Attention in Sequence Model

Jun Chen

Sep 8, 2018

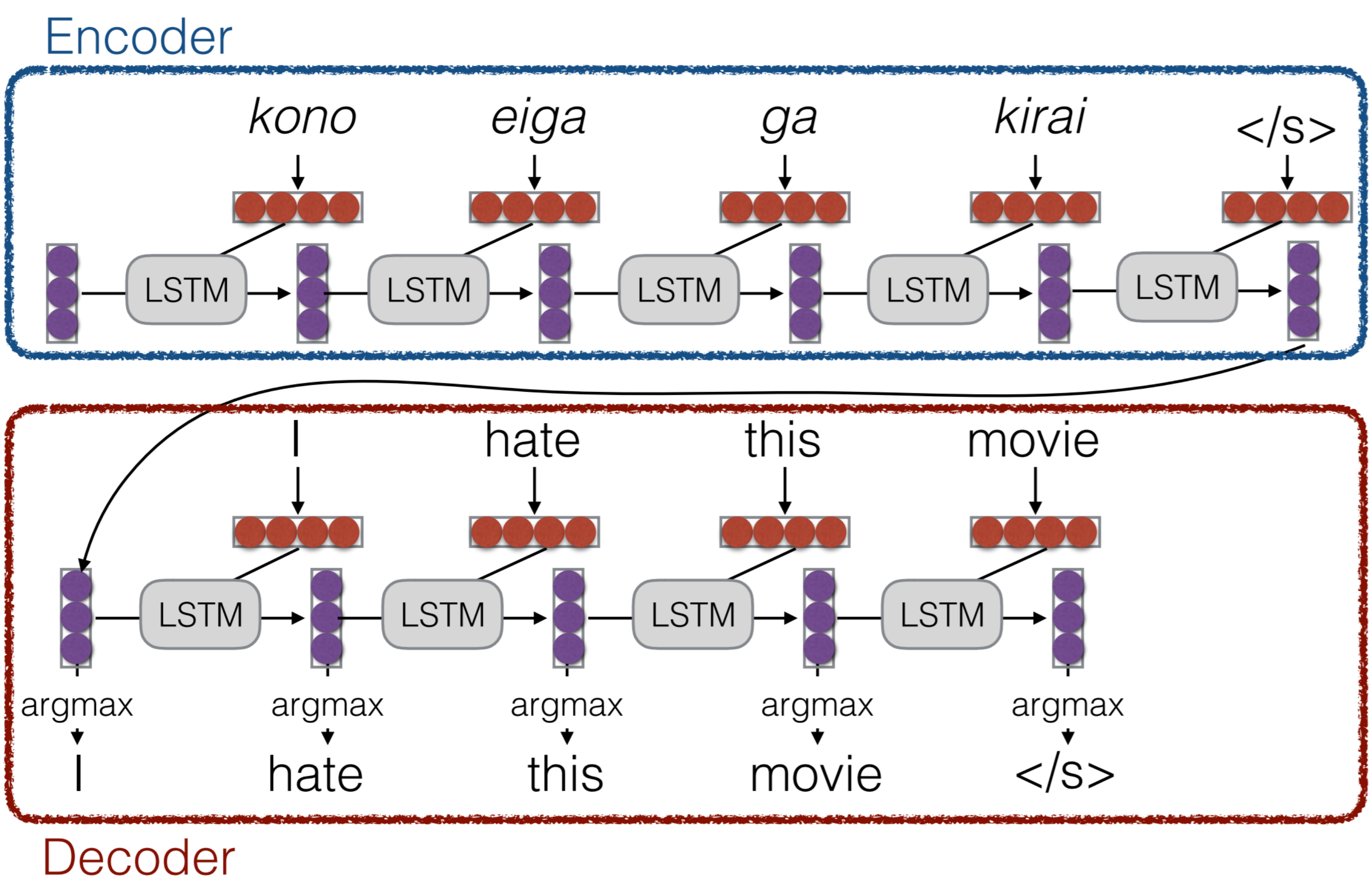
Some slides are modified from slides of CMU CS11-747 Neural Networks for NLP

Outline

- Motivation
- Basic idea
- Variants of attention
  - Addictive attention
  - Multiplicative attention
  - Self-attention
  - Key-value attention
- Case study: Transformer

Motivation

- Encoder-decoder model



Motivation

- Limited representation
- Long distance constrained

Motivation

Hacks:

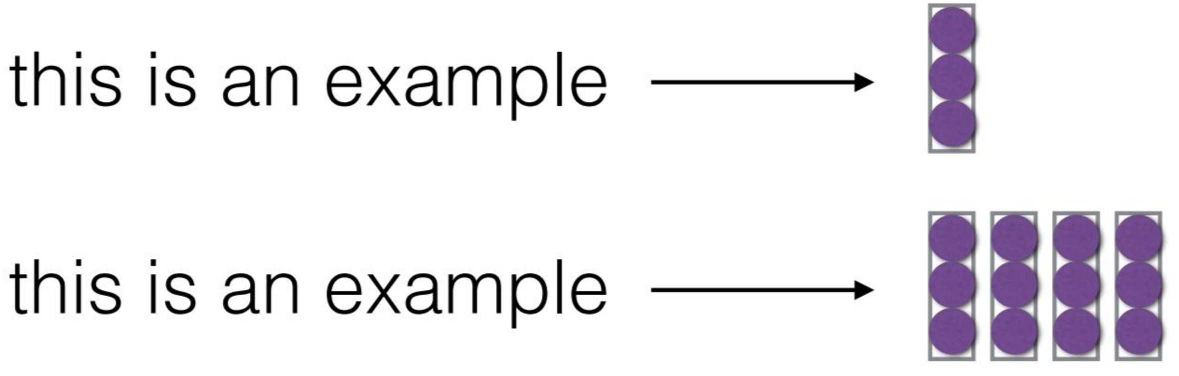
- Reverse the order (Sutskever et al. NIPS’ 14)

- Input twice (Zaremba et al. Arxiv’14)

Make things work better in practice,
but not a principled solution

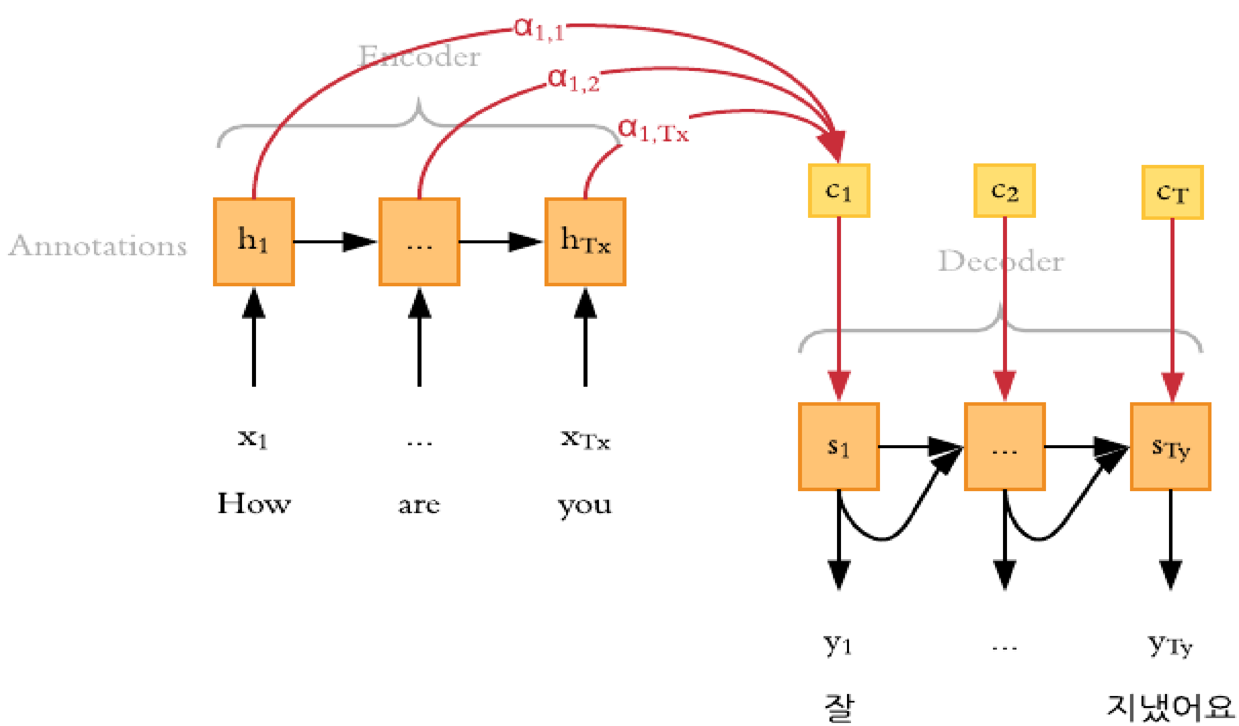
Motivation

- What if we could use multiple vectors, based on the length of the sentence.



