Lecture 15: Attention Model and Transformer

Xuming He
SIST, ShanghaiTech
Fall, 2020

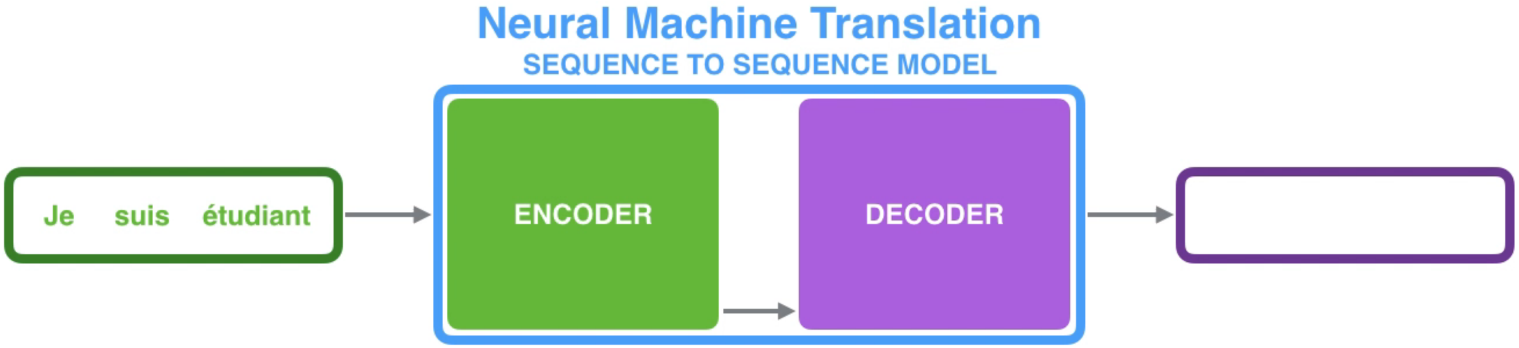
Outline

- Recap and motivation
- Attention models in RNNs
- Transformer

Acknowledgement: Hugo Larochelle's, Mehryar Mohri@NYU's,Yingyu Liang@Princeton's, Bhiksha Raj@CMU's &Feifei Li@Stanford's course notes

Recap

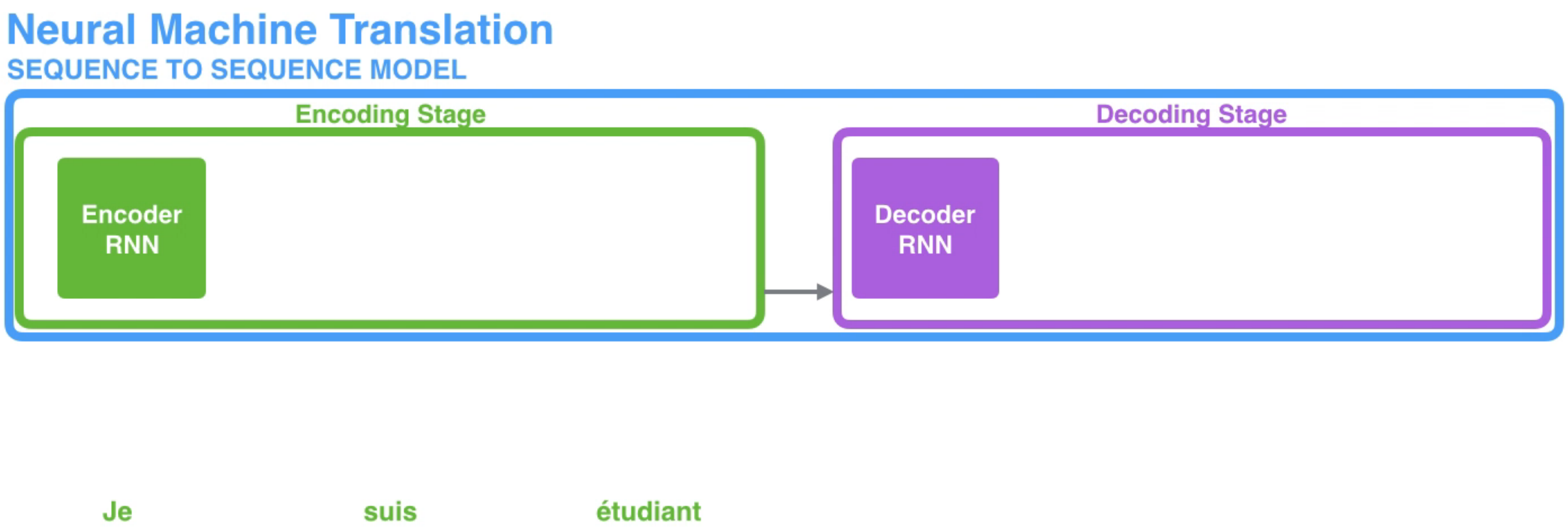
- RNN models
  - Encoder-decoder architecture



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Recap

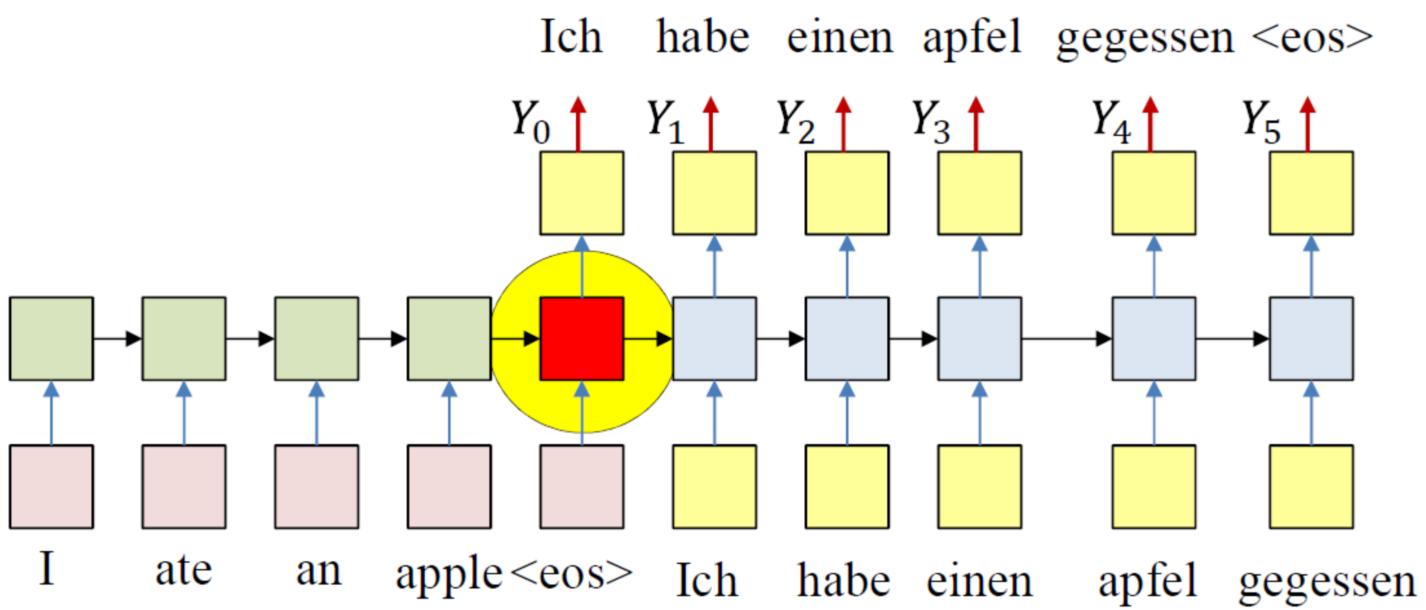
- RNN models
  - Encoder-decoder architecture



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

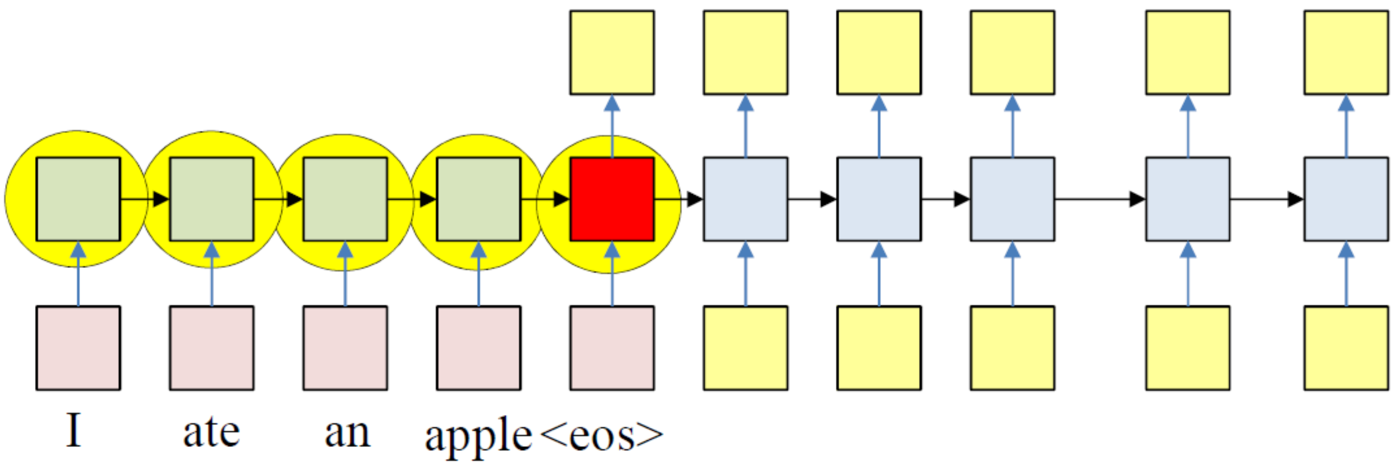
A problem with this framework

- All the information on the input sequence is embedded into a single vector
  - The latent layer at the end of the input sequence
  - This layer is overloaded with information



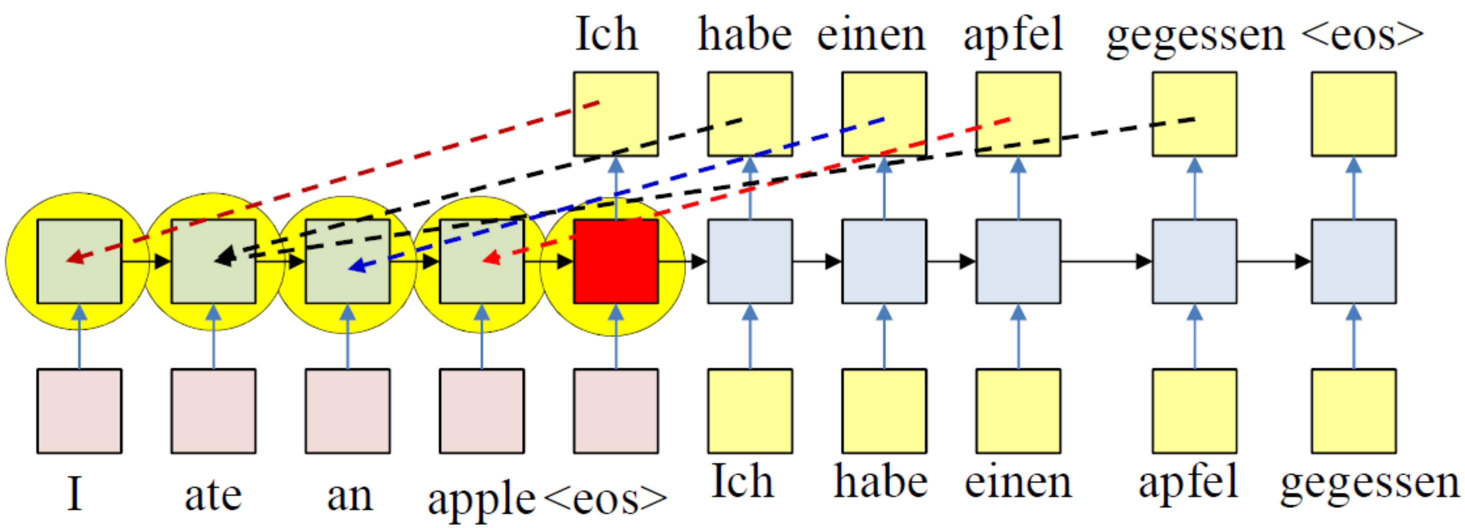
A problem with this framework

- All latent values carry information
  - Some of which may be diluted downstream



