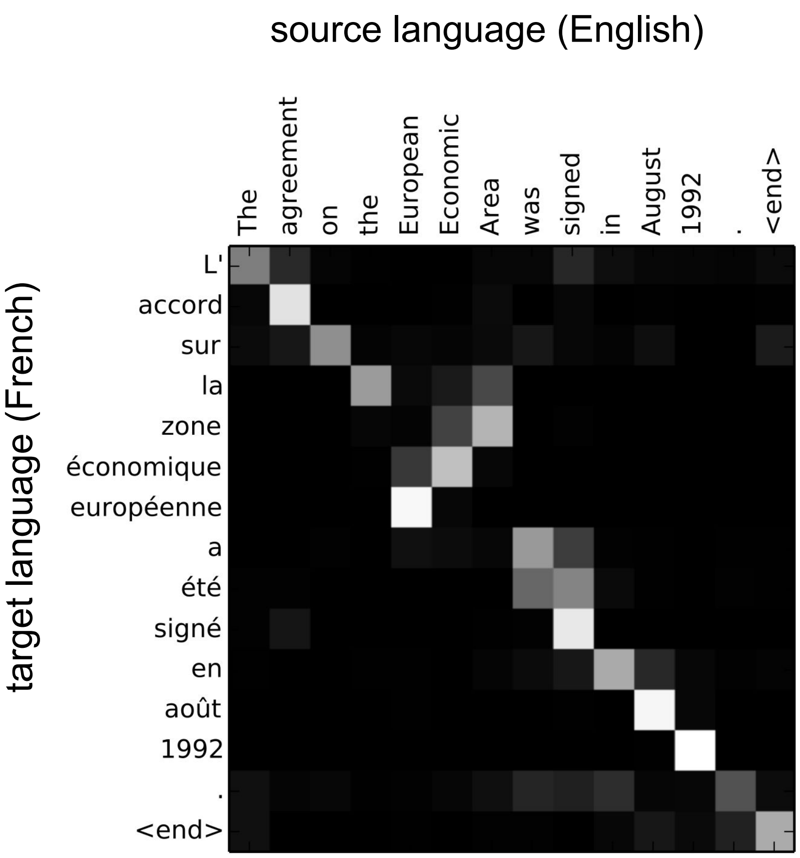
Lecture 04: Attention mechanism

Radoslav Neychev

Words alignment



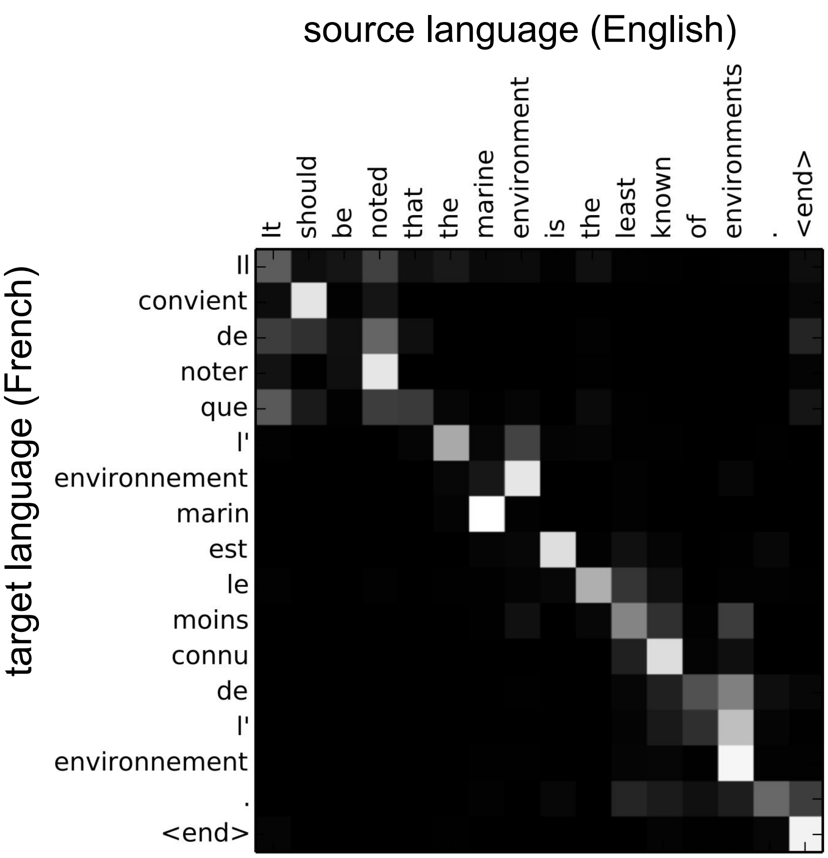


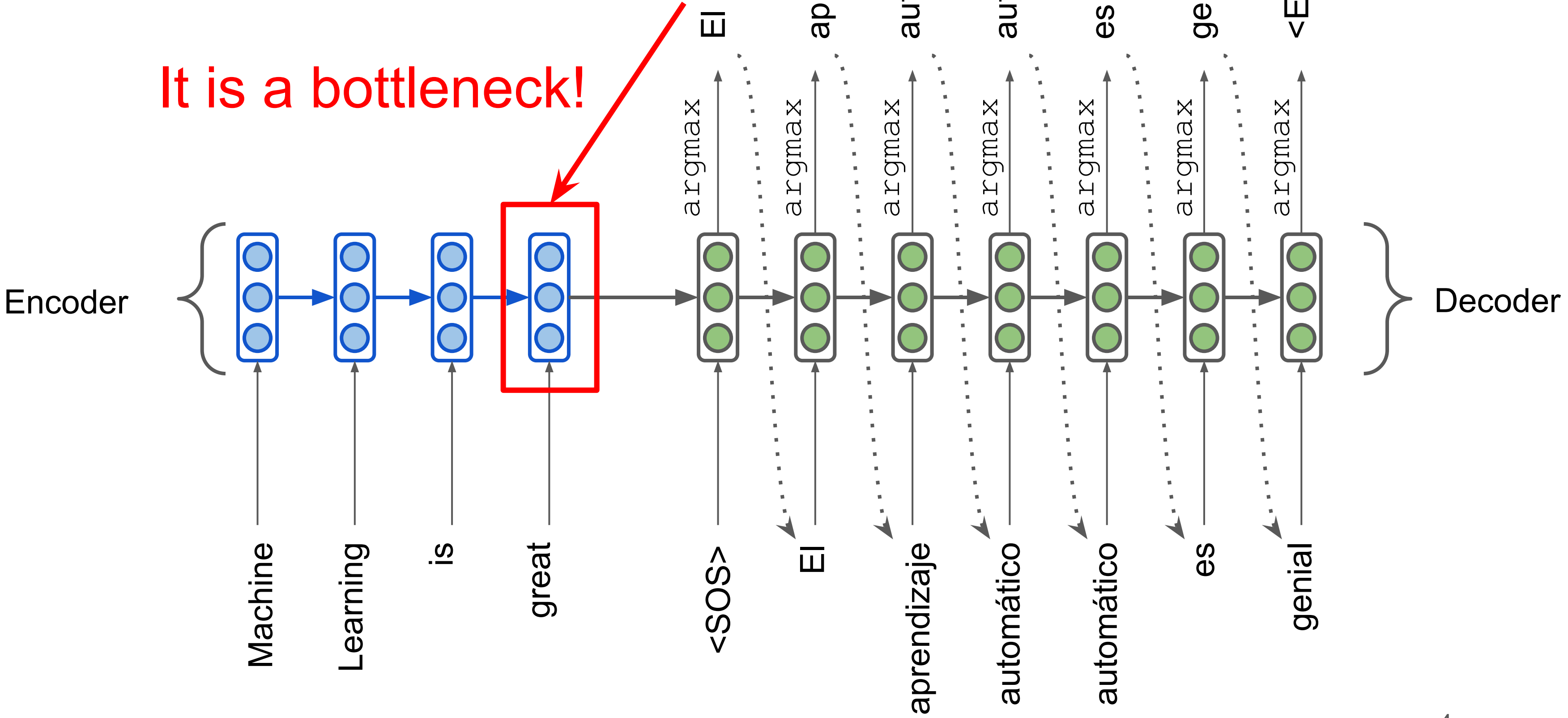
Image source: [Neural Machine Translation by Jointly Learning to Align and Translate]

Attention

Seq2seq NMT

This state encodes the whole sentence

It is a bottleneck!



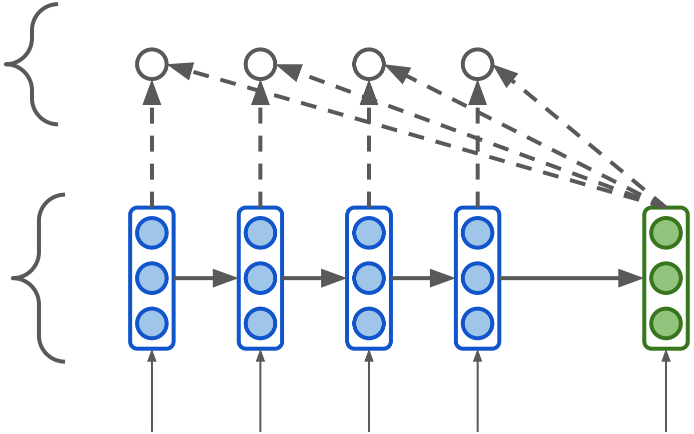
Main idea:

on each step of the **decoder**, use **direct connection to the encoder** to focus on a particular part of the source sequence