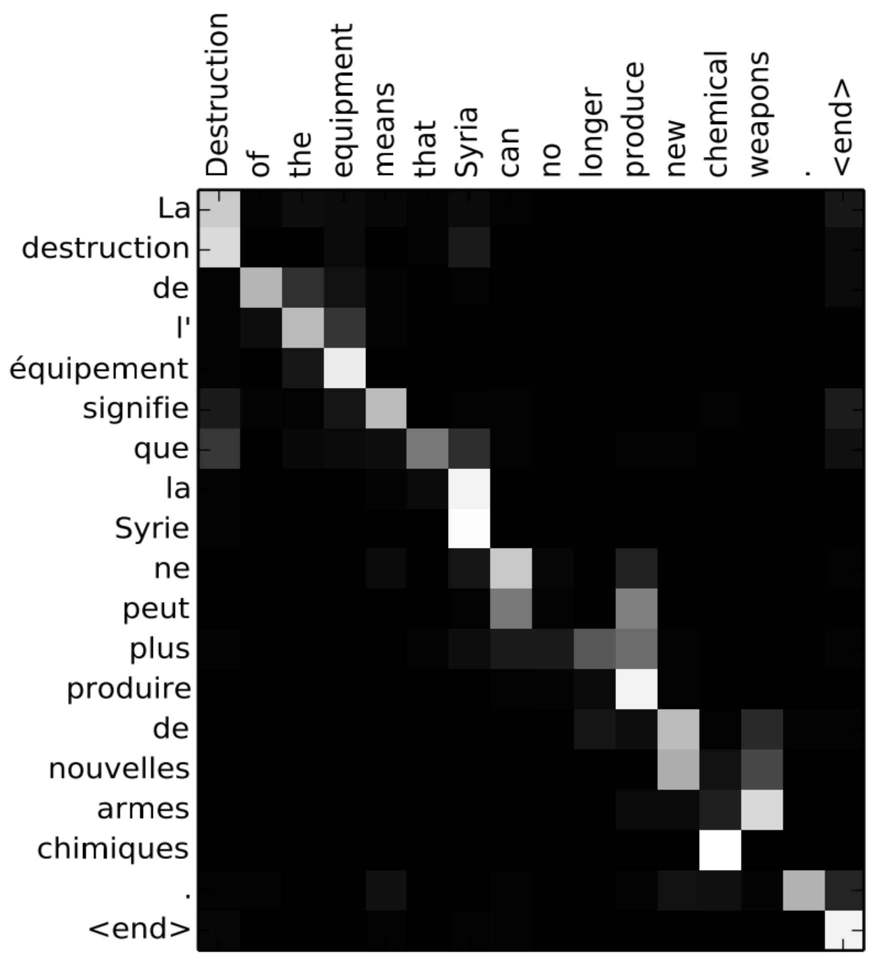
Basic Idea
(Bahdanau et al. 2015)

- Encode each word in the sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination in picking the next word



Basic Idea

- A graphic example：



Attention model

- Using attention, we obtain a context vector c_i based on hidden states $\boldsymbol{s}_i,\ldots,\boldsymbol{s}_m$ that can be used together with the current hidden state h_i for prediction. The context vector $\mathbf{c}_i$ at position is calculated as an average of the previous states weighted with the attention scores $\mathbf{a}_i$:

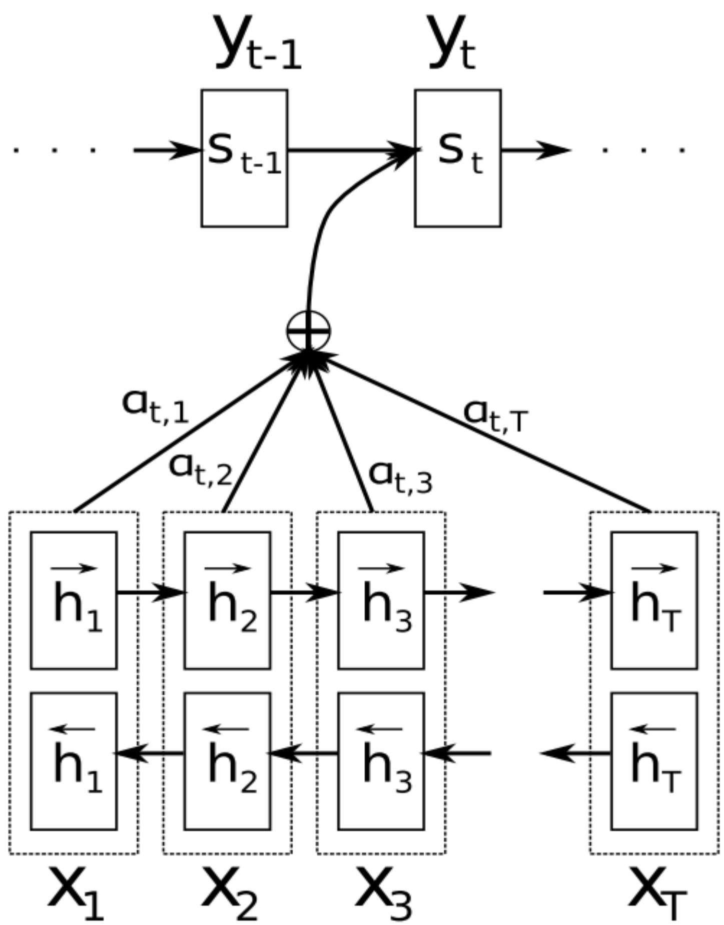
$$\mathbf{c}_i = \sum_j a_{ij}\mathbf{s}_j$$

$$\mathbf{a}_i = \mathrm{softmax}(f_{att}(\mathbf{h}_i, \mathbf{s}_j))$$

- The attention function $f_{att}(\mathbf{h}_i, \mathbf{s}_j)$ calculates an unnormalized alignment score between the current hidden state $\mathbf{h}_i$ and the previous hidden state $\mathbf{s}_j$

Addictive attention

- Bahdanau et al., 2015



Addictive attention

- Use a one-hidden layer feed-forward network to calculate the attention alignment:

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^{\mathrm{T}} \tanh(\mathbf{W}_a[\mathbf{h}_i; \mathbf{s}_j])$$

- where v_a and $\mathbf{W}_a$ are learned attention parameters. Analogously, we can also use matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ to learn separate transformations for $\mathbf{h}_i$ and $\mathbf{s}_j$ respectively, which are then summed:

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^{\mathrm{T}} \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{s}_j)$$

Multiplicative attention

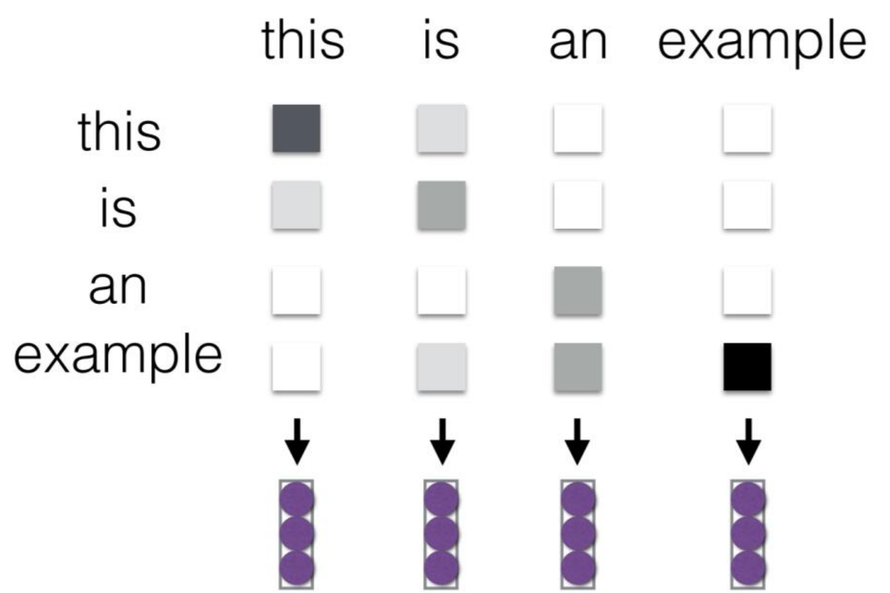
- Simplify the attention operation(Luong et al., 2015):

$$f_{att}(h_i, s_j) = {h_i}^{\mathrm{T}} \mathbf{W}_a s_j$$

- similar in complexity to addictive model
- faster and more space-efficient in practice (can be implemented more efficiently using matrix multiplication)
- scale of dot product increases as dimensions get larger (can be fixed by scaling by size of the vector $1/\sqrt{d_h}$)

