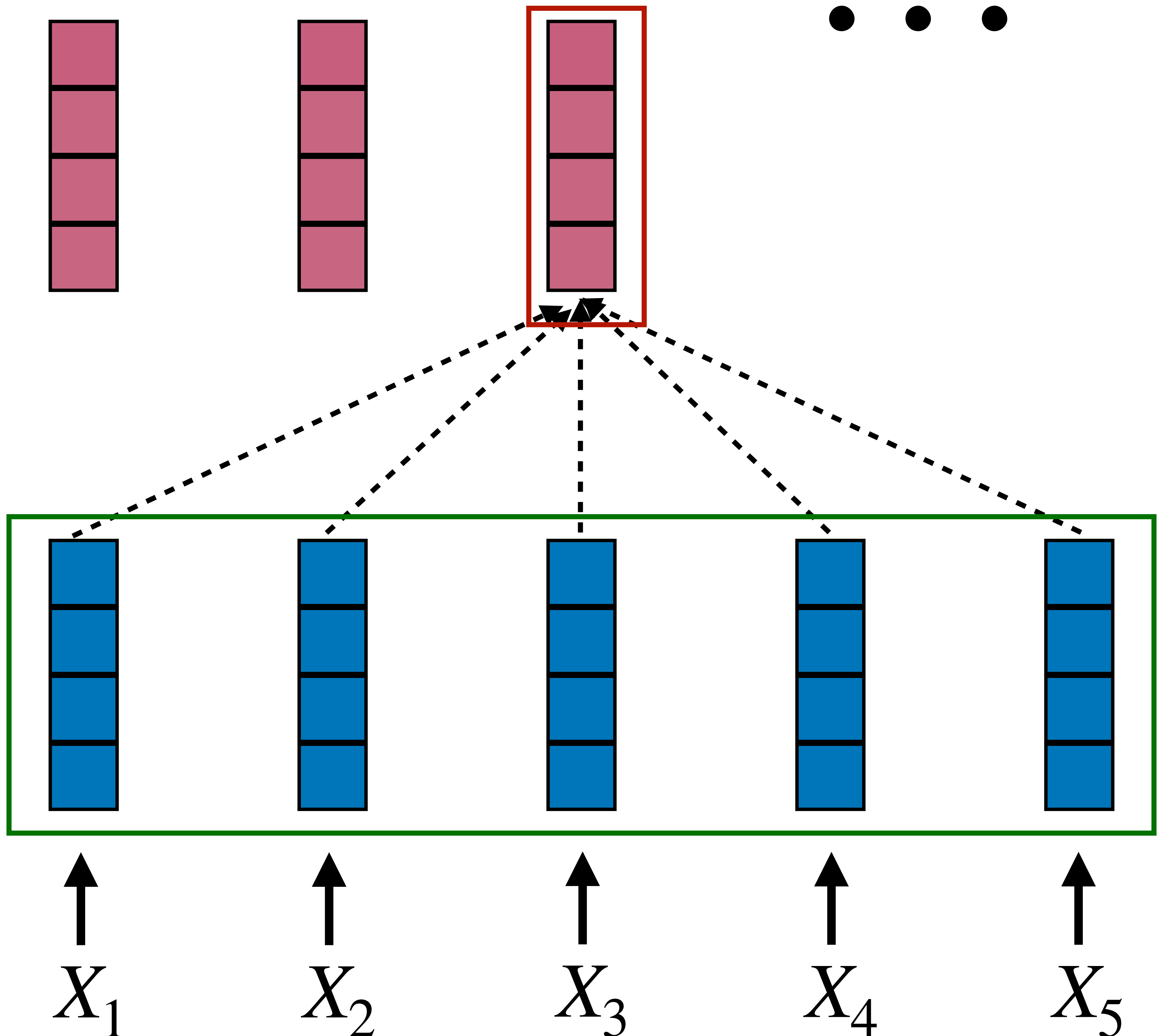
Attention is *General*

- **Given a set of key and value vectors, and a query vector, attention is a technique to compute a weighted sum of the value vectors, dependent on the query and keys**
  - We sometimes say that the query attends to the values via keys
  - In the NMT case, each decoder hidden state (query) attends to all the encoder hidden states (keys and values)

$$\boxed{\mathrm{attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}})V}$$

Attention is *General*

- **Intuition**
  - The weighted sum is a *selective summary* of the information contained in the values, where the query and keys determines which values to focus on
  - Attention is a way to obtain a *fixed-size representation* of an arbitrary set of representations (the values), dependent on some other representation (the query)



Multi-head Attention

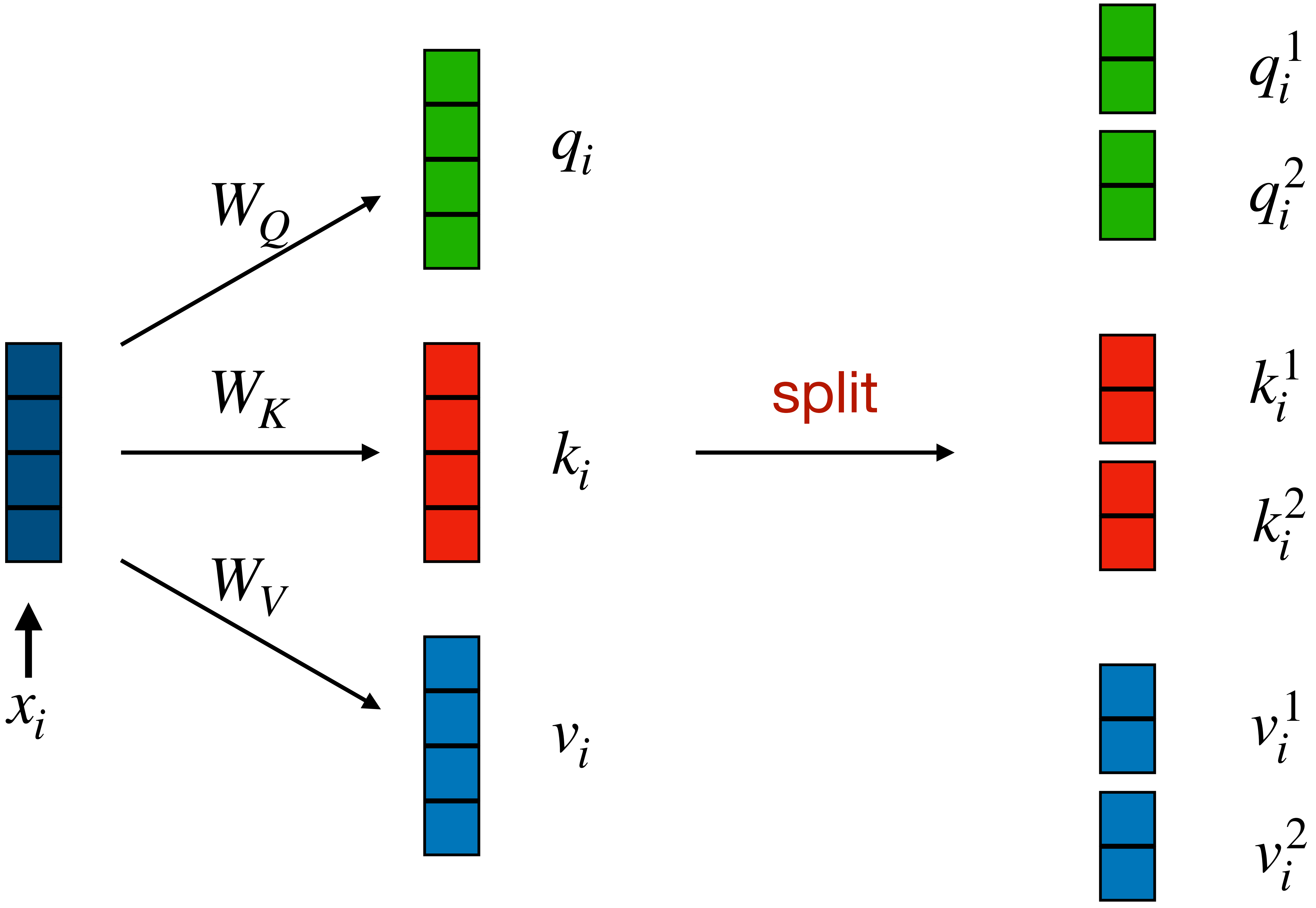
- **Problem with self-attention?**

$$y_i = \sum_{j=1}^{n} a_{i,j} v_j$$

one set of attention weights a_i

- **It is better to use multiple attention weights instead of one!**
  - Each attention can focus on different positions
- **How to do this? Splits queries, keys, values to multiple heads!**

