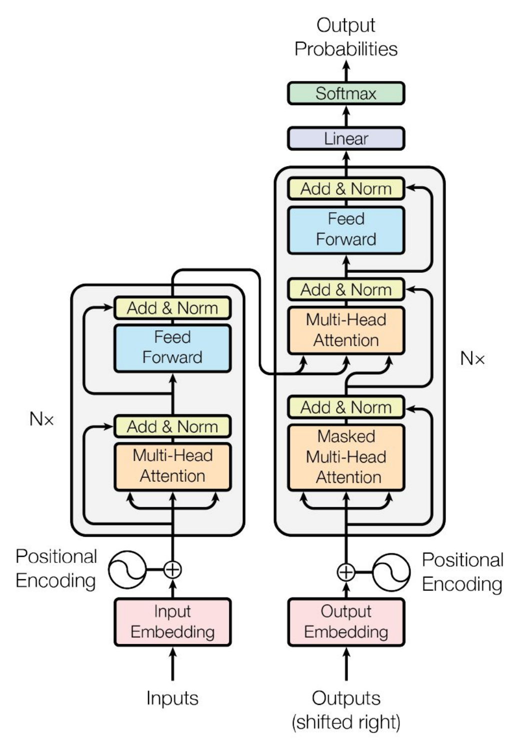
Deep dive into Transformers

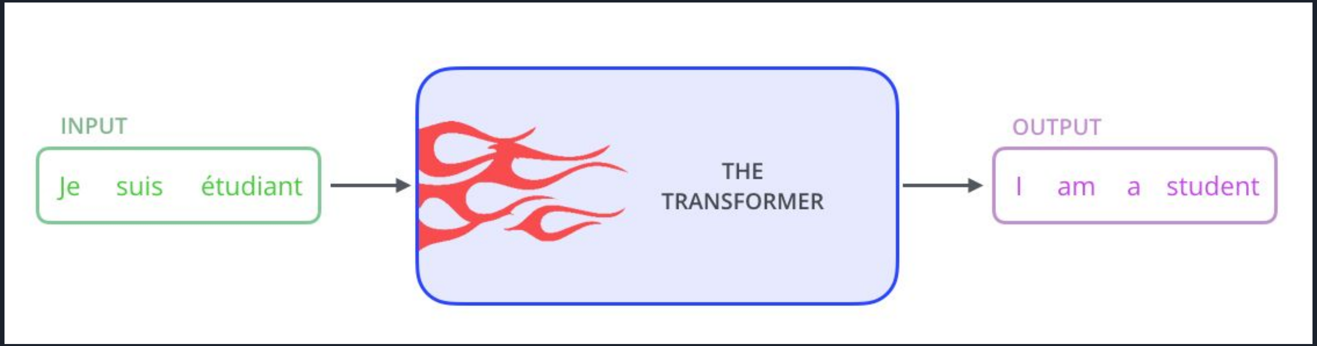
Amirhassan Amirmahani - Gholamreza Dar

Transformer Architecture

- Transformer architecture
- Encoder and Decoder stack
- Self-attention
- Multihead attention
- Positional encoding
- Residual connections
- Encoder - Decoder Attention
- Encoder Attention Vs Decoder Attention (Masking)
- The final Linear and Softmax layers
- Training
    - One Hot Encoding
    - Loss Function
    - Another Training Method

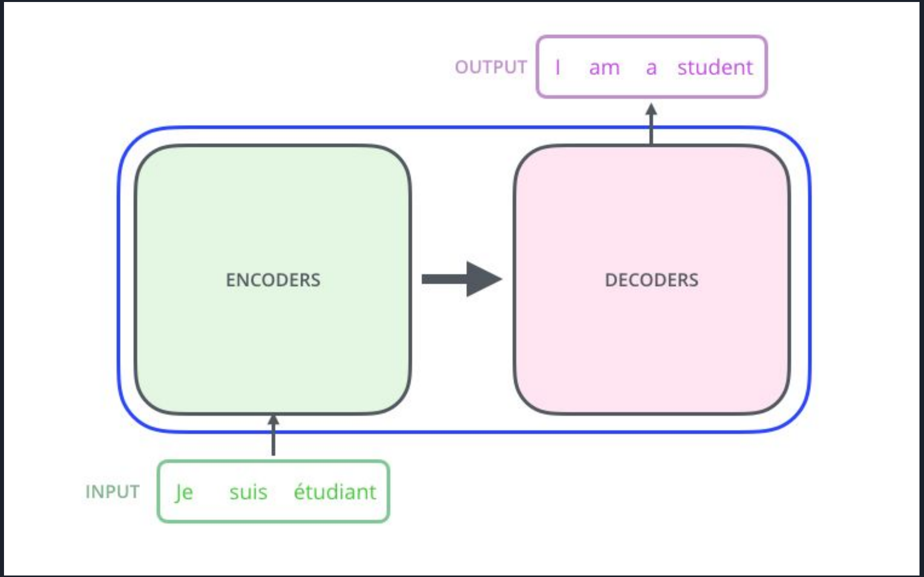


Transformer as a black box



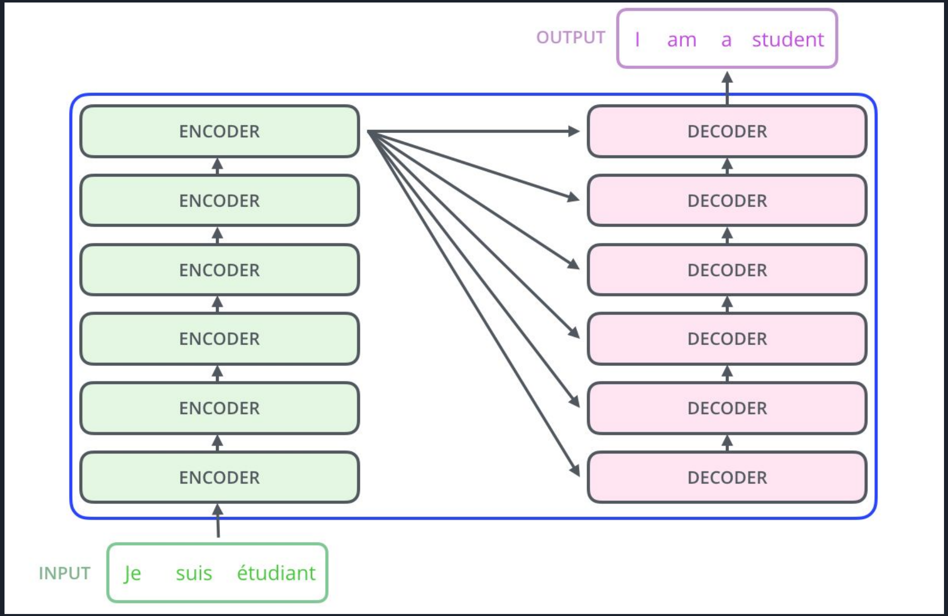
http://jalammar.github.io/illustrated-transformer/