Self attention

- Without any additional information, we can still extract relevant aspects from the sentence by allowing it to attend to itself using self-attention (Lin et al., 2017)
- Each element in the sentence attends to other elements → context sensitive encodings



Self-attention

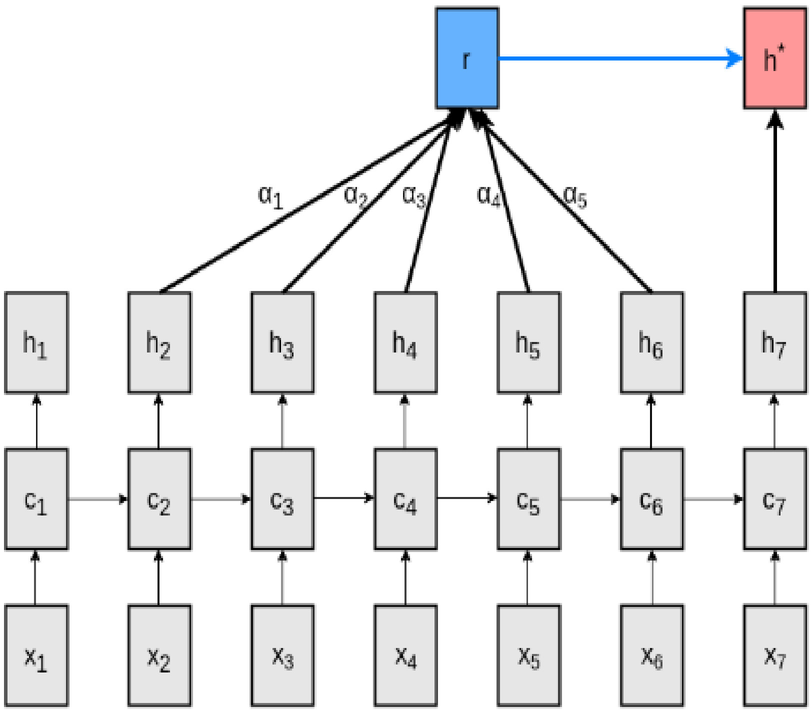
- Simplify additive attention to compute the unnormalized alignment score for each hidden state h_i:

$$f_{att}(h_i) = {v_a}^T \tanh(\mathbf{W}_a h_i)$$

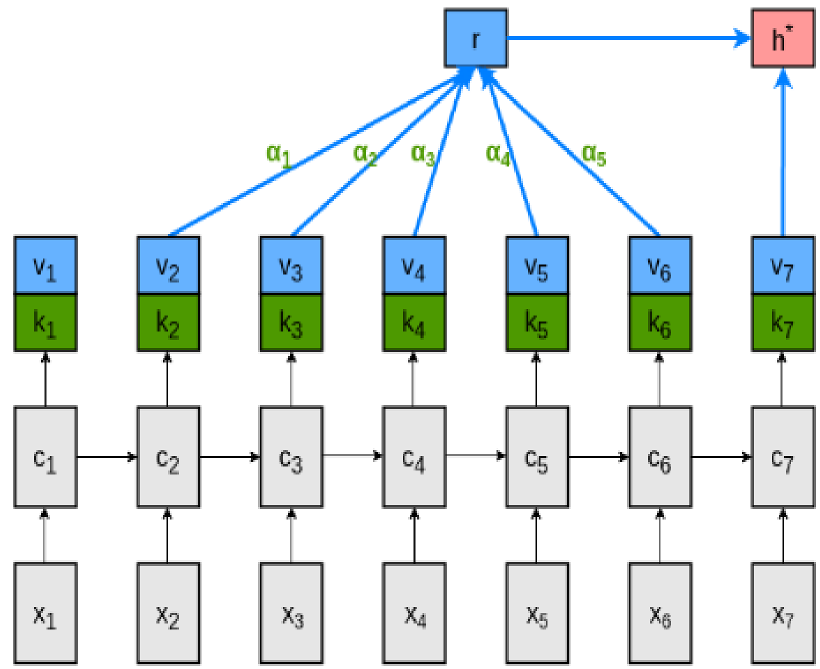
- In matrix form, for hidden states $\mathbf{H} = \mathrm{h}_1, \ldots, \mathrm{h}_\mathrm{n}$ we can calculate the attention vector a and the final sentence representation c as follows:

$$\mathbf{a} = \mathrm{softmax}\big(\mathbf{v}_a \tanh\big(\mathbf{W}_a \mathbf{H}^\mathrm{T}\big)\big)$$
$$\mathbf{c} = \mathbf{H}\boldsymbol{a}^\mathrm{T}$$

Key-value attention
(Daniluk et al., 2017)



(a) Neural language model with attention.



(b) Key-value separation.

Key-value Attention

- Split each hidden vector h_i into a key k_i and a value v_i: $[k_i; v_i] = h_i$

- Keys to calculate the attention distribution a_i using additive attention:
$$a_i = \mathrm{softmax}(\mathbf{v}_a{}^{\mathrm{T}} \tanh(\mathbf{W}_1[\mathbf{k}_{i-L}; \ldots; \mathbf{k}_{i-1}] + (\mathbf{W}_2 \mathbf{s}_j)\mathbf{1}^{\mathrm{T}}))$$

- where L is the length of the attention window and $\mathbf{1}$ is a vector of ones. The values are then used to obtain the context representation $\mathbf{c}_i$:
$$\mathbf{c}_i = [\mathbf{v}_{i-L}; \ldots; \mathbf{v}_{i-1}]\boldsymbol{a}^{\mathrm{T}}$$

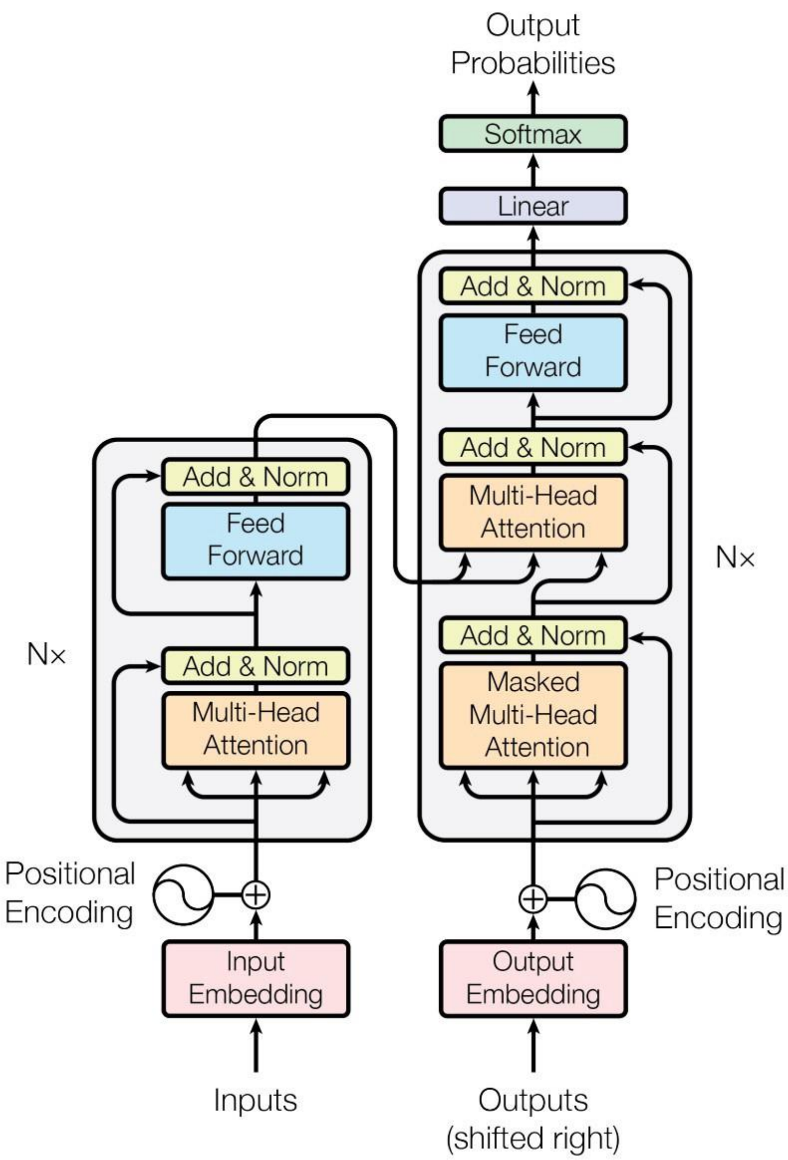
- The context $\mathbf{c}_i$ is used together with the current value $\mathbf{v}_i$ for prediction.

