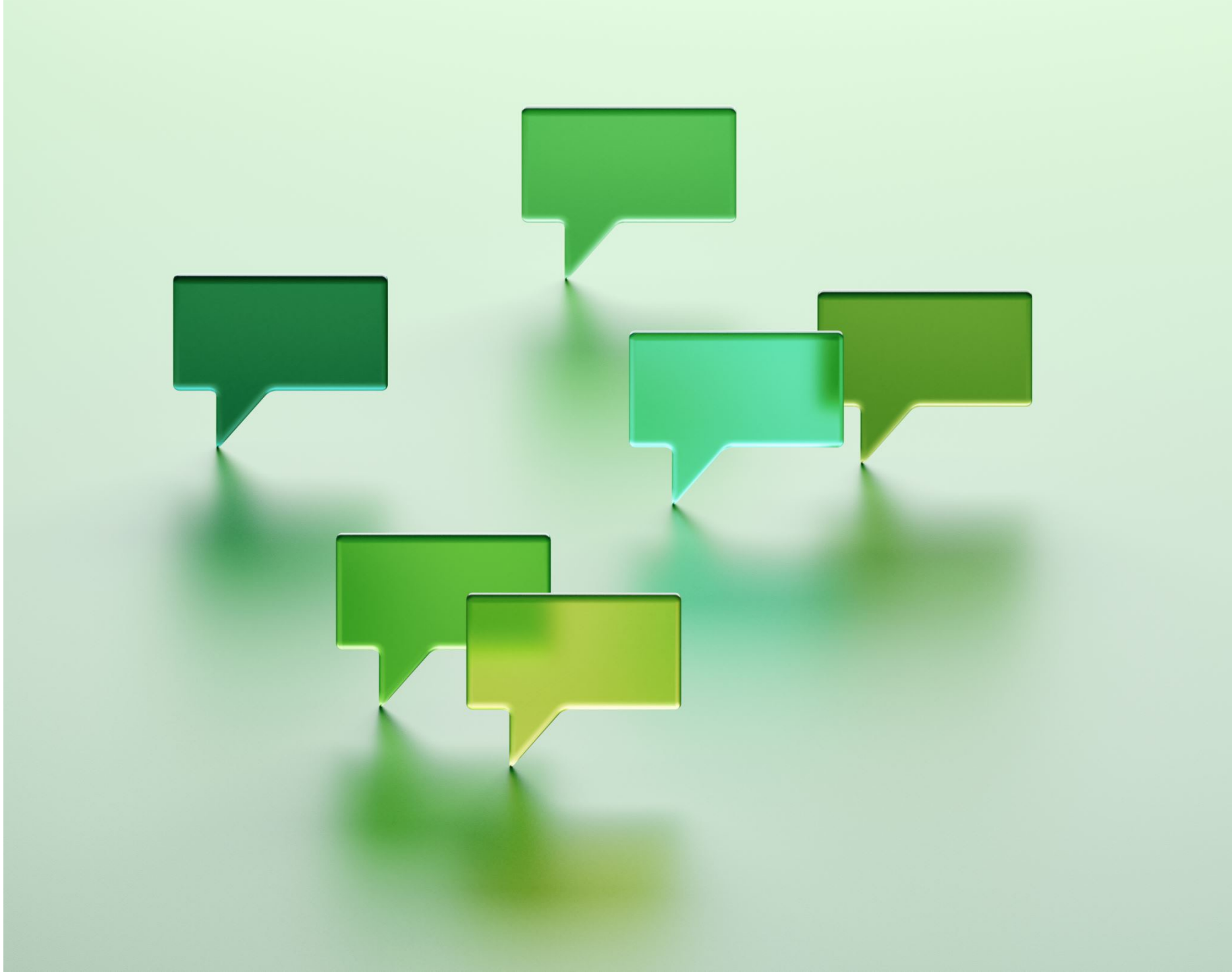
Introduction to Transformers

Dra. Helena Gómez Adorno
IIMAS, UNAM
helena.gomez@iimas.unam.mx



Who am I?

Bachelor, 2001- 2005
Universidad Nacional de Asunción, Paraguay.

Master CS, 2011-2013
Benemérita Universidad Autónoma de Puebla, México.

PhD CS, 2014-2018
Instituto Politécnico Nacional, México.

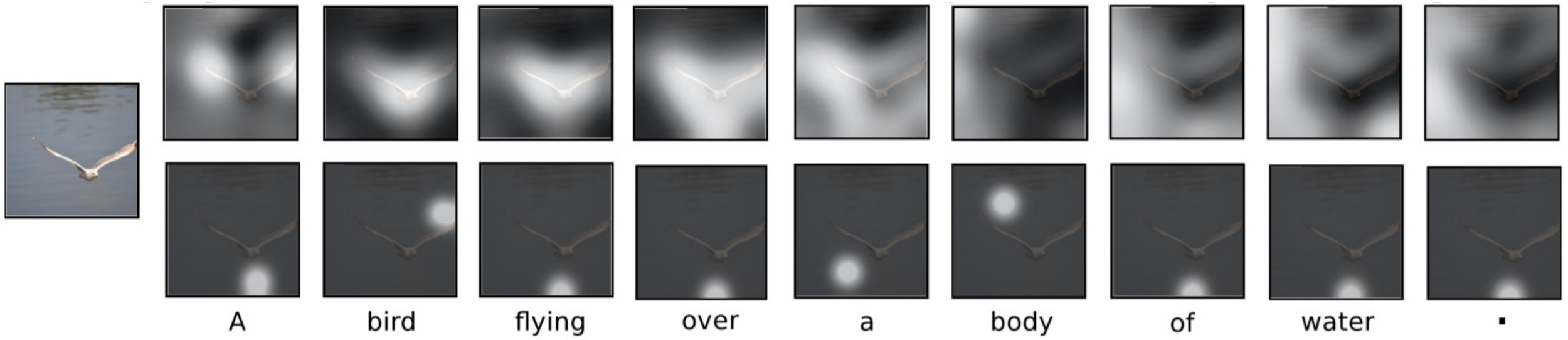
Researcher, 2018-today
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas.

Introduction

Bag of Words → CNN → RNN/LSTM → Transformers

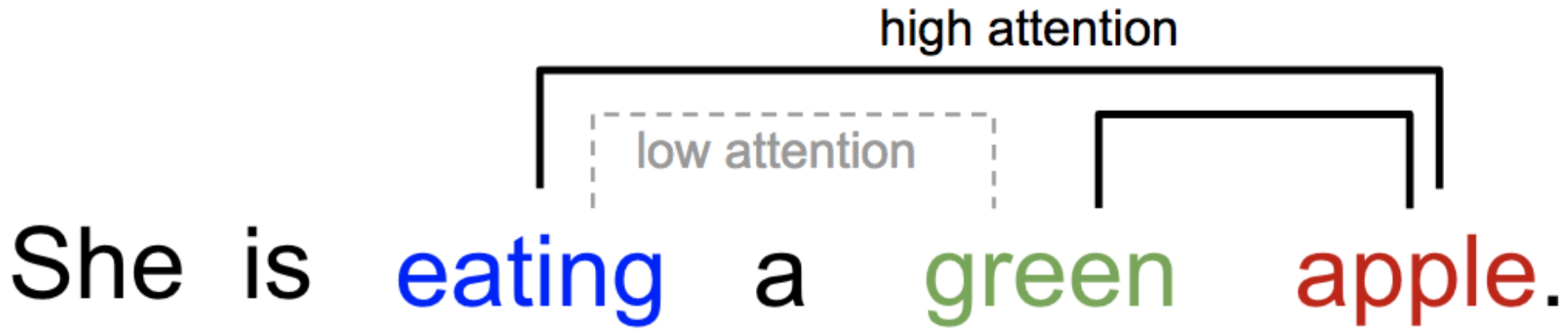
Attention

The attention mechanisms in neural networks are based on the intuition of how humans perceive images and how a sequence of words is interpreted.