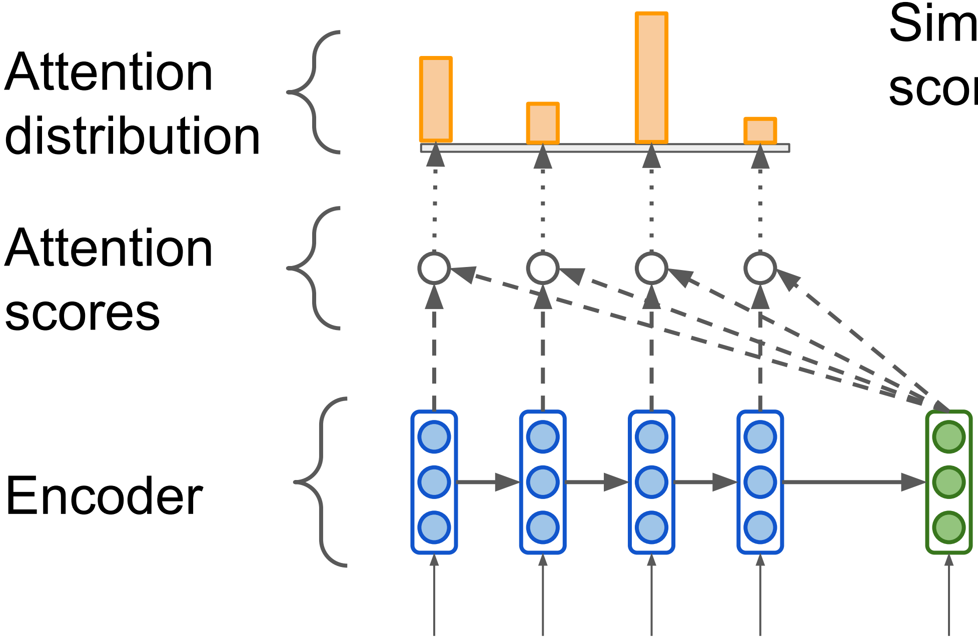
Seq2seq with attention

Attention scores

Encoder

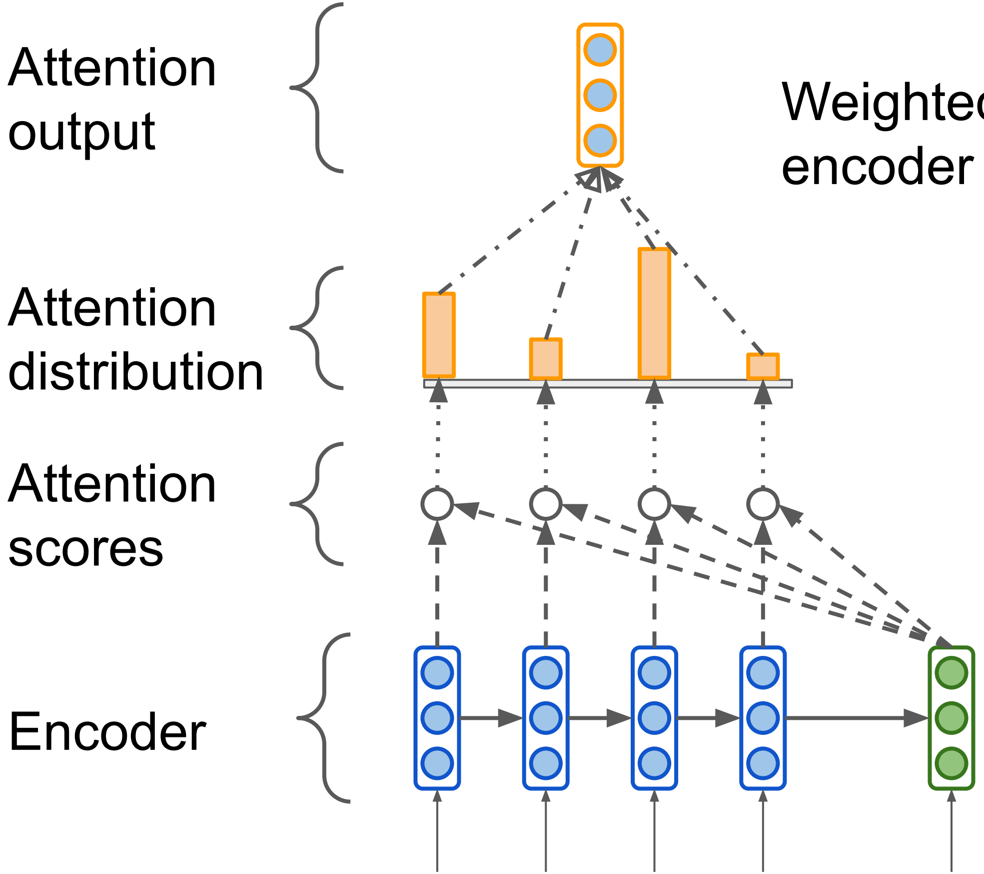


Seq2seq with attention



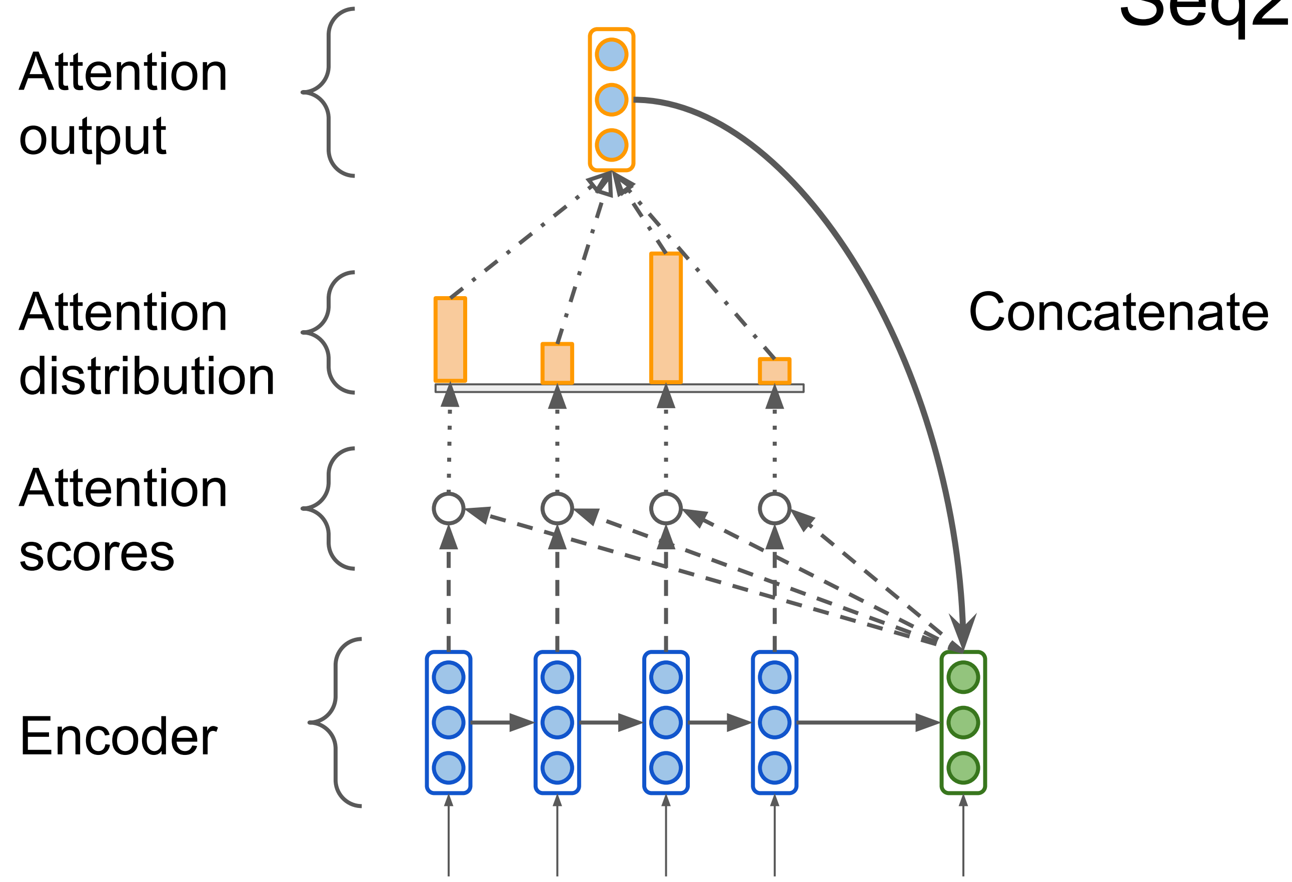
Simply apply softmax to scores

Seq2seq with attention

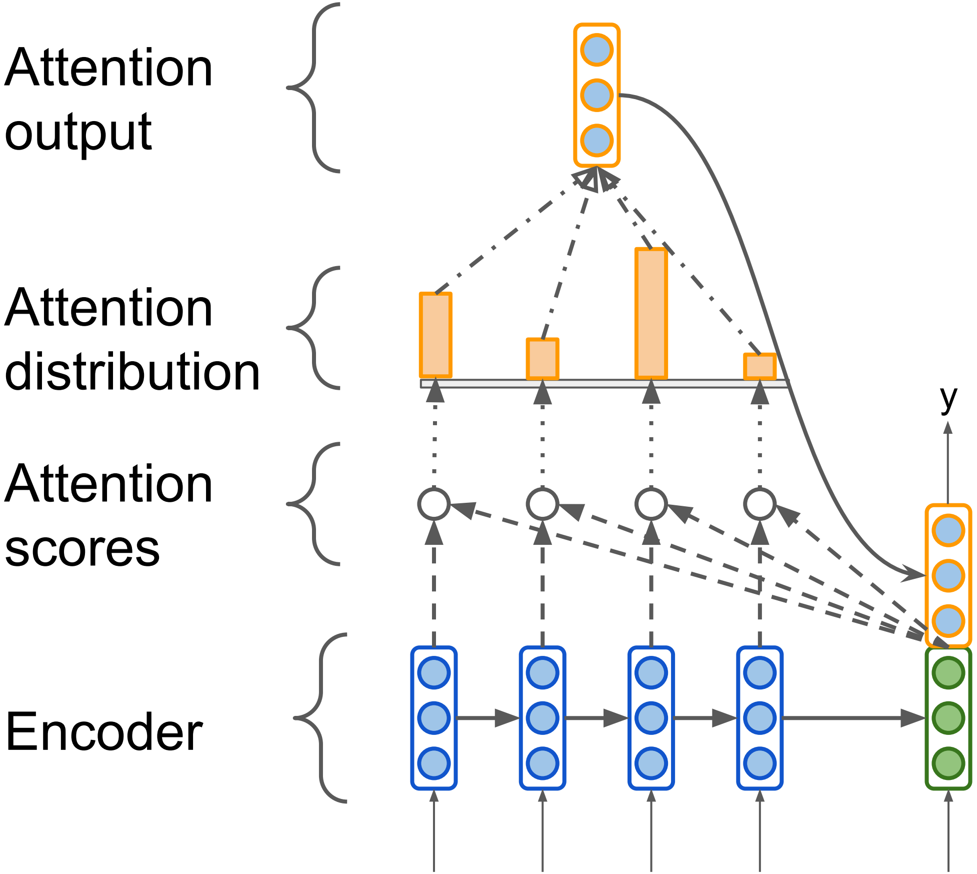


Weighted sum of all encoder states

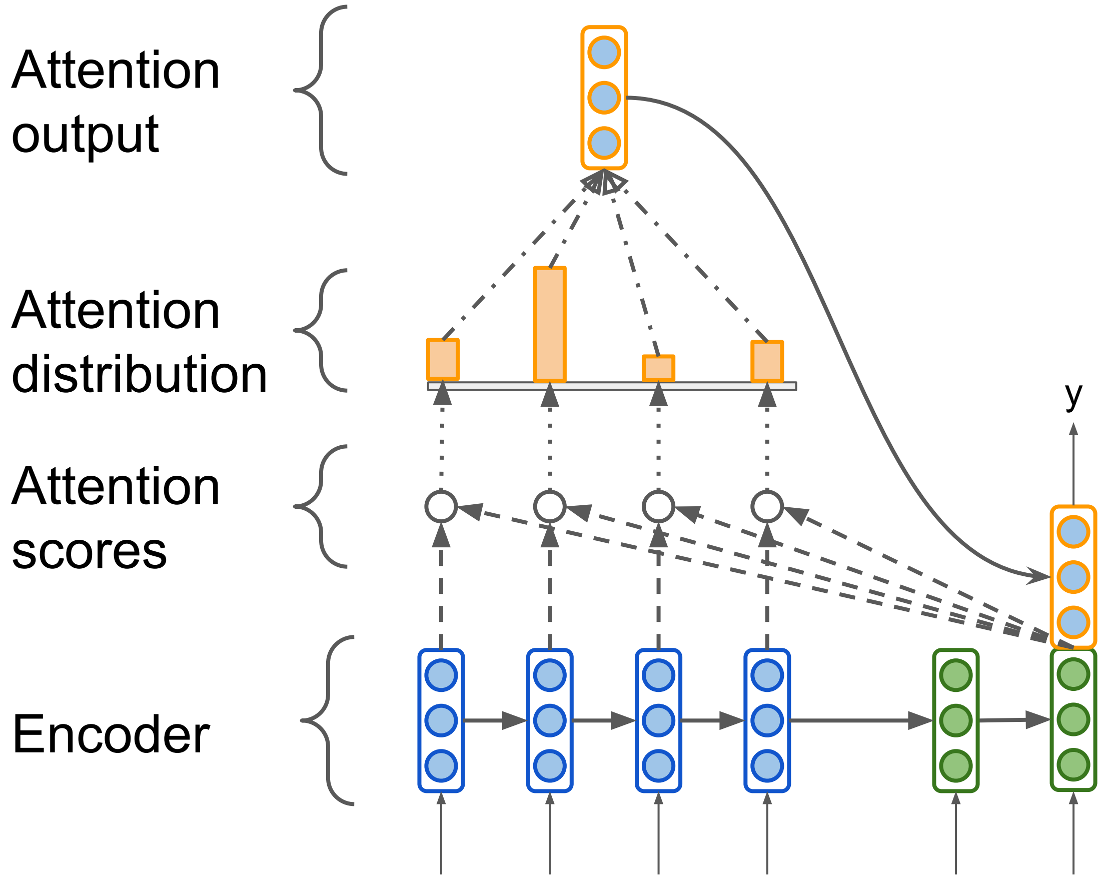
Seq2seq with attention



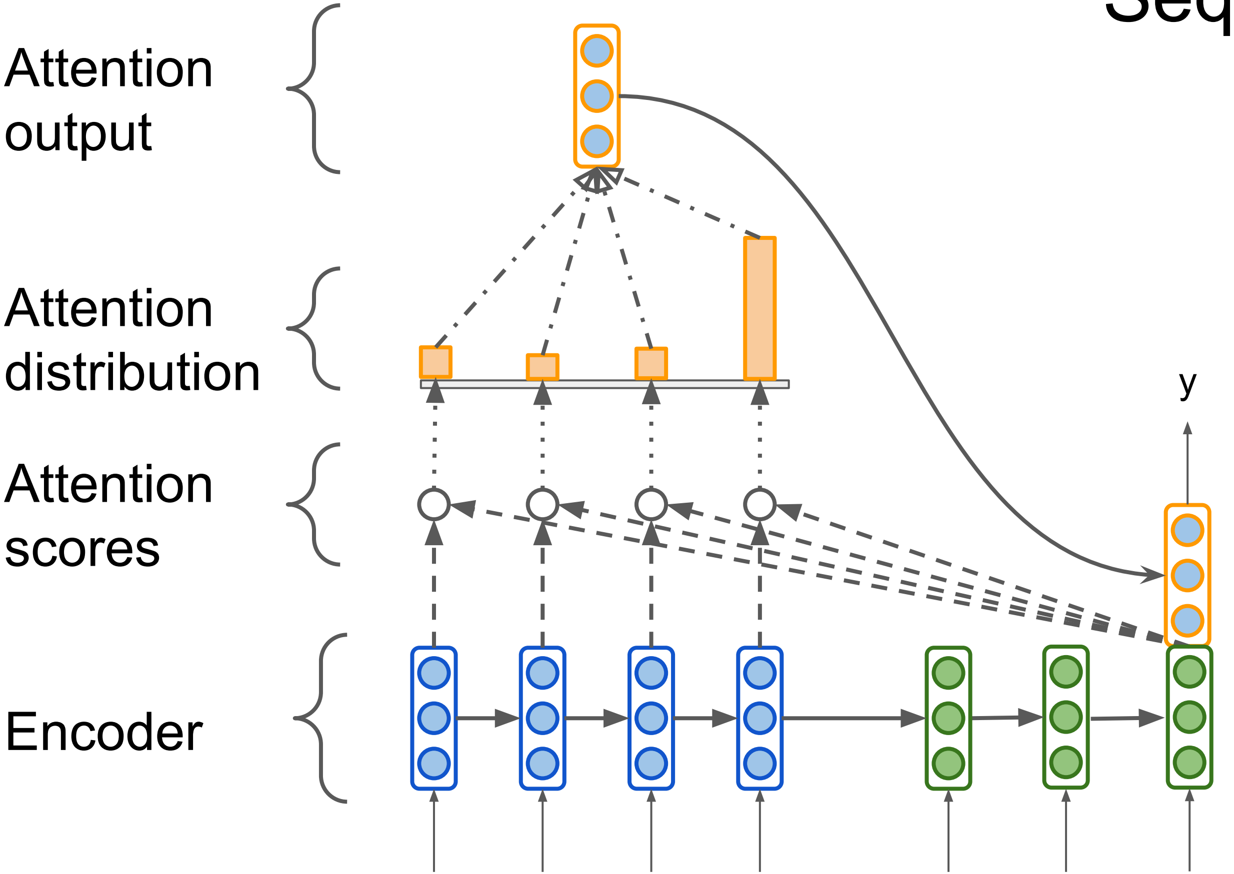
Seq2seq with attention



Seq2seq with attention



Seq2seq with attention



Attention in equations

Denote encoder hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_N \in \mathbb{R}^k$

and decoder hidden state at time step t $\mathbf{s}_t \in \mathbb{R}^k$

The attention scores $\mathbf{e}^t$ can be computed as dot product

$$\mathbf{e}^t = [\mathbf{s}^T \mathbf{h}_1, \ldots, \mathbf{s}^T \mathbf{h}_N]$$

Then the attention vector is a linear combination of encoder states

$$\mathbf{a}_t = \sum_{i=1}^{N} \boldsymbol{\alpha}_i^t \mathbf{h}_i \in \mathbb{R}^k$$, where $$\boldsymbol{\alpha}_t = \mathrm{softmax}(\mathbf{e}_t)$$

Attention provides interpretability

- We may see what the decoder was focusing on
- We get word alignment for free!