A problem with this framework

- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs



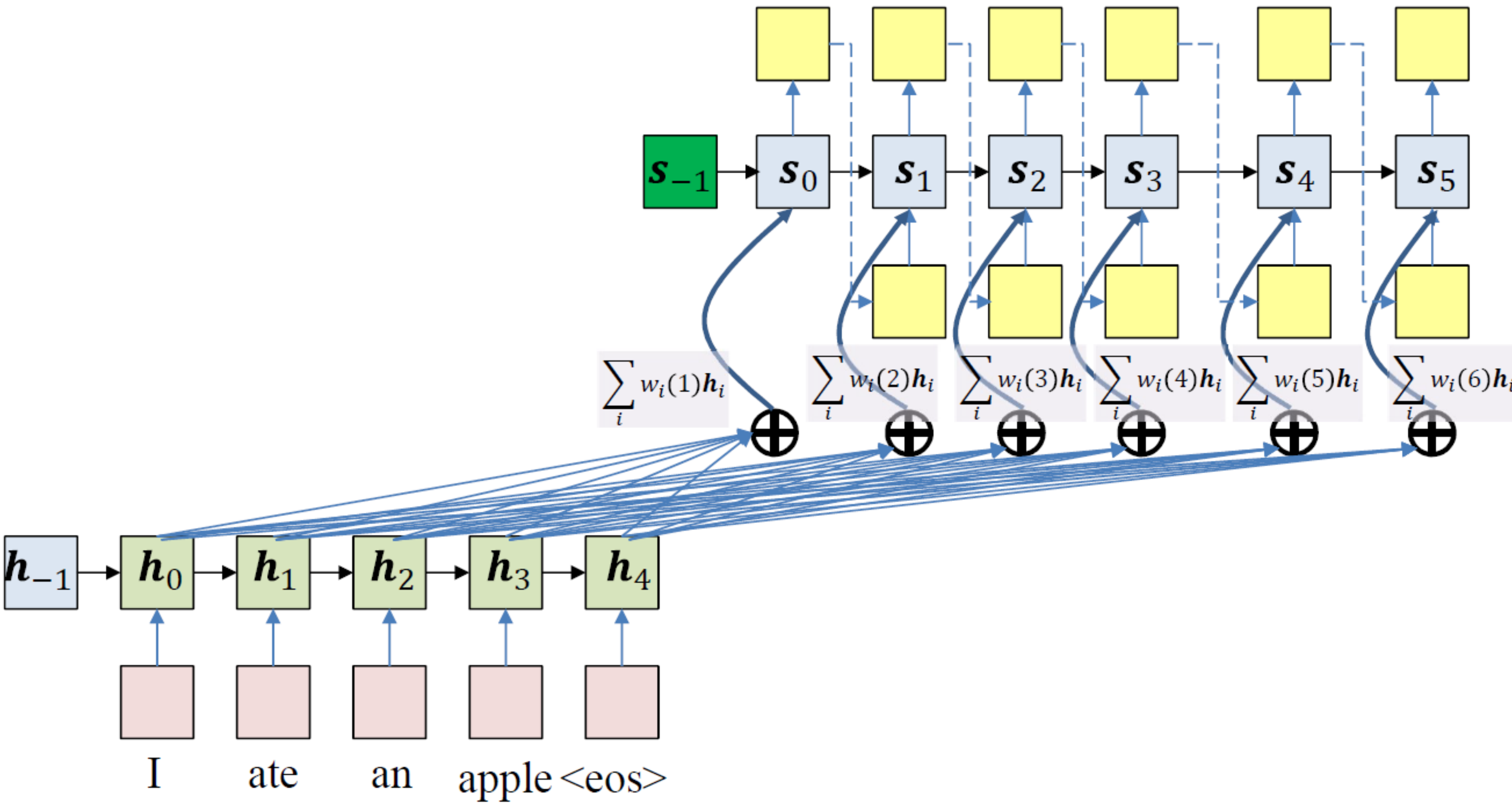
Outline

- Recap and motivation
- Attention models in RNNs
- Transformer
- Graph Neural Networks

Acknowledgement: Hugo Larochelle's, Mehryar Mohri@NYU's,Yingyu Liang@Princeton's, Bhiksha Raj@CMU's &Feifei Li@Stanford's course notes

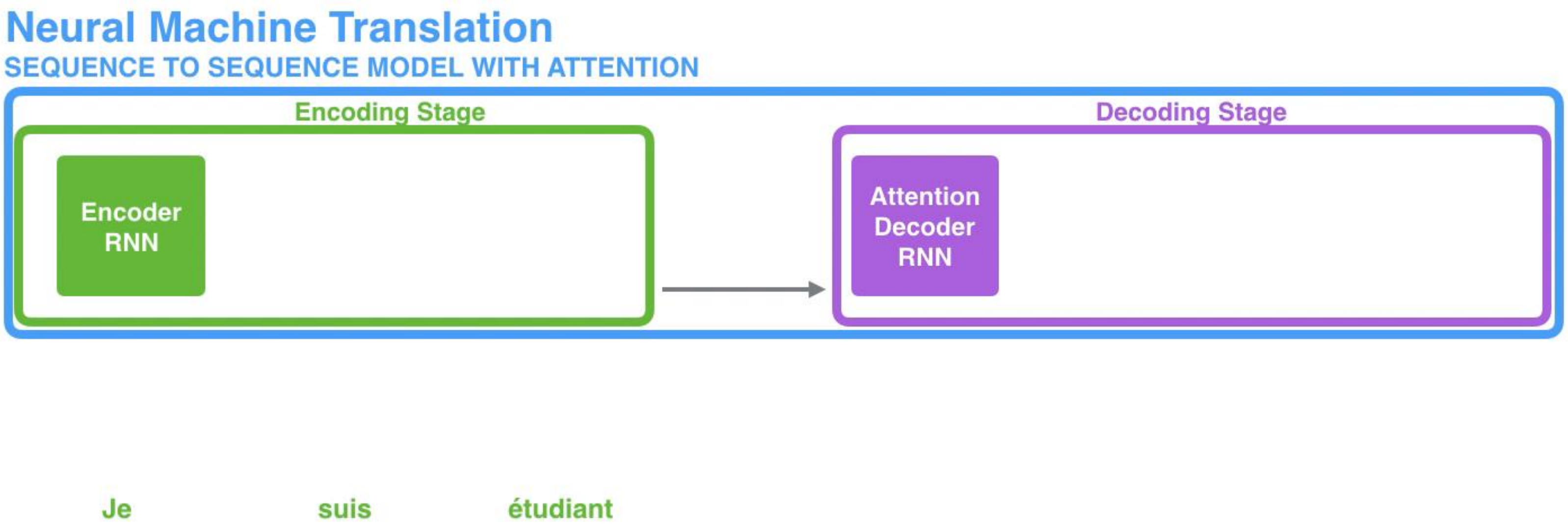
Attention models

- Compute a weighted combination of all the hidden outputs into a single vector



Attention models

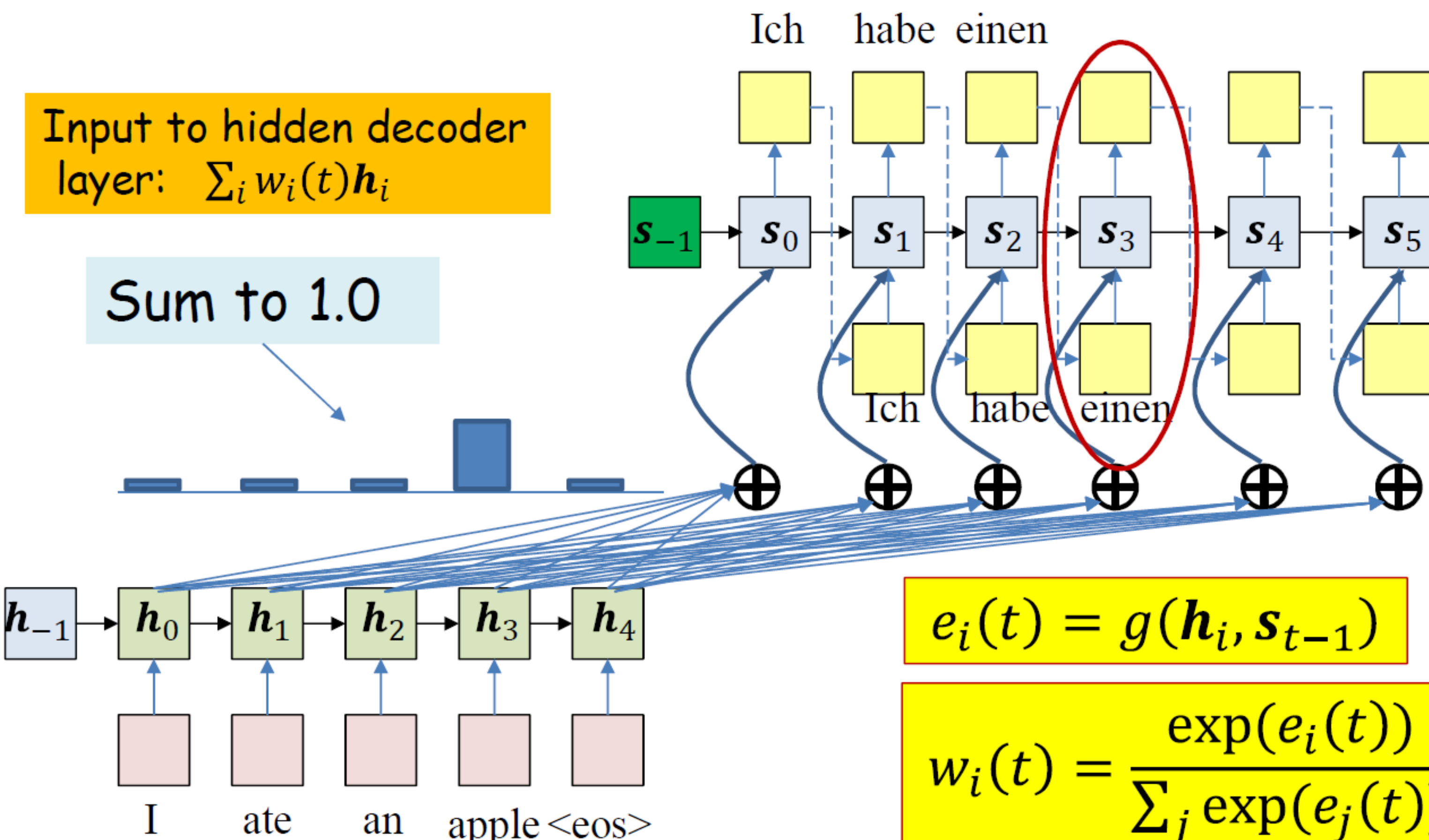
- Compute a weighted combination of all the hidden outputs into a single vector



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention models

- The weights are a distribution over the input
  - A function g() on two hidden states followed by a softmax



$$e_i(t) = g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})$$

$$w_i(t) = \frac{\exp(e_i(t))}{\sum_j \exp(e_j(t))}$$

Attention models

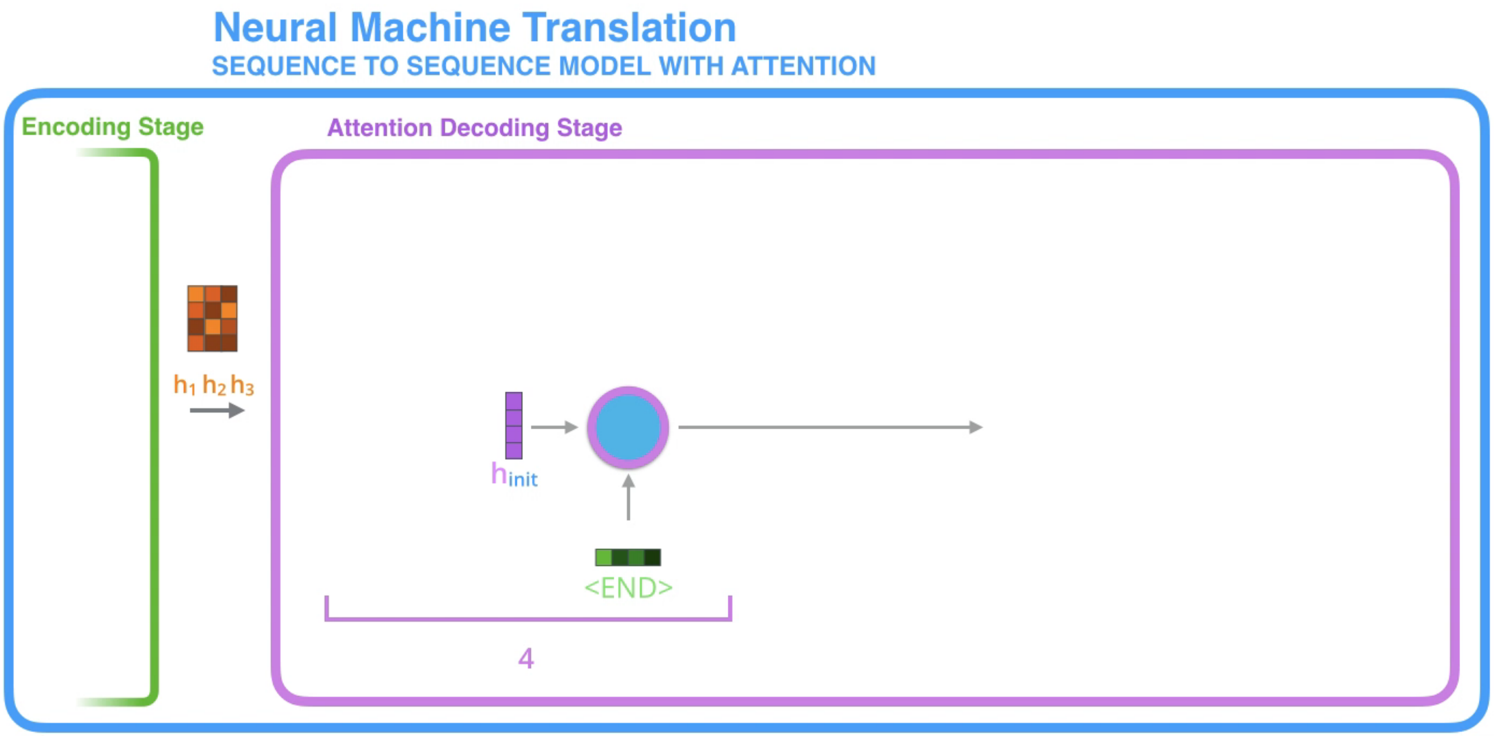
- Compute a weighted combination of all the hidden outputs into a single vector

Attention at time step 4

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention models

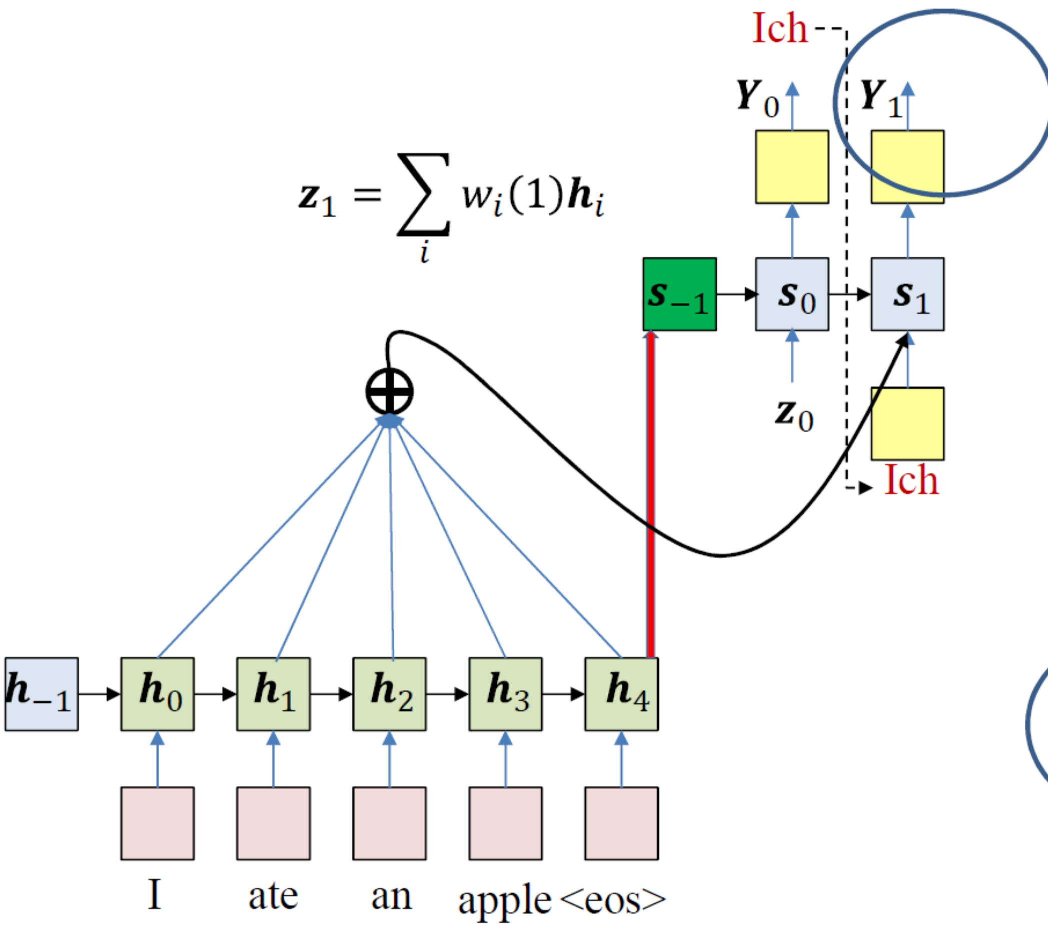
- Compute a weighted combination of all the hidden outputs into a single vector



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

What does the attention learn?