Multi-head Attention: Head Split

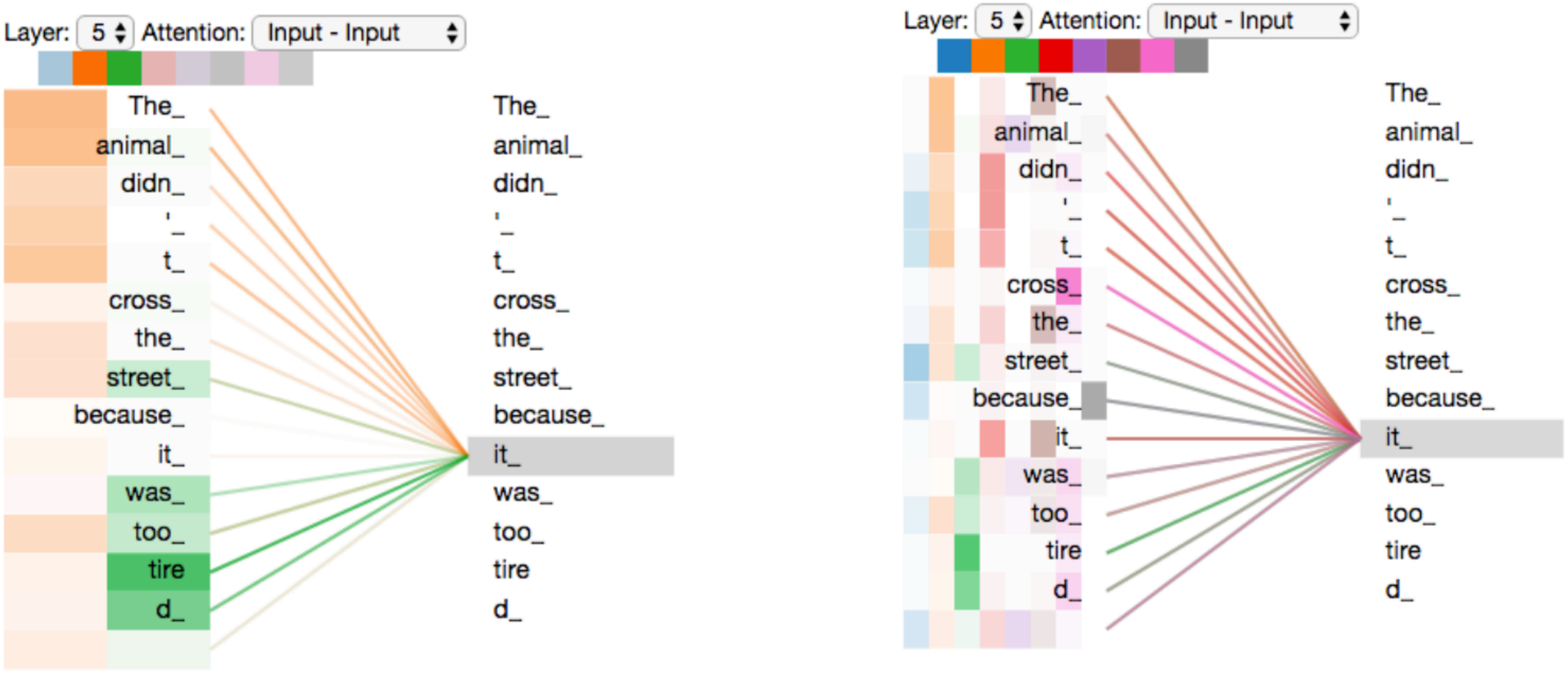


$$h_1 = \mathrm{attn}(Q_1, K_1, V_1) = \mathrm{softmax}(\frac{Q_1K_1^T}{\sqrt{d/2}})V_1$$

$$h_2 = \mathrm{attn}(Q_2, K_2, V_2) = \mathrm{softmax}(\frac{Q_2K_2^T}{\sqrt{d/2}})V_2$$

$$Y = \mathrm{concat}(h_1,\ h_2)W_O$$

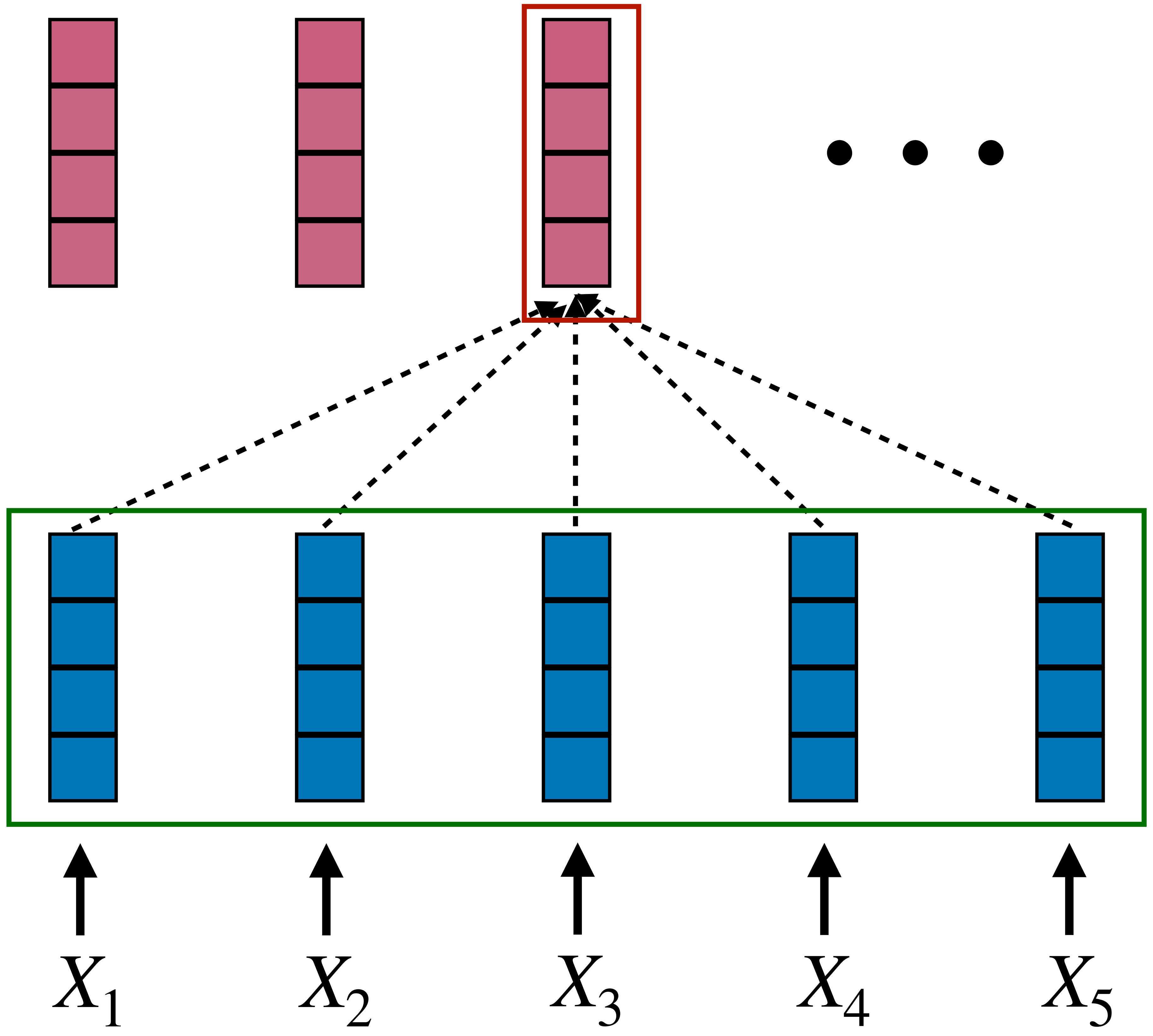
What does multi-head attention learn?



https://github.com/jessevig/bertviz

Self-Attention

- **Self-attention: attention within on single sequence**
  - Contexts and queries are drawn from the same source
- **Contextual information via self-attention**