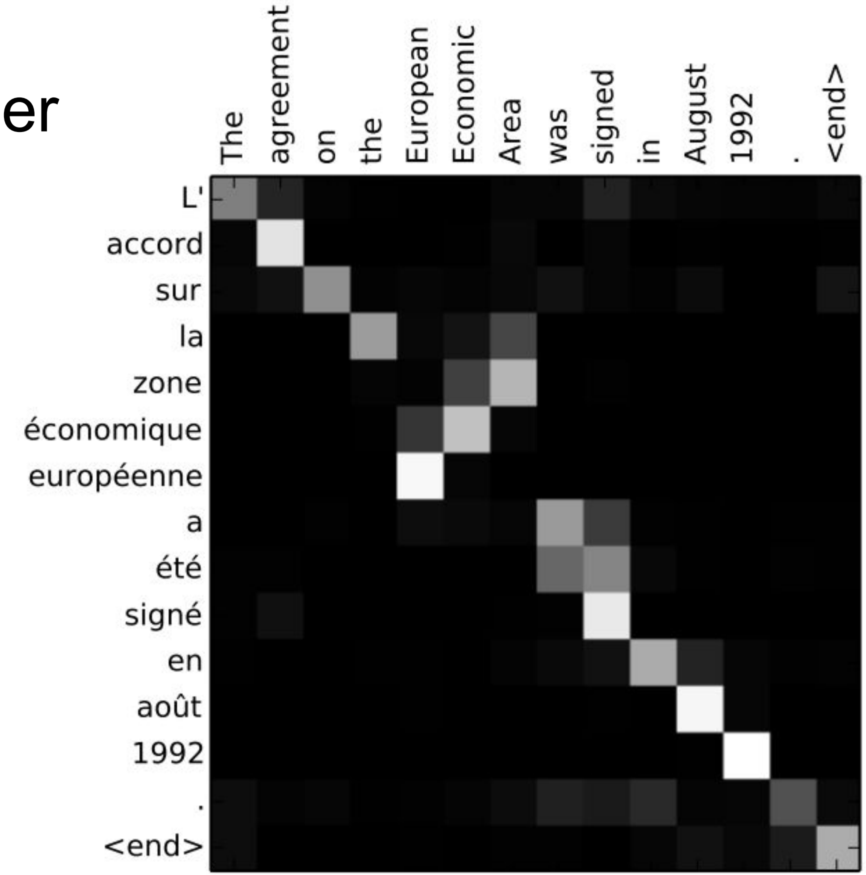
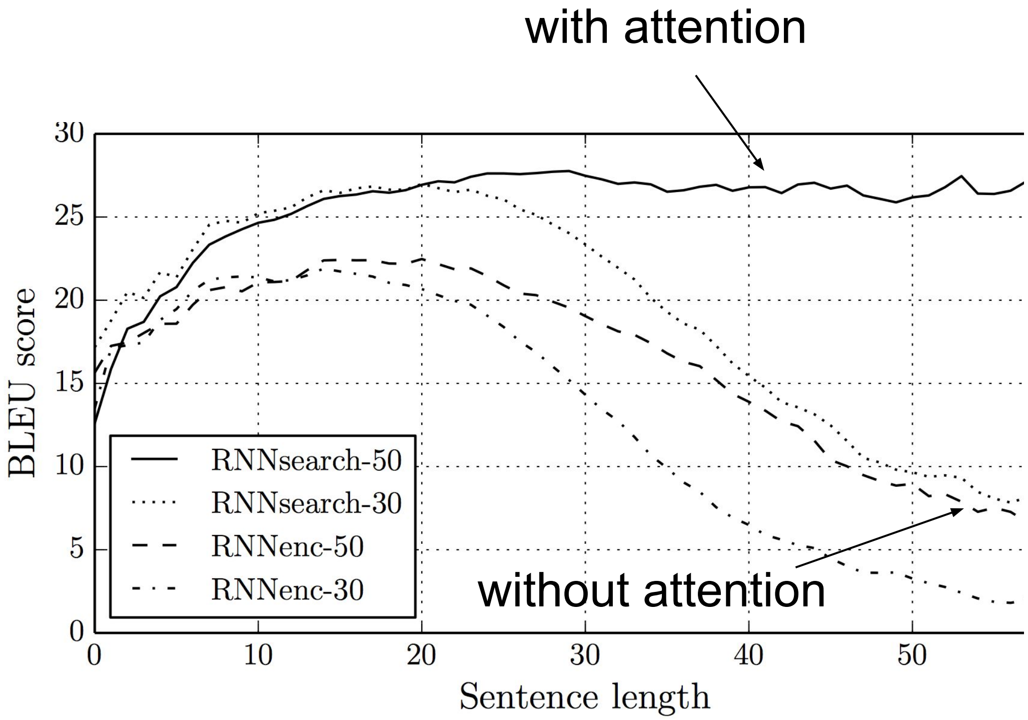


Image source: [Neural Machine Translation by Jointly Learning to Align and Translate]

Attention advantages

- "Free" word alignment
- Better results on long sequences



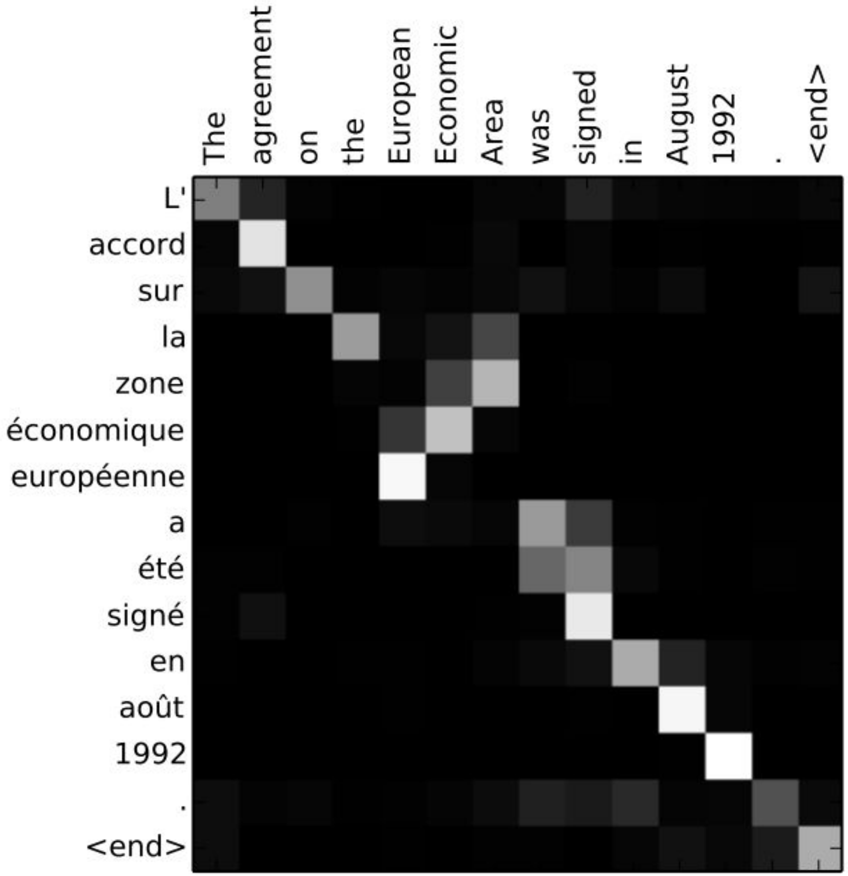


Image source: [Neural Machine Translation by Jointly Learning to Align and Translate]()

Attention variants

- Basic dot-product (the one discussed before): $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ - weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ - weight matrices
  - $\boldsymbol{v} \in \mathbb{R}^{d_3}$ - weight vector

Self-Attention

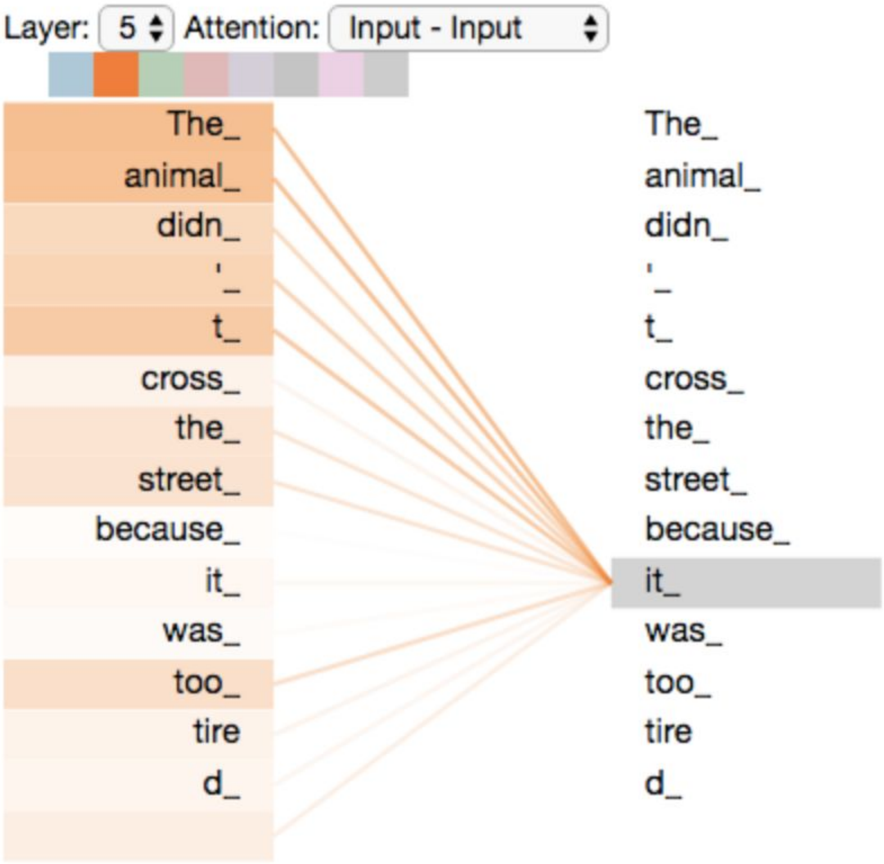
Self-Attention at a High Level

”`The animal didn't cross the street because it was too tired`”

- What does “it” in this sentence refer to?
- We want self-attention to associate “it” with “animal”

- Self-attention is the method the Transformer uses to bake the “understanding” of other relevant words into the one we’re currently processing

Self-Attention at a High Level



Self-Attention: detailed explanation

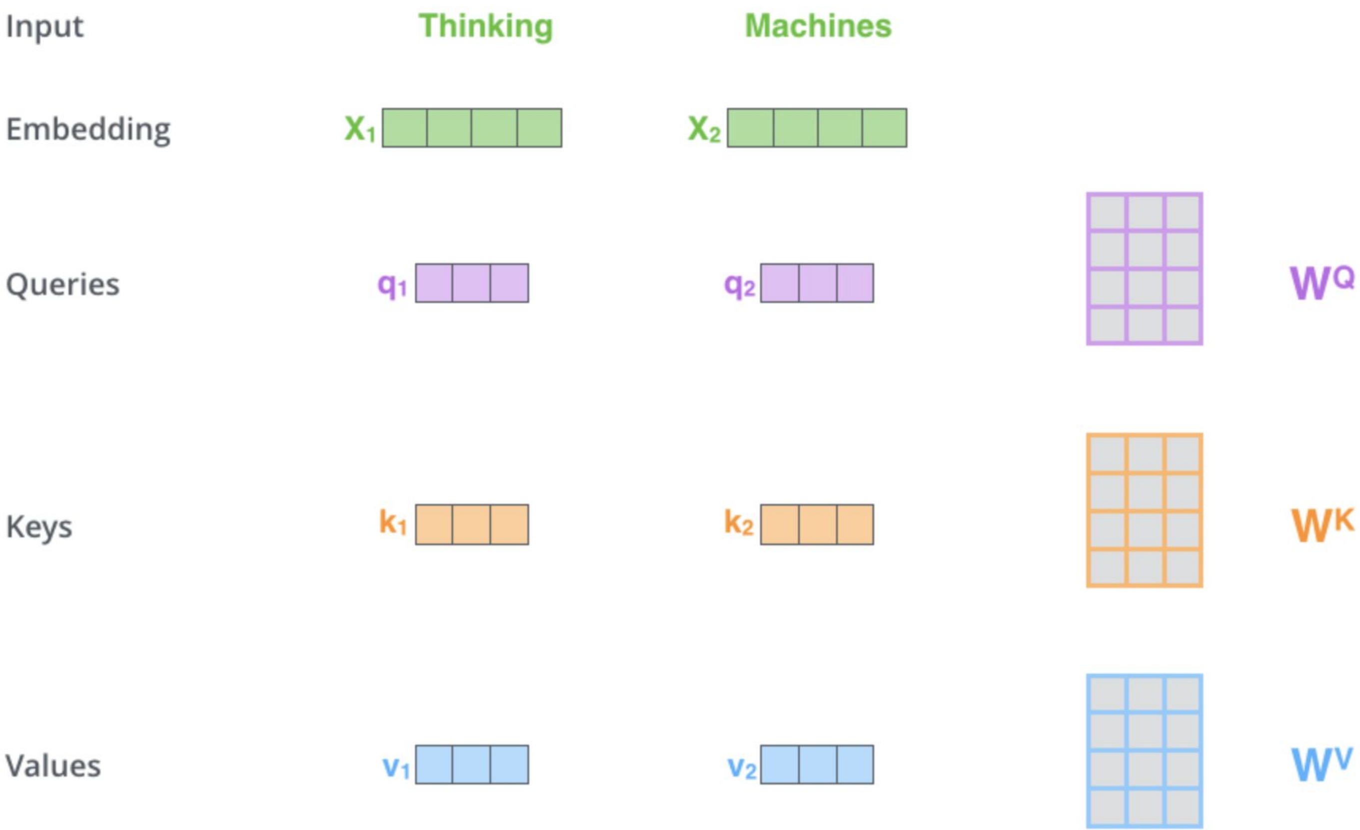


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

STEP 1:

create 3 vectors (**query**, **key**, **value**)

from each of the encoder's input vectors

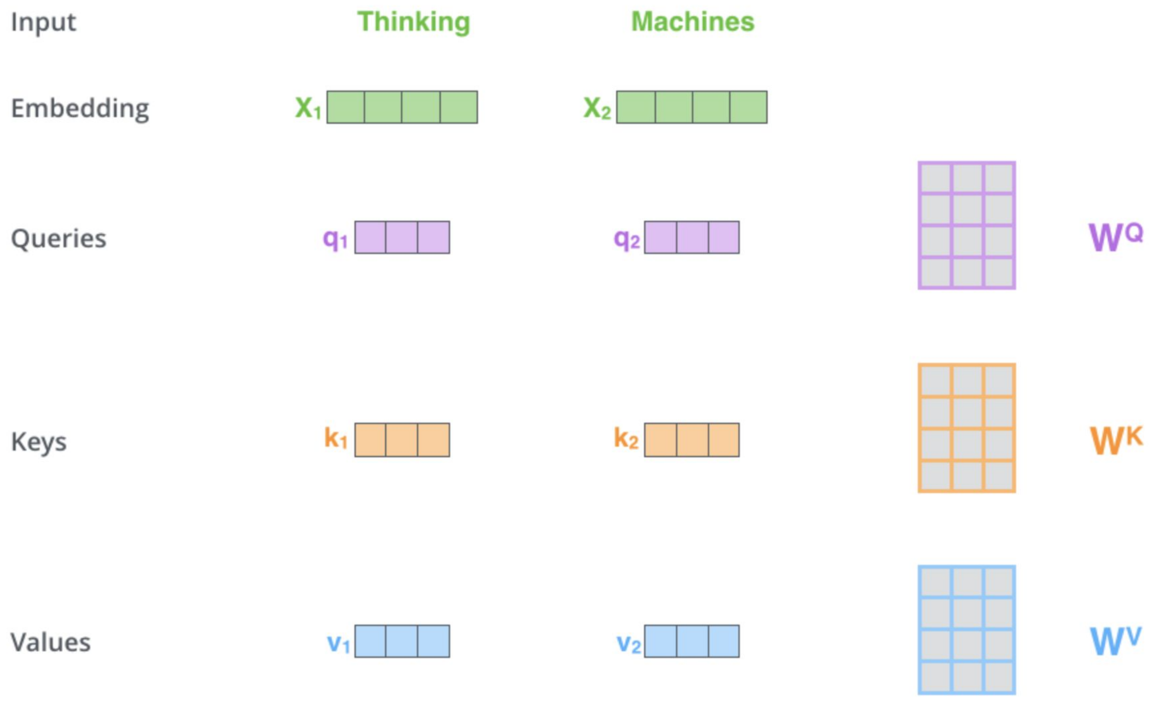


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

What are the **query**, **key**, **value** vectors?

They're abstractions that are useful for calculating and thinking about attention.

Self-Attention: detailed explanation

STEP 2:

calculate a score

(score each word of the input sentence against the current word)

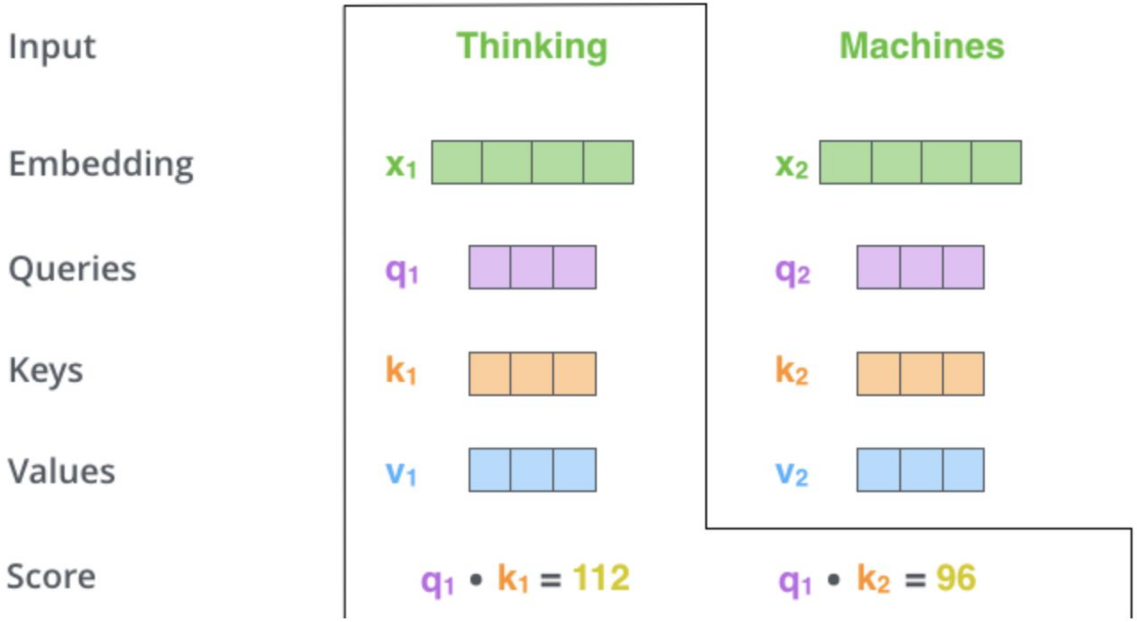


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

<u>STEP 3:</u>

divide the scores by 8

(the square root of the dimension of the key vectors)

<u>STEP 4:</u>

softmax

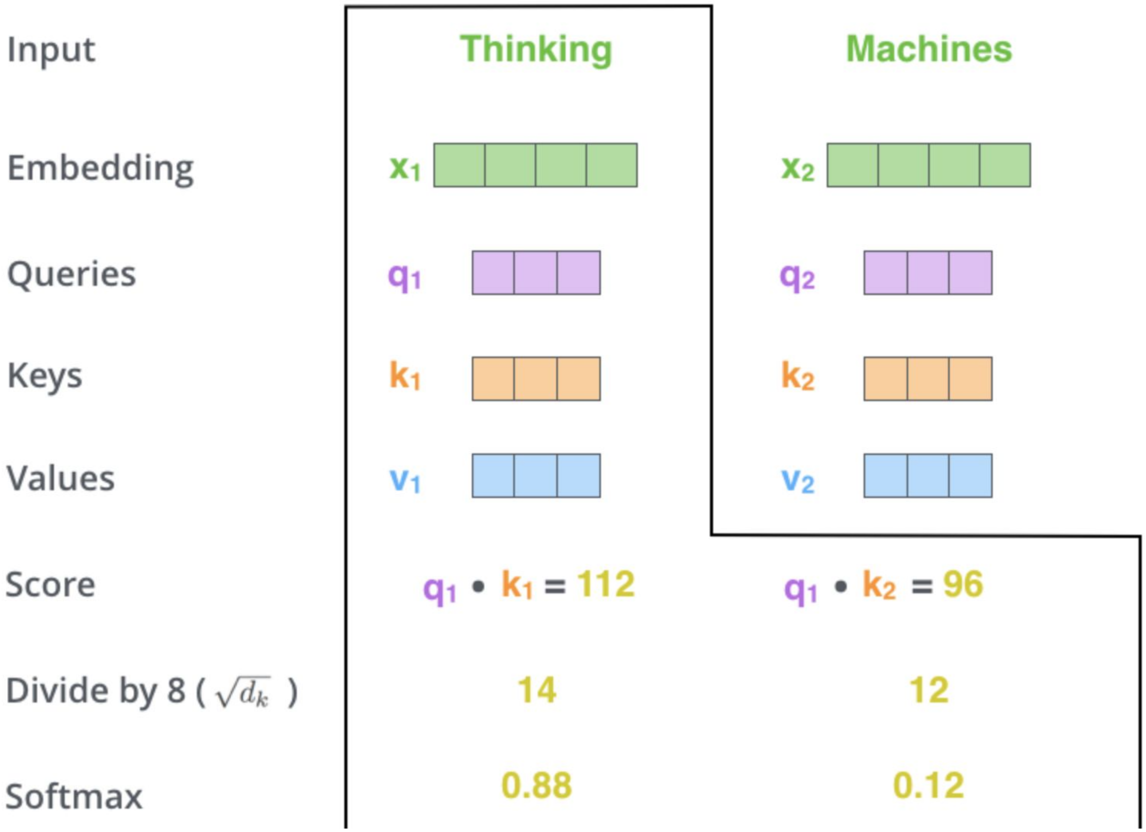


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

STEP 5:

multiply each value vector by the softmax score

STEP 6:

sum up the weighted value vectors

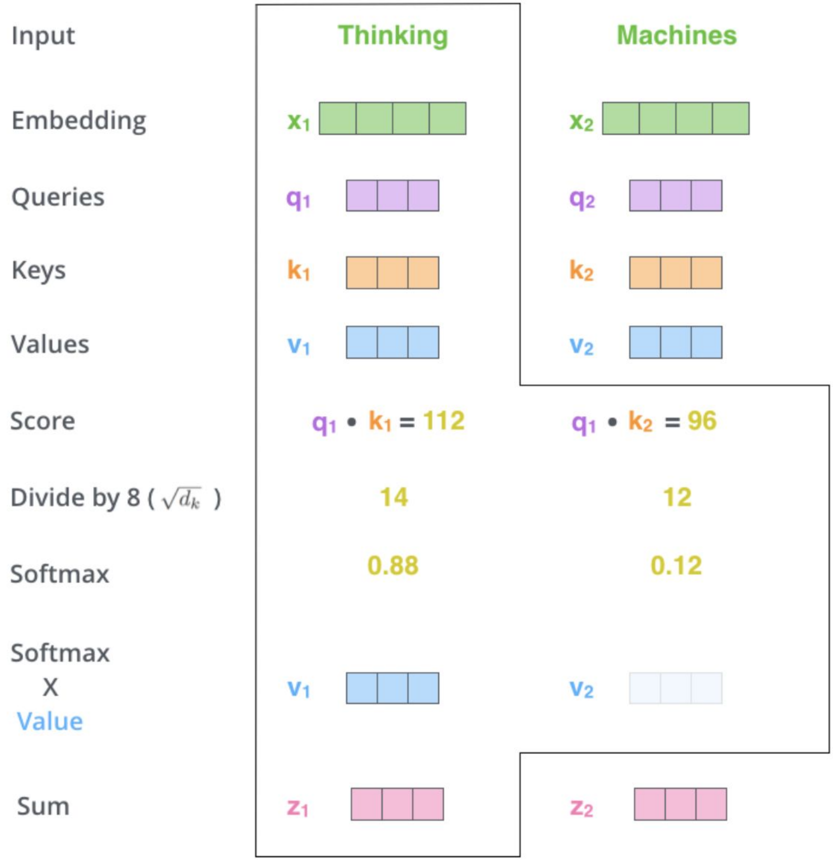
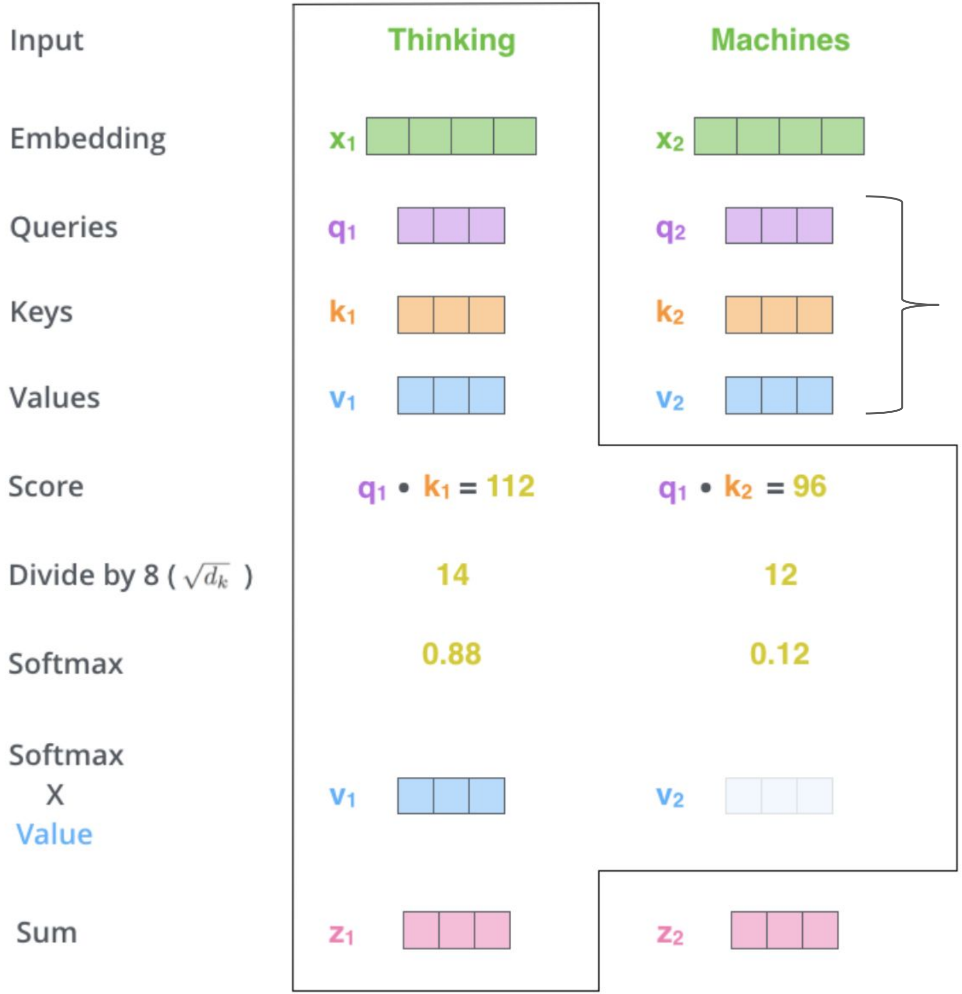