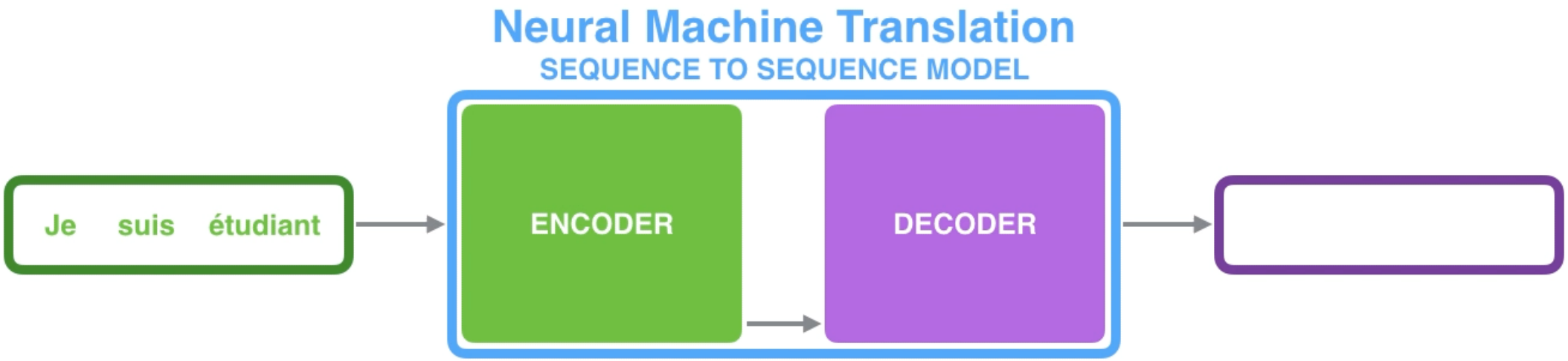
Attention

Attention consists of defining a vector of weights that allows weighting the different levels of inputs.



Seq2seq: Encoder-Decoder architecture

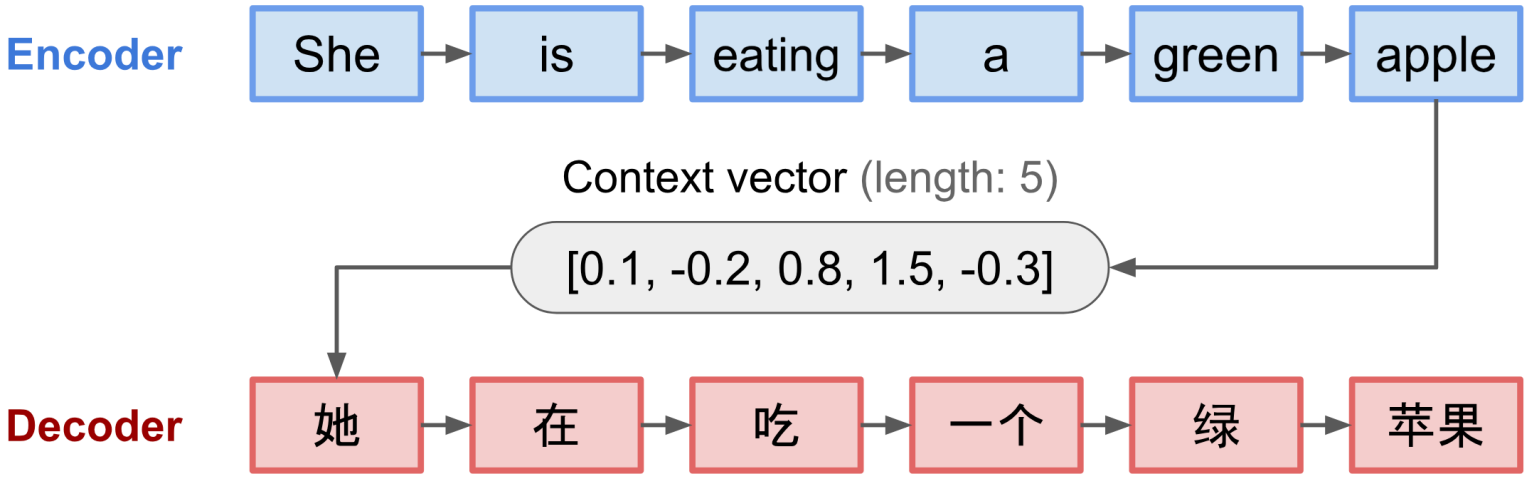
They are models that receive an input sequence and produce an output sequence. Both sequences can be of arbitrary lengths and not necessarily of the same magnitude. This type of architecture is used for problems such as machine translation, question-answering systems and automatic summarization.



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

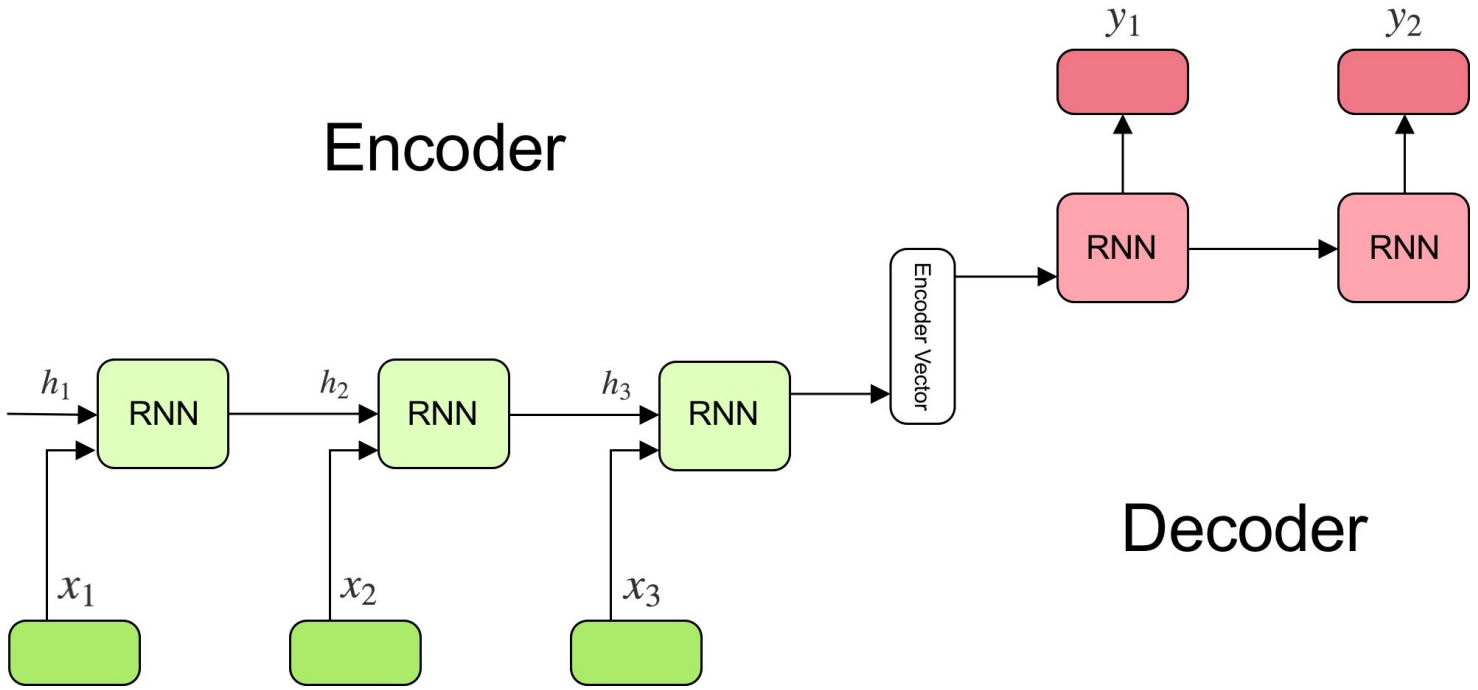
Encoder-Decoder architecture

After processing the entire input sequence, the encoder sends the context to the decoder, which starts producing the output sequence element by element. The context vector is basically the number of hidden units in the encoder RNN.

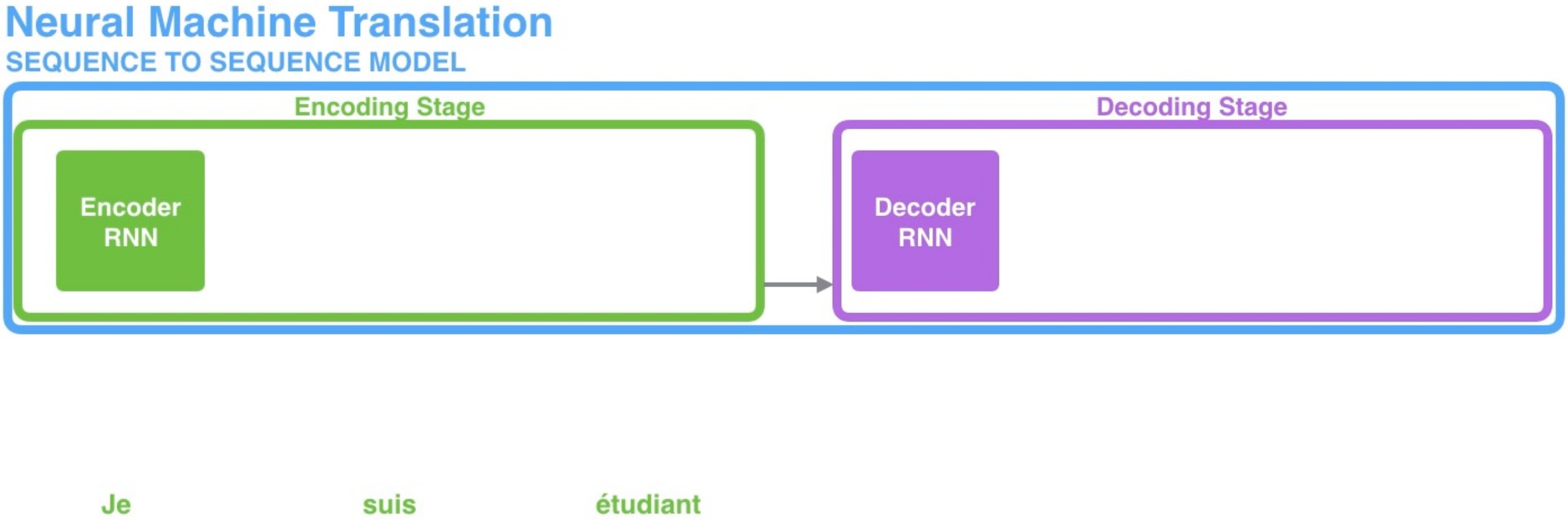


Encoder-Decoder architecture

- They allow modeling relationships between elements of a sequence.
- Both, the encoder and the decoder can be implemented as an RNN.
- A context vector representing the input sequence is generated.
- It is complicated that the context vector captures all the properties of very long input sequences.