- The key component of this model is the attention weight
  - It captures the relative importance of each position in the input to the current output



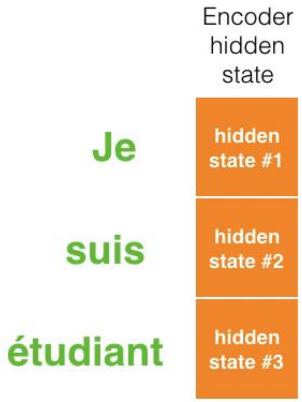
$$g(h_i, s_0) = h_i^T W_g s_0$$

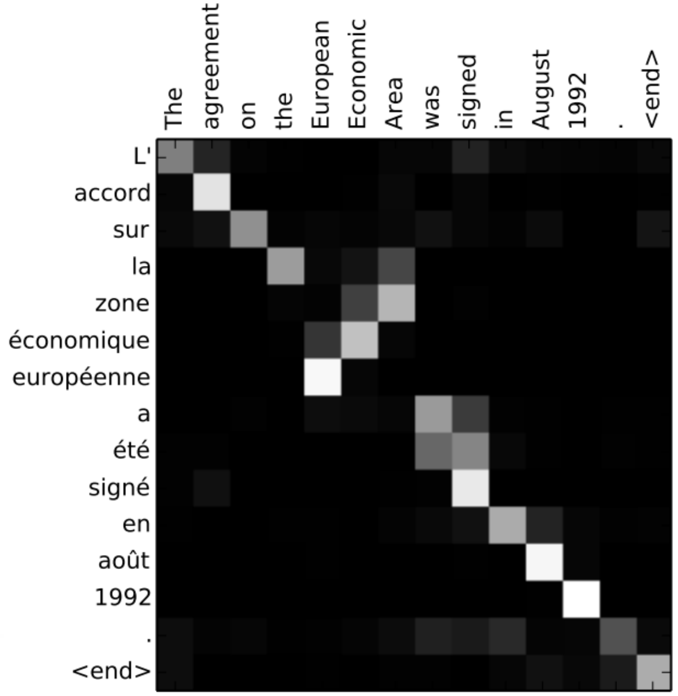
$$e_i(1) = g(h_i, s_0)$$

$$w_i(1) = \frac{\exp(e_i(1))}{\sum_j \exp(e_j(1))}$$

Attention models

- Compute a weighted combination of all the hidden outputs into a single vector

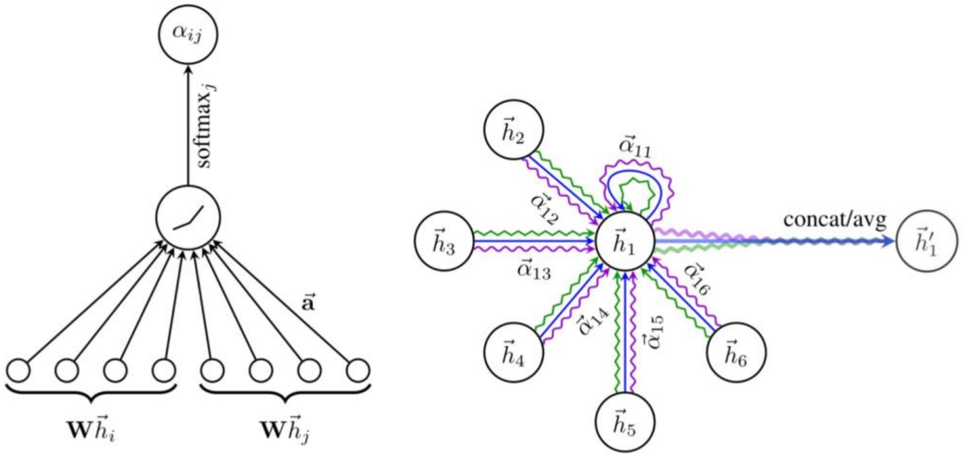




https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Graph Neural Networks (GNNs) with **Attention**

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



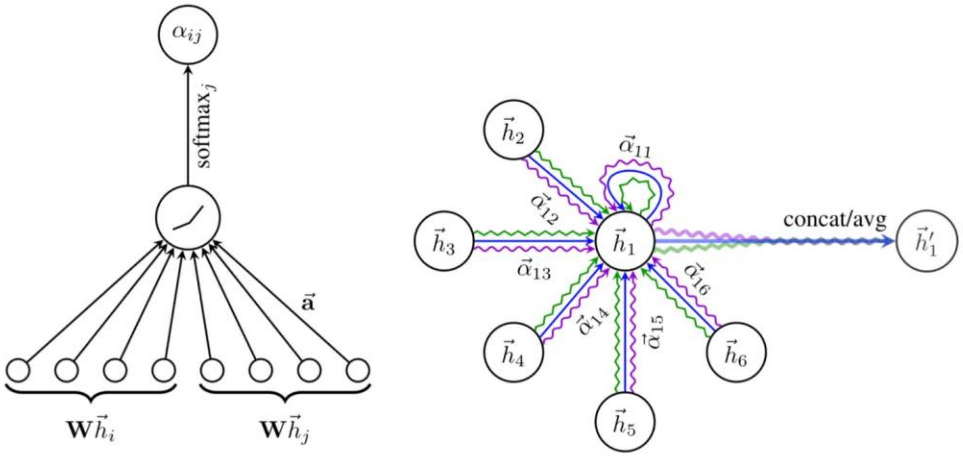
[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

* slide from Thomas Kipf,
University of Amsterdam

Graph Neural Networks (GNNs) with **Attention**

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



[Figure from Veličković et al. (ICLR 2018)]

Pros:

- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

Cons:

- (Most likely) less expressive than GNNs with edge embeddings
- Can be more difficult to optimize

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_k]\right)\right)}$$

Outline

- Recap and motivation
- Attention models in RNNs
- Transformer
- Graph Neural Networks

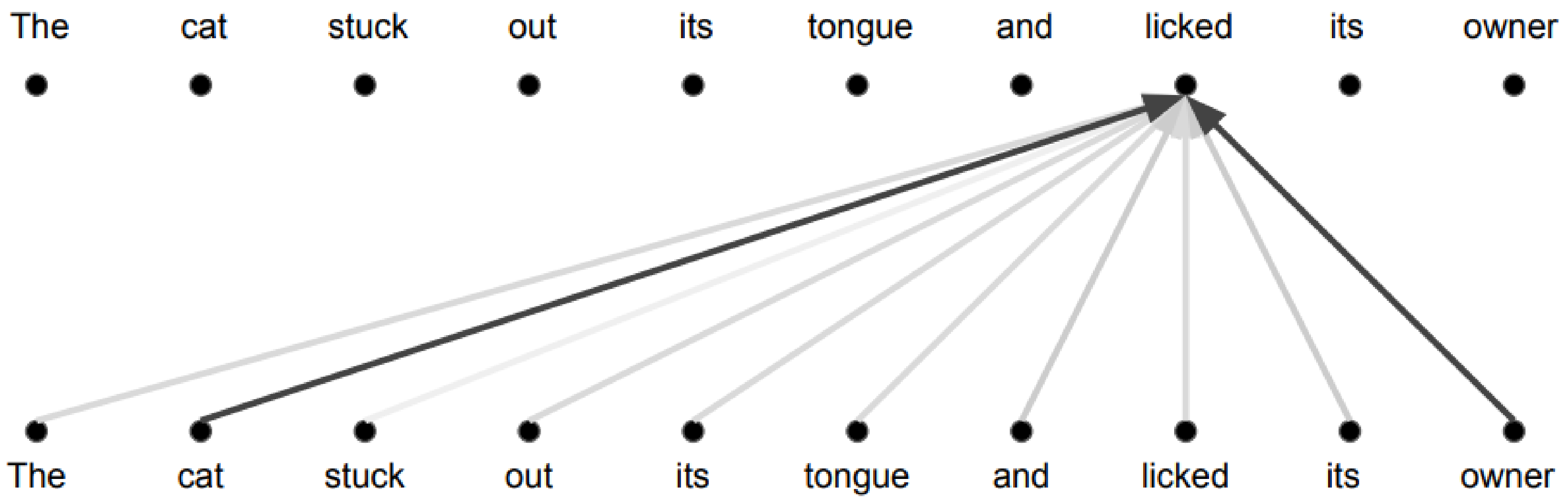
Acknowledgement: Hugo Larochelle's, Mehryar Mohri@NYU's,Yingyu Liang@Princeton's, Bhiksha Raj@CMU's &Feifei Li@Stanford's course notes

Limitations of RNNs

- RNNs involve sequential computation
  - Cannot parallelize = time-consuming
- RNNs “forget” past information
  - LSTM helps to some degree, but not too long
  - No explicit modeling of long and short range dependencies

Self-Attention

- Use (self-)attention as representation?

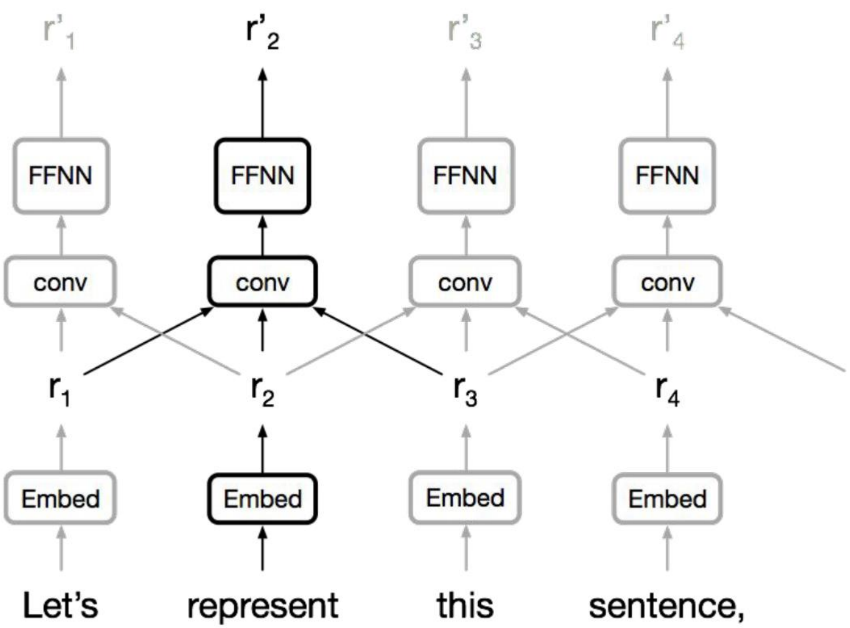


Ashish Vaswani & Anna Huang, Stanford CS224n

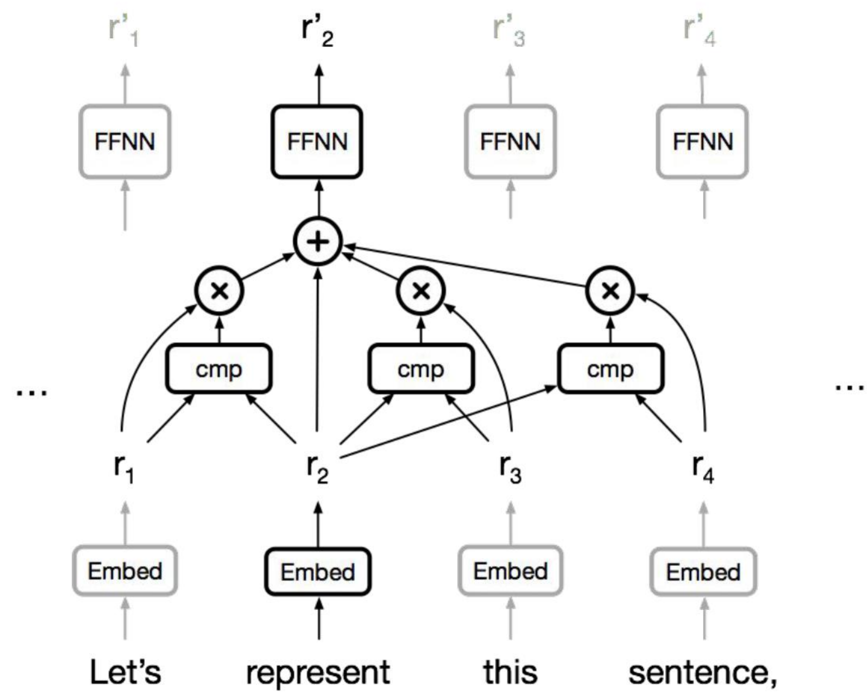
Self-Attention

- Use (self-)attention as representation?