Inside a Transformer



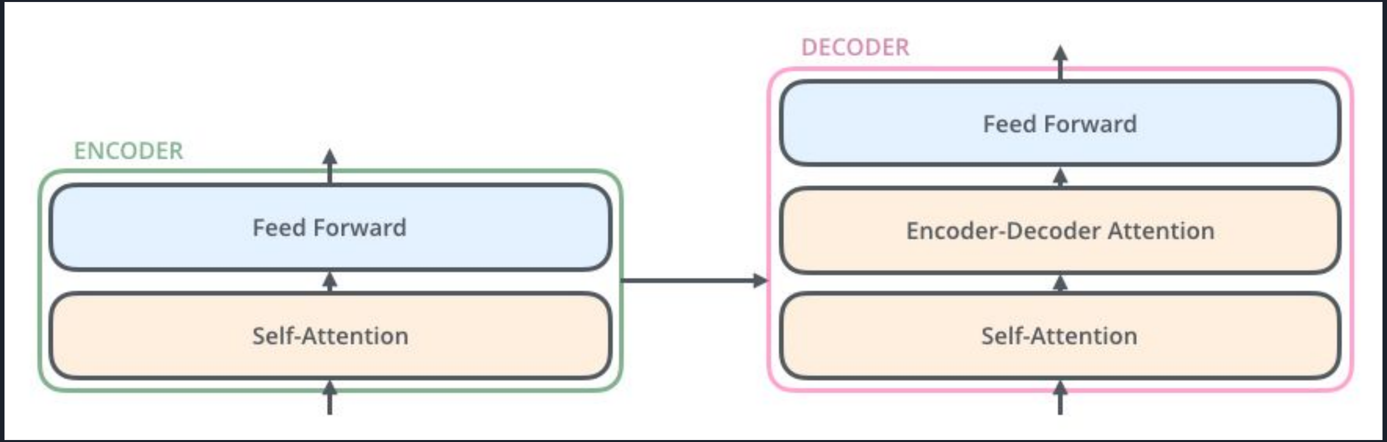
http://jalammar.github.io/illustrated-transformer/

Encoder and Decoder Stacks



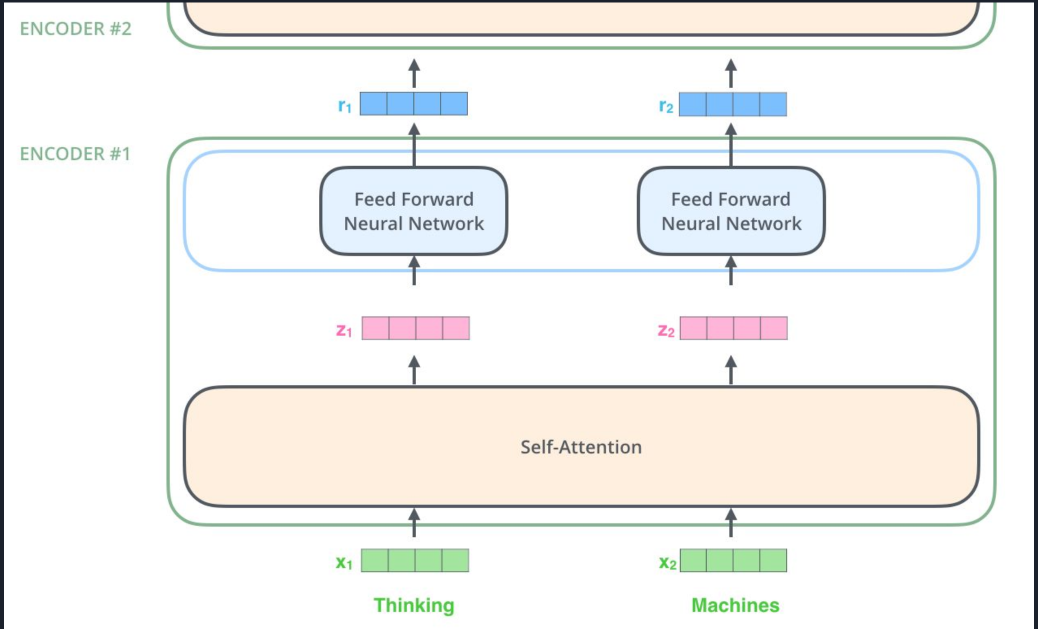
http://jalammar.github.io/illustrated-transformer/

Inside Encoder and Decoder Units



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

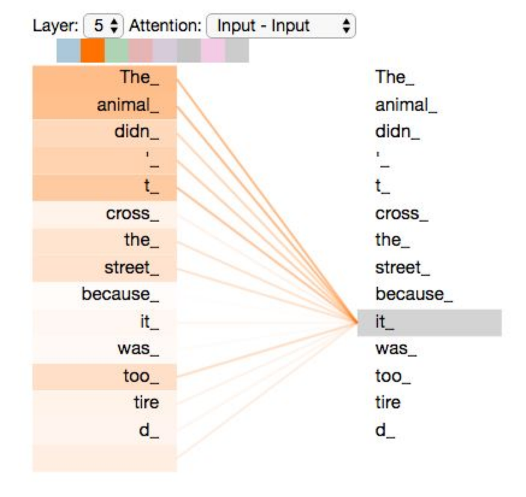
Encoding



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

Self-attention

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

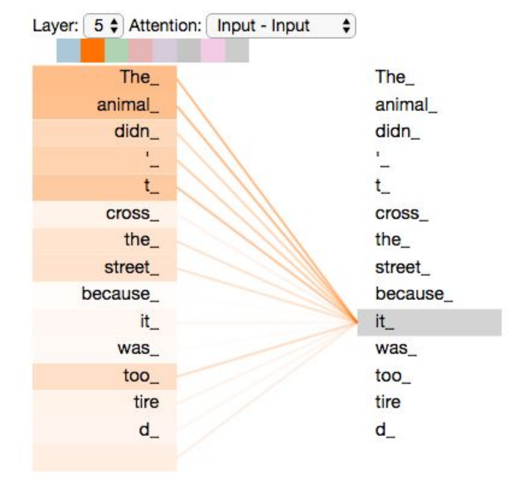


Layer: 5 Attention: Input - Input

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

Tensor2Tensor by Google Brain team

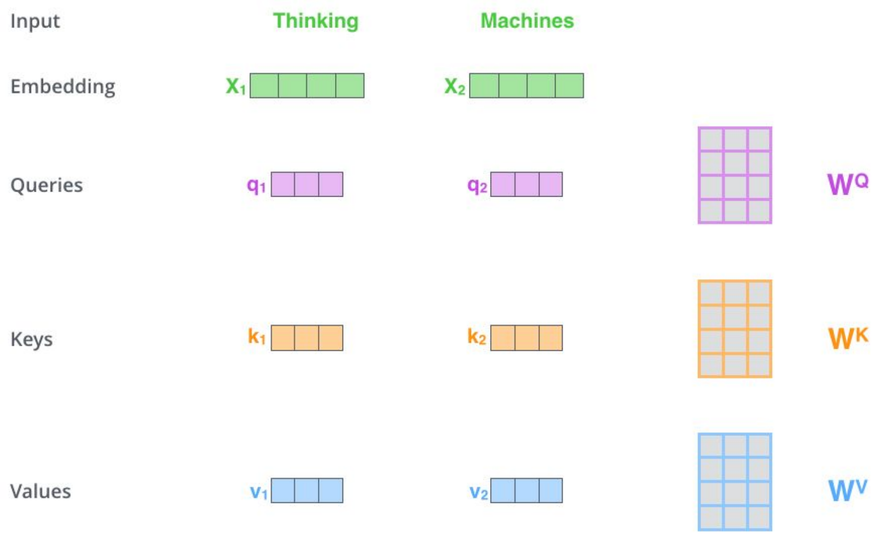
Tensor2Tensor, or T2T for short, is a library of deep learning models and datasets designed to make deep learning more accessible and accelerate ML research.