- **How to apply to auto-regressive case?**

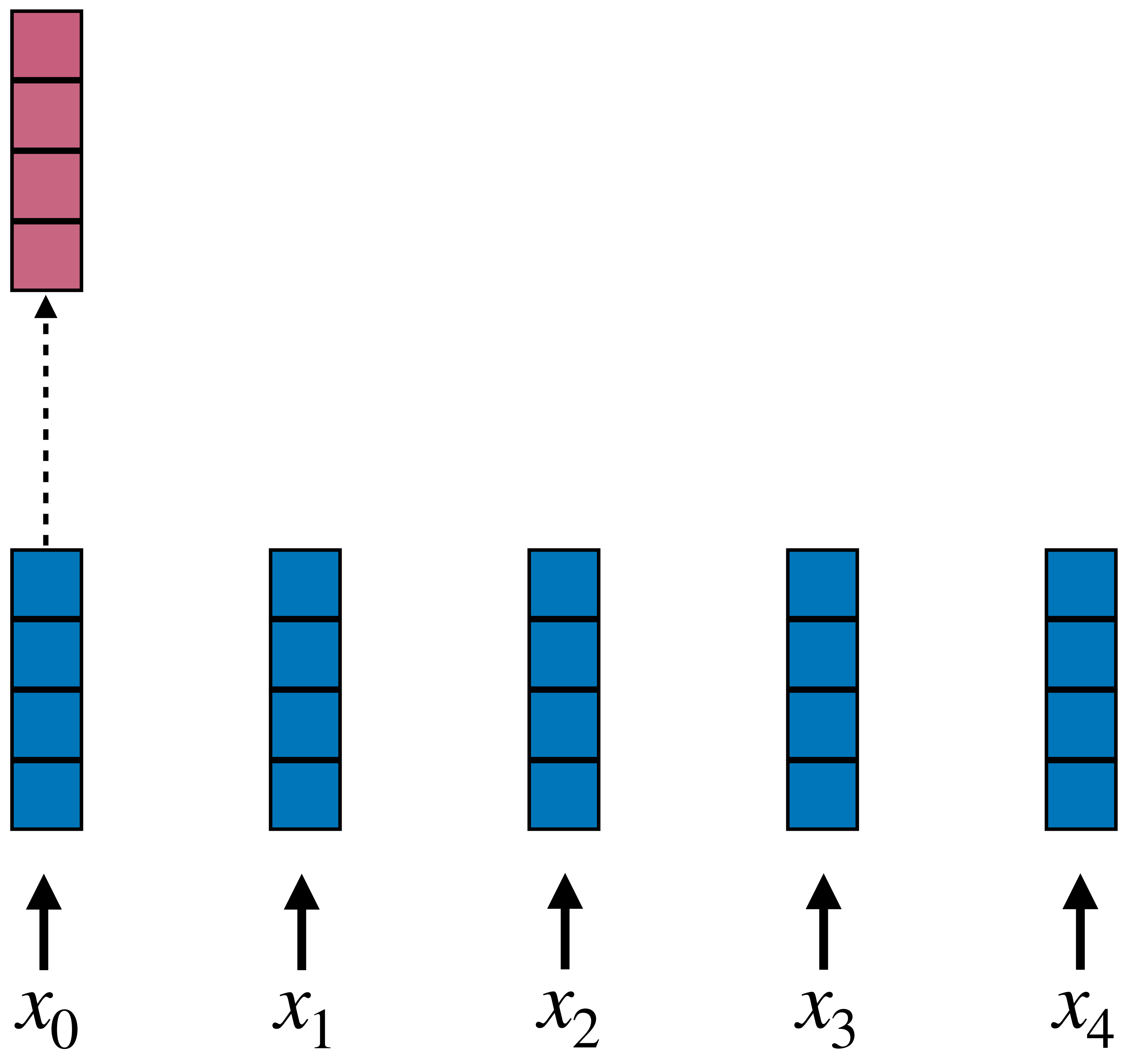
$$p_\theta(Y|X) = \prod_{t=1}^{T} p_\theta(y_t | y_{\boxed{<t}}, X)$$

Next Token

history

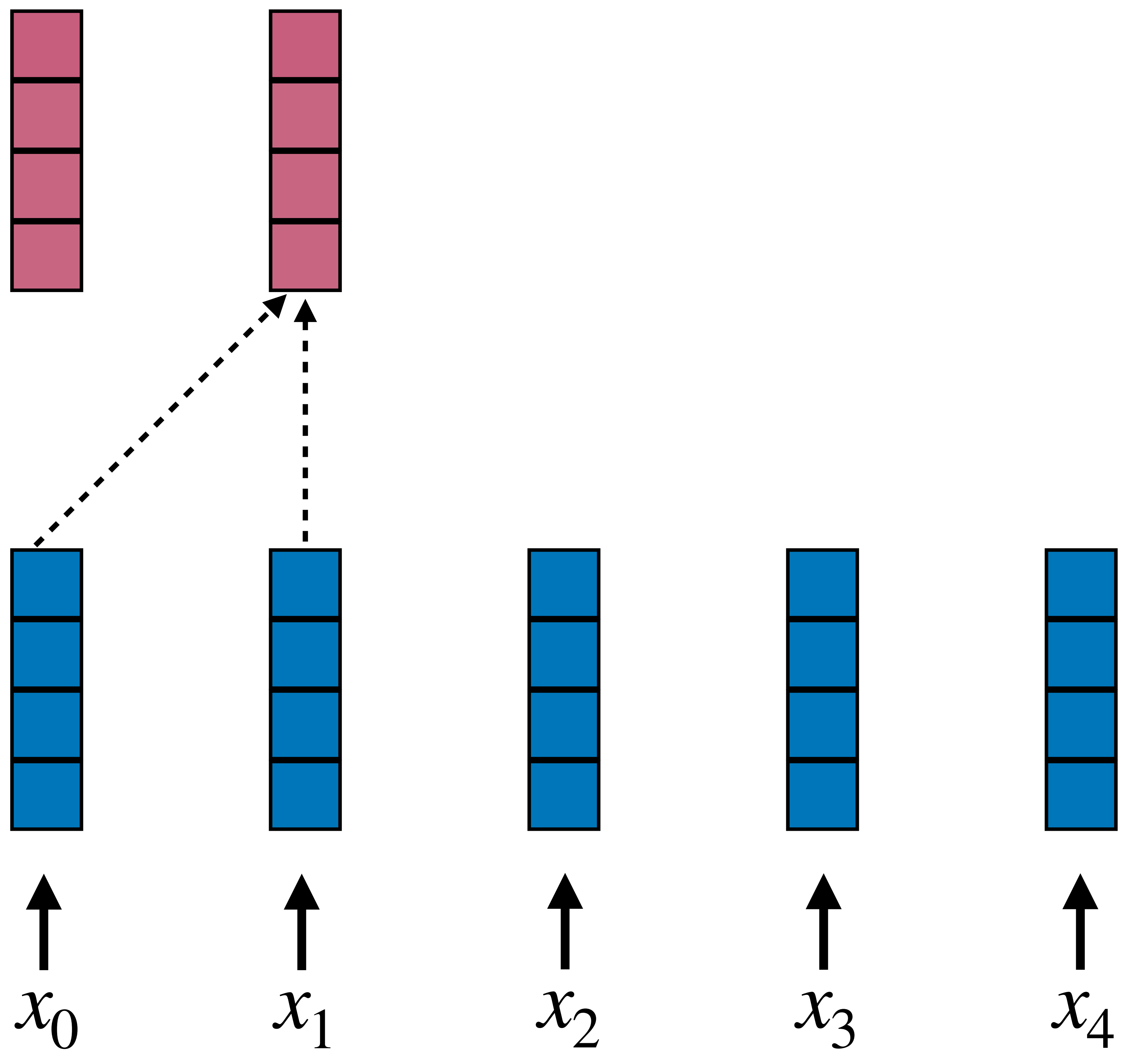
Masked Multi-Head Attention

- **Key point:** we cannot see the future words in decoder
- **Solution:** for every q_i, only attend to $\{(k_j, v_j)\}, j \leq i$