CNN



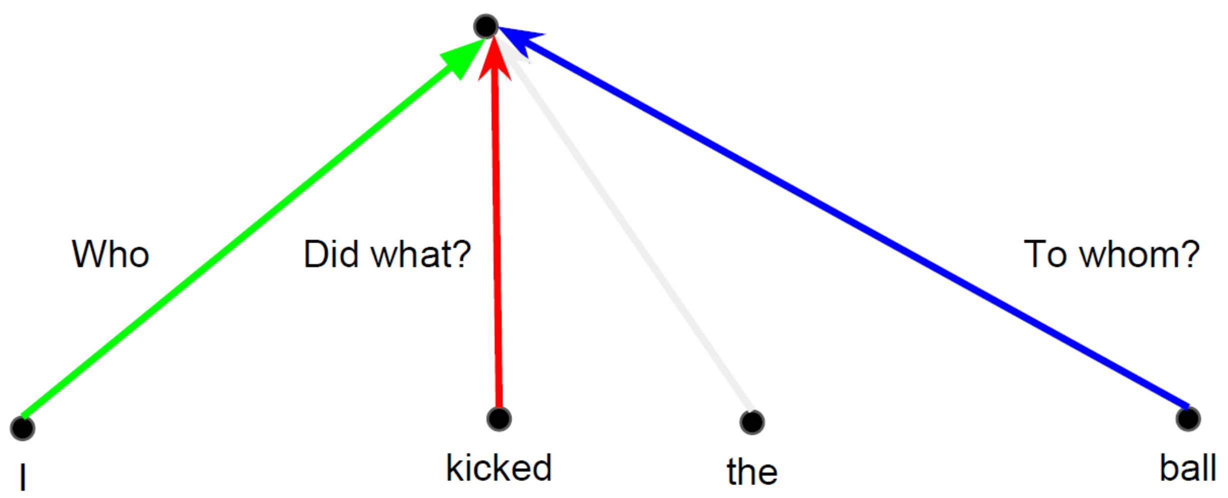
Self-attention



Ashish Vaswani & Anna Huang, Stanford CS224n

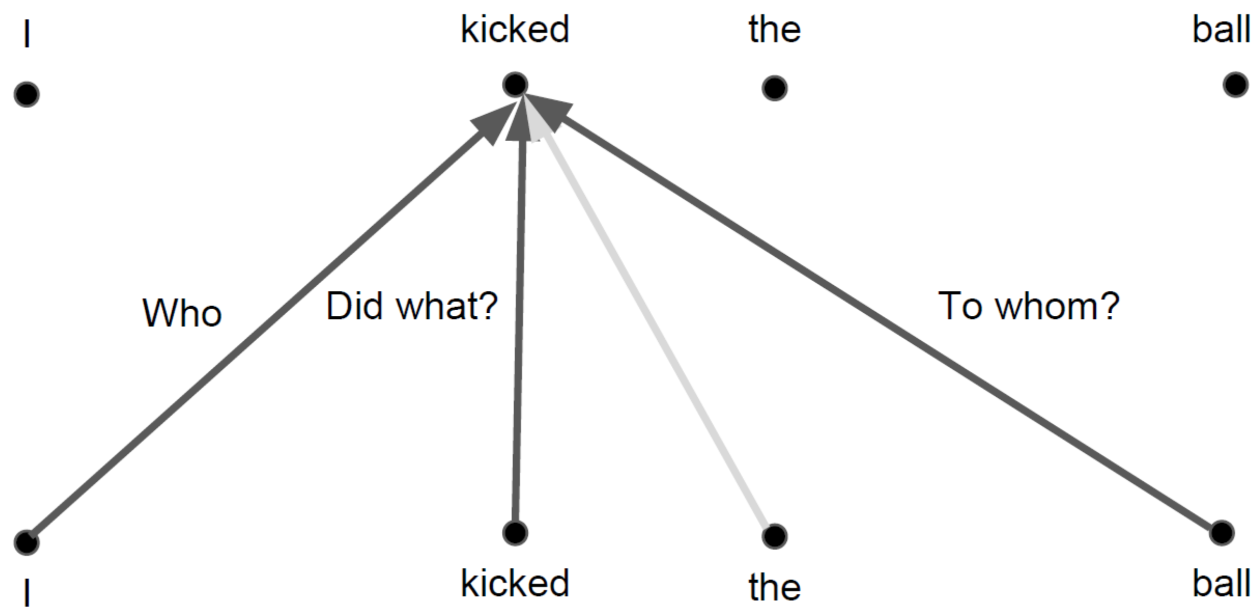
Convolution

- Convolution as representation



Self-Attention

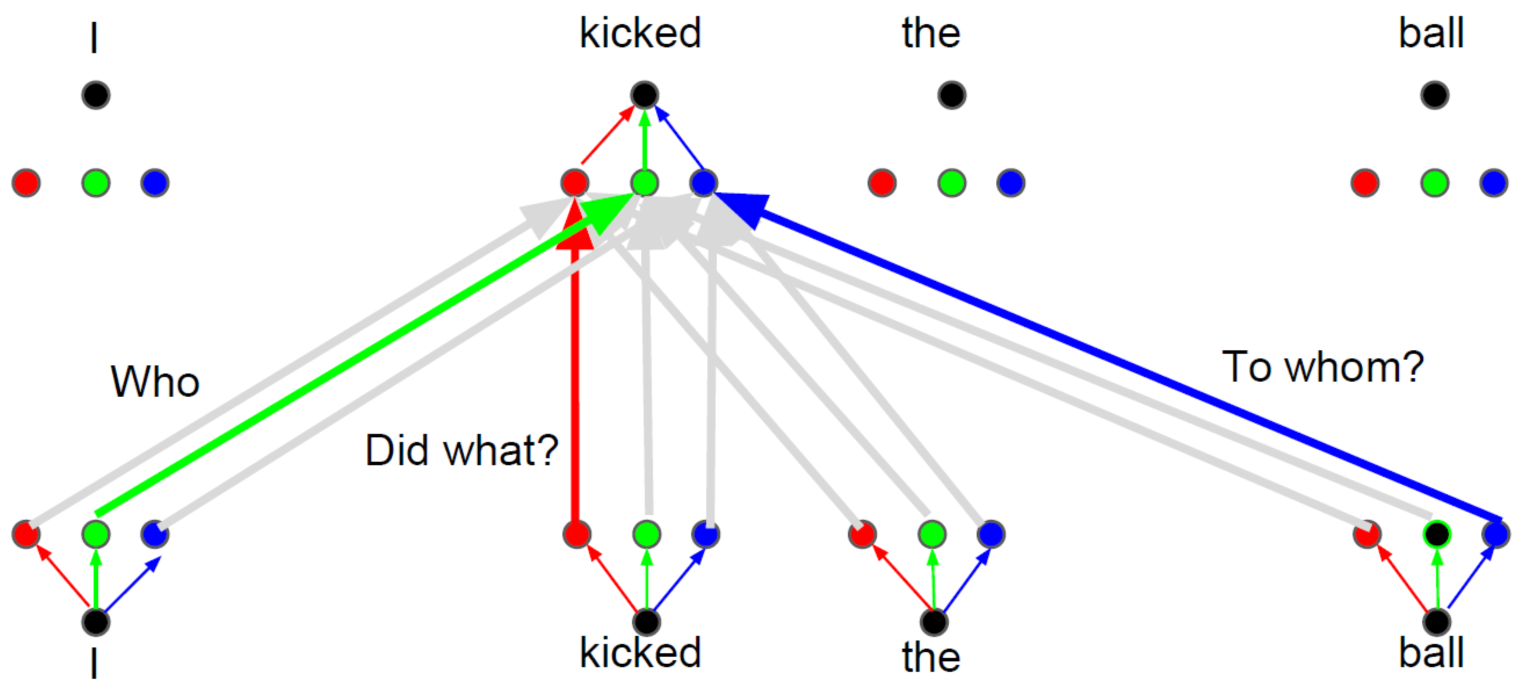
- Self-attention as representation



Ashish Vaswani & Anna Huang, Stanford CS224n

Self-Attention

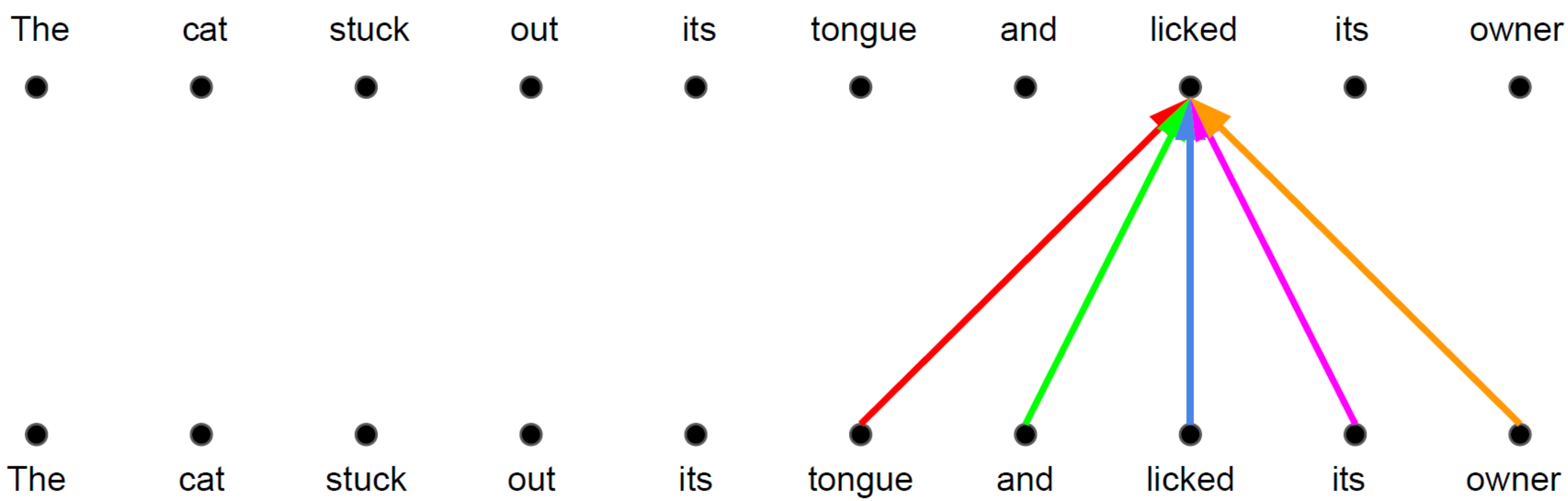
- Parallel attention heads



Ashish Vaswani & Anna Huang, Stanford CS224n

Convolution

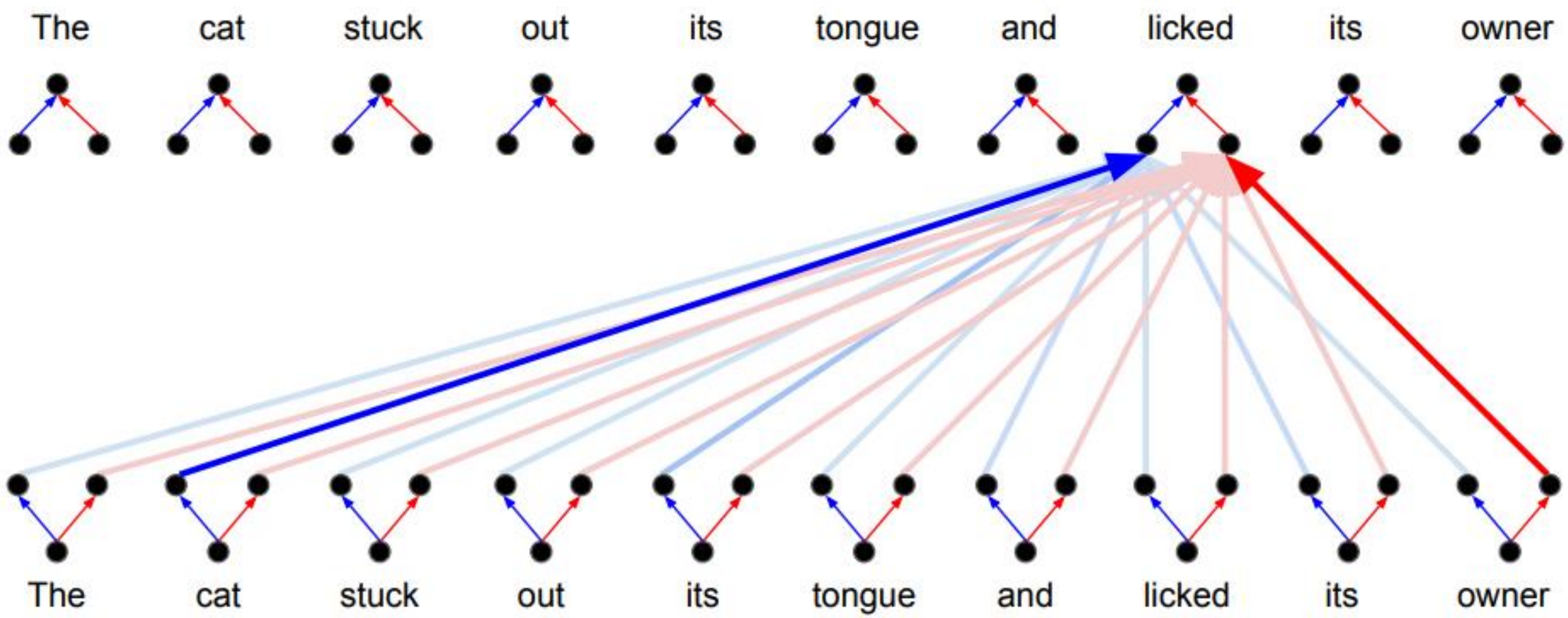
- Different linear transformations by relative position



Ashish Vaswani & Anna Huang, Stanford CS224n

Self-Attention

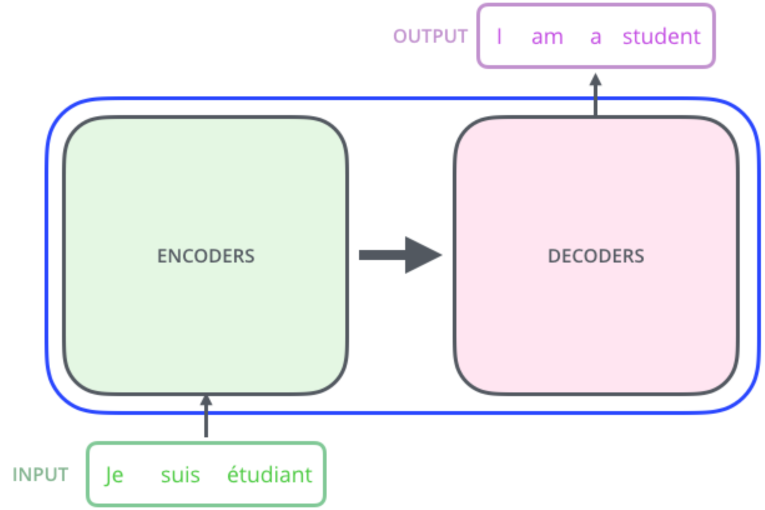
- Multi-head attention
  - Parallel attention layers with different linear transformations on input and output.