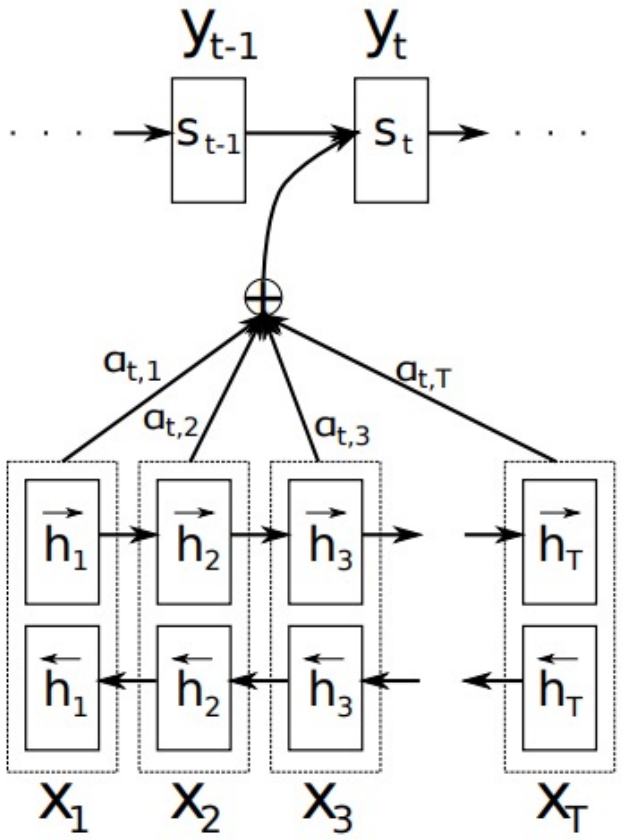
Encoder-Decoder architecture



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

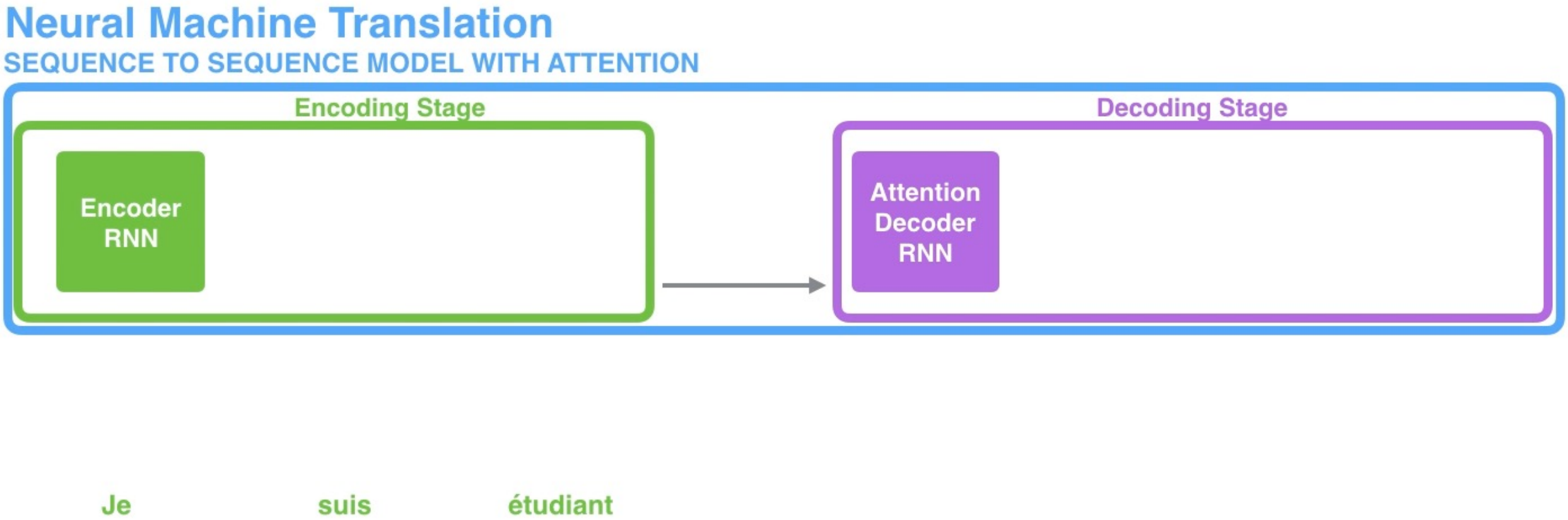
Encoder-Decoder architecture - Attention

Bahdanau 2015, proposed to use all the hidden states of the encoder to create a custom context for each element of the output sequence.



Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate.” ICLR 2015.