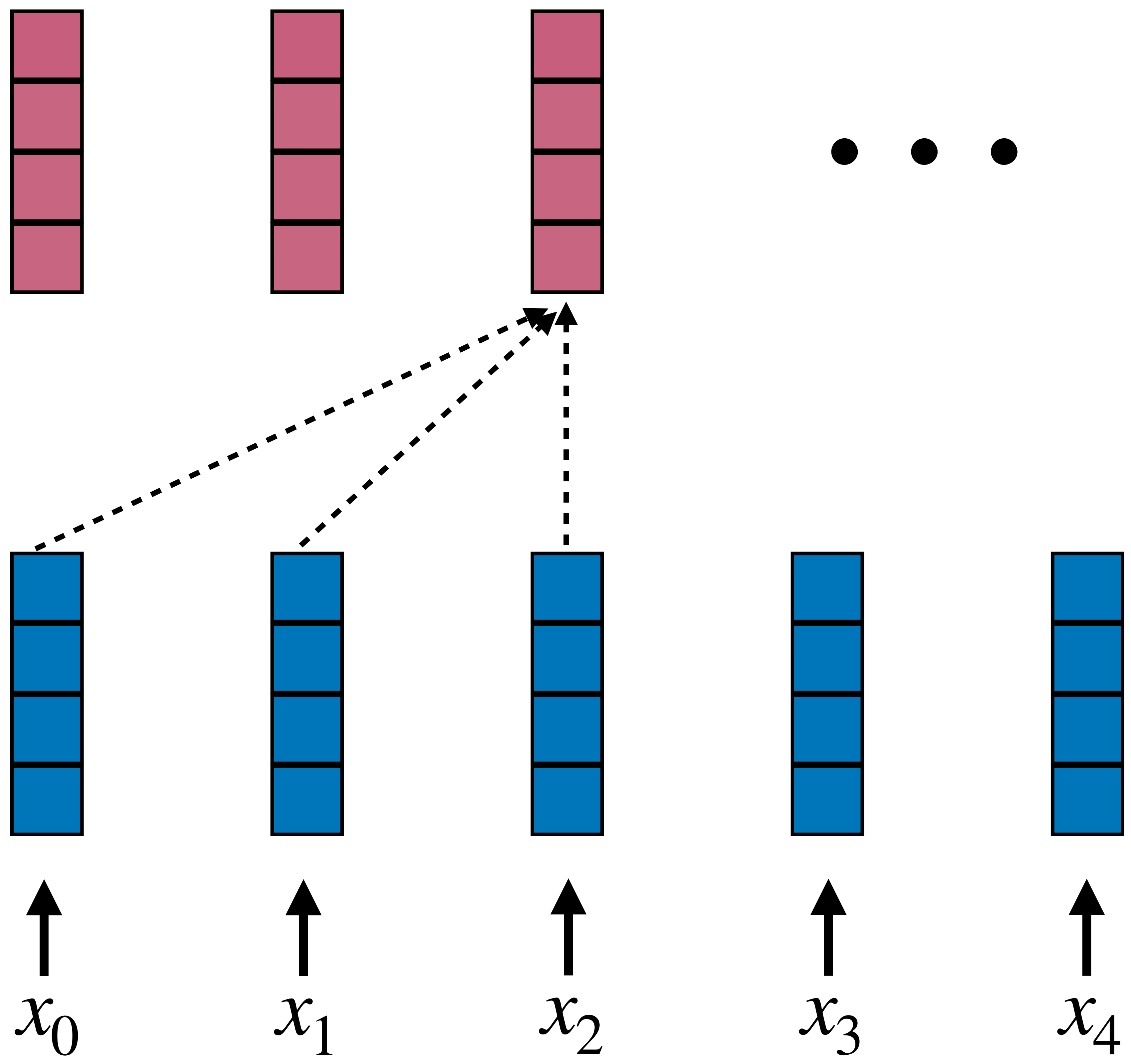
Masked Multi-Head Attention

- **Key point:** we cannot see the future words in decoder
- **Solution:** for every q_i, only attend to $\{(k_j, v_j)\}, j \leq i$



Masked Multi-Head Attention

- **Key point:** we cannot see the future words in decoder
- **Solution:** for every q_i, only attend to $\{(k_j, v_j)\}, j \leq i$

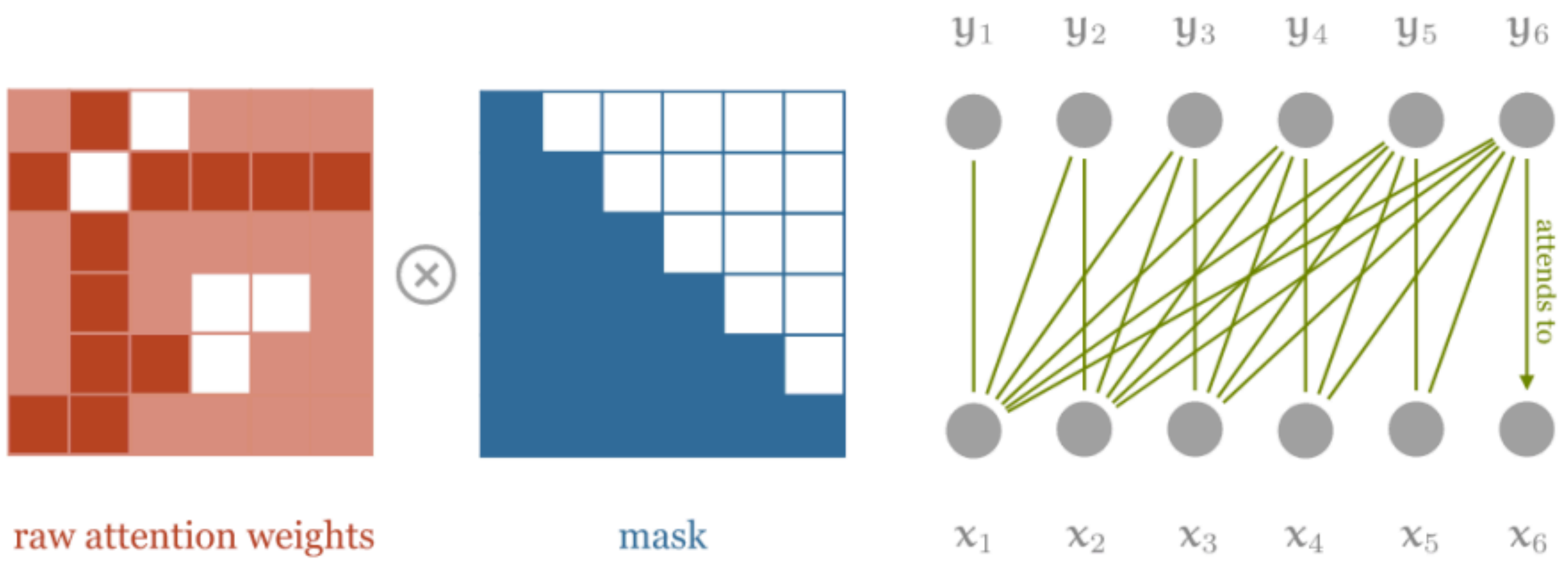


How to vectorize?

Masked Multi-Head Attention

$$q_i = W_Q x_i,\ k_i = W_K x_i,\ v_i = W_V x_i$$

$$a_{i,j} = \mathrm{softmax}\left(\frac{q_i^T k_j}{\sqrt{d}}\right)$$



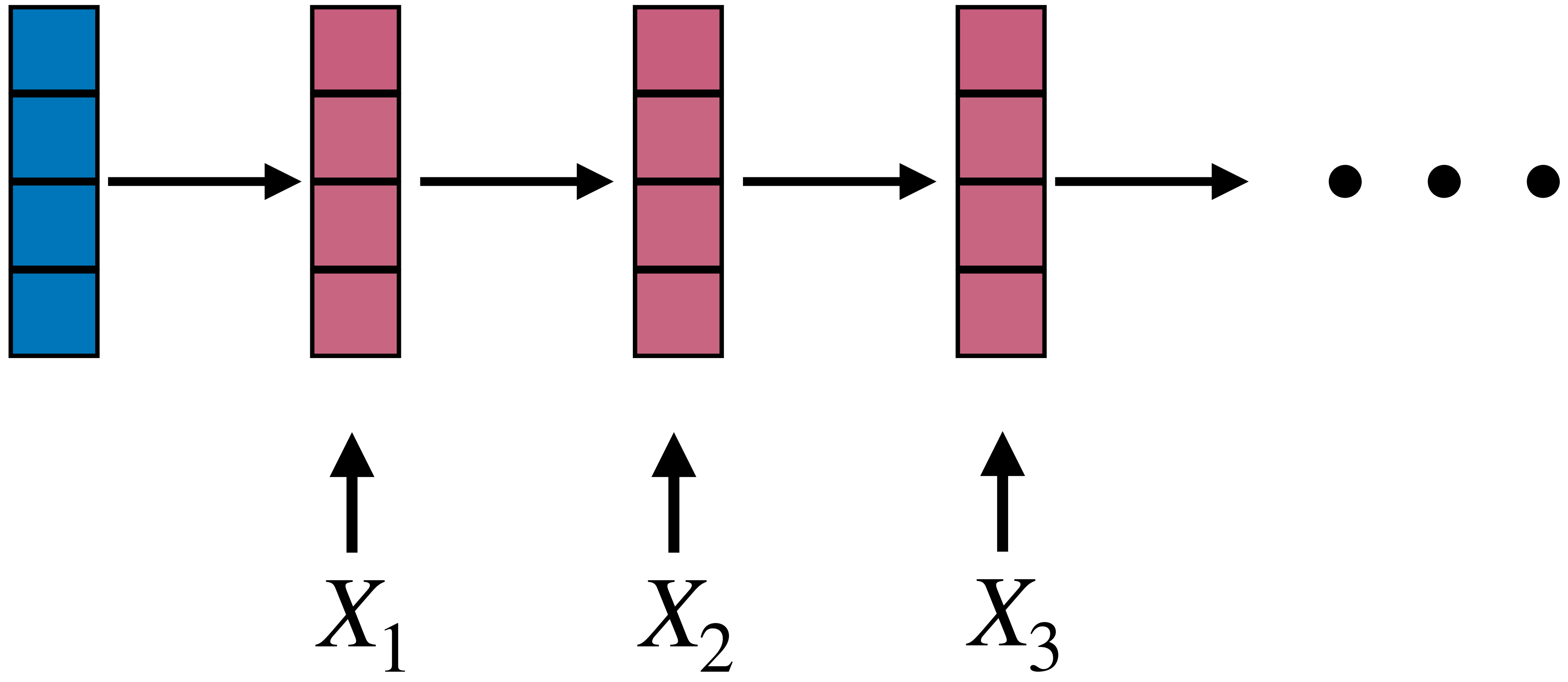
Efficient implementation: compute attention as we normally do, mask out attention to future words by setting attention scores to $-\infty$

```
dot = torch.bmm(queries, keys.transpose(1, 2))

indices = torch.triu_indices(t, t, offset=1)
dot[:, indices[0], indices[1]] = float('-inf')

dot = F.softmax(dot, dim=2)
```