Layer: 5 Attention: Input - Input

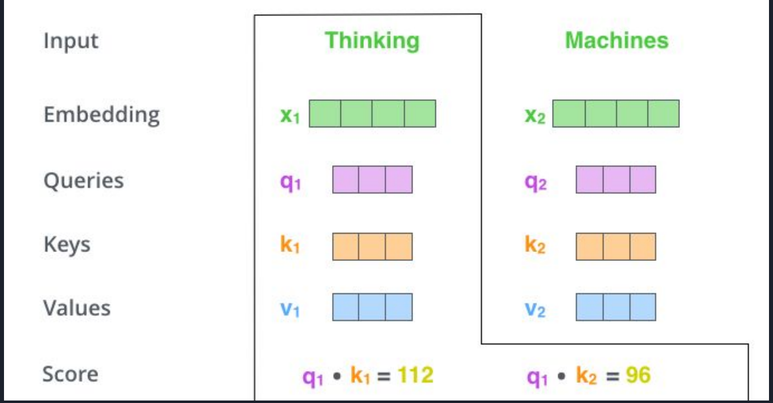
https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

Self-attention in more detail



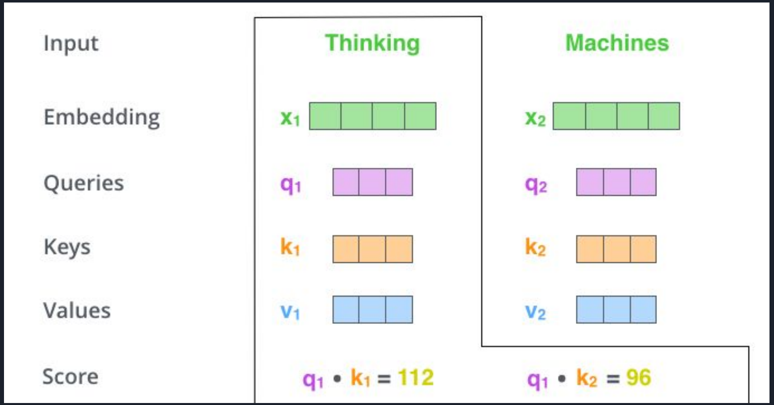
http://jalammar.github.io/illustrated-transformer/

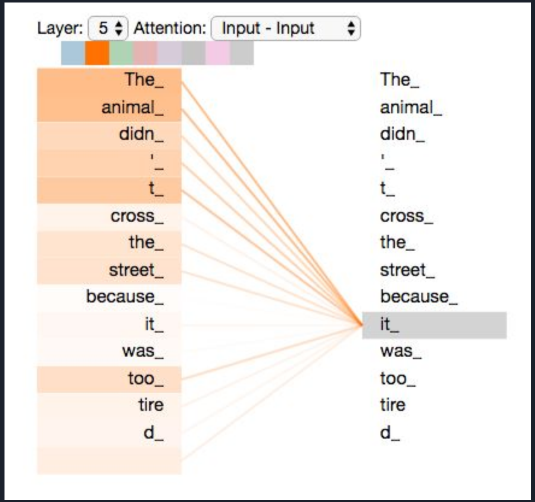
Second step : Score



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

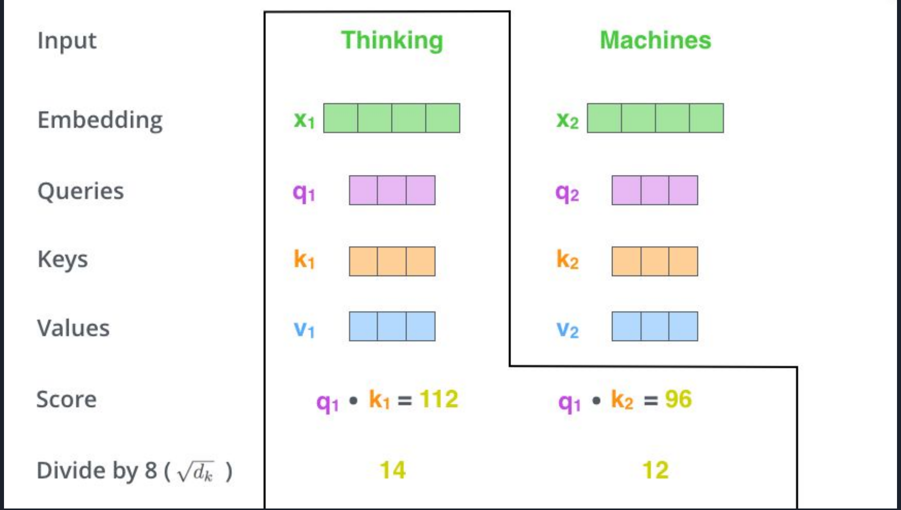
Second step : Score




Layer: 5 Attention: Input - Input

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

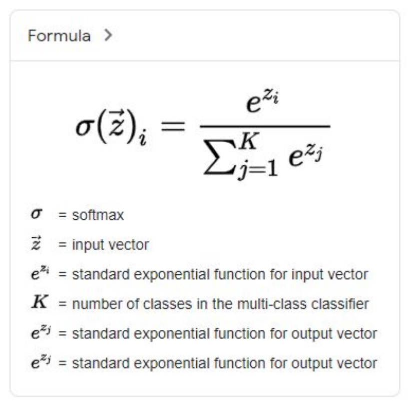
Third step : Divide by 8



Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12

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

Fourth step : Softmax



Formula >

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ = softmax

$\vec{z}$ = input vector

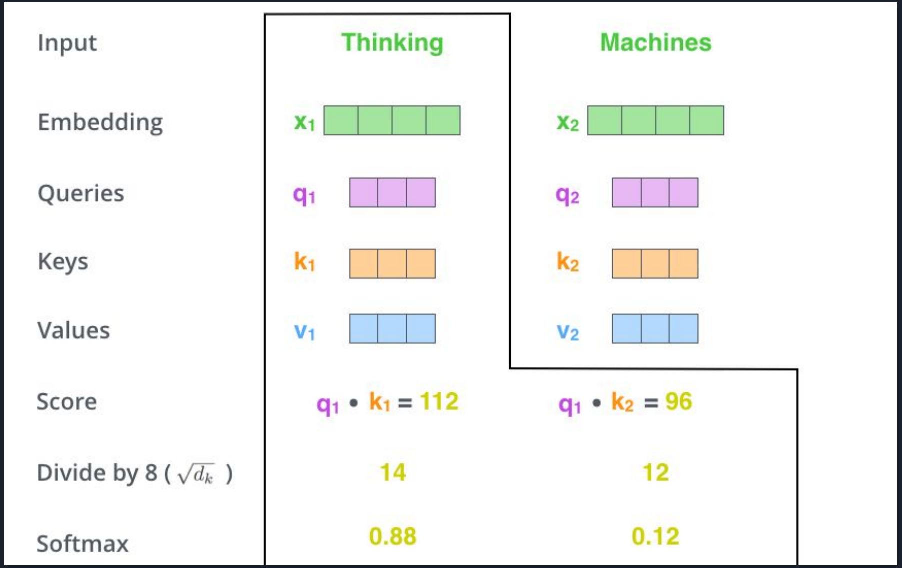
e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

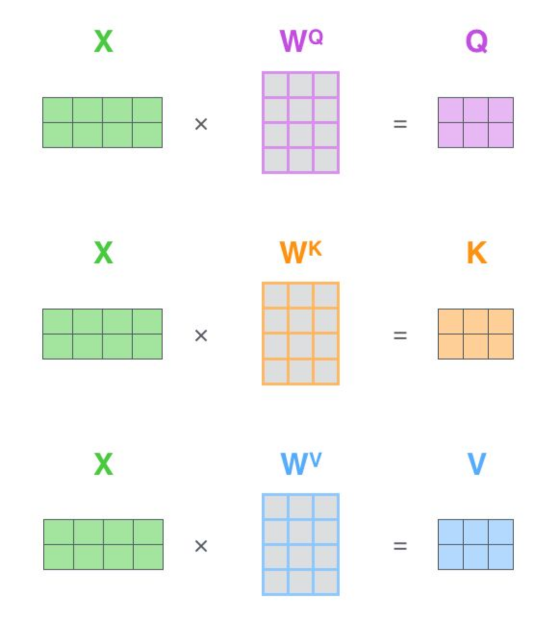
google.com



Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

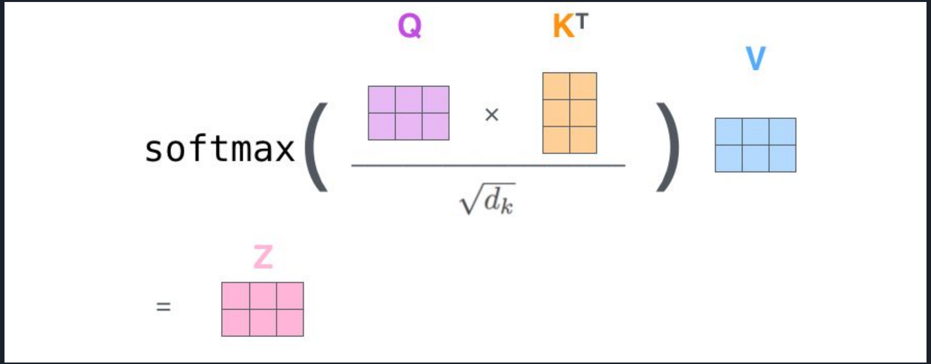
http://jalammar.github.io/illustrated-transformer/