Now let’s put attention



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

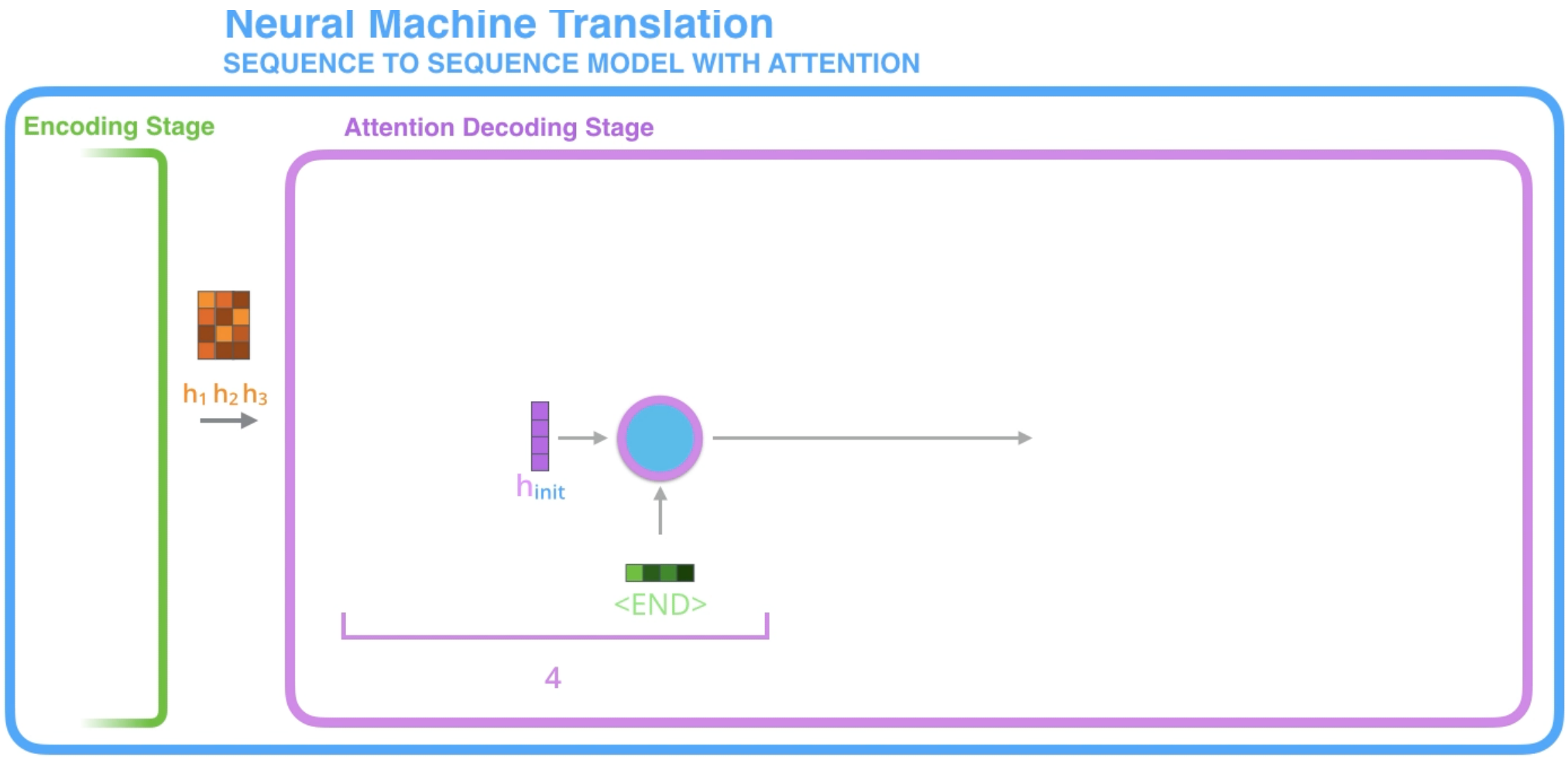
Now let’s put attention

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

Attention – Score Functions

	Score	Artículo
Content based	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves, 2014
Summative	$score(s_t, h_i) = v_a \tanh(W_a[s_t;\ h_i])$	Bahdanau, 2015
Location Based	$\text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$	Luong, 2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t \mathbf{W}_a \boldsymbol{h}_i$	Luong, 2015
Dot product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t \boldsymbol{h}_i$	Luong, 2015
Scaled Dot Product	$score(s_t, h_i) = \frac{s_t h_i}{\sqrt{n}}$	Vaswani, 2017

Seq2seq model with attention



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

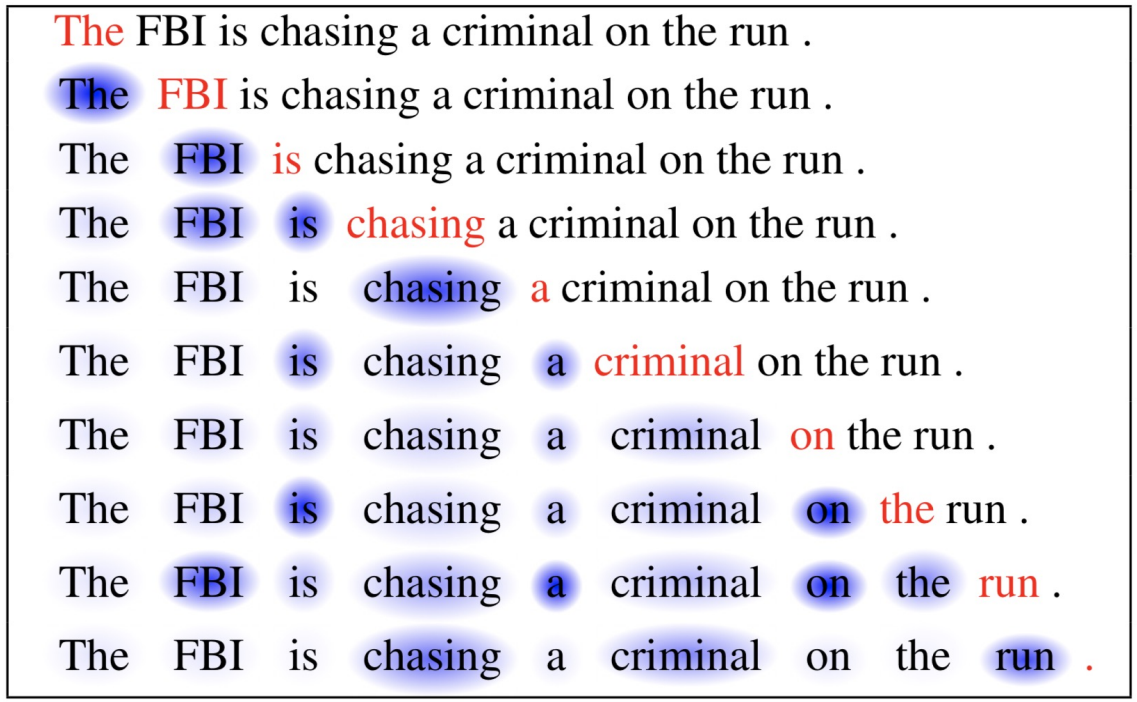
Attention – Variants

- **Global.** The complete sequence is taken into account.
- **Local.** A portion of the sequence is attended.
- **Self-attention.** Evaluates the weights considering the same sequence.

For all these variants any **score** function can be used, what varies is which parts of the sequence are accessed.

Auto-Attention

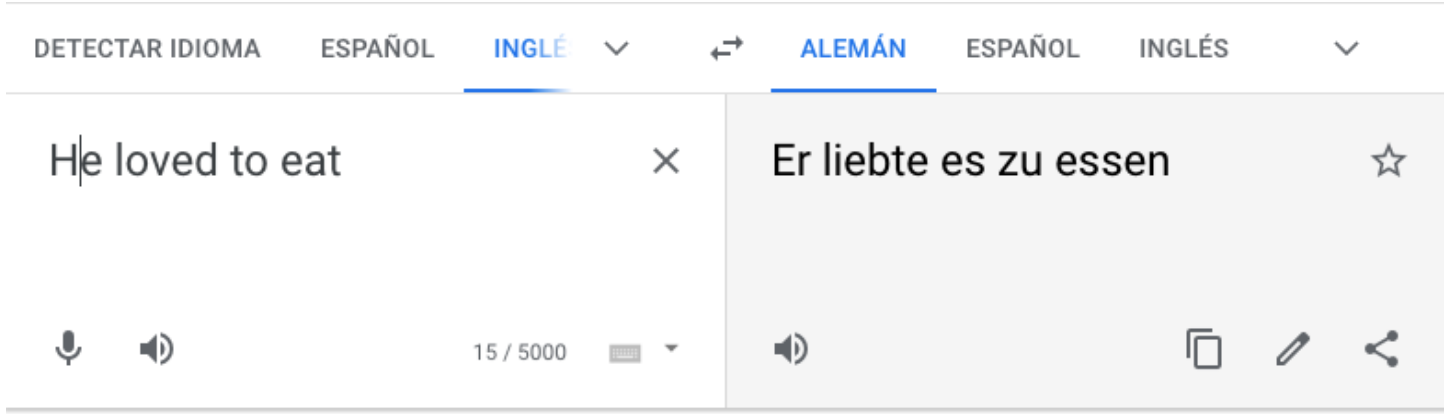
This mechanism consists of calculating an attention score by taking into account an element of a sequence and evaluating it against another element of the same sequence.
In the case of text, this mechanism allows to evaluate the relationships that exist between words of the same sequence, not only between the **input** and **output** sequence.

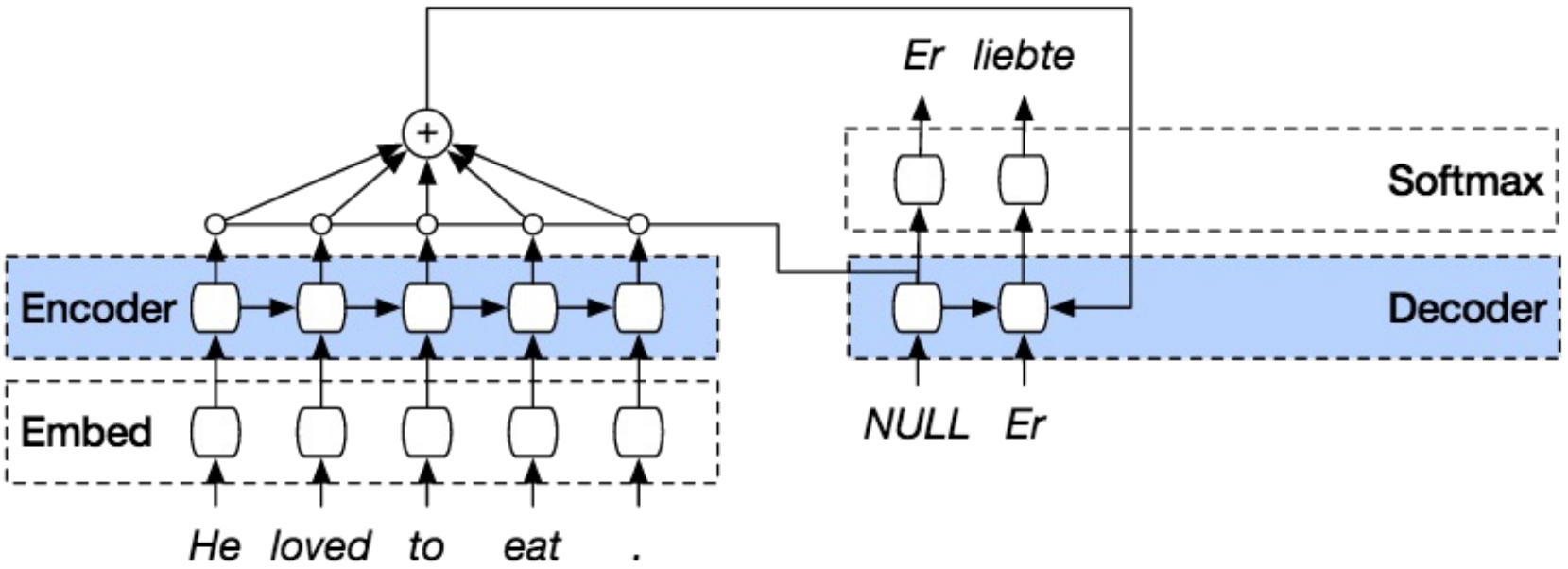


Jianpeng Cheng, Li Dong and Mirella Lapata, "Long Short-Term Memory-Networks for Machine Reading", 2016