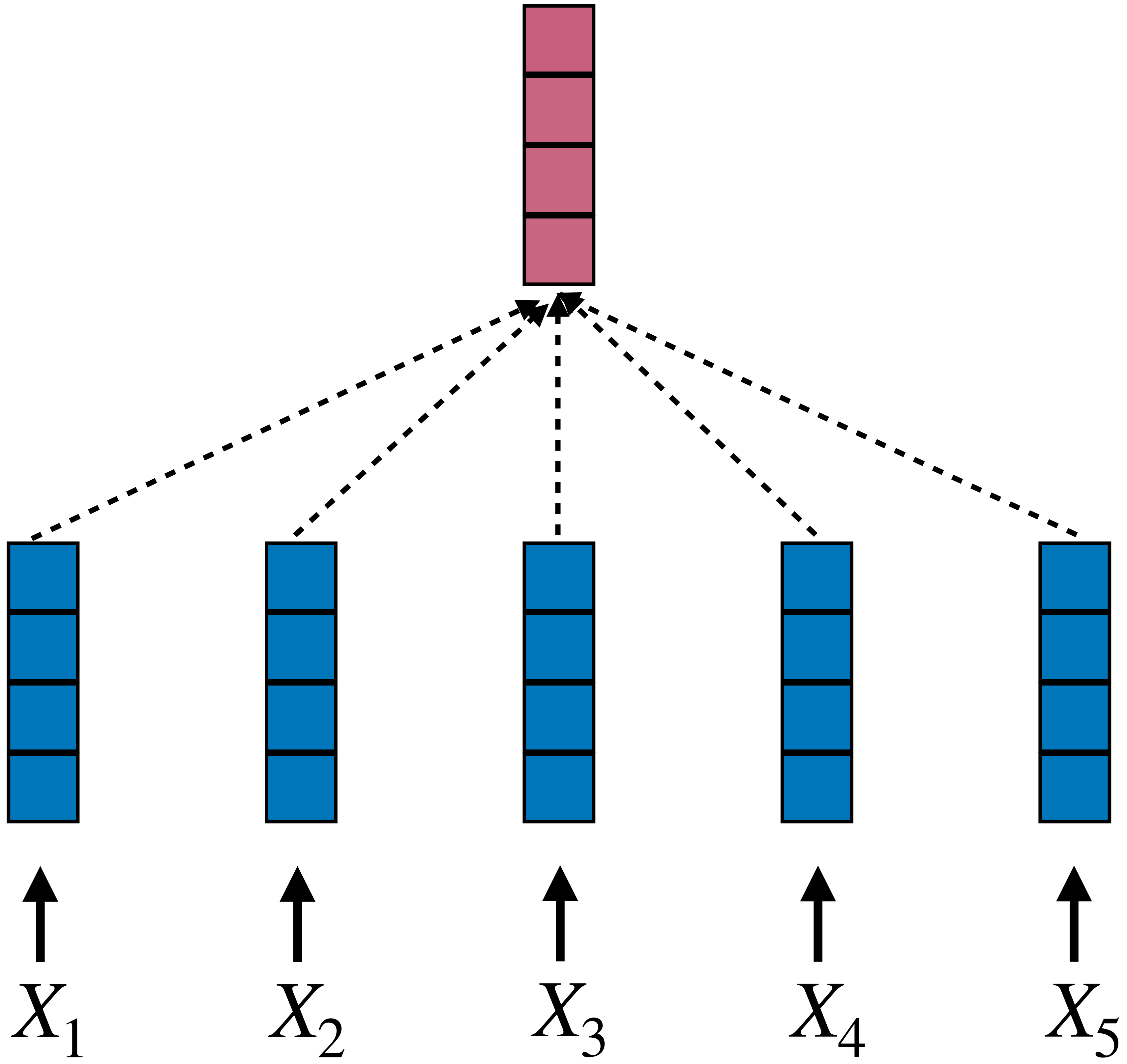
http://peterbloem.nl/blog/transformers

Missing Piece: Positional Information

RNN



Self-attention



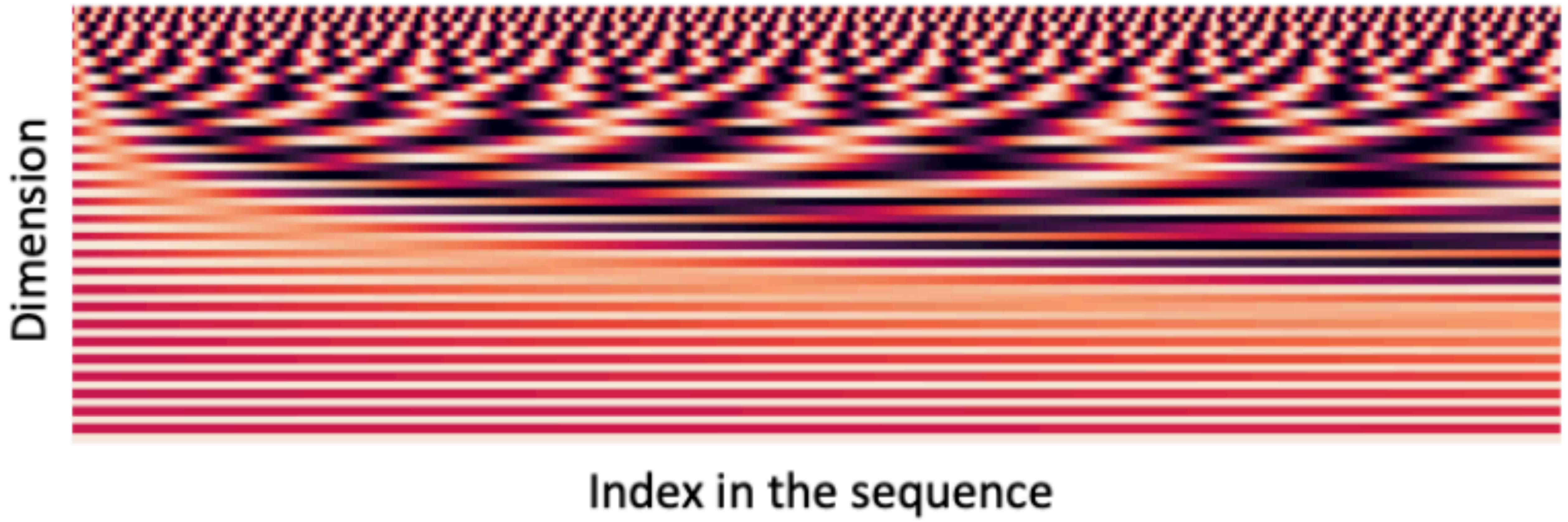
Missing Piece: Positional Information

- **Unlike RNNs, self-attention does not build in order information**
  - Encode the order of the sentence into the input $x_1, \ldots, x_n$
- **Solution: add positional encoding to the input embeddings**

$$x_i \leftarrow x_i + p_i$$

- **Use sine and cosine functions of different frequencies (not learnable)**

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



Transformer: Pros and Cons

- **Easier to capture dependencies:** we draw attention between every pair of words!
- **Easier to parallelize:**

$$\boxed{\begin{gathered}\mathrm{MultiHead}(X) = \mathrm{concat}(h_1, \ldots, h_k)W_O \\ h_i = \mathrm{attn}(Q_i, K_i, V_i) \\ Q_i = (XW_Q)^i, K_i = (XW_K)^i, V_i = (XW_V)^i\end{gathered}}$$