Matrix Representations



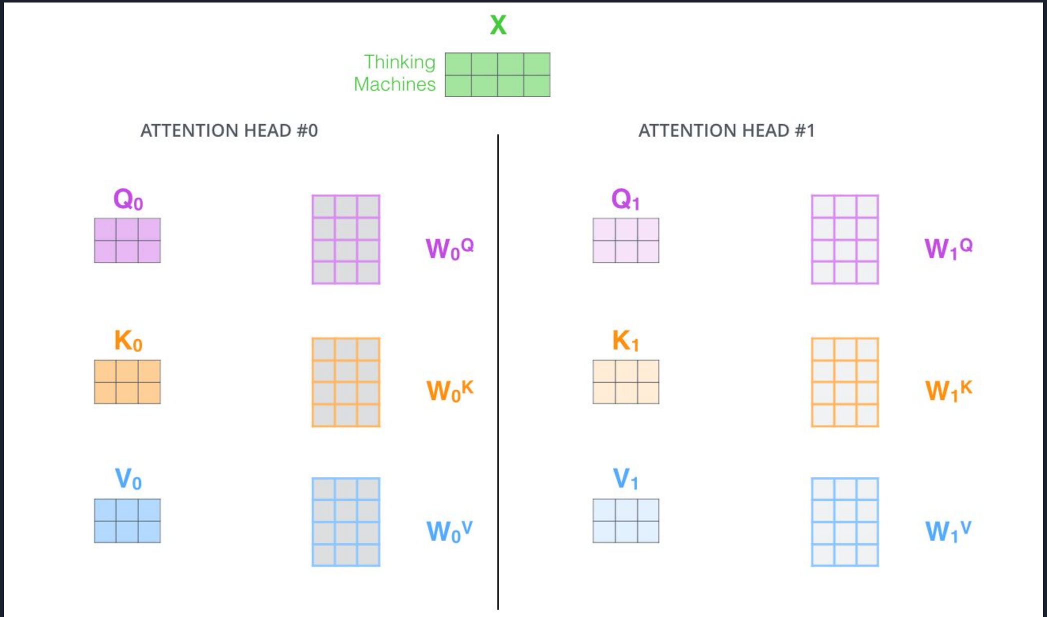
http://jalammar.github.io/illustrated-transformer/

Self-attention Formula



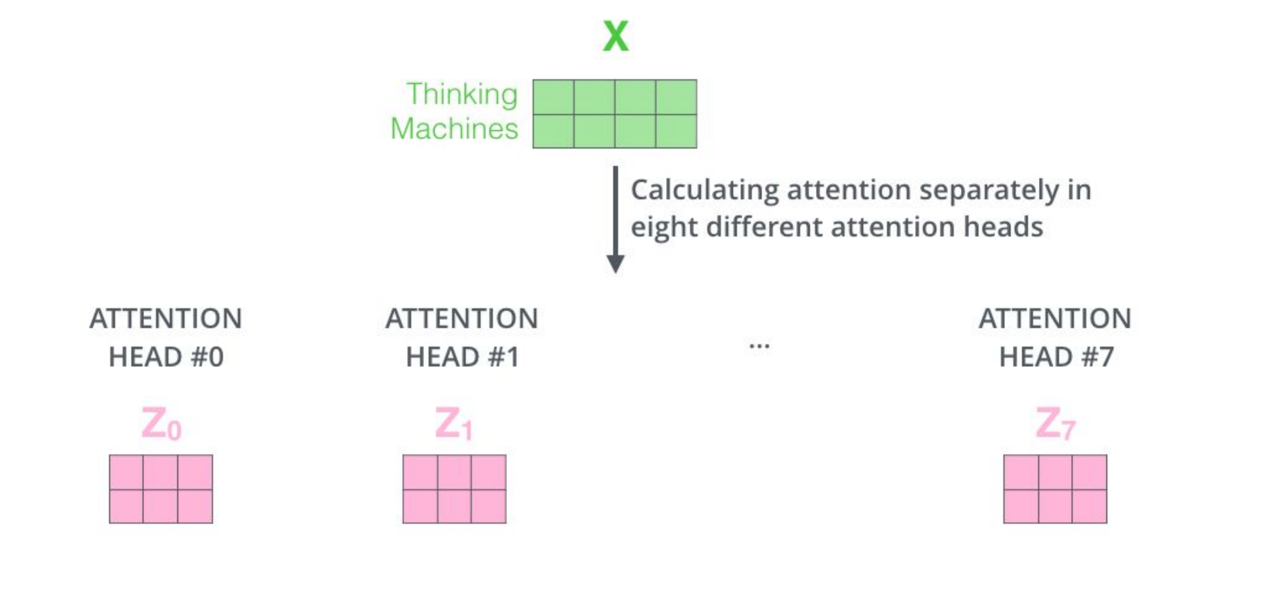
http://jalammar.github.io/illustrated-transformer/

Multi-headed attention



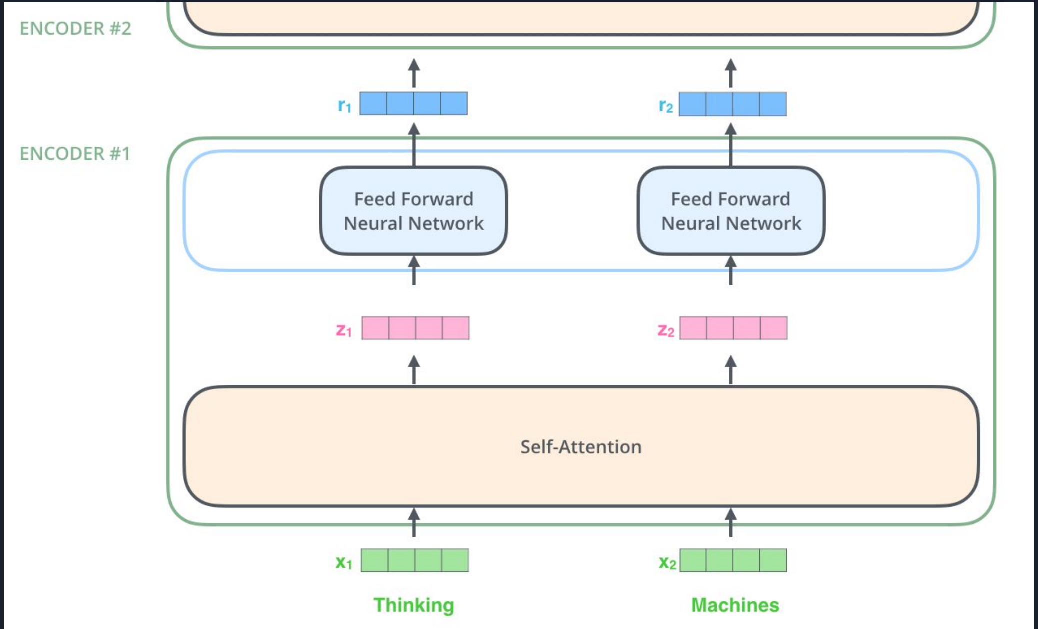
http://jalammar.github.io/illustrated-transformer/