Example – Automatic Translation

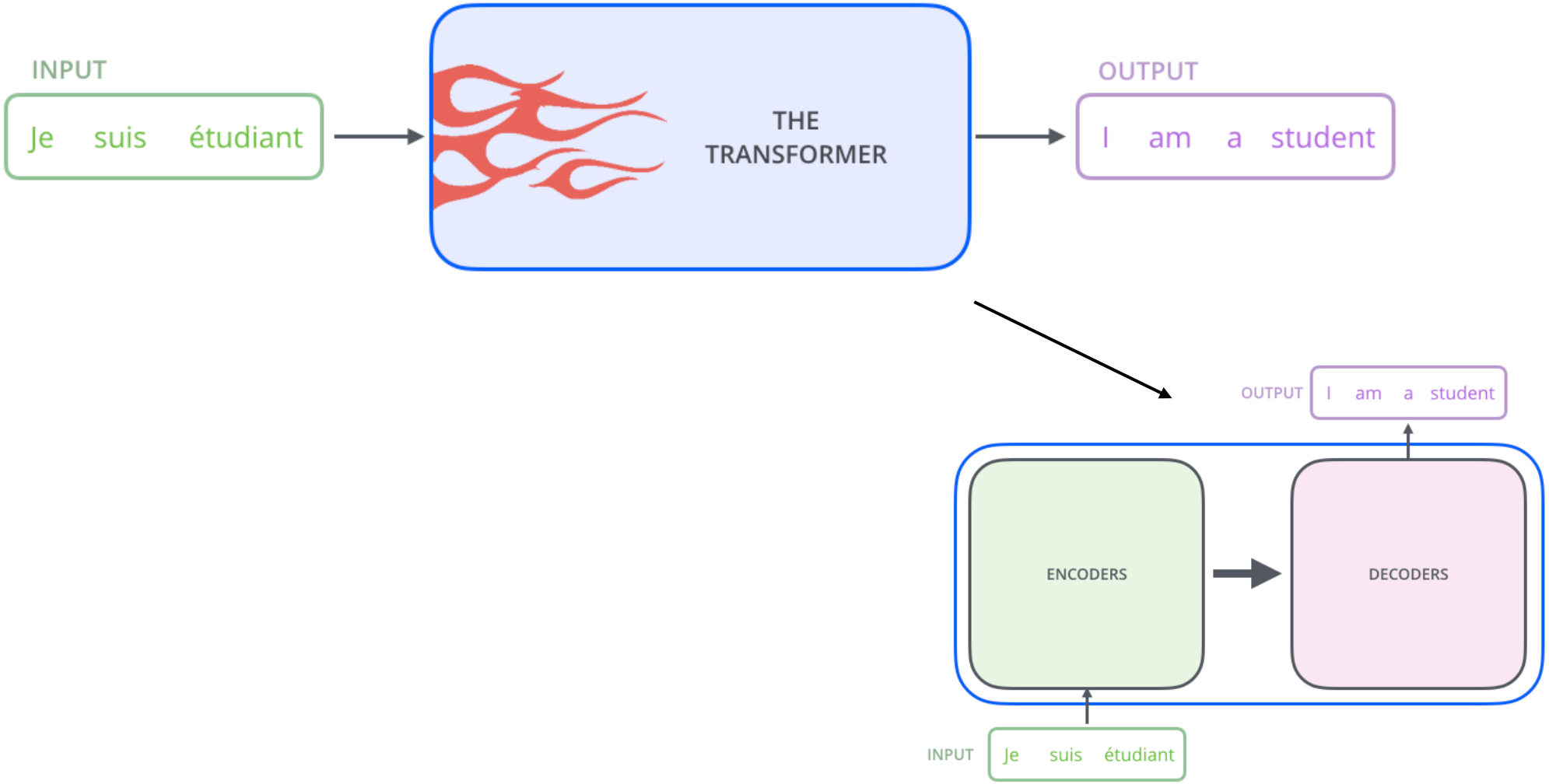
The task of translating a text from one language to another using a computer.