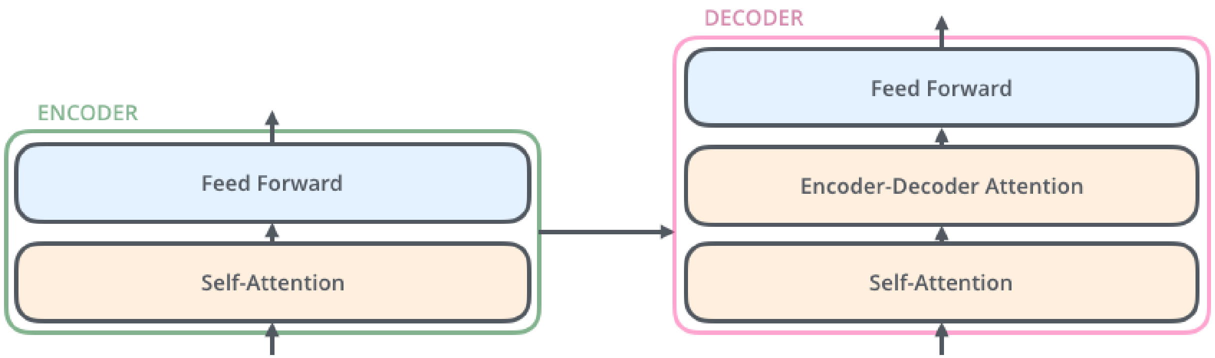


Ashish Vaswani & Anna Huang, Stanford CS224n

The Transformer

- Encoder-decoder network

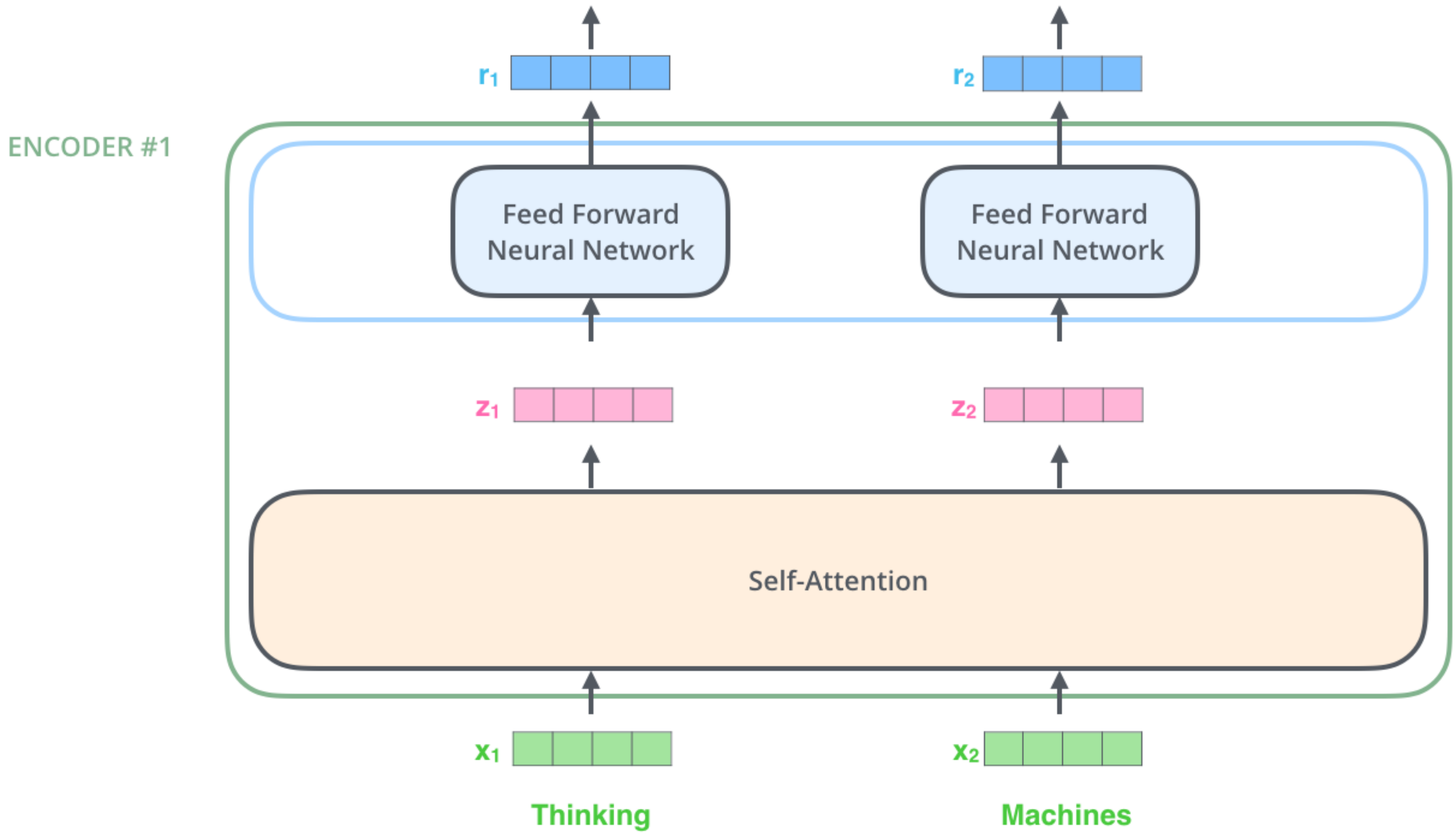




http://jalammar.github.io/illustrated-transformer/

The Transformer

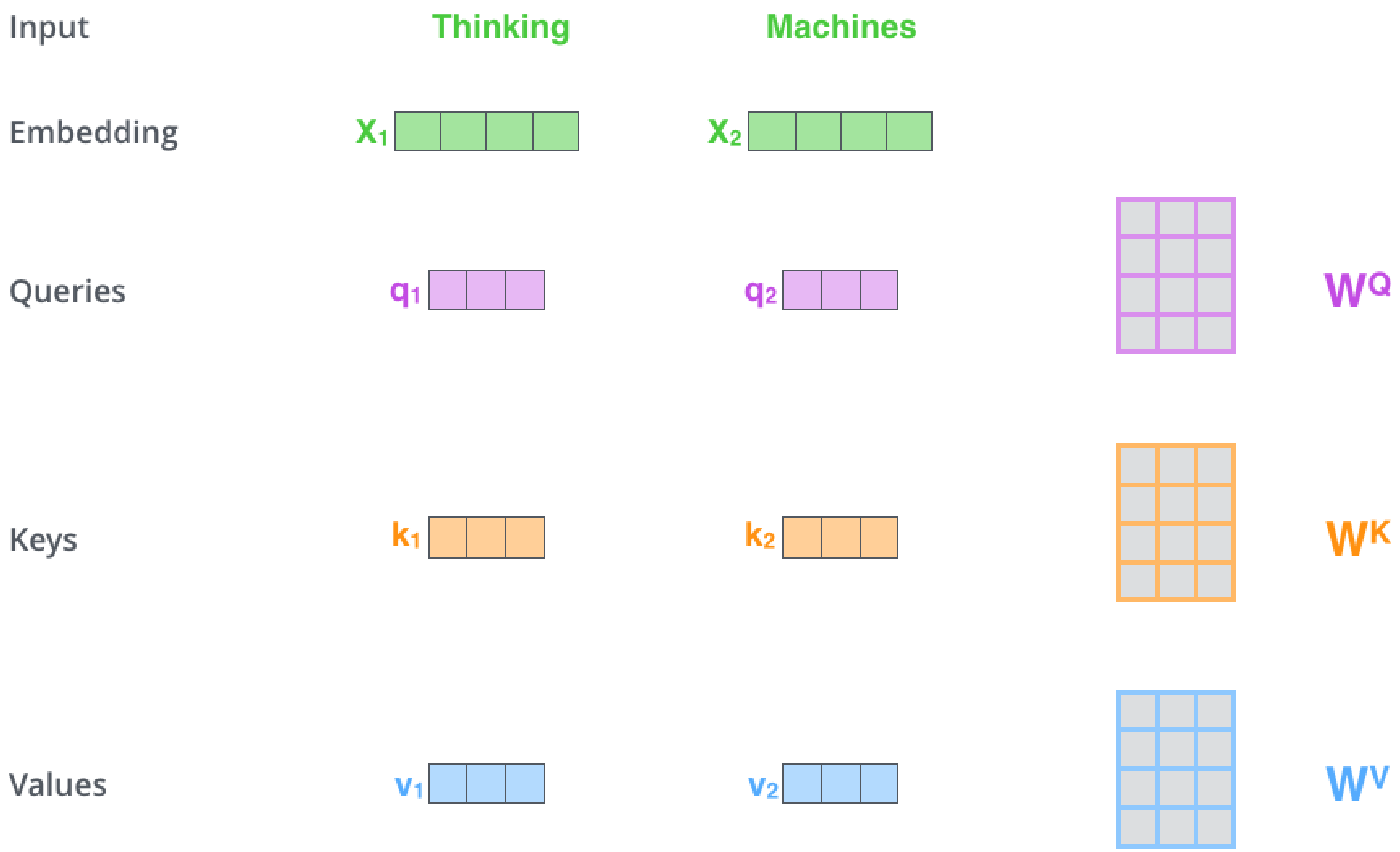
- Encoder network



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

The Transformer

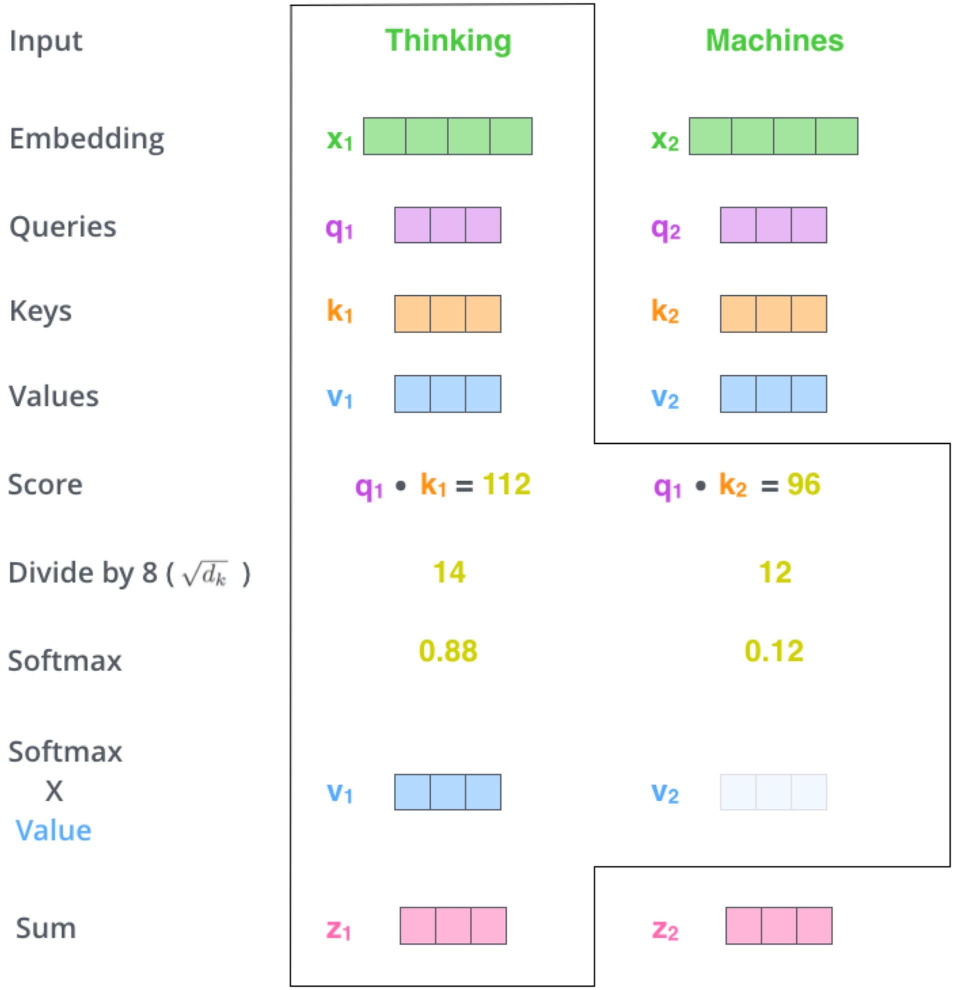
- Self-attention



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

The Transformer

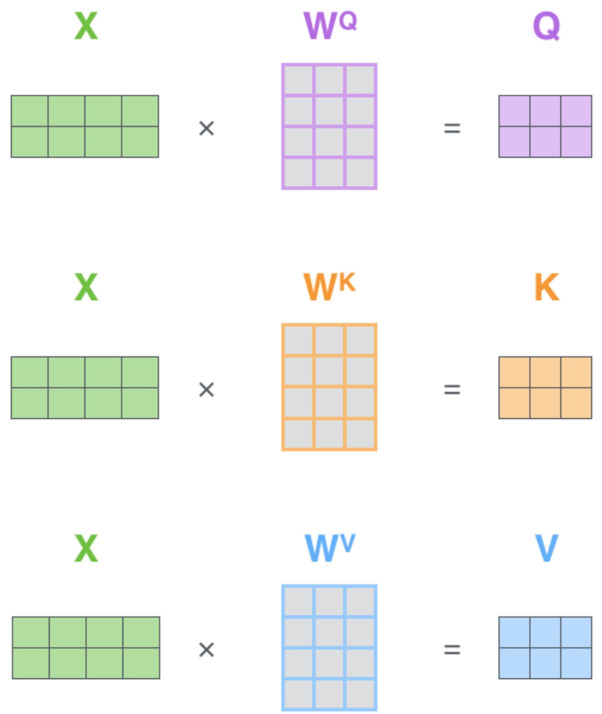
- Self-attention

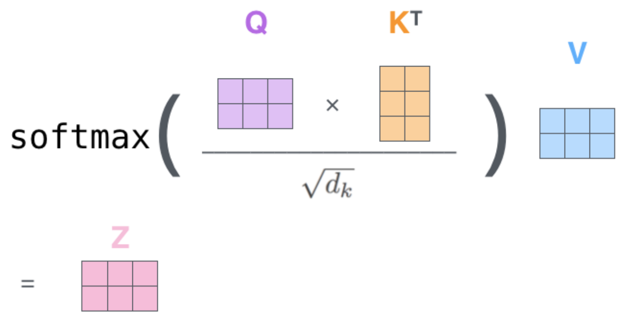


http://jalammar.github.io/illustrated-transformer/

The Transformer

- Self-attention in matrix calculation

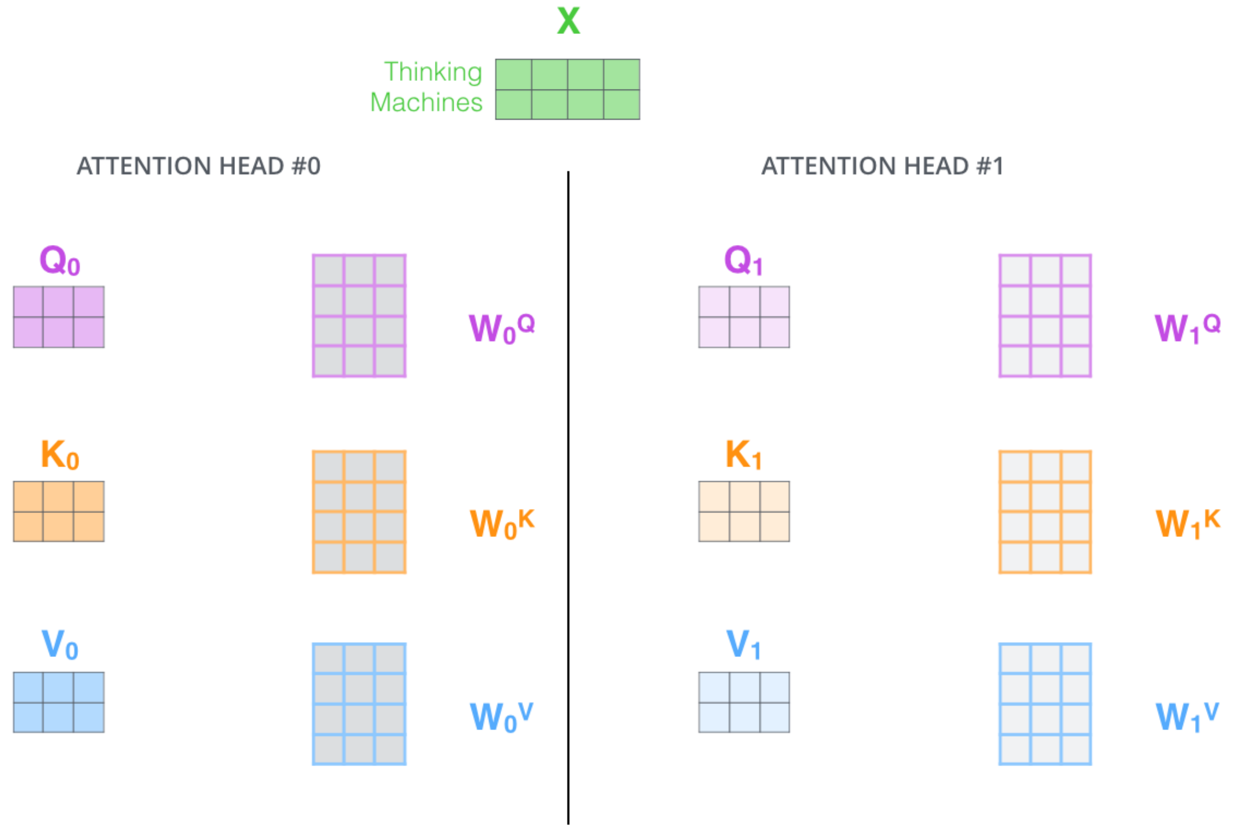




http://jalammar.github.io/illustrated-transformer/

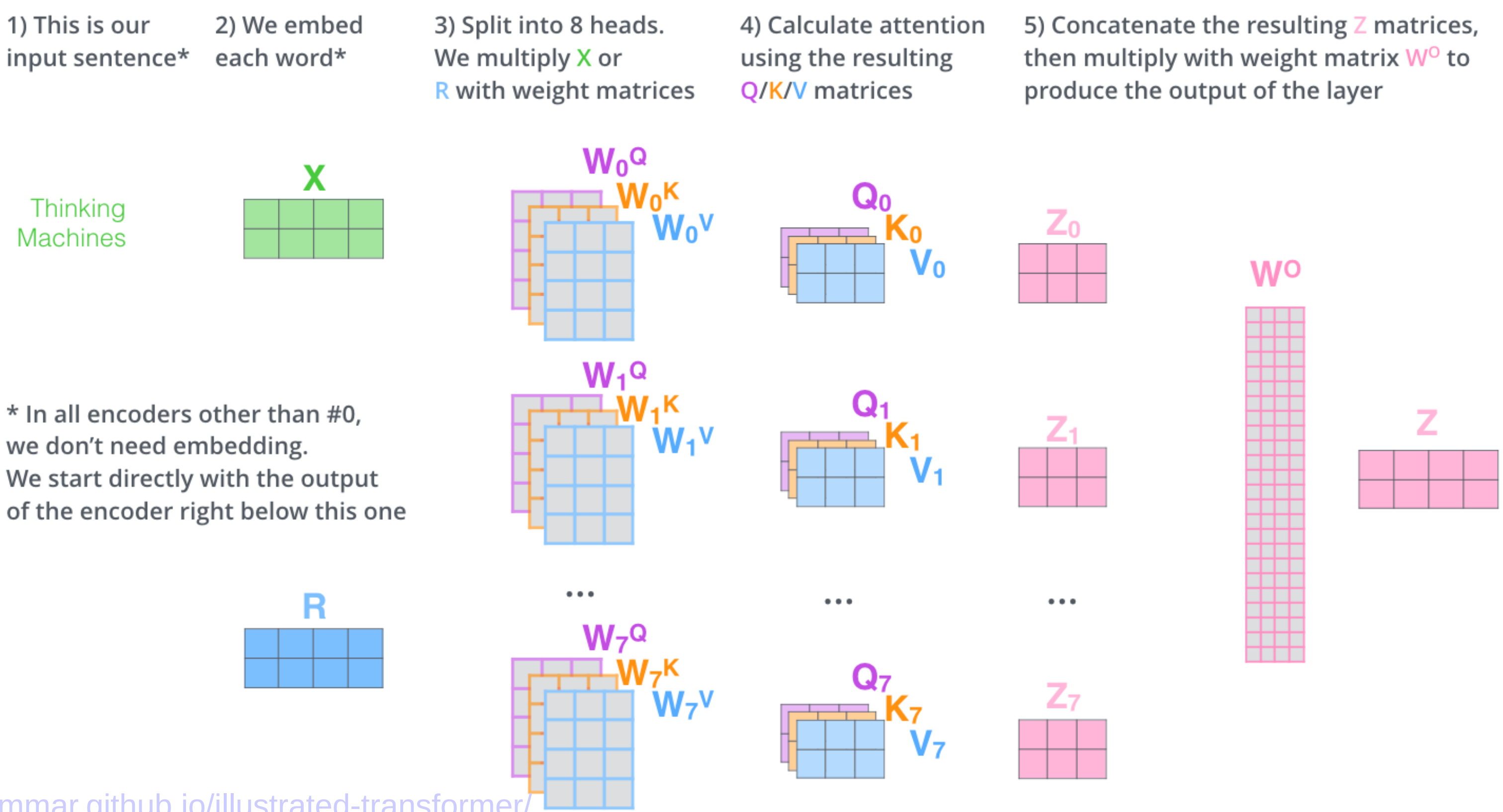