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention



STEP 1: create Query, Key, Value

STEP 2: calculate scores

STEP 3: divide by $\sqrt{d_k}$

STEP 4: softmax

STEP 5: multiply each value vector by the softmax score

STEP 6: sum up the weighted value vectors

Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: Matrix Calculation

Pack embeddings into matrix **X**

Multiply **X** by weight matrices we've trained (**Wk, Wq, Wv**)

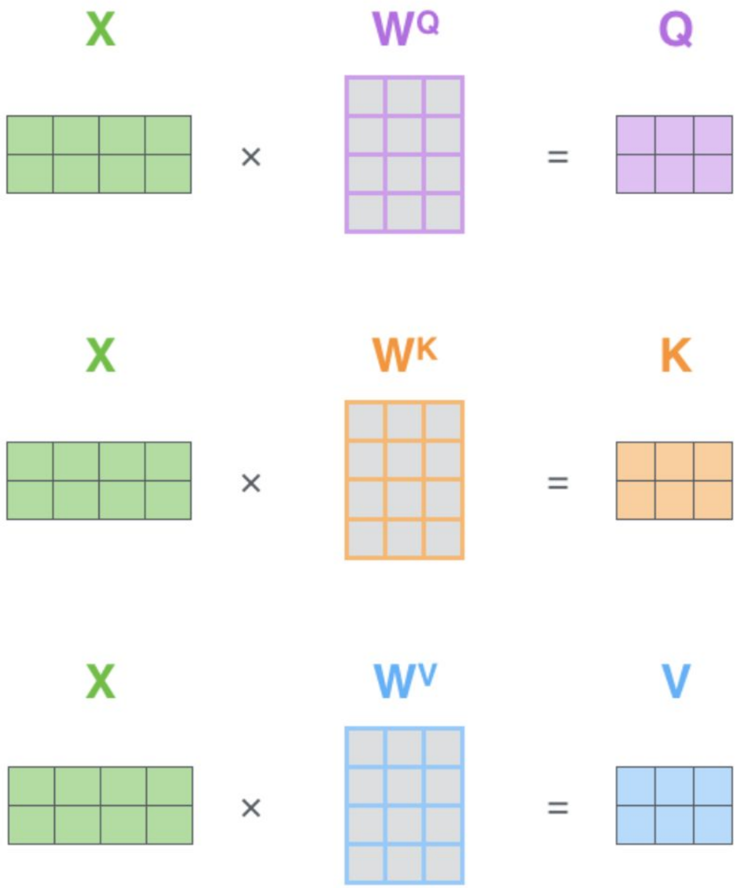


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: Matrix Calculation