Multi-headed attention



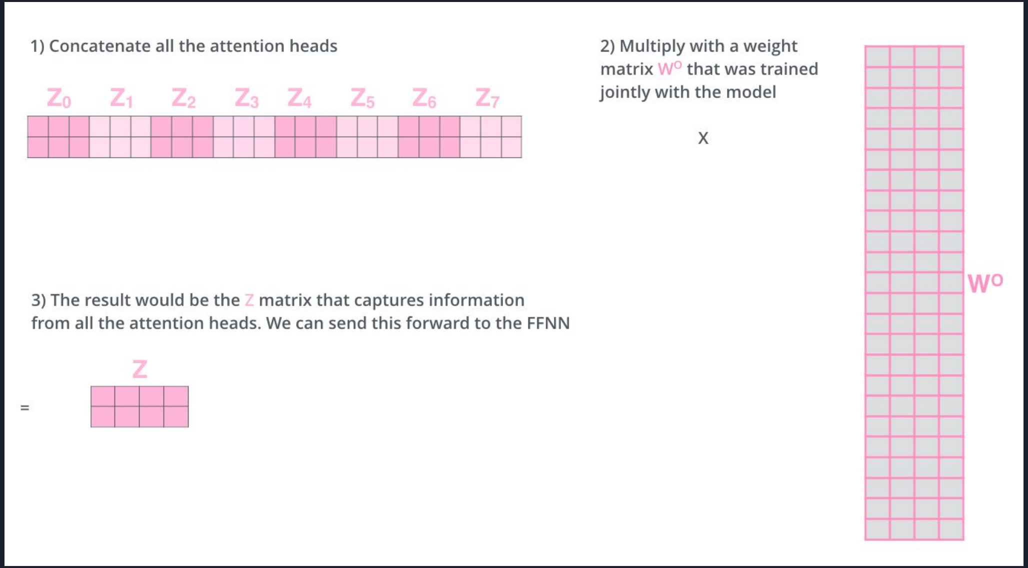
http://jalammar.github.io/illustrated-transformer/

We only need one Z



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

Concatenation and Wo weight matrix



1) Concatenate all the attention heads

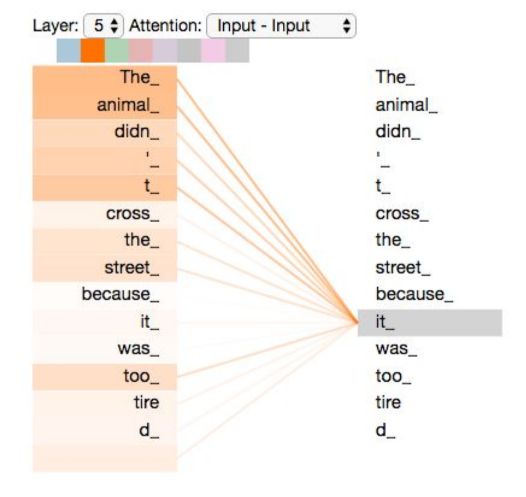
2) Multiply with a weight matrix W^O that was trained jointly with the model