Transformer
(Vaswani et al. 2017)

Summary:

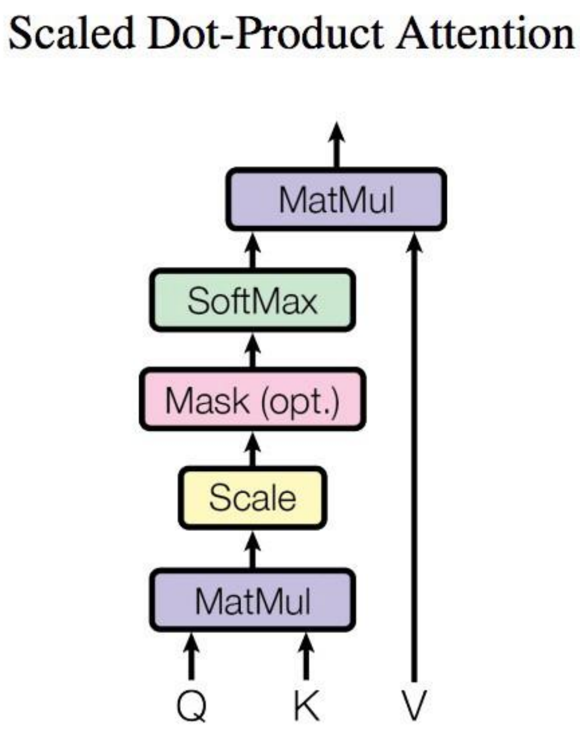
Attention is all you need

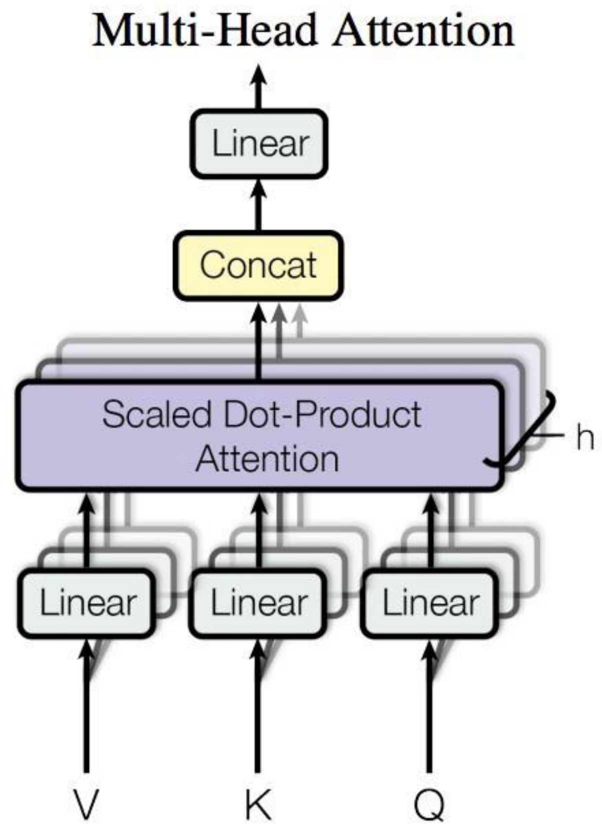
- A sequence-to-sequence model based entirely on attention
- Strong results on standard WMT datasets
- Fast: only matrix multiplications



Transformer
(Vaswani et al. 2017)

- Attention is all you need





Idea: multiple attention “heads” focus on different parts of the sentence

Attention Tricks

- **Self Attention**: Each layer combines words with Others
- **Multi-headed Attention**: 8 attention heads learned Independently
- **Normalized Dot-product Attention**: Remove bias in dot product when using large networks
- **Positional Encodings**: Make sure that even if we don’t have RNN, can still distinguish positions