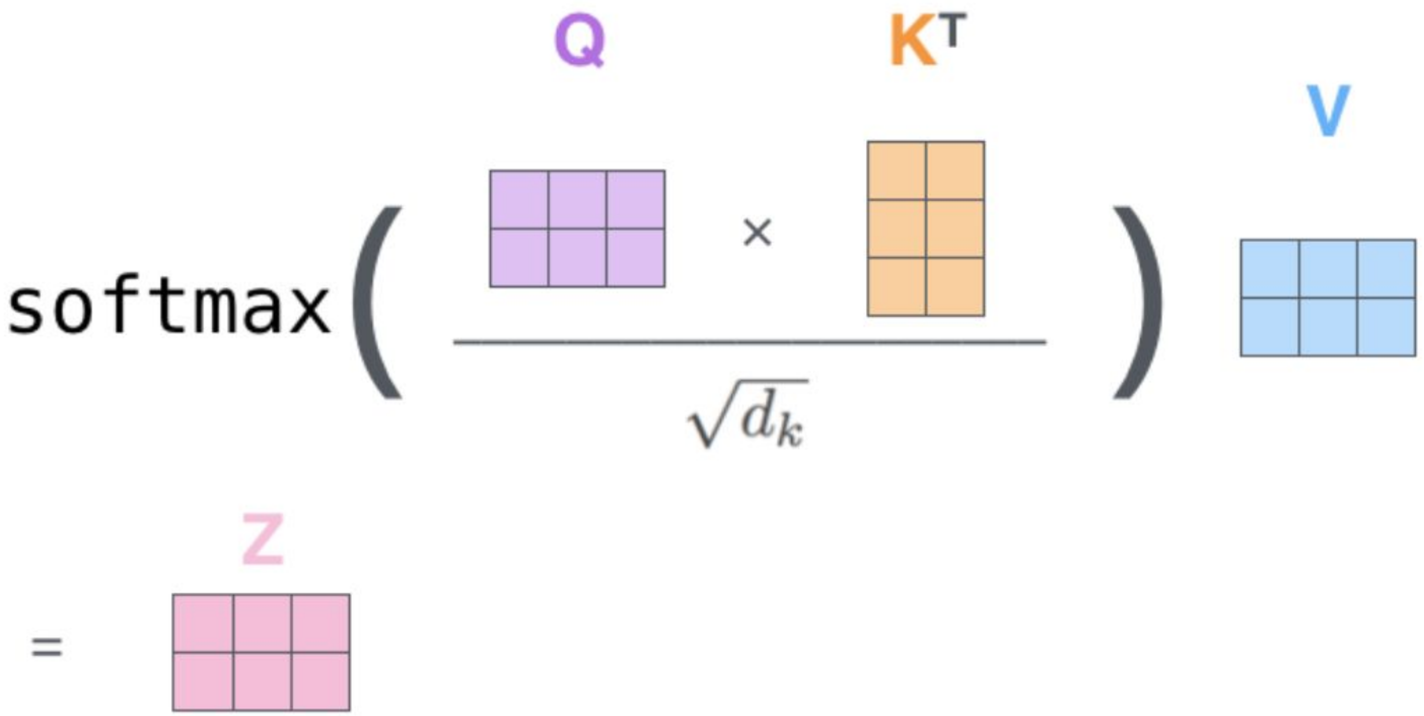


Image source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

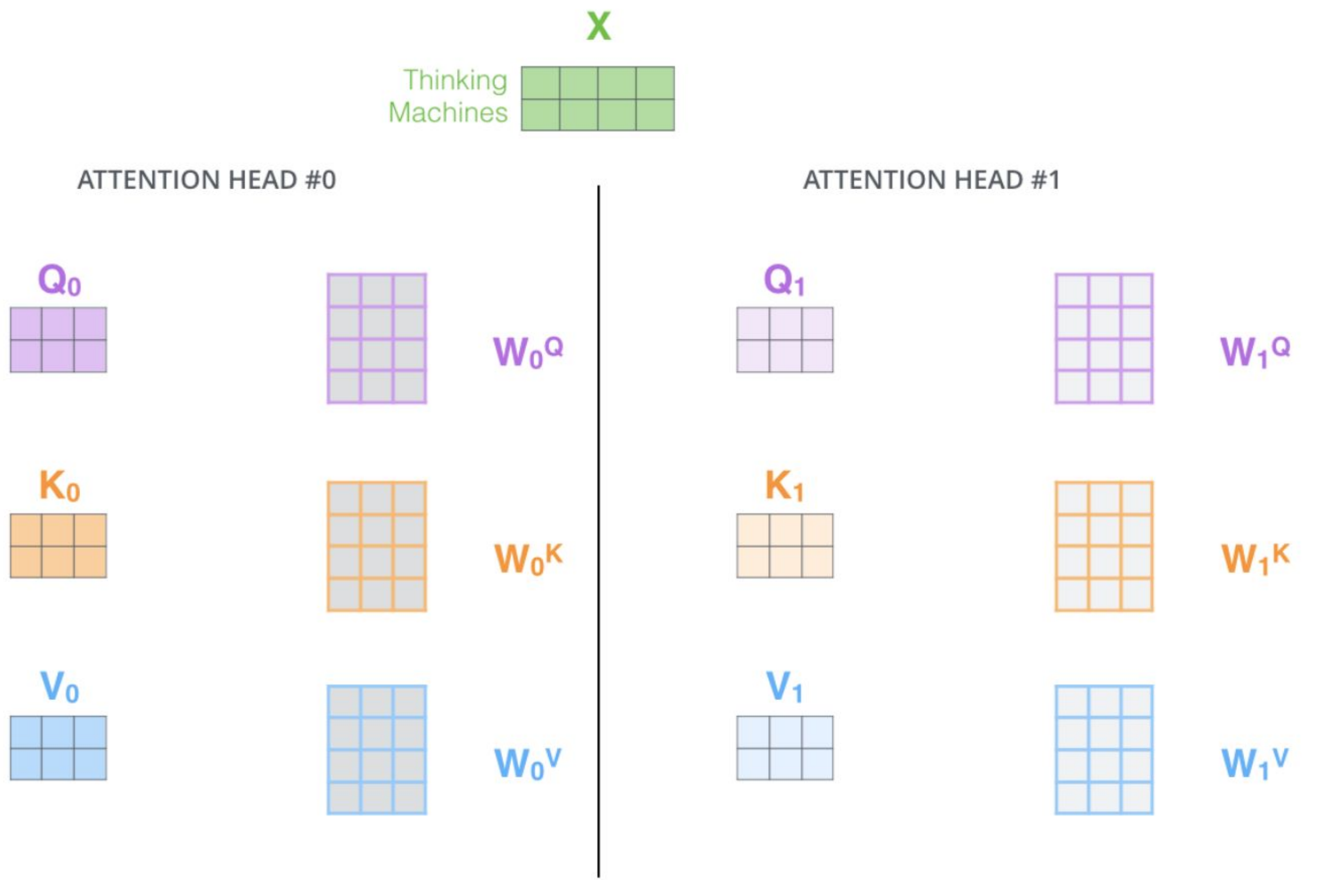


Image source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

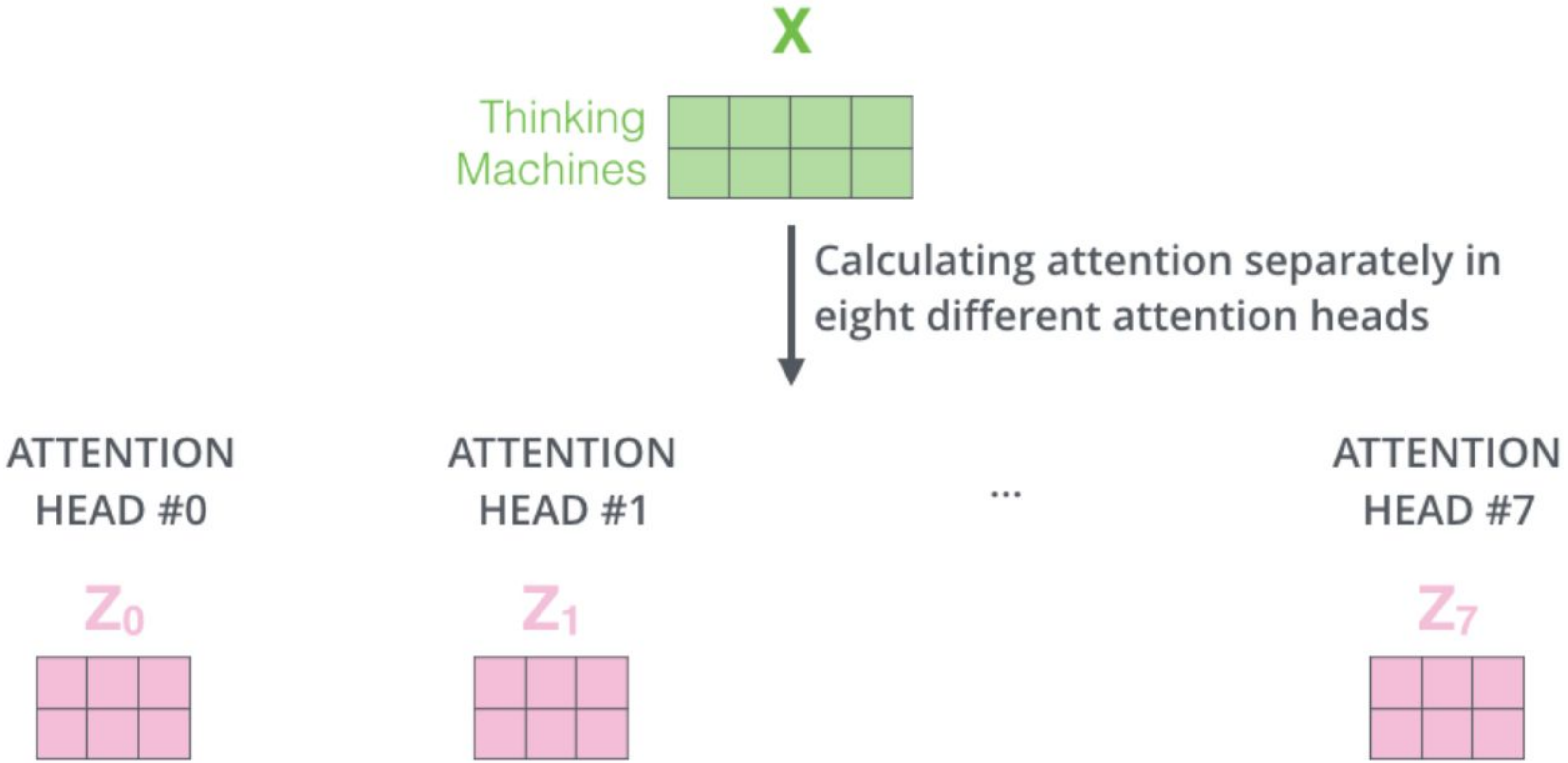


Image source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

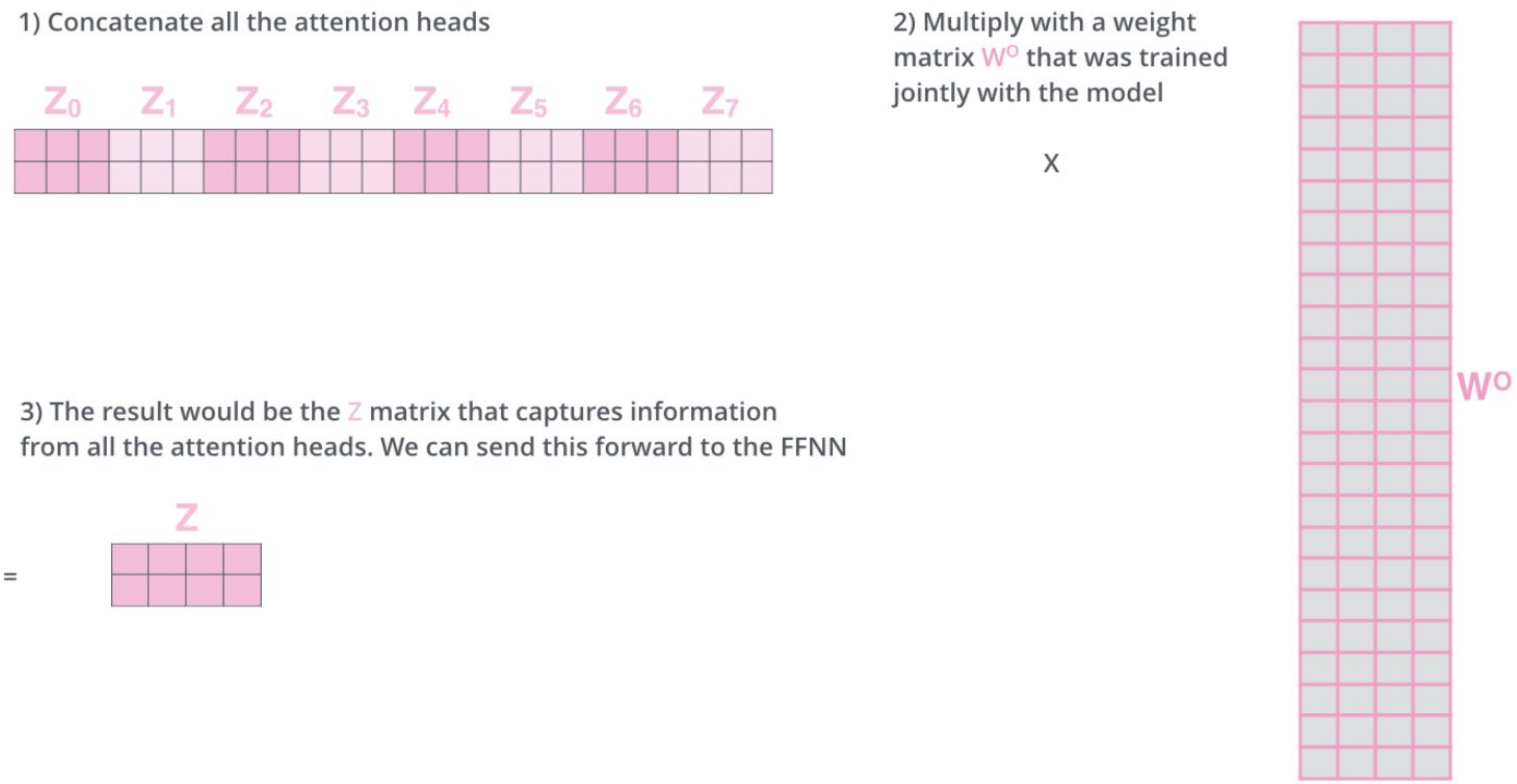


Image source: https://jalammar.github.io/illustrated-transformer/

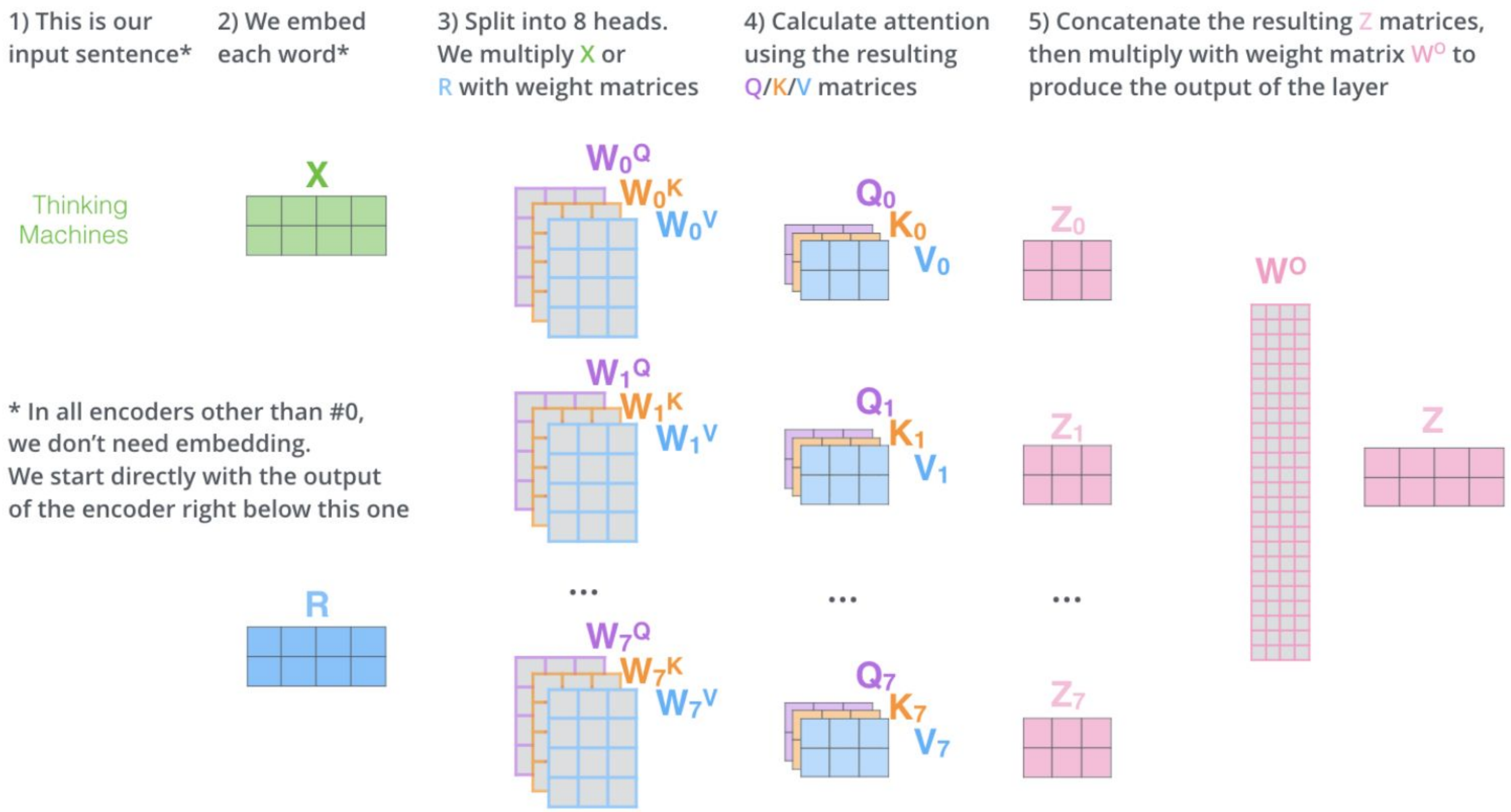
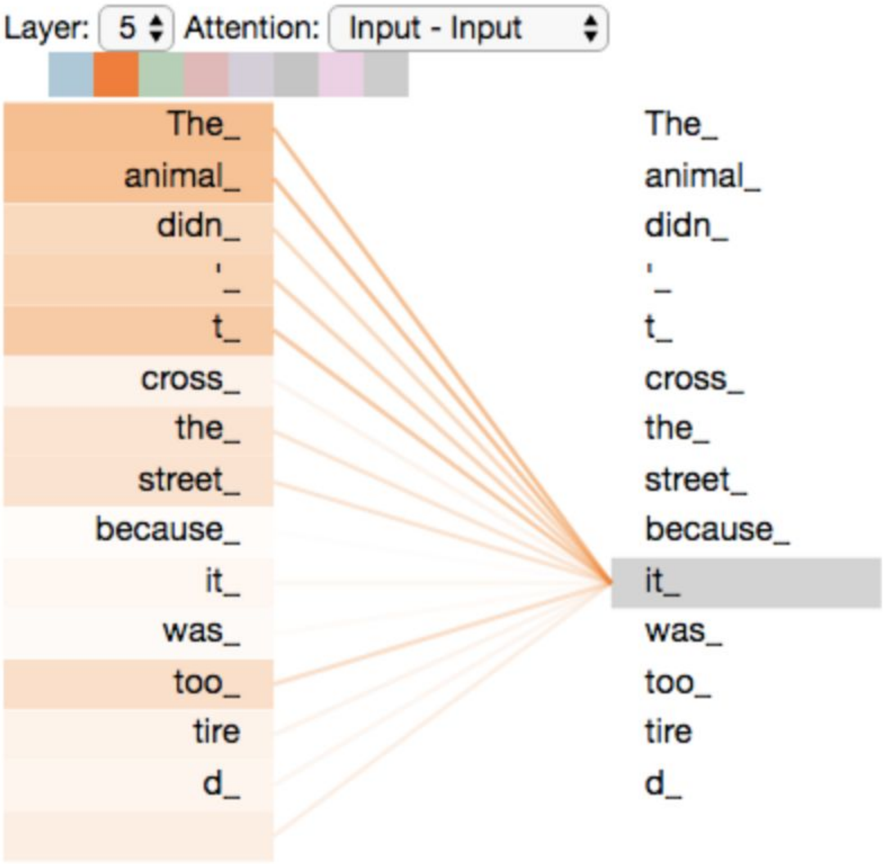
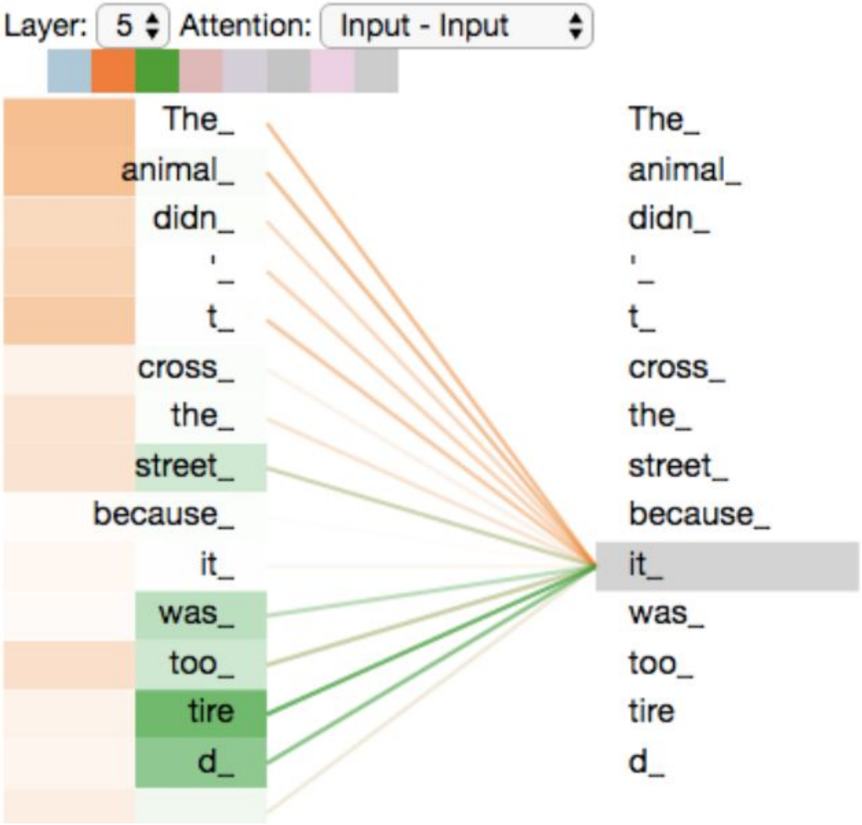