The Transformer

- Multi-head attention
  - Parallel attention layers with different linear transformations on input and output.



Attention

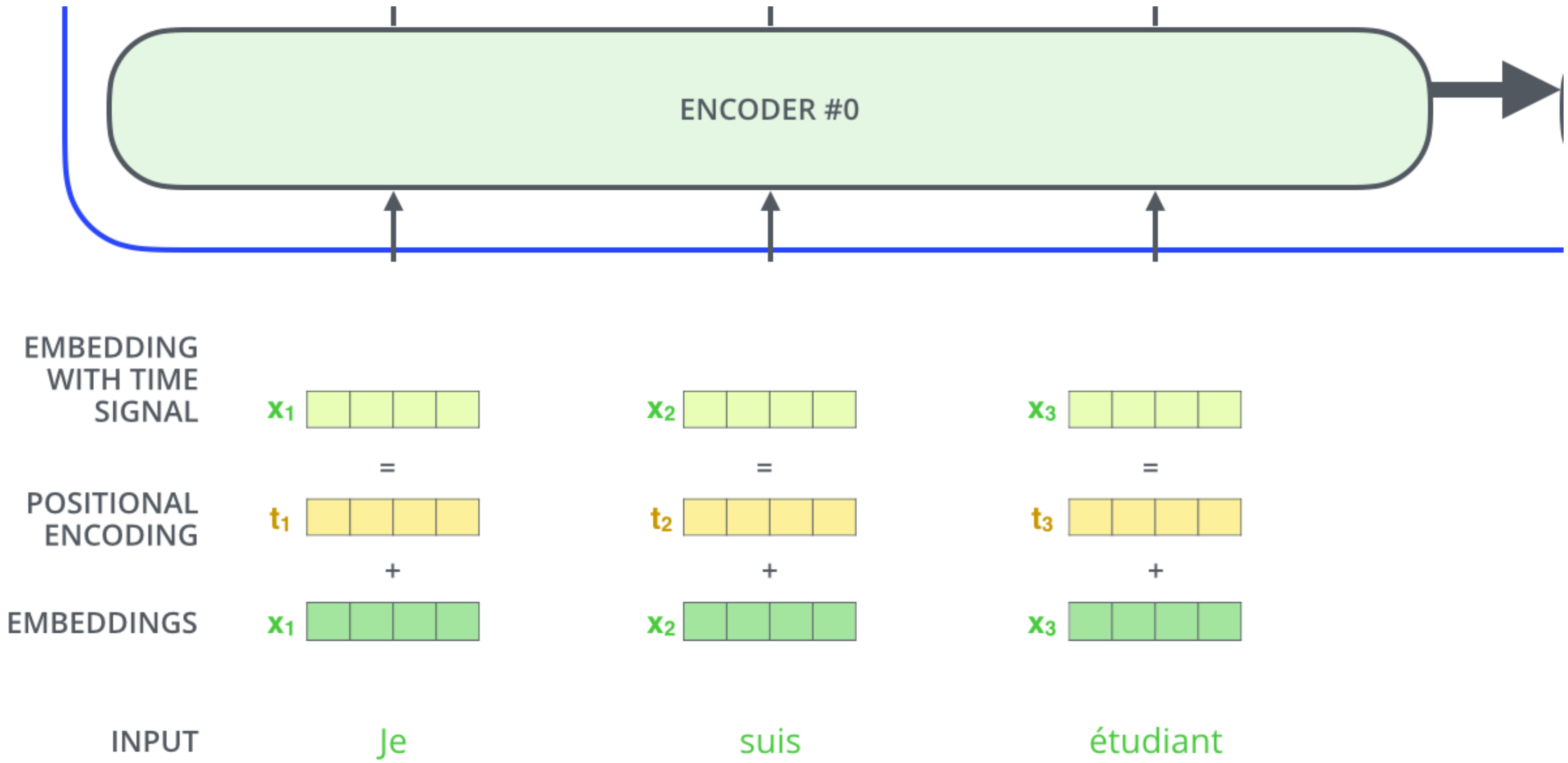
- Multi-head attention
  - Parallel attention layers with different linear transformations on input and output.



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

The Transformer

- Encoder network: the order information
  - Positional encoding



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

The Transformer

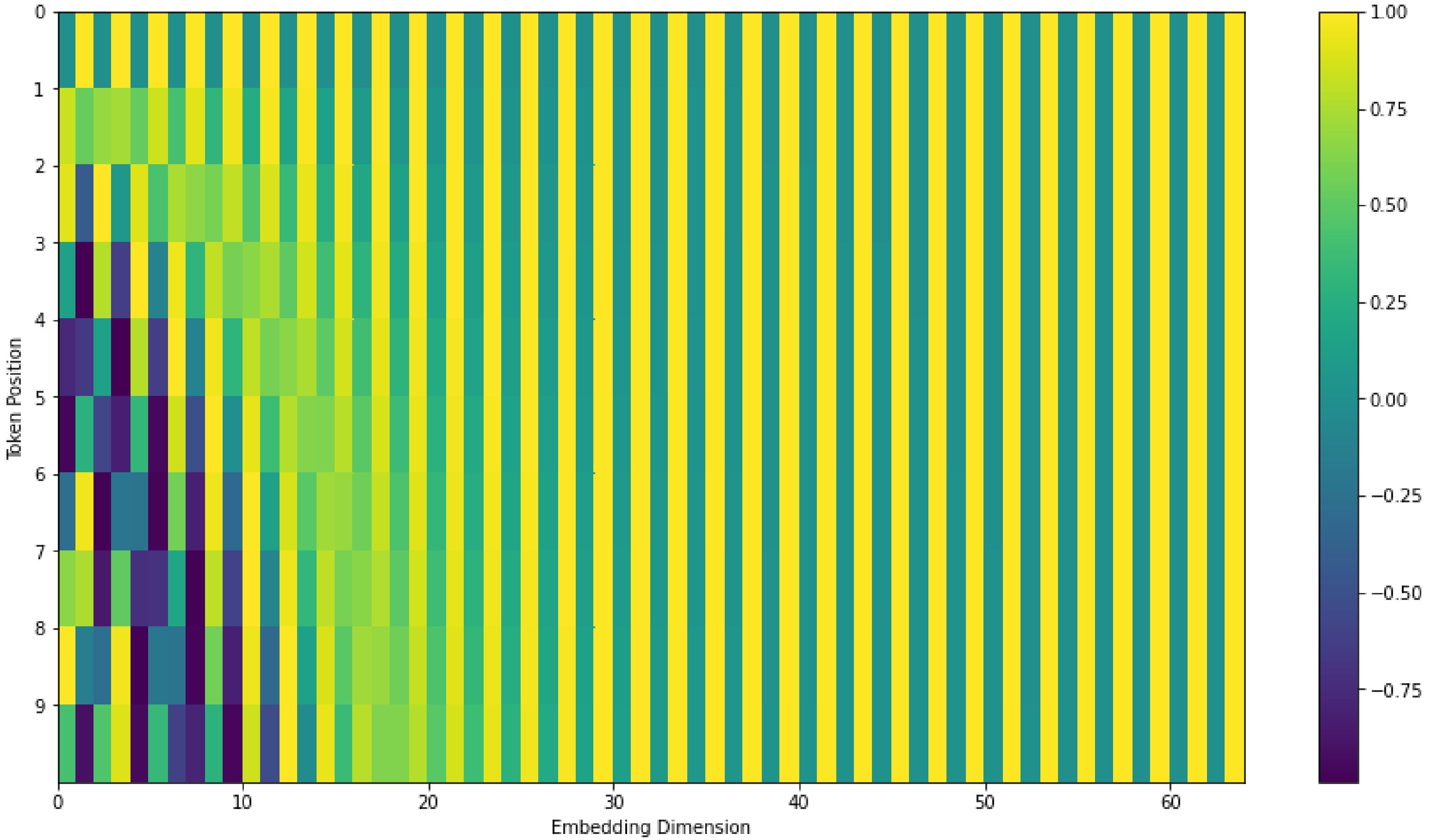
- Positional encoding
  - Sinusoid
    - Can extrapolate beyond max. sequence length at test-time
    - Represent periodicity of positions: a continuous way of binary encoding of position

$$\mathrm{PE}_{pos,2i} = sin(pos/10000^{2i/d_{emb}}),$$
$$\mathrm{PE}_{pos,2i+1} = cos(pos/10000^{2i/d_{emb}}),$$