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

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

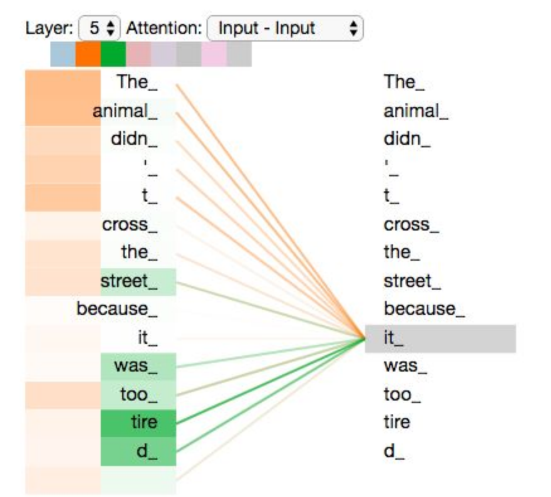
Multi-headed attention visualization

1 Head



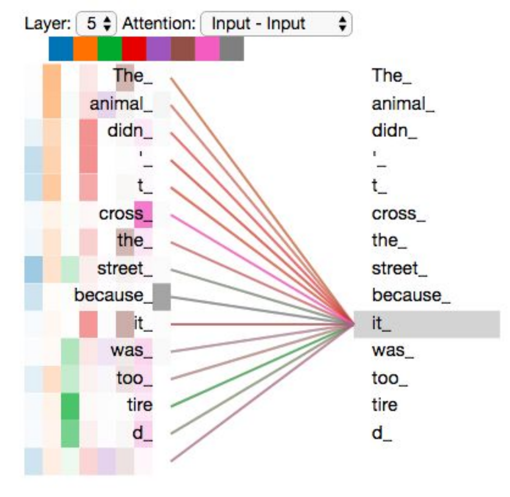
Layer: 5 Attention: Input - Input

2 Heads



Layer: 5 Attention: Input - Input

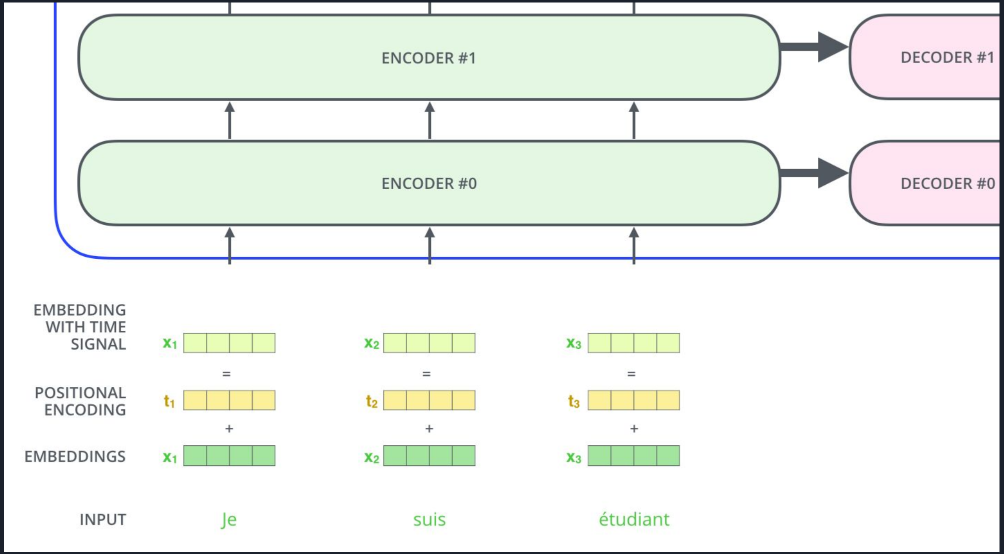
8 Heads

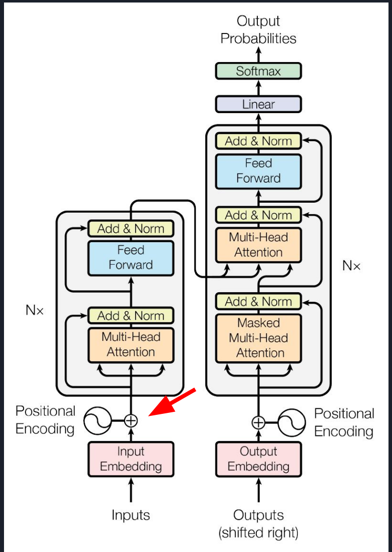


Layer: 5 Attention: Input - Input

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

Positional Encoding

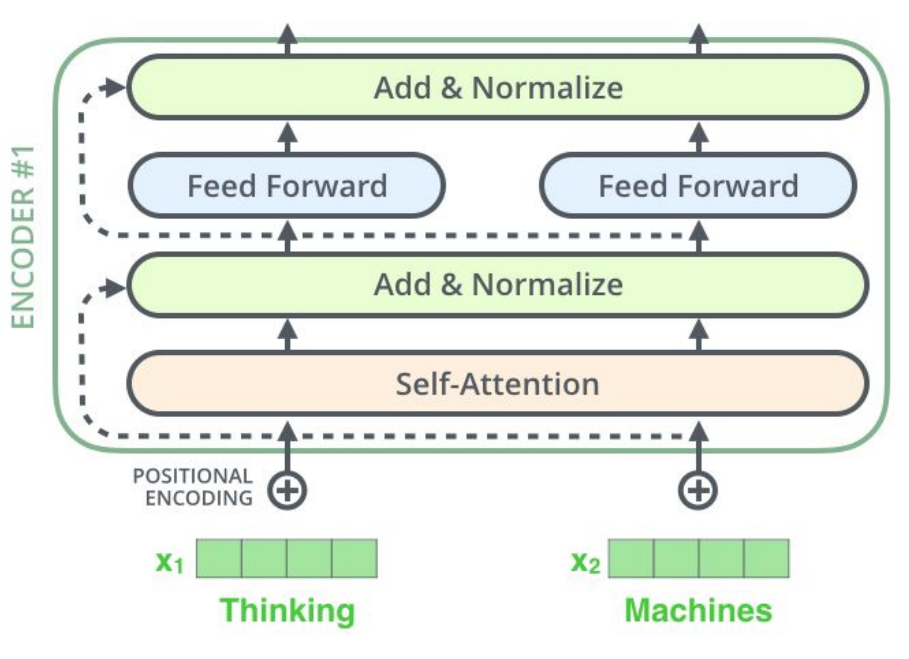




http://jalammar.github.io/illustrated-transformer/

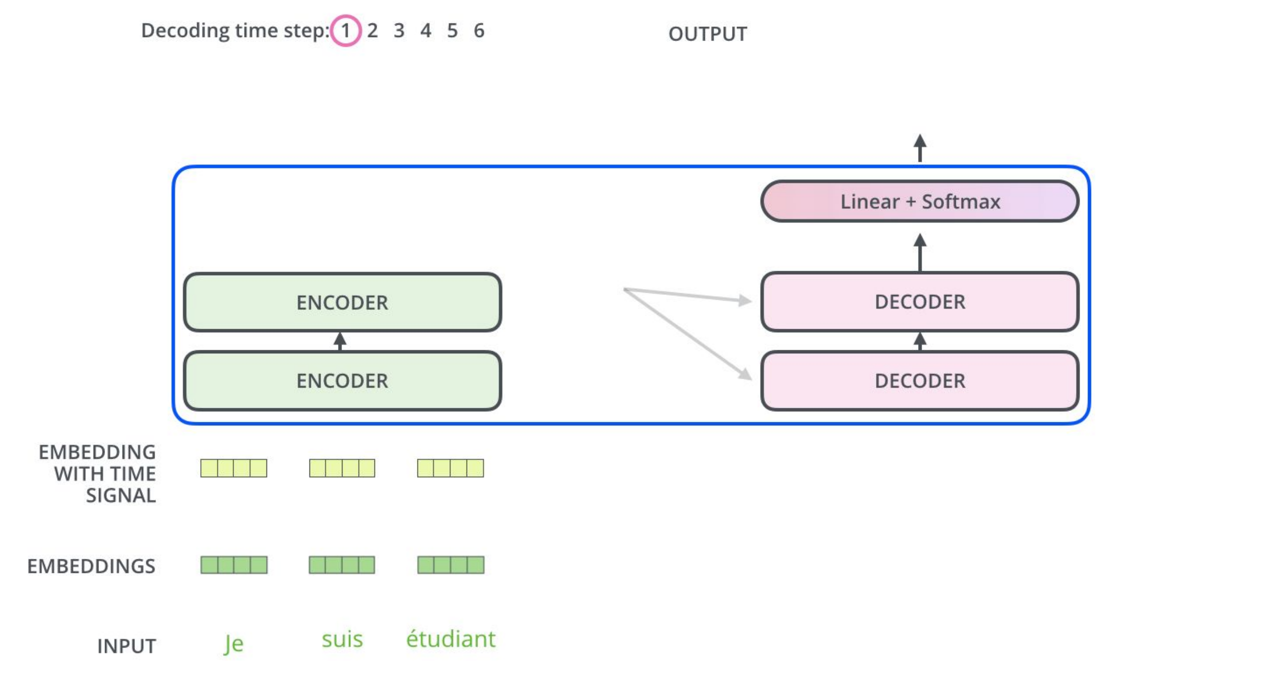
https://arxiv.org/abs/1706.03762

Residual Connections



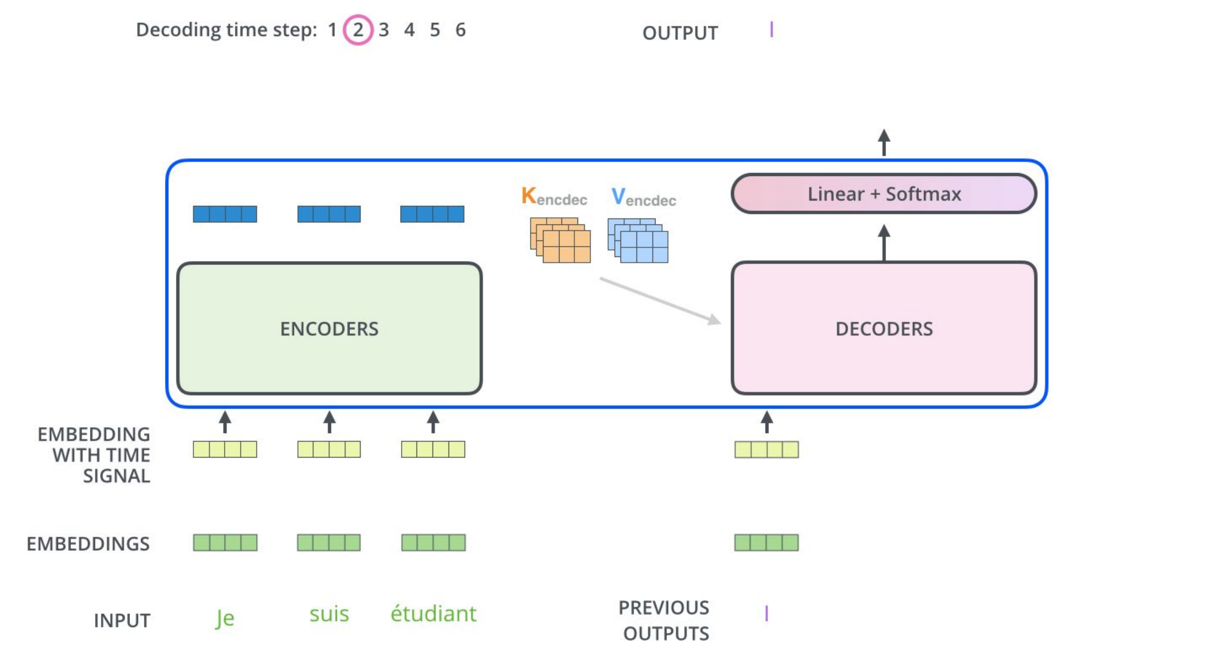
http://jalammar.github.io/illustrated-transformer/