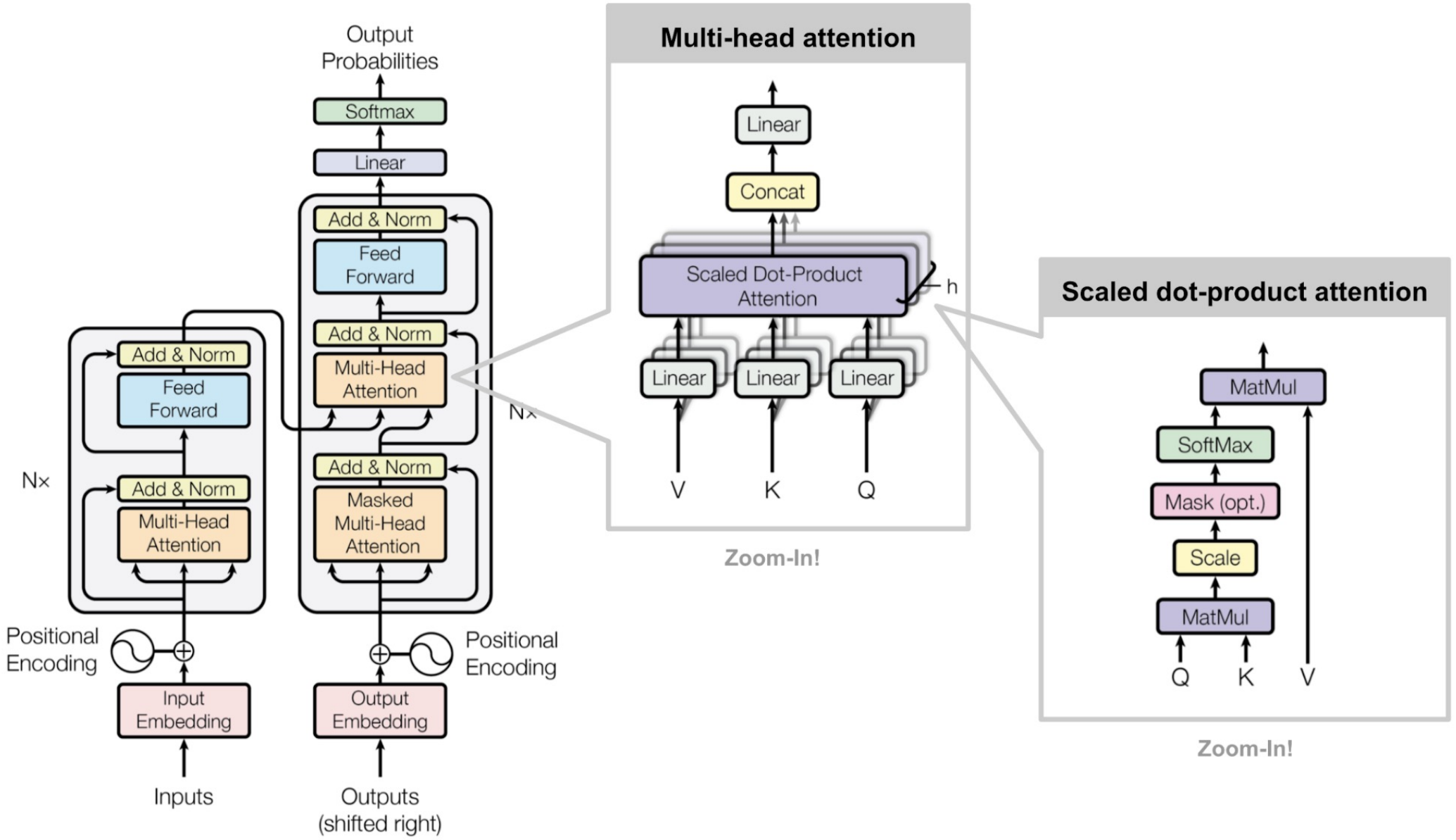
Transformers

Vaswani et al. “Attention Is All You Need”, 2017



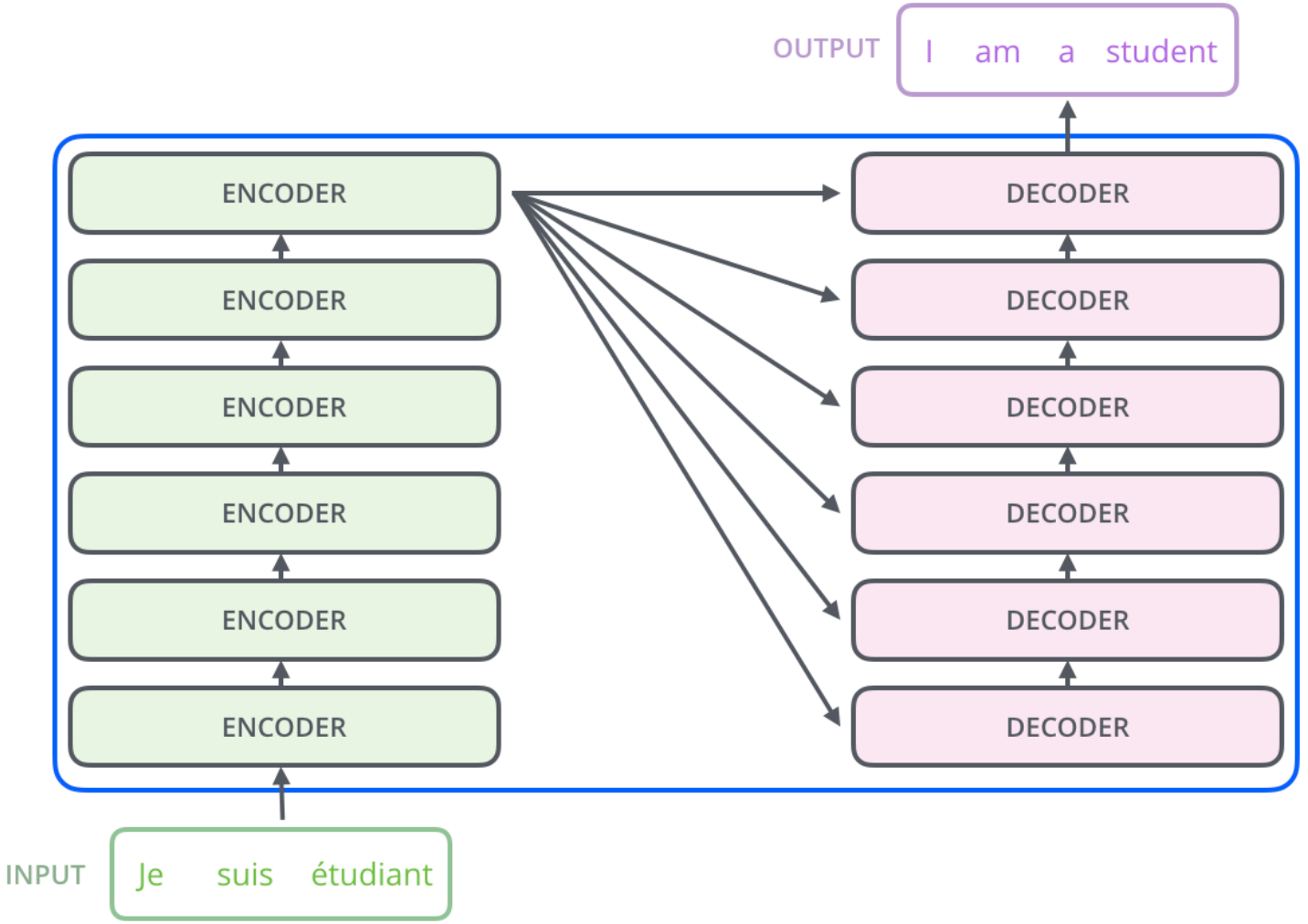
http://jalammar.github.io/illustrated-transformer/

Transformers



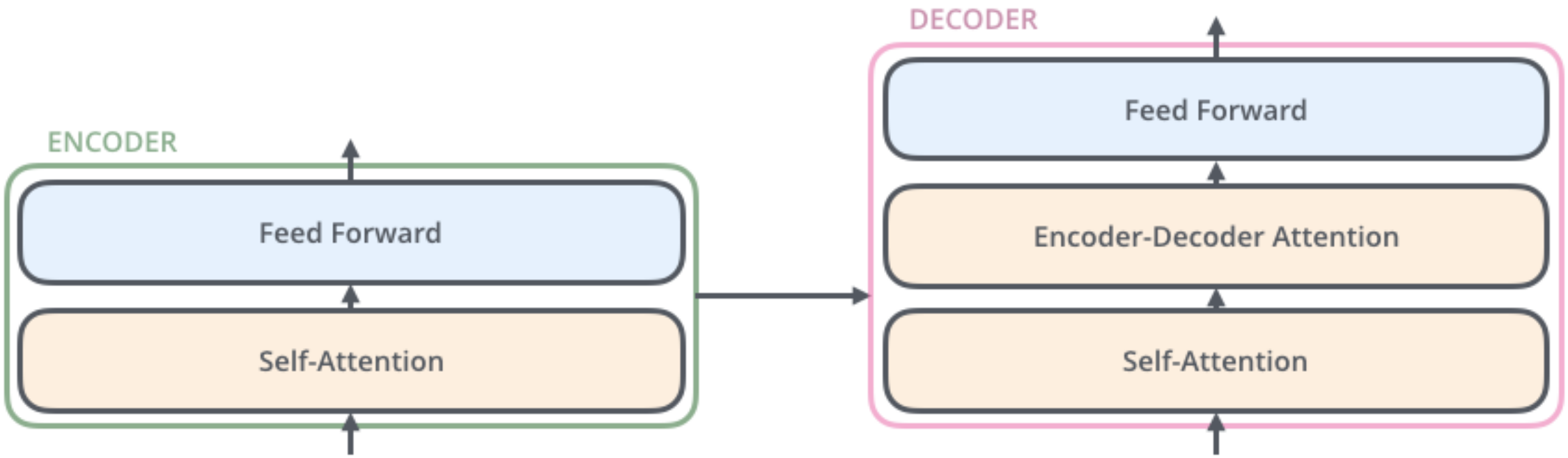
http://jalammar.github.io/illustrated-transformer/

Transformers



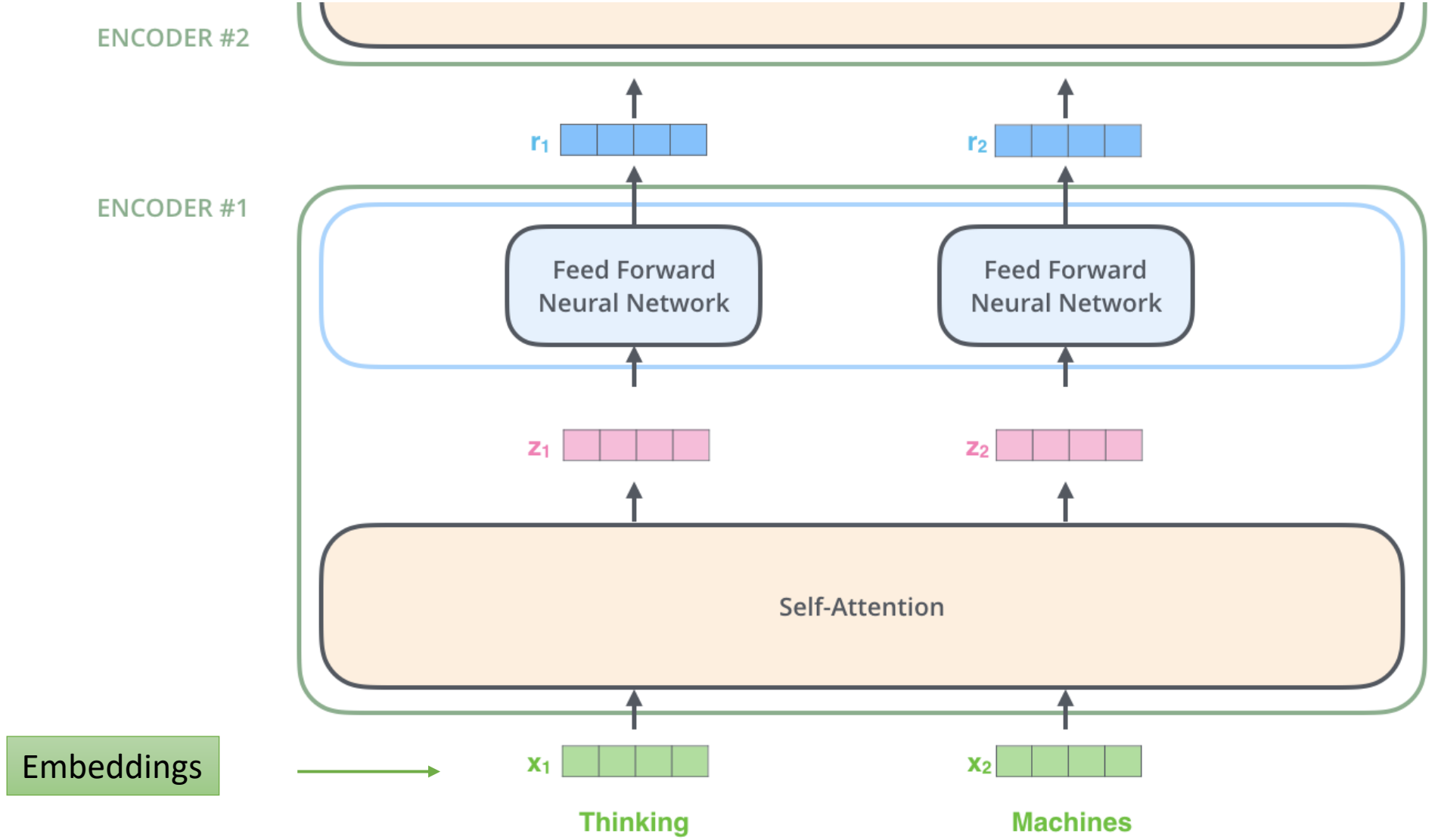
http://jalammar.github.io/illustrated-transformer/