$$\boxed{\mathrm{attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}})V}$$

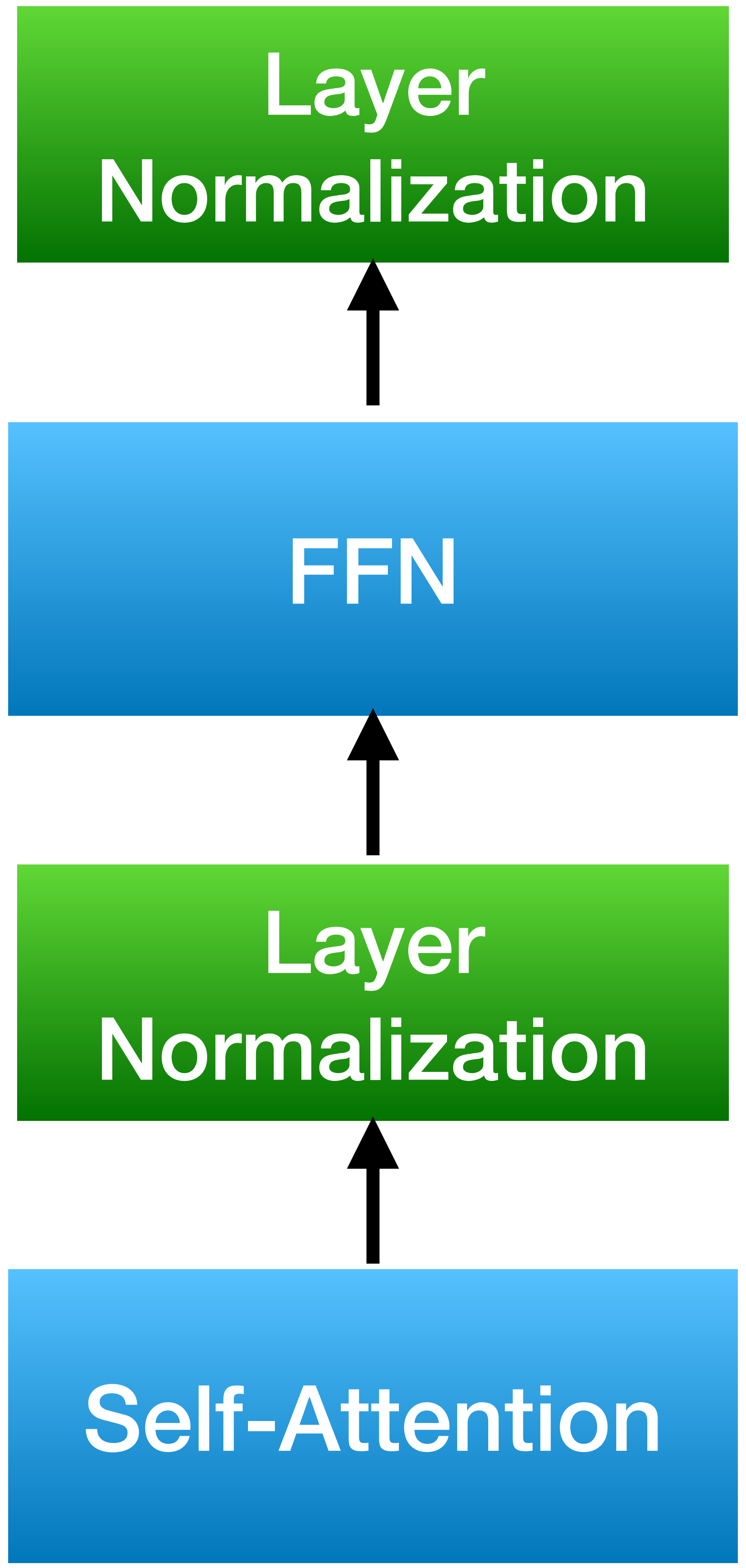
- **Quadratic computation in self-attention:**
  - Can be very expensive when the sequence is very long: $O(hn^2 + nd)$
- **Harder to model positional information**

Transformer vs. RNN

	RNN/LSTM	Transformer
• **Time Complexity**	$O(n)$	$O(n^2)$
• **Memory Complexity**	$O(n)$	$O(n^2)$
• **Training Speed**	Slow	Fast
• **Decoding Speed**	Fast	Slow

Transformers

Transformer Encoder Block



- **Three Key Components**
  - (Masked) Multi-head Self-Attention
  - Layer Normalization
  - Position-wise Feed-Forward Network

Layer Normalization

Layer Normalization

- **Motivation:** normalize each vector individually to control vector scale

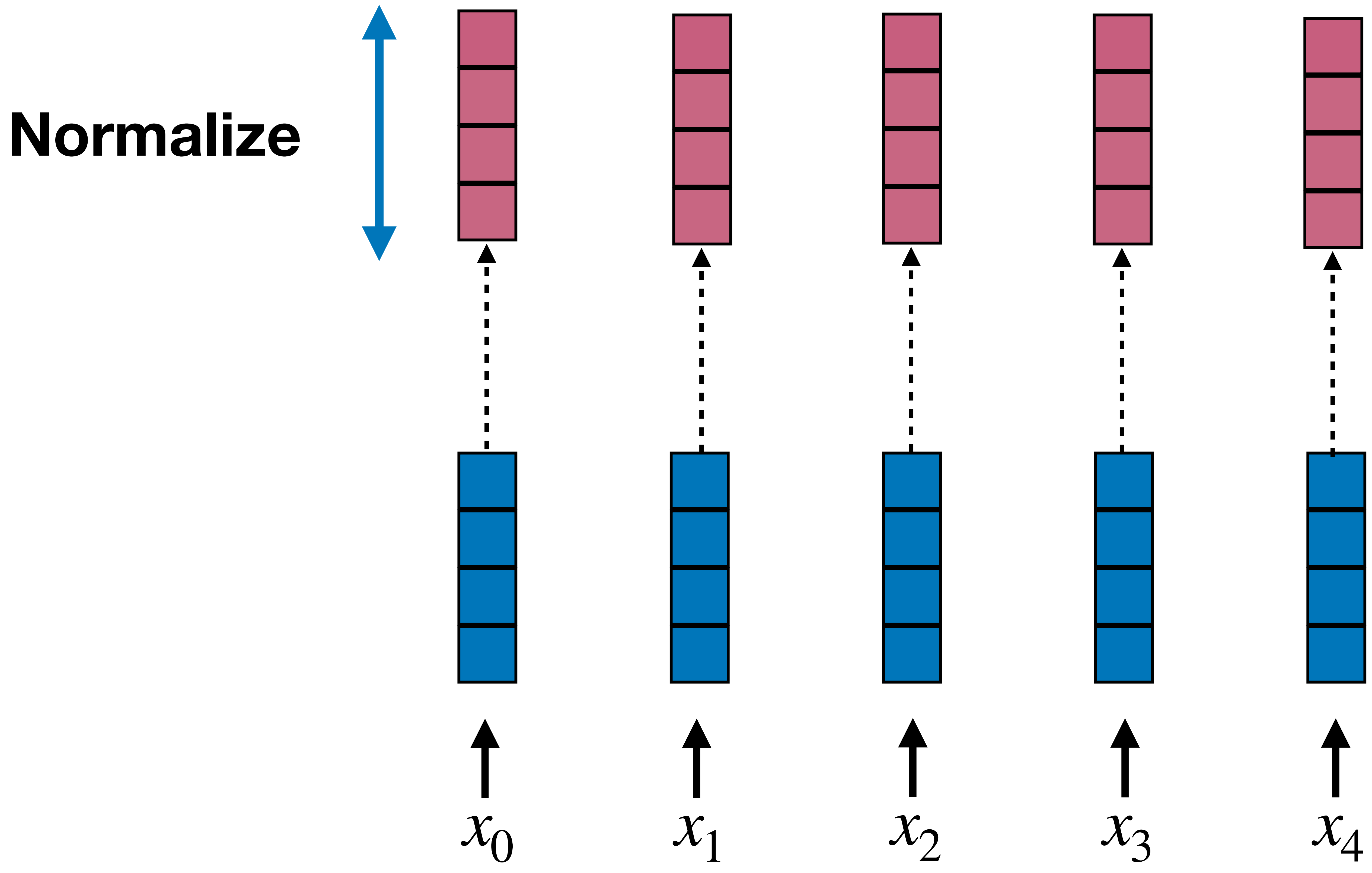
$$Y = \frac{X - \mathrm{E}[X]}{\sqrt{\mathrm{Var}[X] + \epsilon}} * \gamma + \beta$$

Initialization:

$\gamma = 1, \beta = 0$

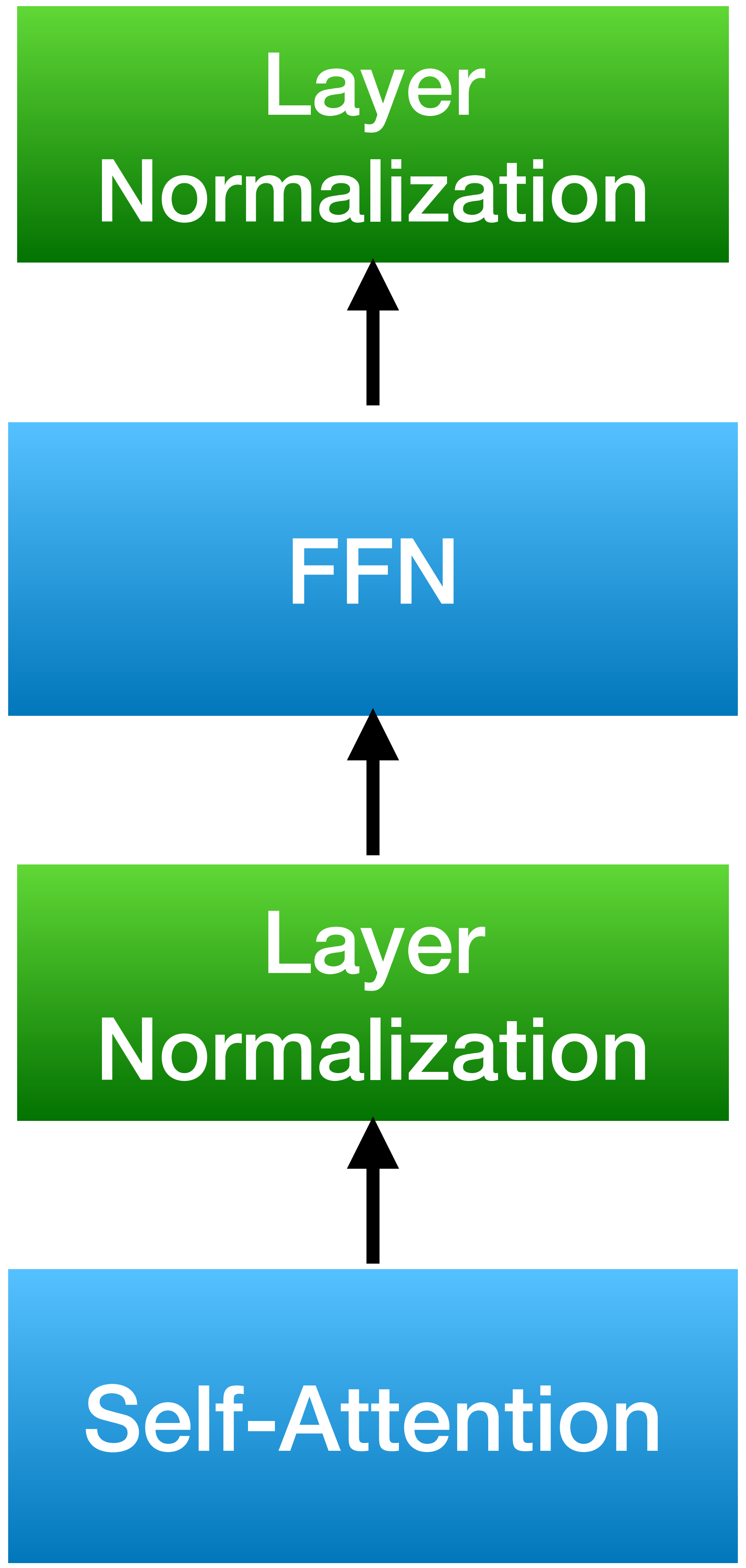
What is the range of Y?

$(-\sqrt{d}, \sqrt{d})$



Transformers

Transformer Encoder Block



- **Three Key Components**
  - (Masked) Multi-head Self-Attention
  - Layer Normalization
  - Position-wise Feed-Forward Network

Position-wise Feed Forward Network

Position-wise Feed Forward Network

- There is no elementwise nonlinearities in self-attention; stacking more self-attention layers just re-averages value vectors

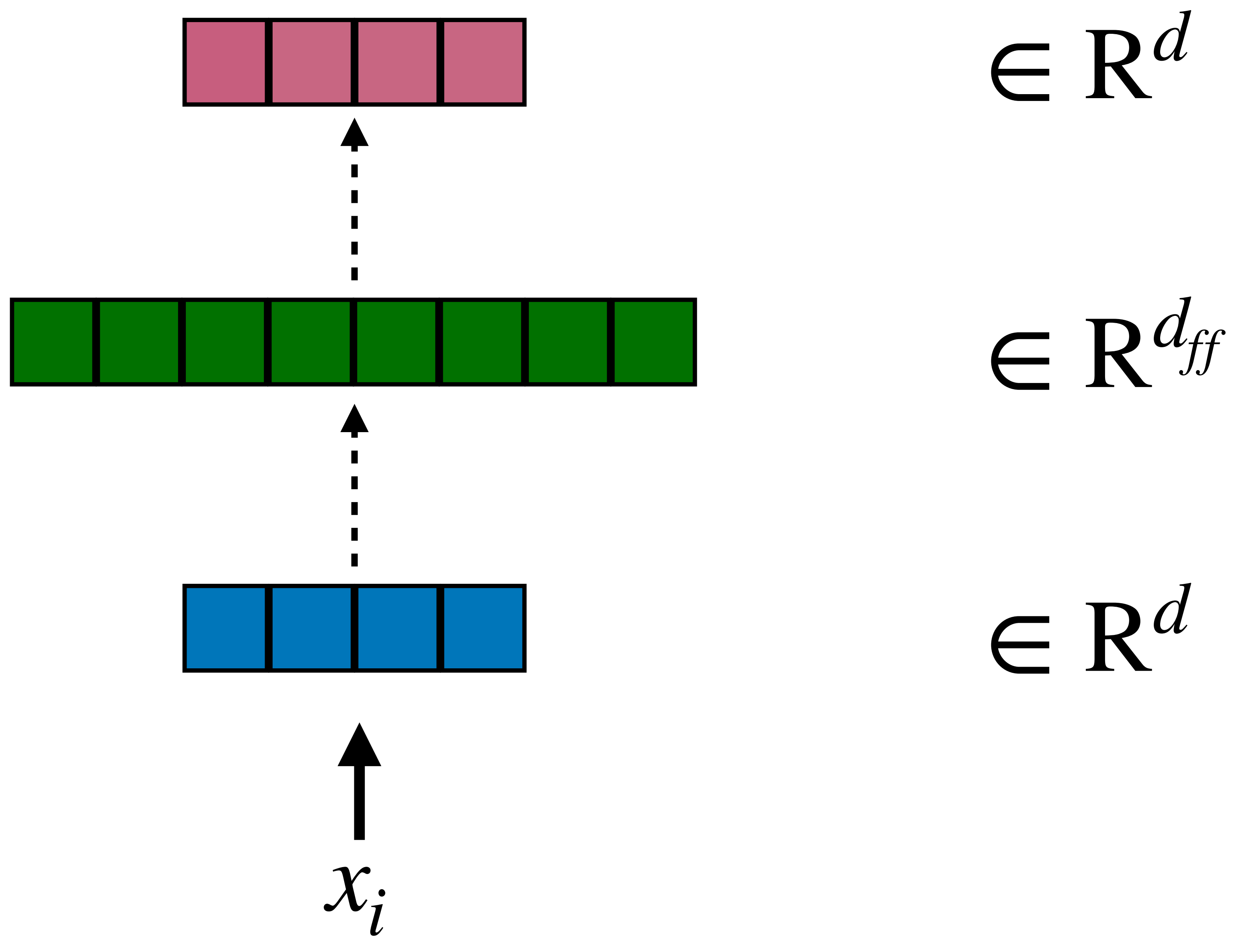
- Simple fix: add a feed-forward network to post-process each output vector

$$\mathrm{FFN}(\mathbf{x}_i) = W_2 \mathrm{ReLU}(W_1 \mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2$$

A large number of parameters

$$\boxed{\begin{array}{l} W_1 \in \mathbb{R}^{d_{ff} \times d}, \mathbf{b}_1 \in \mathbb{R}^{d_{ff}} \\ W_2 \in \mathbb{R}^{d \times d_{ff}}, \mathbf{b}_2 \in \mathbb{R}^{d} \end{array}}$$