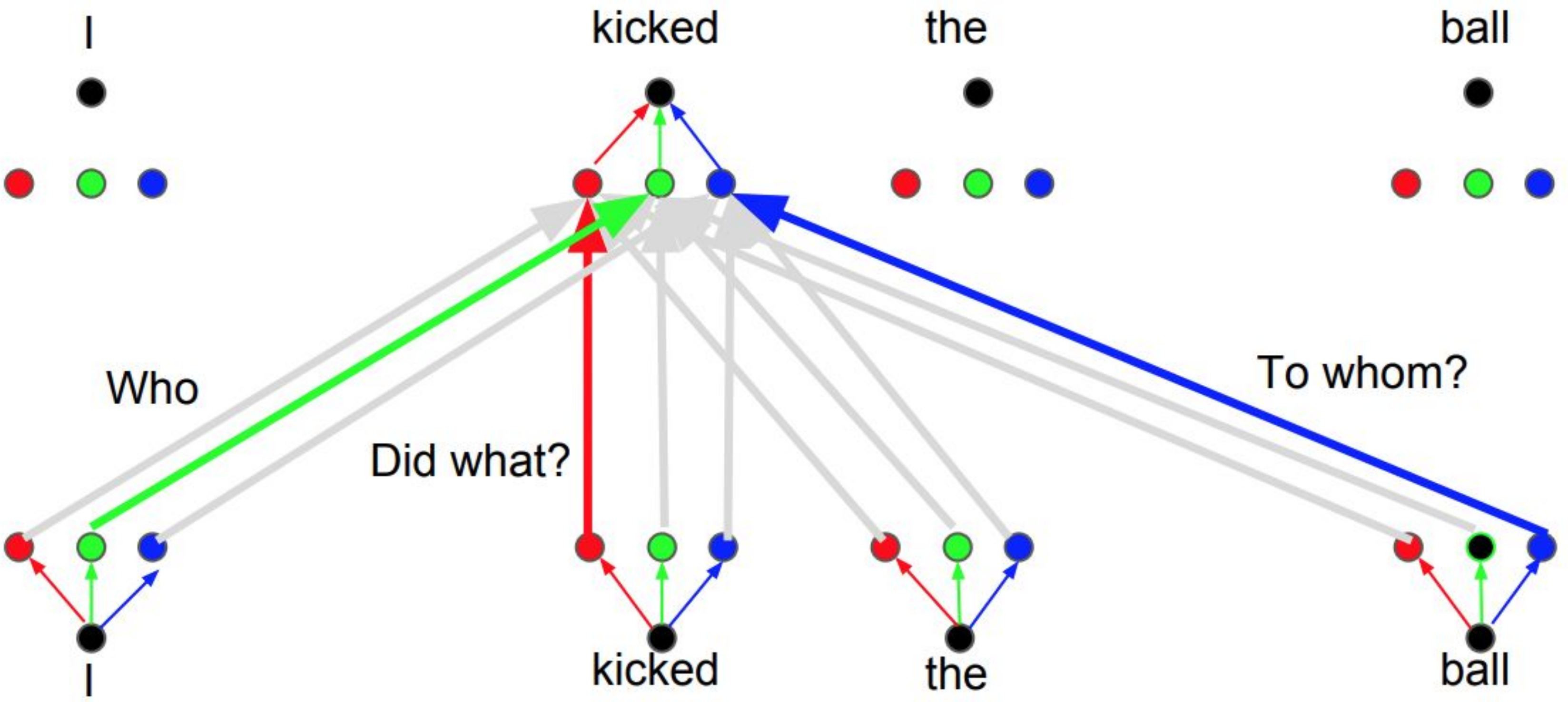
Image source: https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention

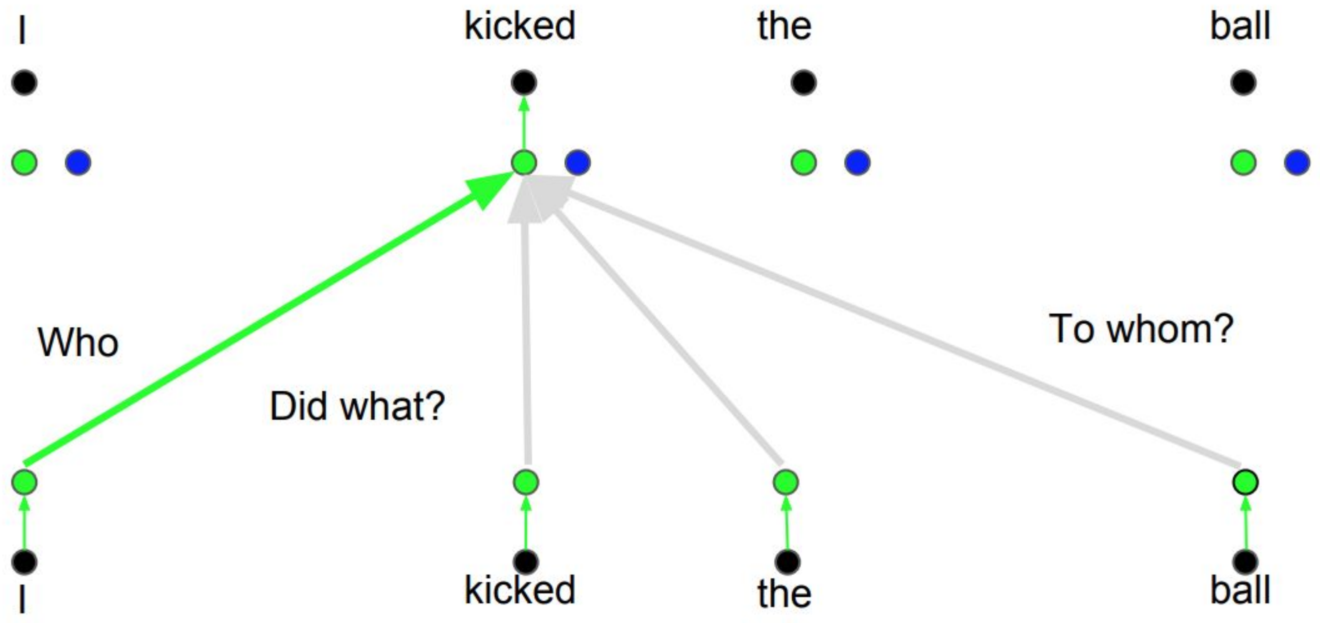




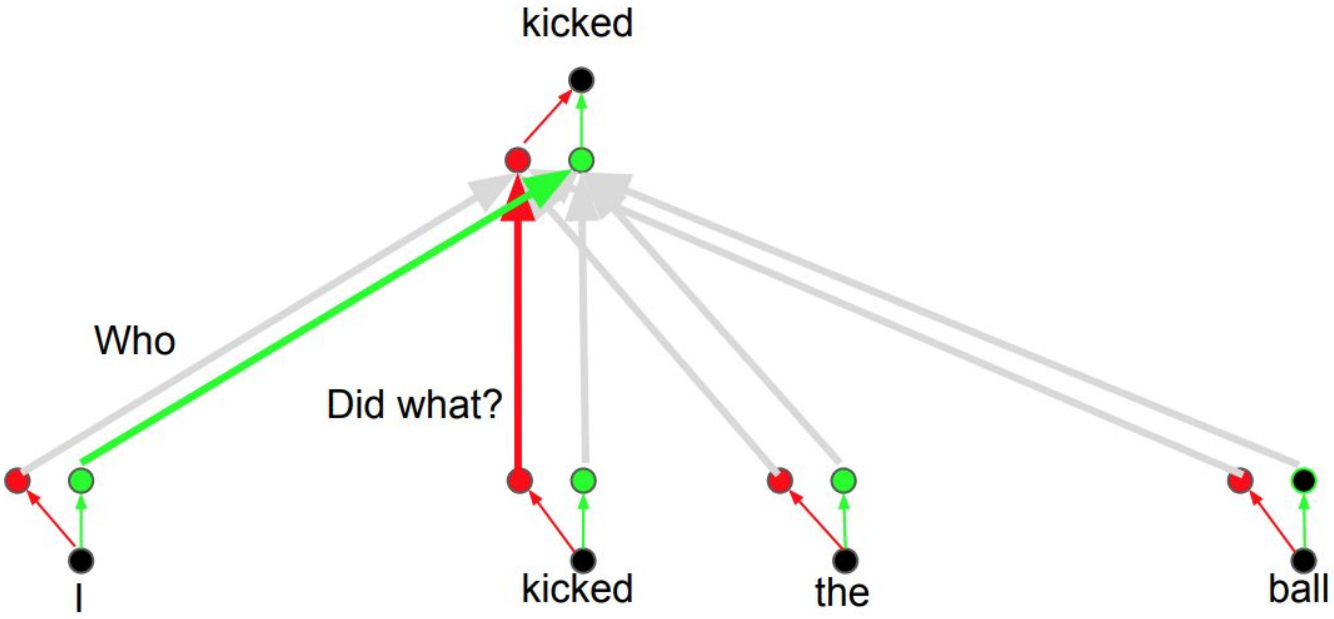
Why Multi-Head Attention?



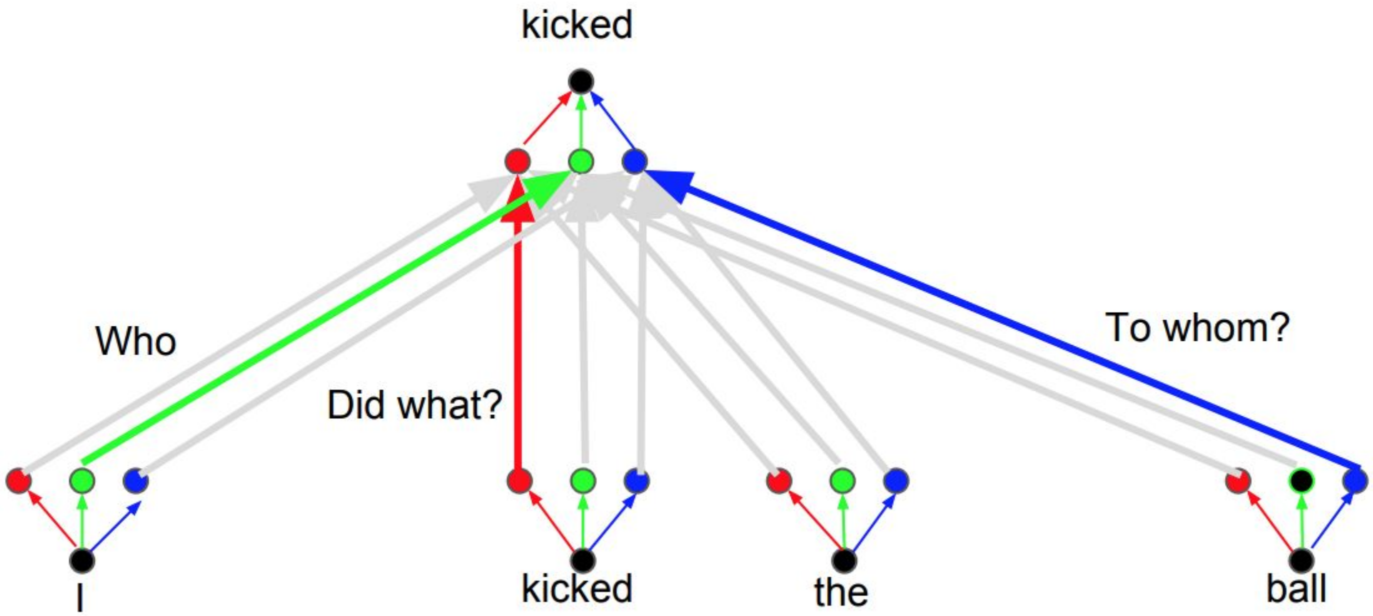
Attention head: Who



Attention head: Did What?

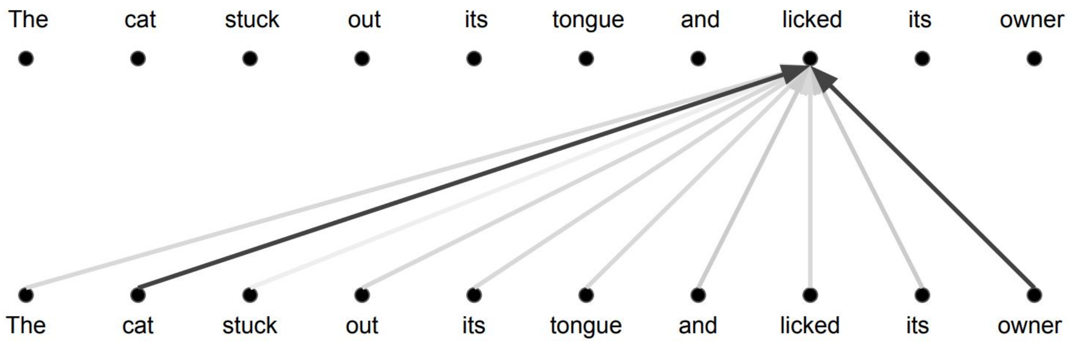


Attention head: To Whom?



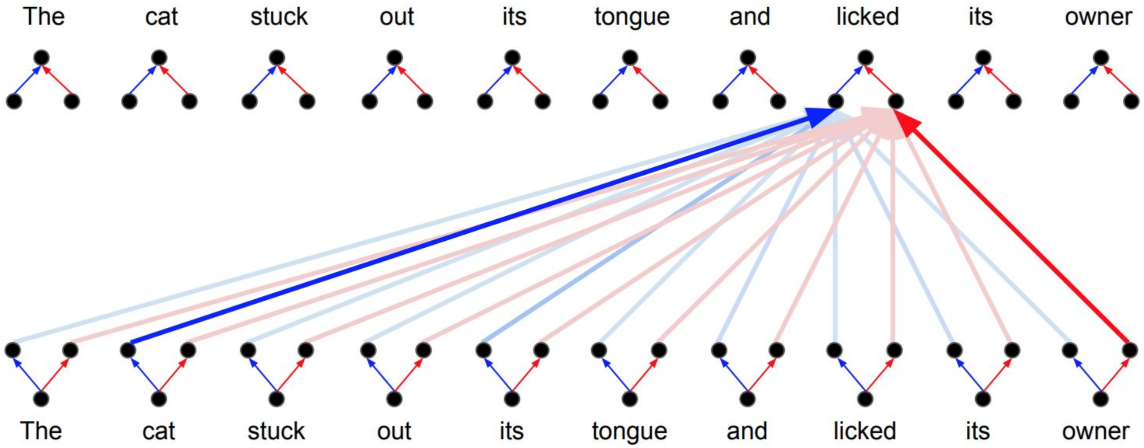
Attention vs. Multi-Head Attention

<u>Attention:</u> a weighted average



<u>Multi-Head Attention:</u>

parallel attention layers with different linear transformations on input and output.