Transformers



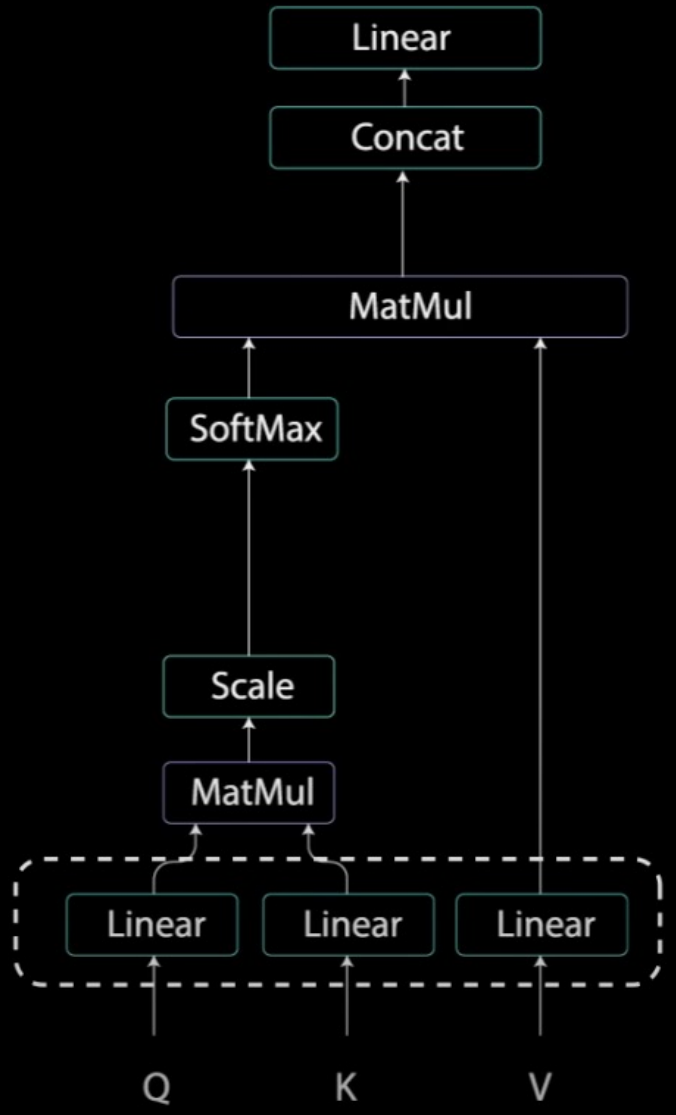
http://jalammar.github.io/illustrated-transformer/

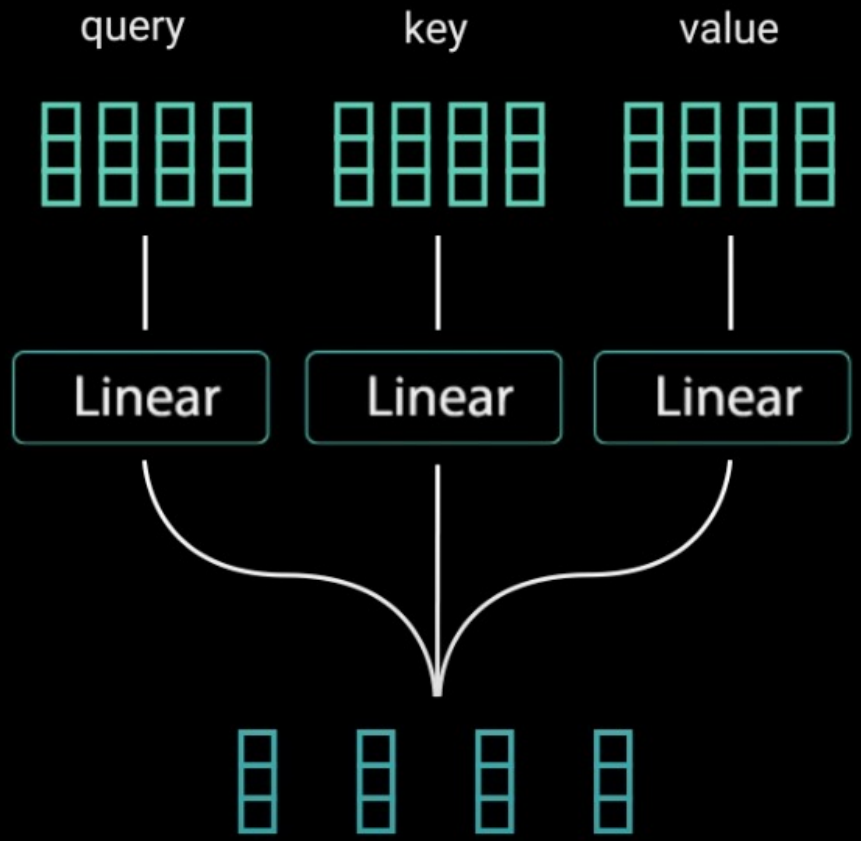
Encoders



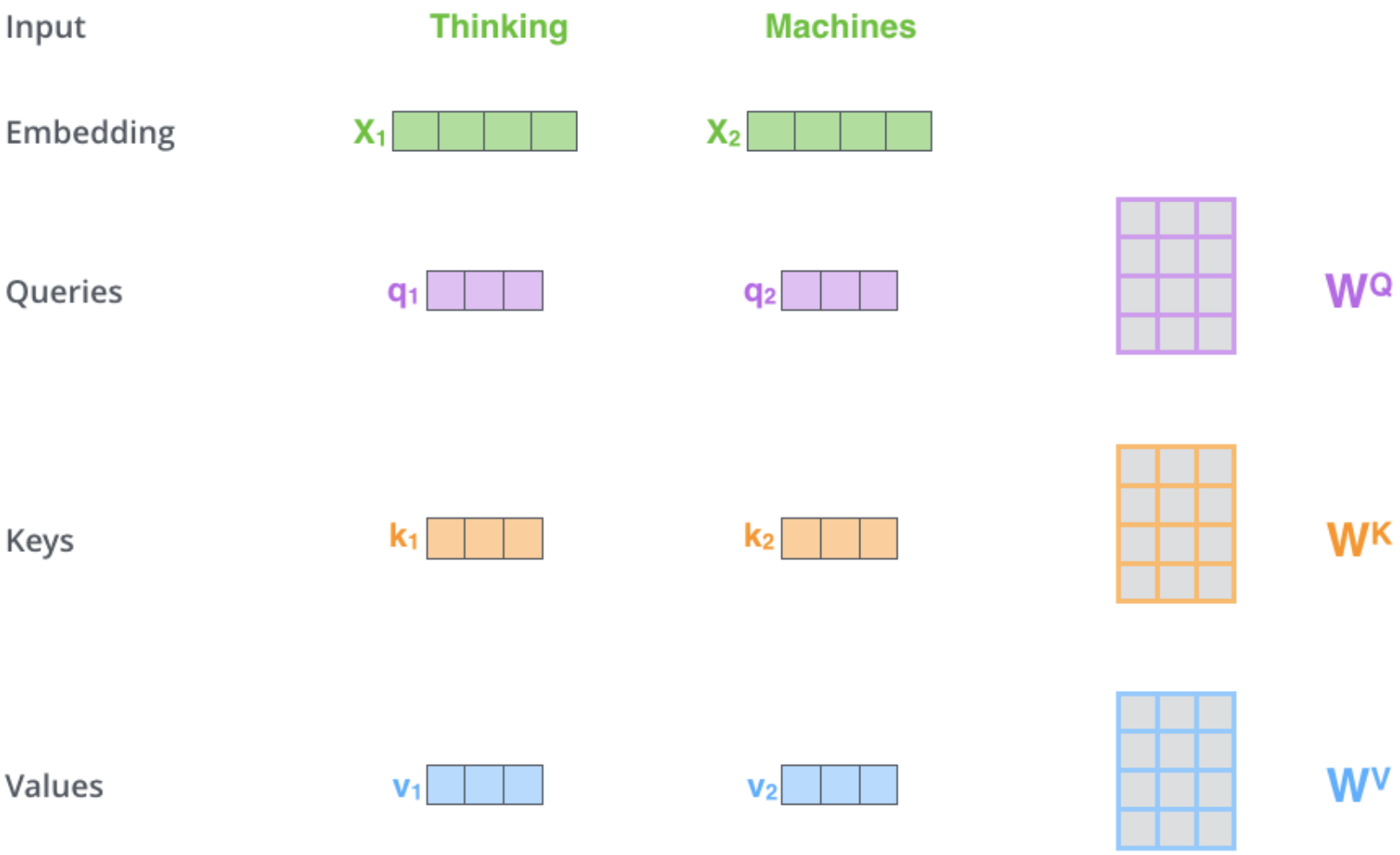
http://jalammar.github.io/illustrated-transformer/