Encoder Decoder Attention



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

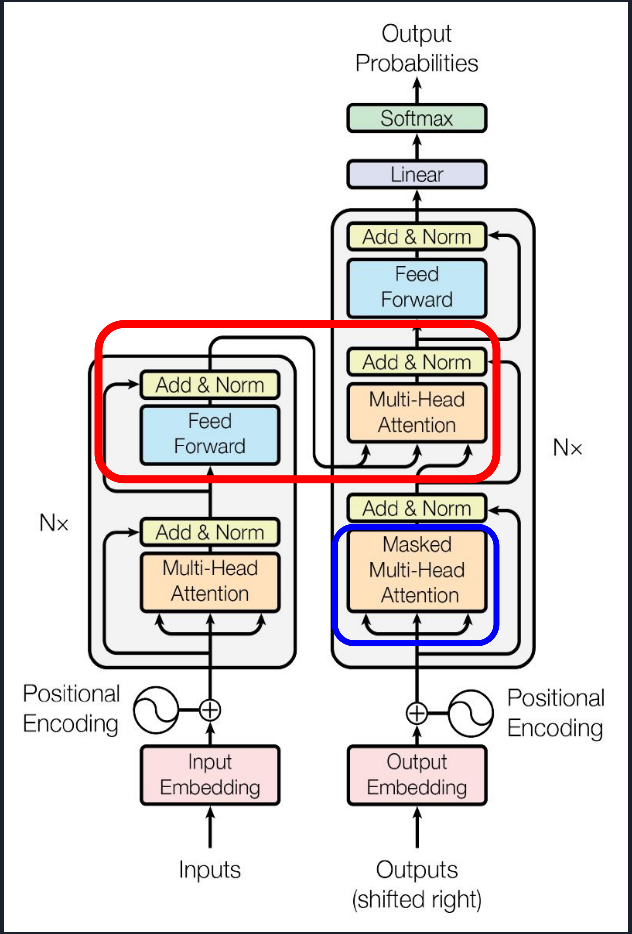
Encoder Decoder Attention



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

Encoder Decoder Attention

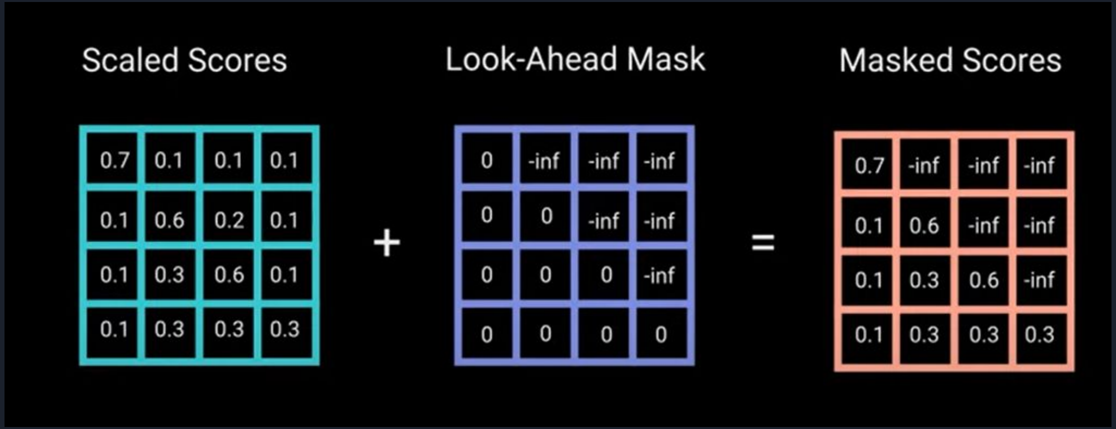
Creates Queries matrix from the layer below it, and takes the Keys and Values matrix from the encoder stack.



https://arxiv.org/abs/1706.03762

Encoder Attention
Vs
Decoder Attention

Masking



Scaled Scores

0.7	0.1	0.1	0.1
0.1	0.6	0.2	0.1
0.1	0.3	0.6	0.1
0.1	0.3	0.3	0.3

+

Look-Ahead Mask

0	-inf	-inf	-inf
0	0	-inf	-inf
0	0	0	-inf
0	0	0	0

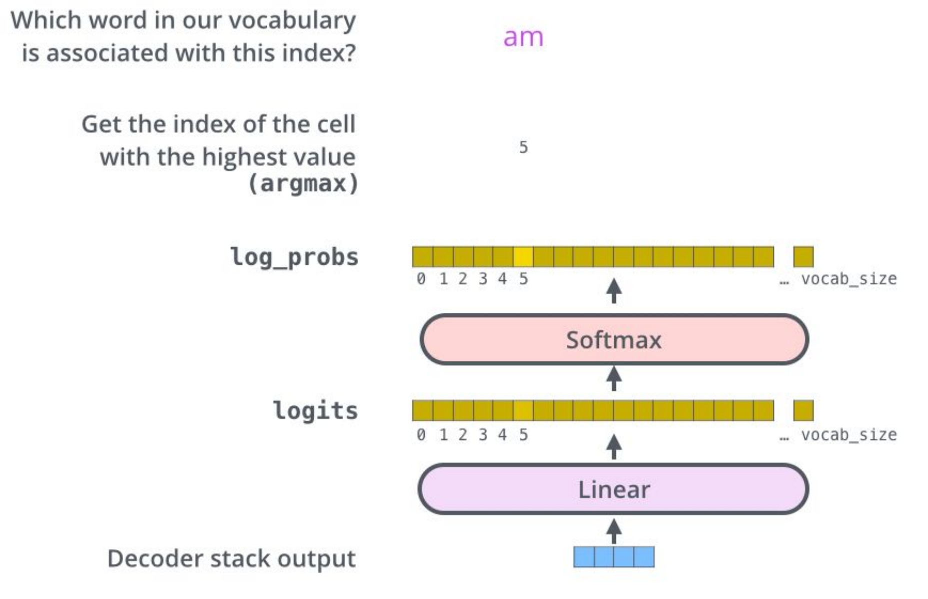
=

Masked Scores

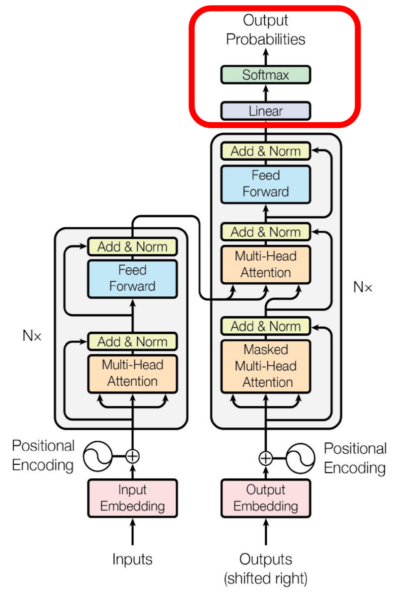
0.7	-inf	-inf	-inf
0.1	0.6	-inf	-inf
0.1	0.3	0.6	-inf
0.1	0.3	0.3	0.3

YT - The A.I. Hacker - Michael Phi

The Final Linear and Softmax Layer


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value (argmax)
5
log_probs
0 1 2 3 4 5
… vocab_size
Softmax
logits
0 1 2 3 4 5
… vocab_size
Linear
Decoder stack output