In practice, they use $d_{ff} = 4d$



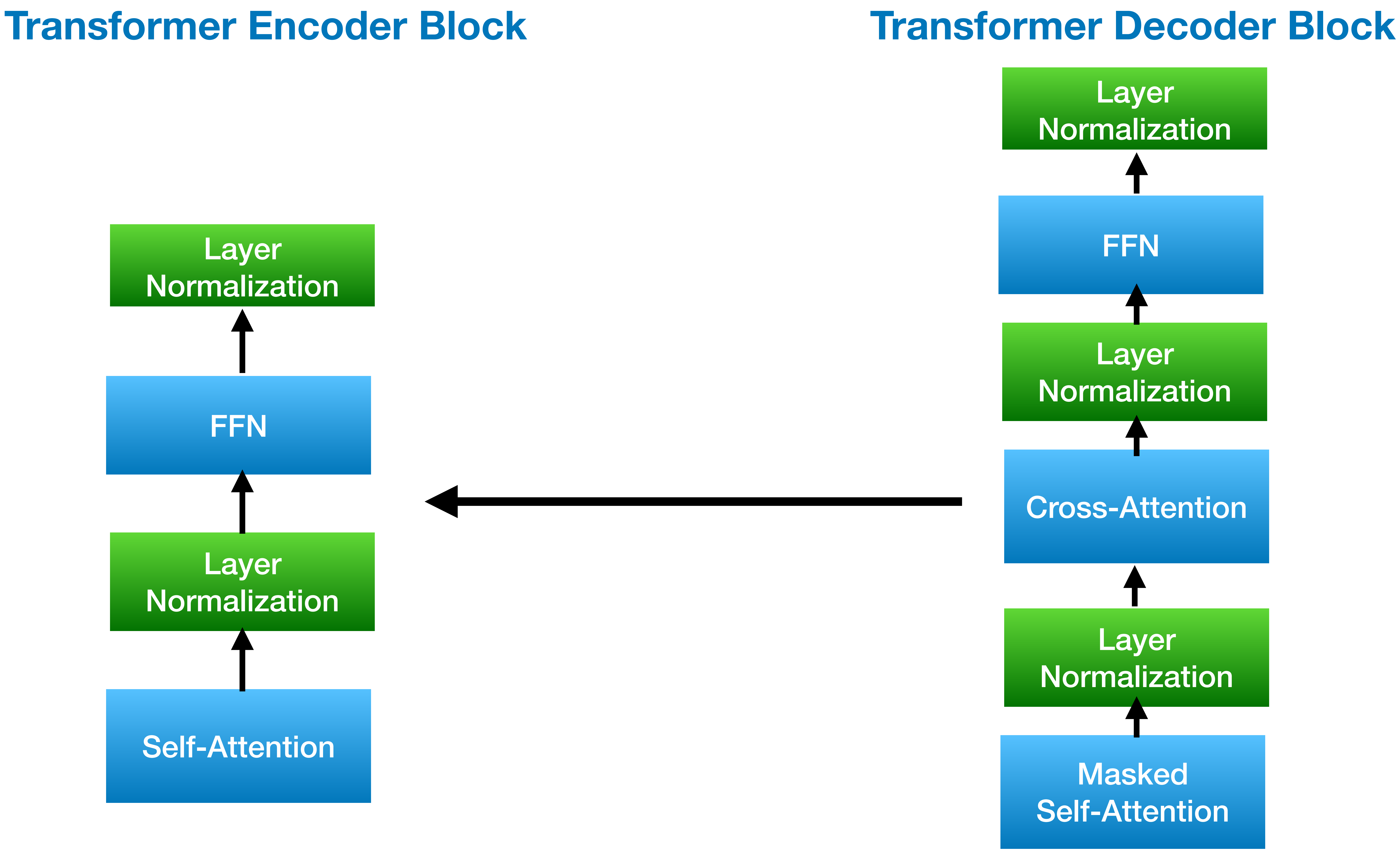
Slide credit: Danqi Chen COS 484/584

Feed-Forward Layers

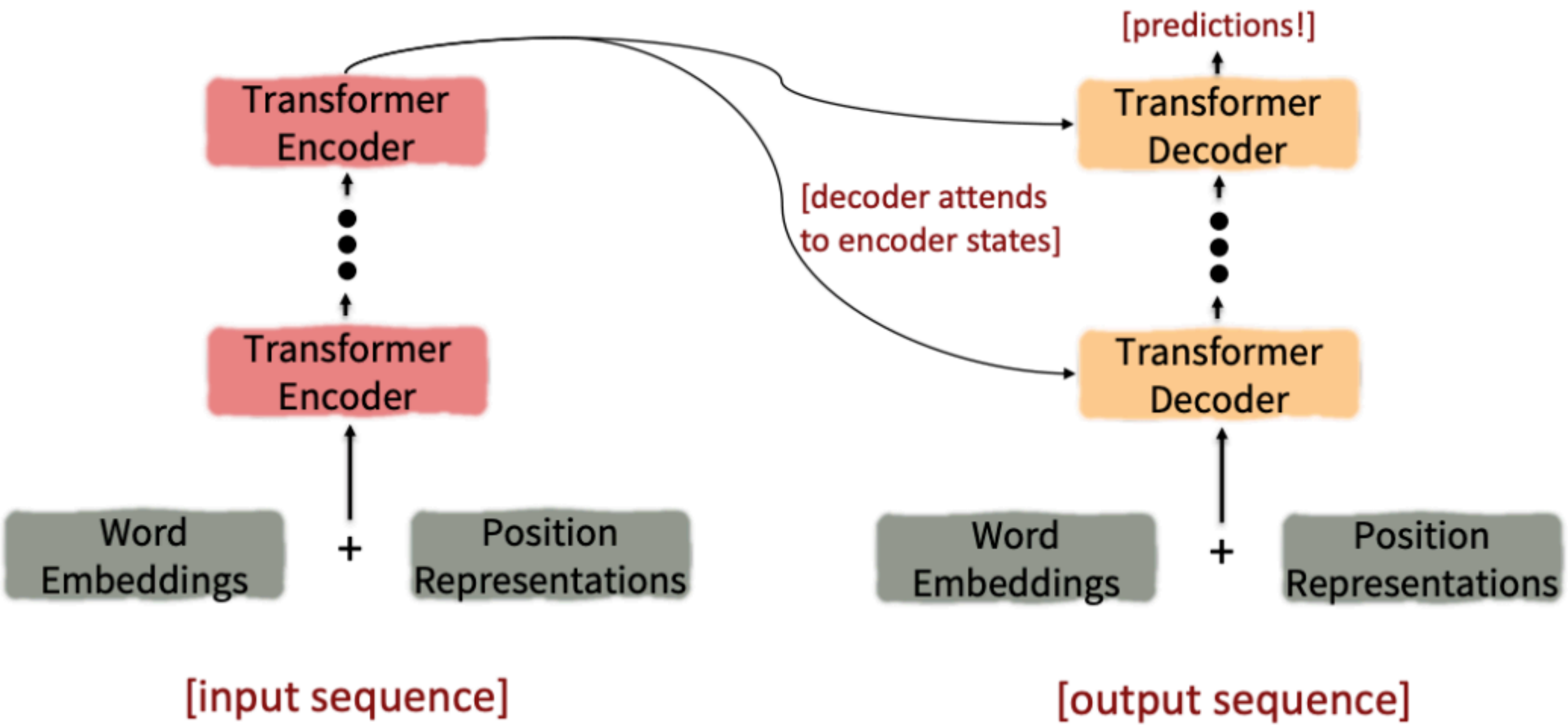
- **Feed-forward layers constitute two-thirds of parameters**
- **Operates as memories of textual patterns** (Gova et al., 2021)

Key	Pattern	Example trigger prefixes
$\mathbf{k}^{1}_{449}$	Ends with "*substitutes*" (shallow)	*At the meeting, Elton said that "for artistic reasons there could be no substitutes* *In German service, they were used as substitutes* *Two weeks later, he came off the substitutes*
$\mathbf{k}^{6}_{2546}$	Military, ends with "*base*"/"*bases*" (shallow + semantic)	*On 1 April the SRSG authorised the SADF to leave their bases* *Aircraft from all four carriers attacked the Australian base* *Bombers flying missions to Rabaul and other Japanese bases*
$\mathbf{k}^{10}_{2997}$	a "part of" relation (semantic)	*In June 2012 she was named as one of the team that competed* *He was also a part of the Indian delegation* *Toy Story is also among the top ten in the BFI list of the 50 films you should*
$\mathbf{k}^{13}_{2989}$	Ends with a time range (semantic)	*Worldwide, most tornadoes occur in the late afternoon, between 3 pm and 7* *Weekend tolls are in effect from 7:00 pm Friday until* *The building is open to the public seven days a week, from 11:00 am to*
$\mathbf{k}^{16}_{1935}$	TV shows (semantic)	*Time shifting viewing added 57 percent to the episode's* *The first season set that the episode was included in was as part of the* *From the original NBC daytime version , archived*

Transformers



Putting the pieces together



Slide credit: Danqi Chen COS 484/584

Transformer: Machine Translation

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Vaswani et al., 2017: Attention Is All You Need

Transformer: Document Generation

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

Significant gains compared to seq2seq-attention with LSTMs

Liu et al., 2018: Generating Wikipedia by Summarizing Long Sequences