Transformer outro

The Transformer

Image source: https://jalammar.github.io/illustrated-transformer/

The Transformer

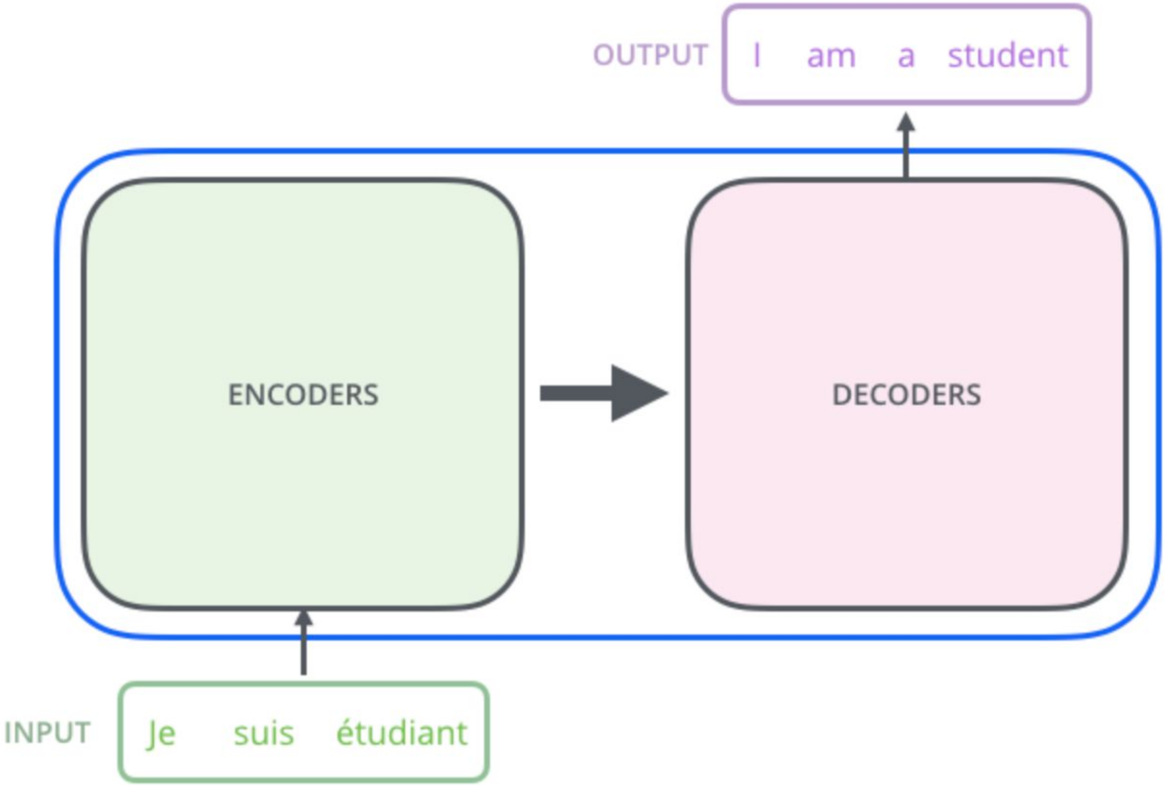


Image source: https://jalammar.github.io/illustrated-transformer/

The Transformer

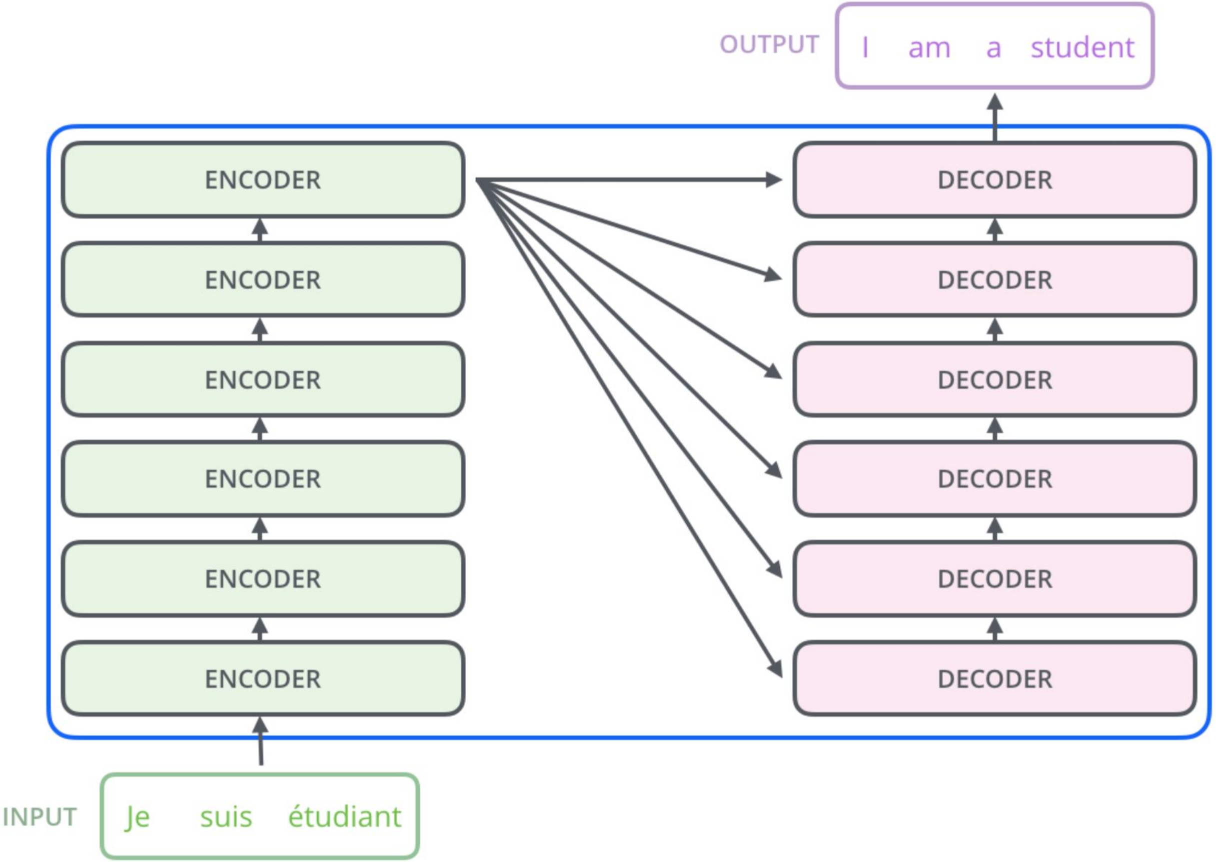
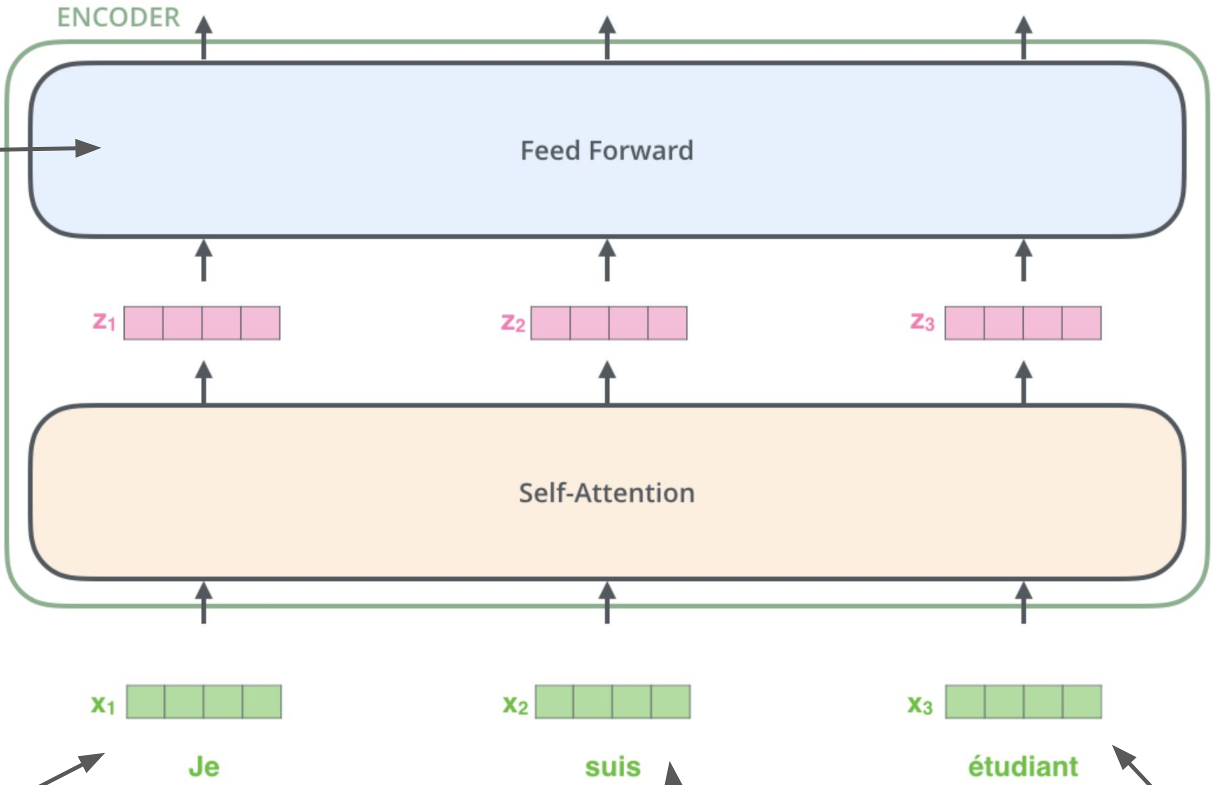


Image source: https://jalammar.github.io/illustrated-transformer/

The Encoder Side

Can be parallelized



the word in each position flows through its own path in the encoder

Image source: https://jalammar.github.io/illustrated-transformer/

Outro

- Attention mechanism allows to “attend all positions” in the original sequence (or any other input with internal structure)
- Attention mechanism requires more computational resources than original seq2seq models
- Change of the model architecture affects the training procedure, so be careful with intuitive explanations

More on translation quality evaluation

BLEU (Bilingual Evaluation Understudy) compares the machine-written translation to human-written translation, and computes a similarity score based on:

- n-gram precision
- penalty for too-short system translations (brevity penalty)

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^{n} \text{precision}_i\right)^{1/n} \cdot 100\%$$

$$\text{brevity penalty} = min\left(1, \frac{\text{output length}}{\text{reference length}}\right)$$

Perplexity

$$PP(W) = P(w_1, w_2, .., w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, .., w_N)}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i | w_1, .., w_{i-1})}}$$

WER (Word Error Rate)

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N = S + D + C)

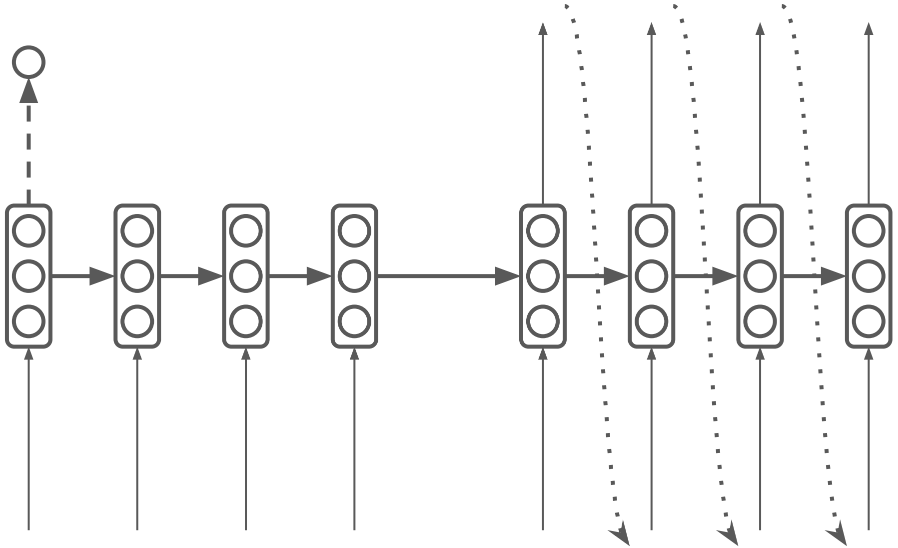
ROUGE

- **ROUGE** Recall-Oriented Understudy for Gisting Evaluation
- Recall in the context of ROUGE means how much of the reference summary is the system summary recovering or capturing
- **BLEU** is focusing on **precision:**
    - overlapping_words / total_words_in_**system**_summary
- **ROUGE** is focusing on **recall:**
    - overlapping_words / total_words_in_**reference**_summary

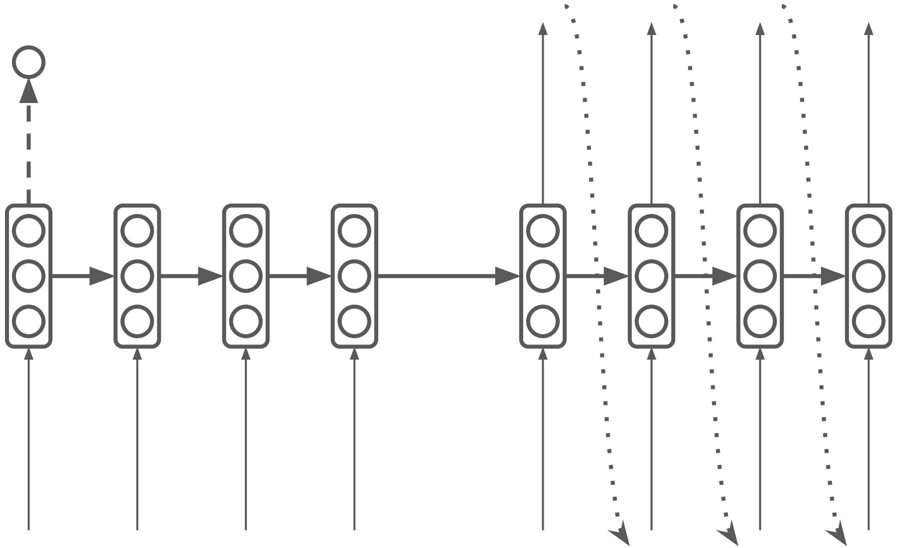
ROUGE - Recall-Oriented Understudy for Gisting Evaluation

- **ROUGE-N:** Overlap of N-grams between the system and reference summaries.
- **ROUGE-L:** Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.
- **ROUGE-W:** Weighted LCS-based statistics
- etc.

Seq2seq with attention



Seq2seq with attention



Machine Translation