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

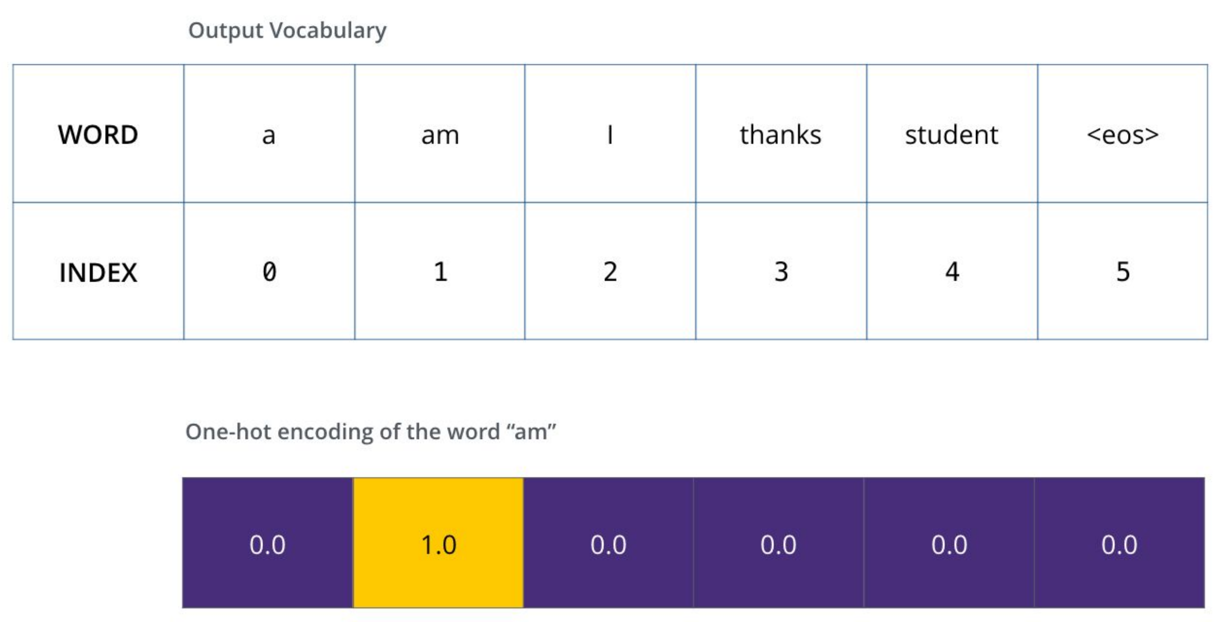


https://arxiv.org/abs/1706.03762

Training

- One Hot Encoding
- Loss Function
- Another Training Method

One Hot Encoding



Output Vocabulary

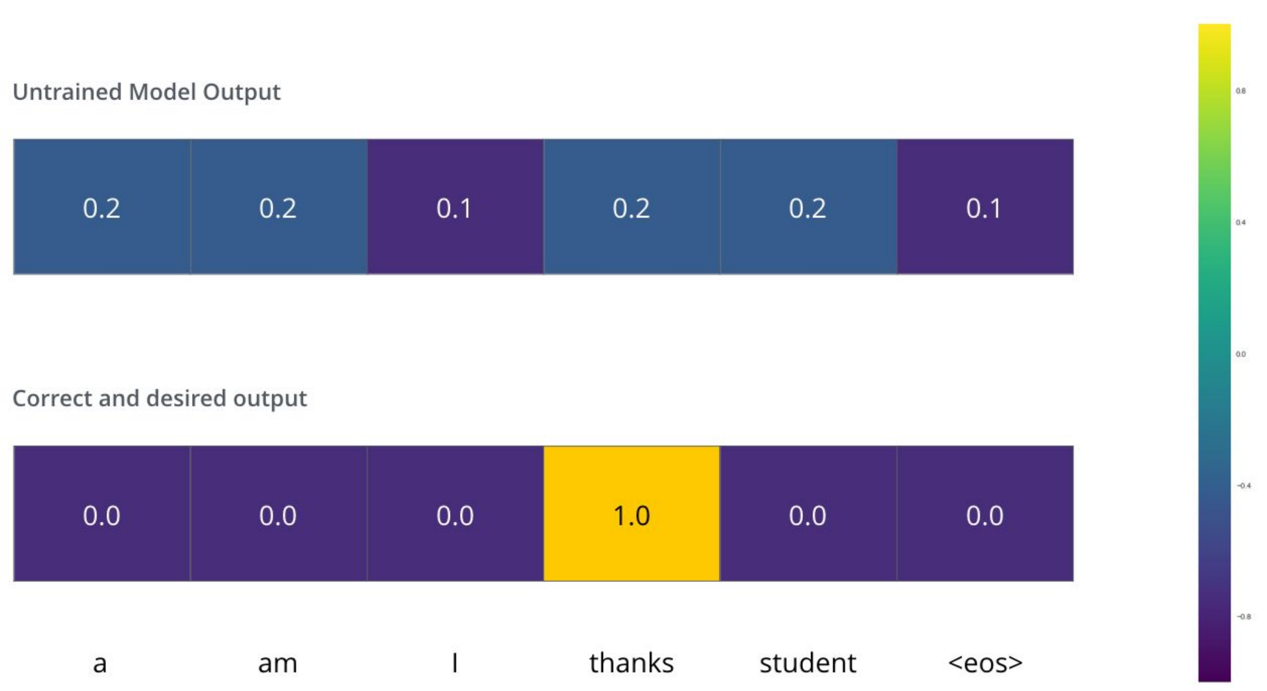
WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word “am”

0.0	1.0	0.0	0.0	0.0	0.0

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

Loss Function



Untrained Model Output

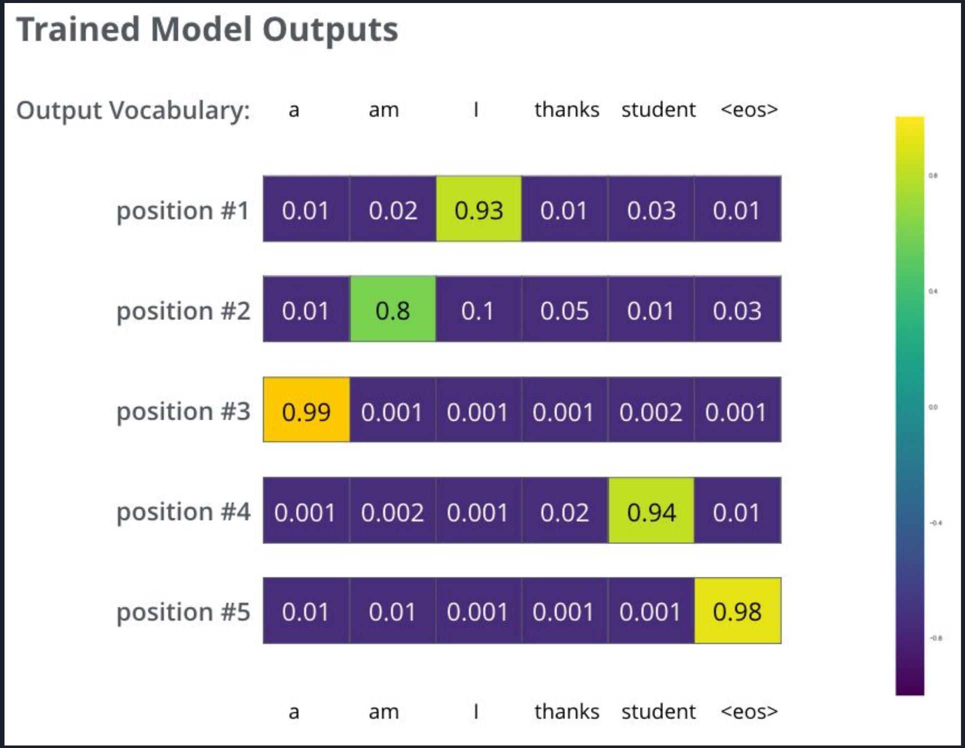
0.2	0.2	0.1	0.2	0.2	0.1

Correct and desired output

a	am	I	thanks	student	<eos>
0.0	0.0	0.0	1.0	0.0	0.0

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

Loss Function



Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98

a
am
I
thanks
student
<eos>
0.8
0.4
0.0
-0.4
-0.8

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

Another Training Method

Another way to do it would be to hold on to, say, the top two words (say, ‘I’ and ‘a’ for example), then in the next step, run the model twice

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

Resources

Illustrated-transformer

Illustrated Guide to Transformers Neural Network: A step by step explanation

A Deep Dive Into the Transformer Architecture – The Development of Transformer Models

Attention Is All You Need

Transformers and Language Models - YouTube Playlist

Thanks

hasanmahani08@gmail.com

rezadar1378@gmail.com