  - Learned
    - Rather straightforward
    - Cannot extrapolate

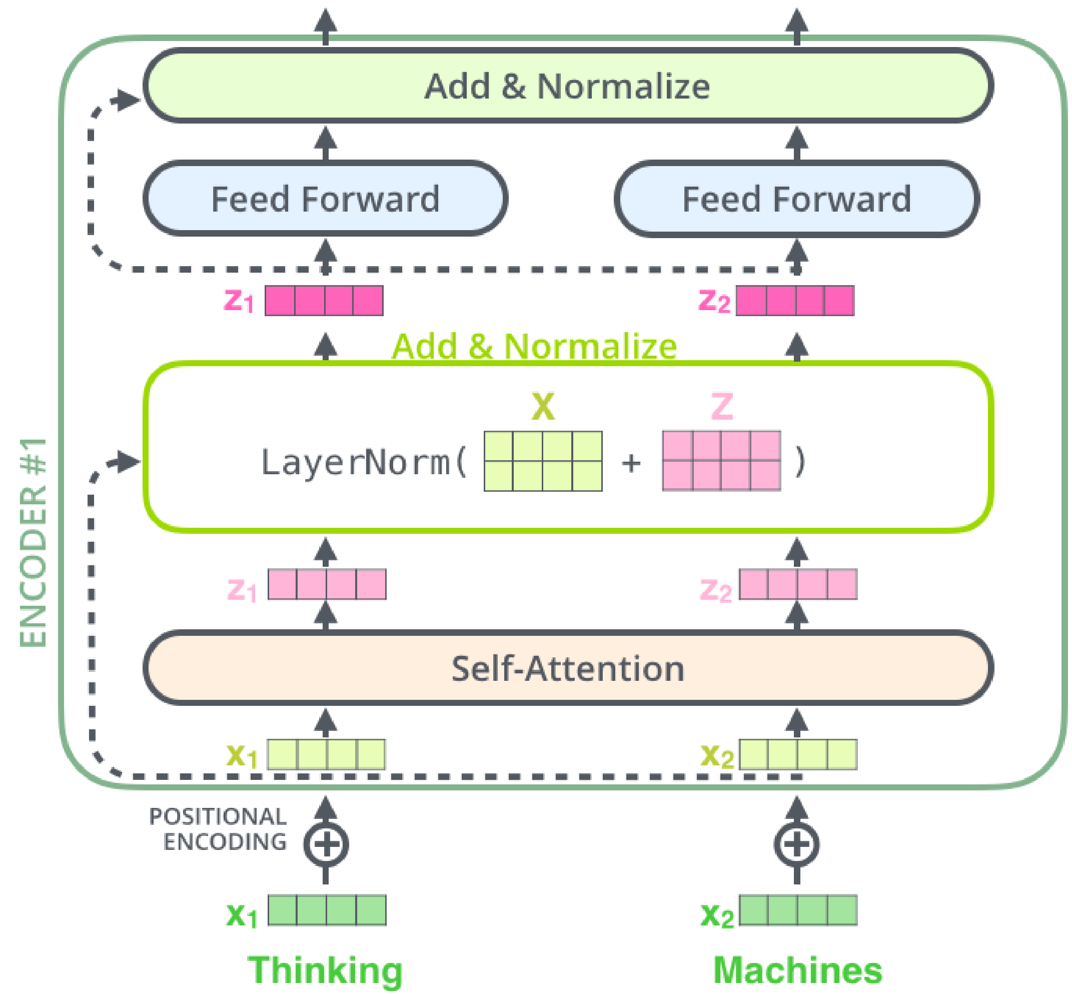
The Transformer

- Positional encoding
  - Sinusoid



The Transformer

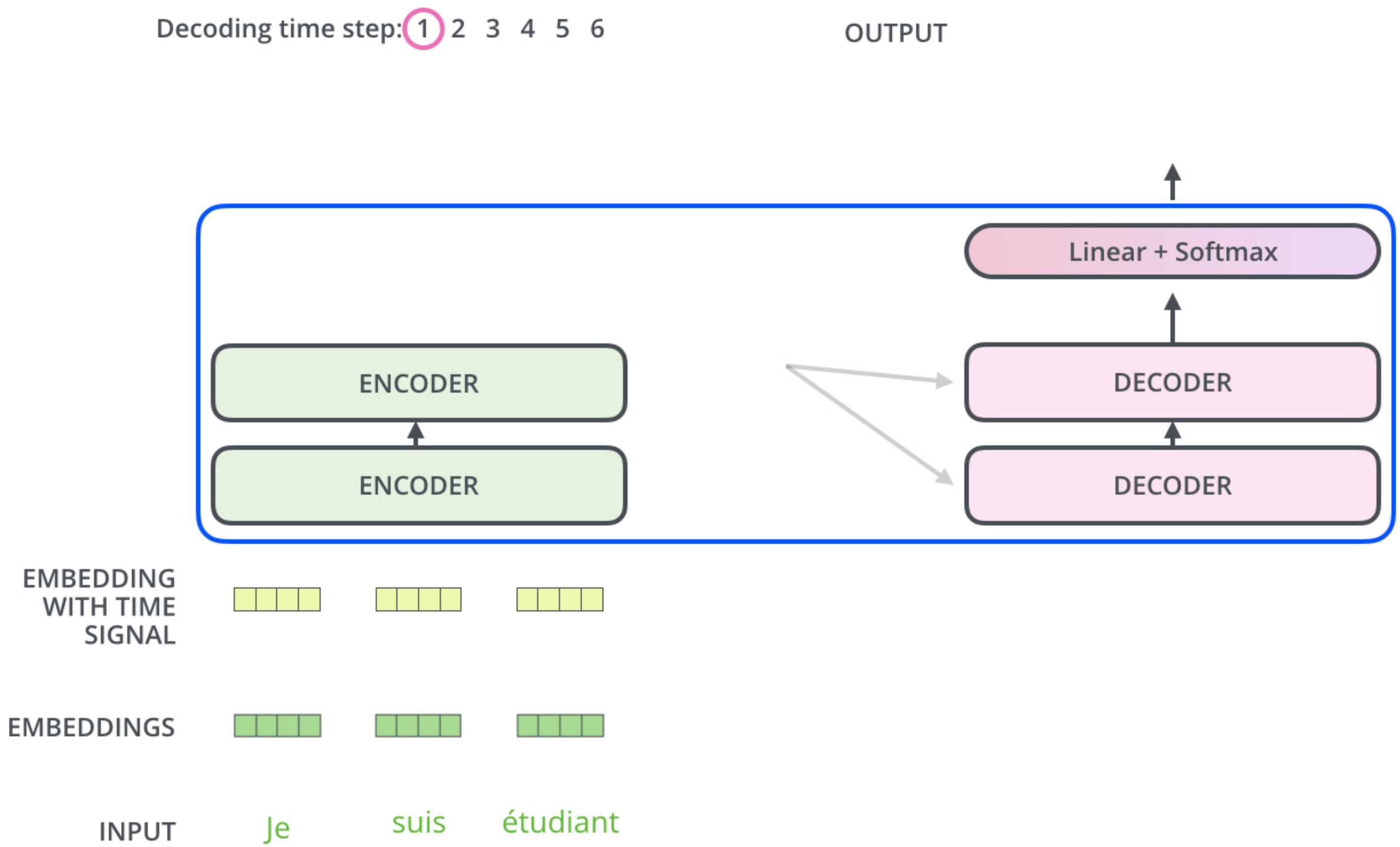
- Residual connections



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

The Transformer

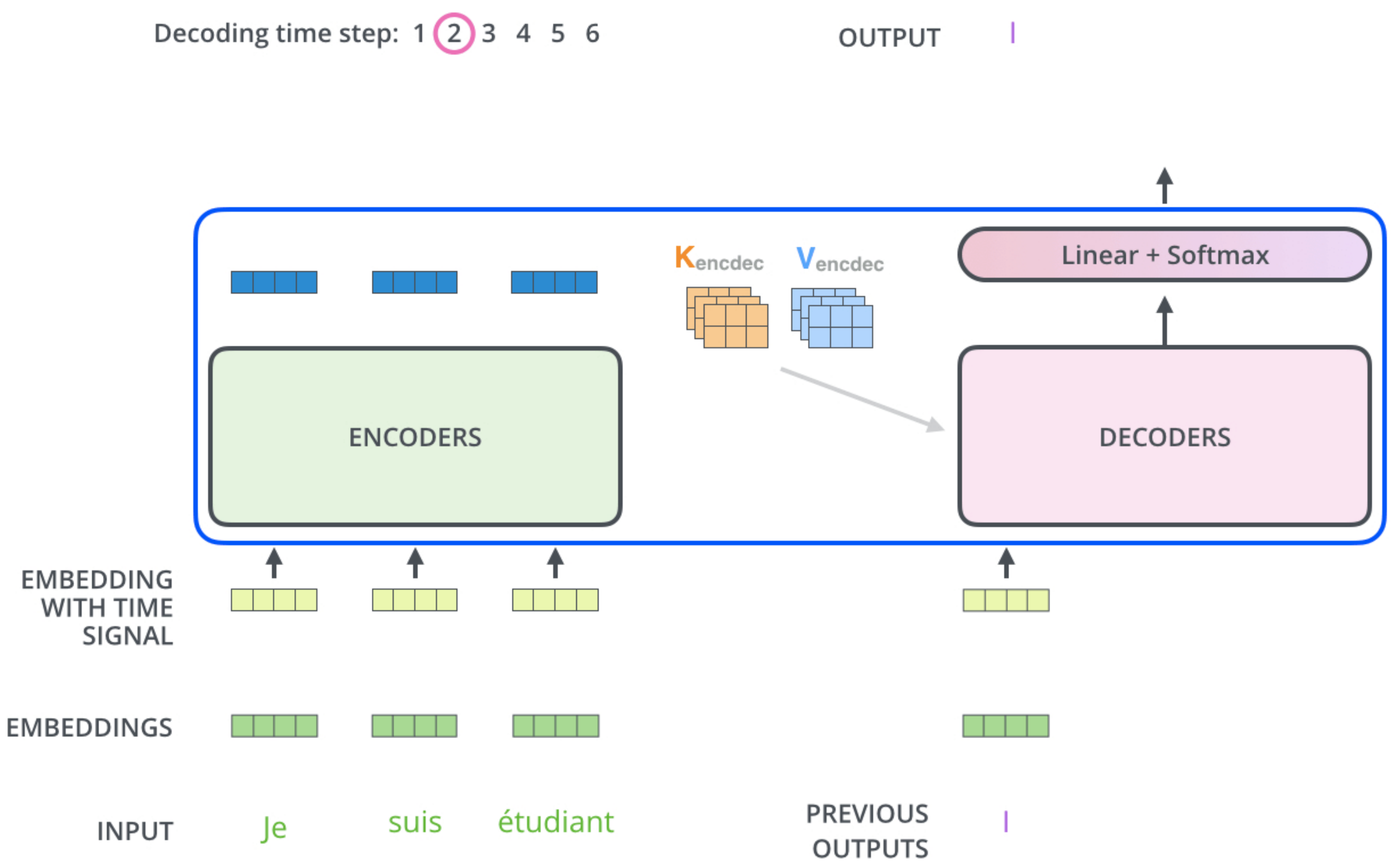
- Decoder network



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

The Transformer

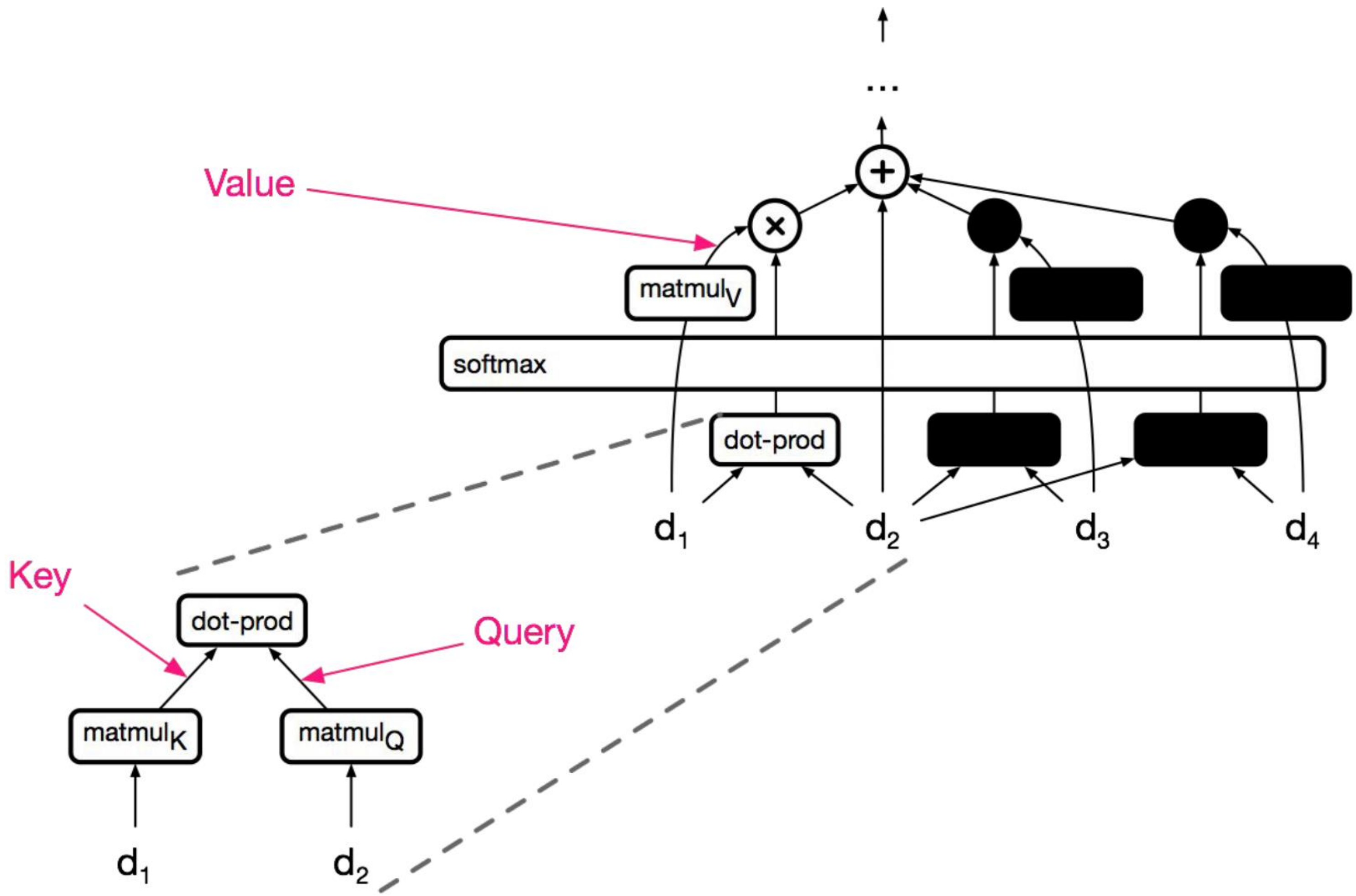
- Decoder network



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

The Transformer

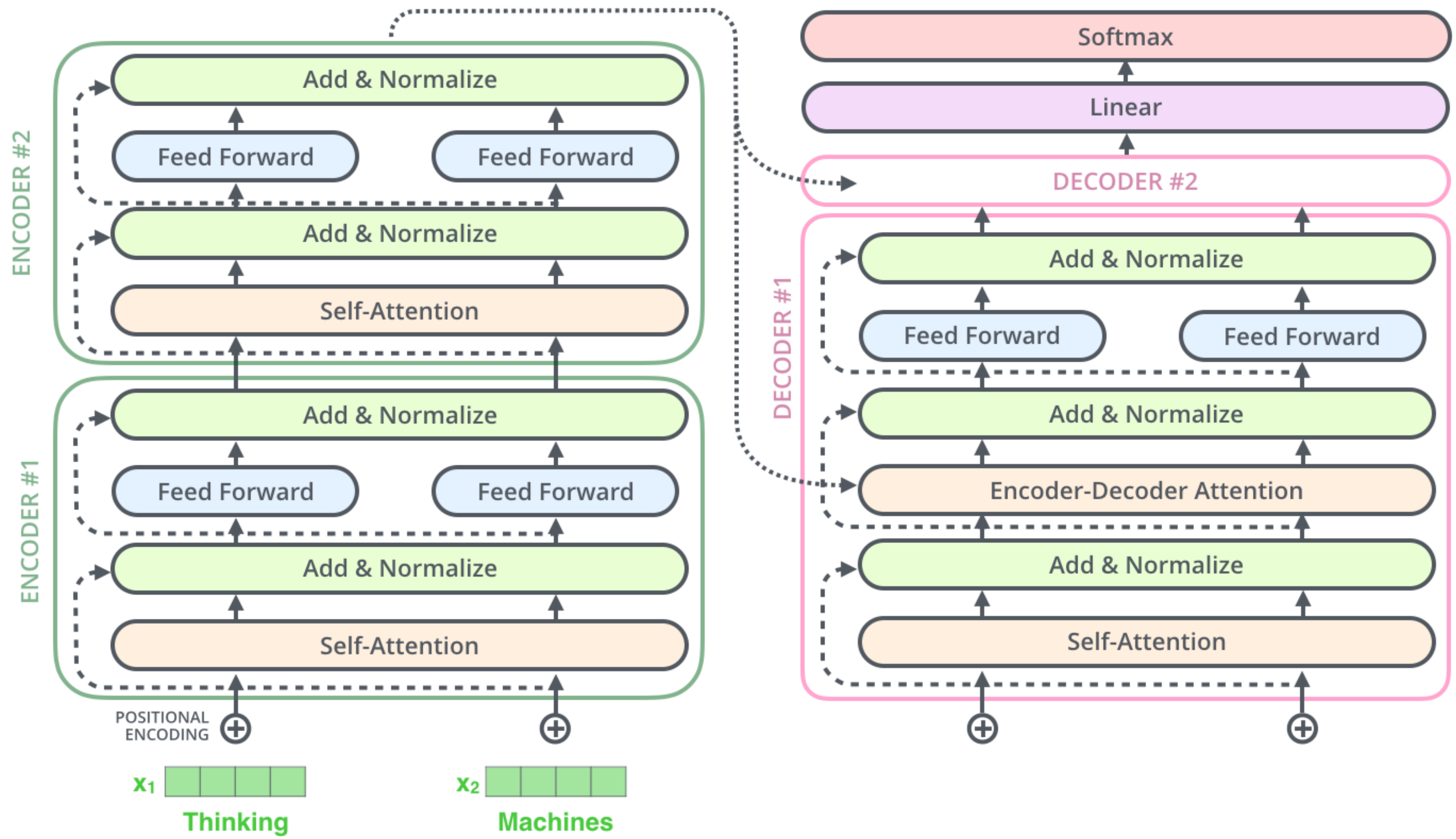
- Decoder network



Ashish Vaswani & Anna Huang, Stanford CS224n

The Transformer

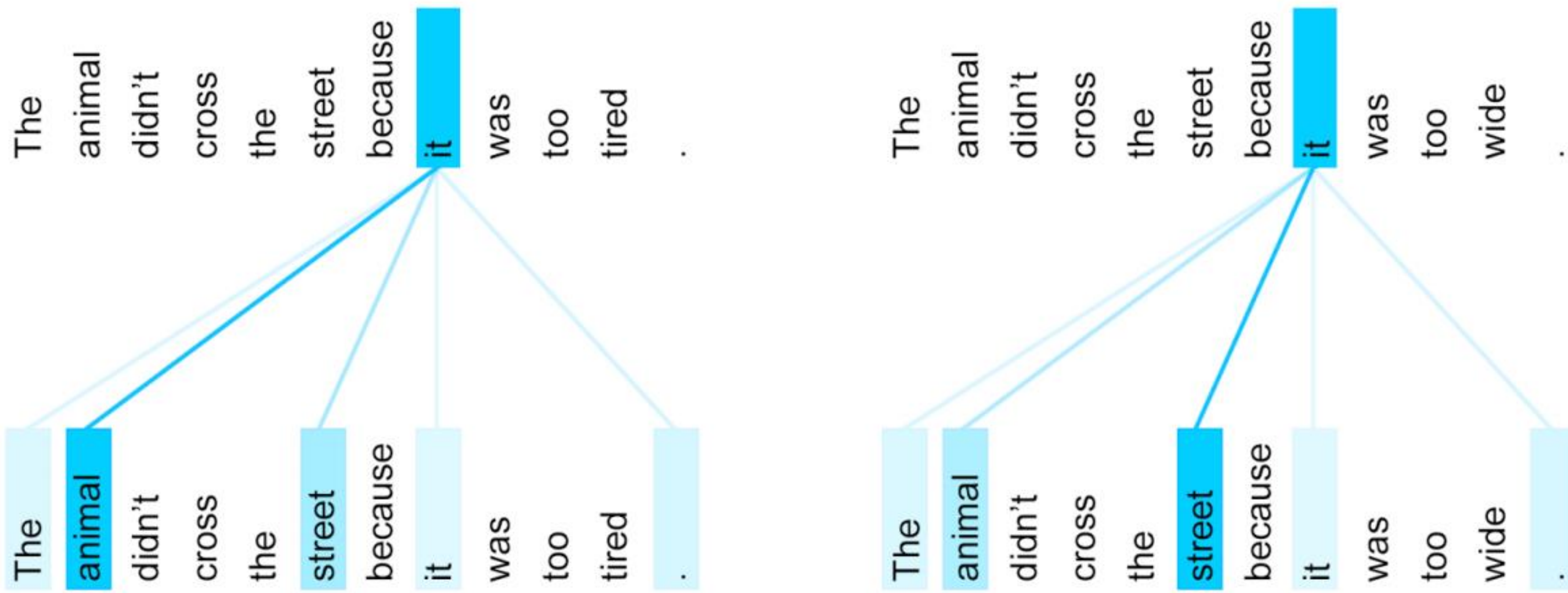
- Overall architecture



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

The Transformer

- Self-attention example
  - Self-attention layers learnt "it" could refer to different entities in the different contexts



Visualization of the 5th to 6th self-attention layer in the encoder

The Transformer

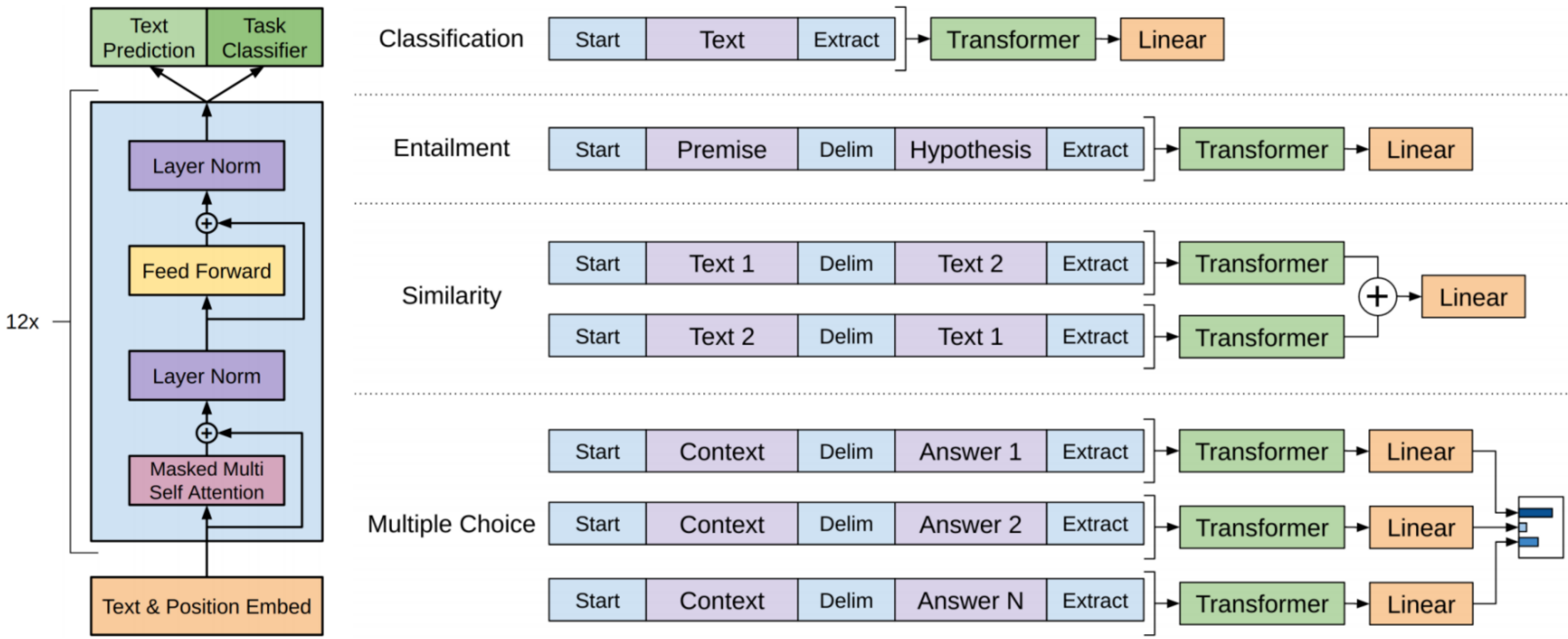
- Attention is all you need
  - BLEU scores of state-of-the-art models on the WMT14 English-to-German translation task

Translation Model	Training time	BLEU (diff. from MOSES)
Transformer (large)	3 days on 8 GPU	**28.4 (+7.8)**
Transformer (small)	1 day on 1 GPU	24.9 (+4.3)
GNMT + Mixture of Experts	1 day on 64 GPUs	26.0 (+5.4)
ConvS2S (FB)	18 days on 1 GPU	25.1 (+4.5)
GNMT	1 day on 96 GPUs	24.6 (+4.0)

Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.

Transformer Language Pre-training

- We can pre-train a language model for NLP tasks.
  - The pre-trained model is then fine-tuned on target tasks.



Radford, Alec, et al. "Improving Language Understanding by Generative Pre-Training." 2018

Transformer Language Pre-training

- We can pre-train a language model for NLP tasks.
  - Generated texts

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

Radford, Alec, et al. "Language Models are Unsupervised Multitask Learners." 2019

Summary

- Self-Attention
  - Constant ‘path length’ between any two positions.
  - Unbounded memory.
  - Trivial to parallelize (per layer).
  - Models Self-Similarity.
  - Relative attention provides expressive timing, equivariance, and extends naturally to graphs.