Auto-attention Layer



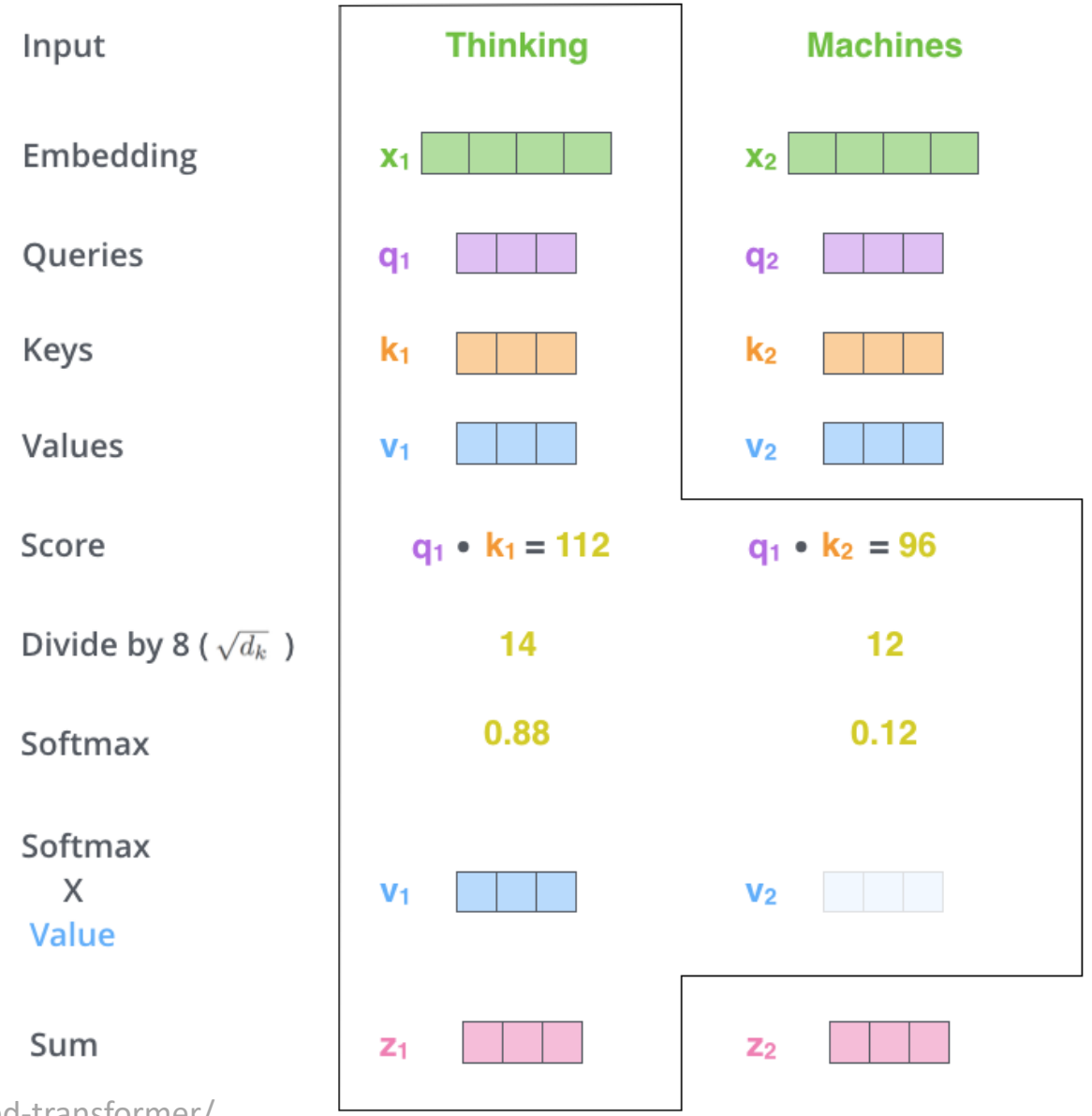


Auto-attention Layer



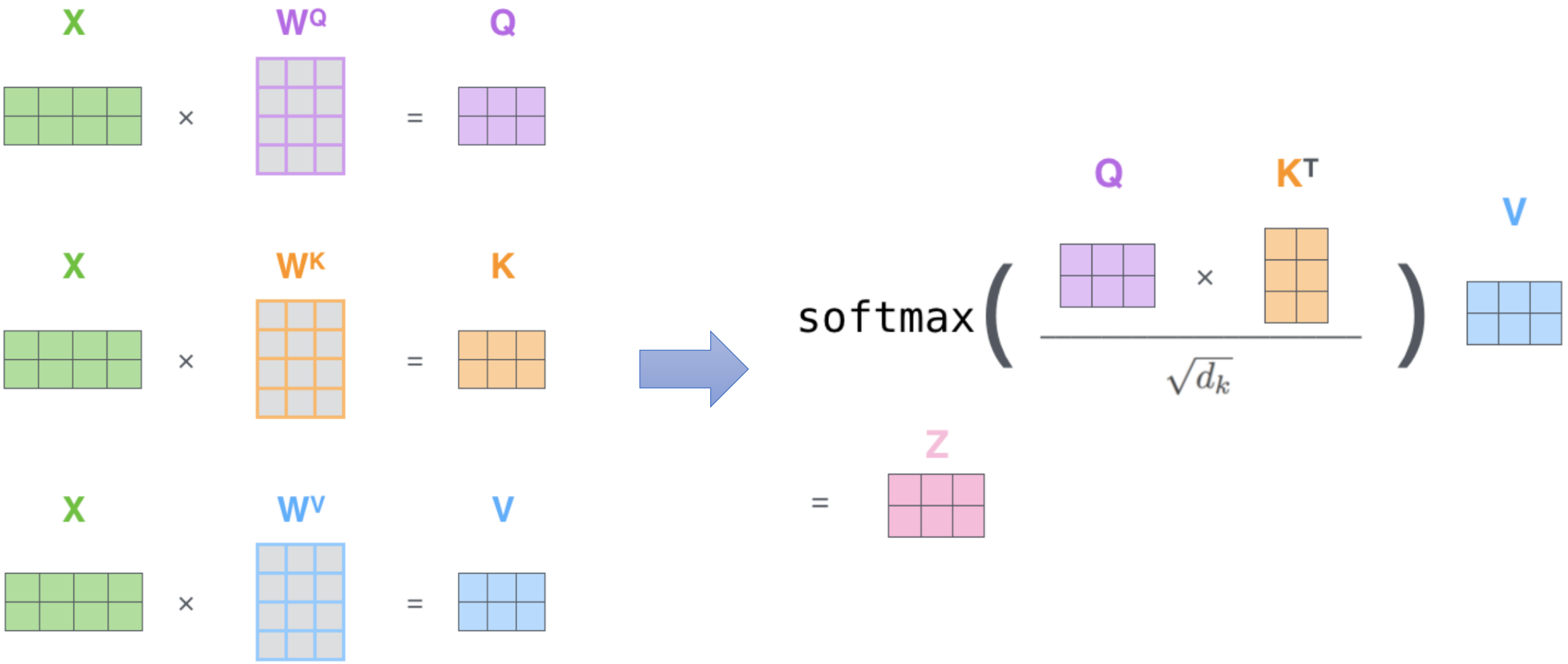
http://jalammar.github.io/illustrated-transformer/

Auto-attention Layer



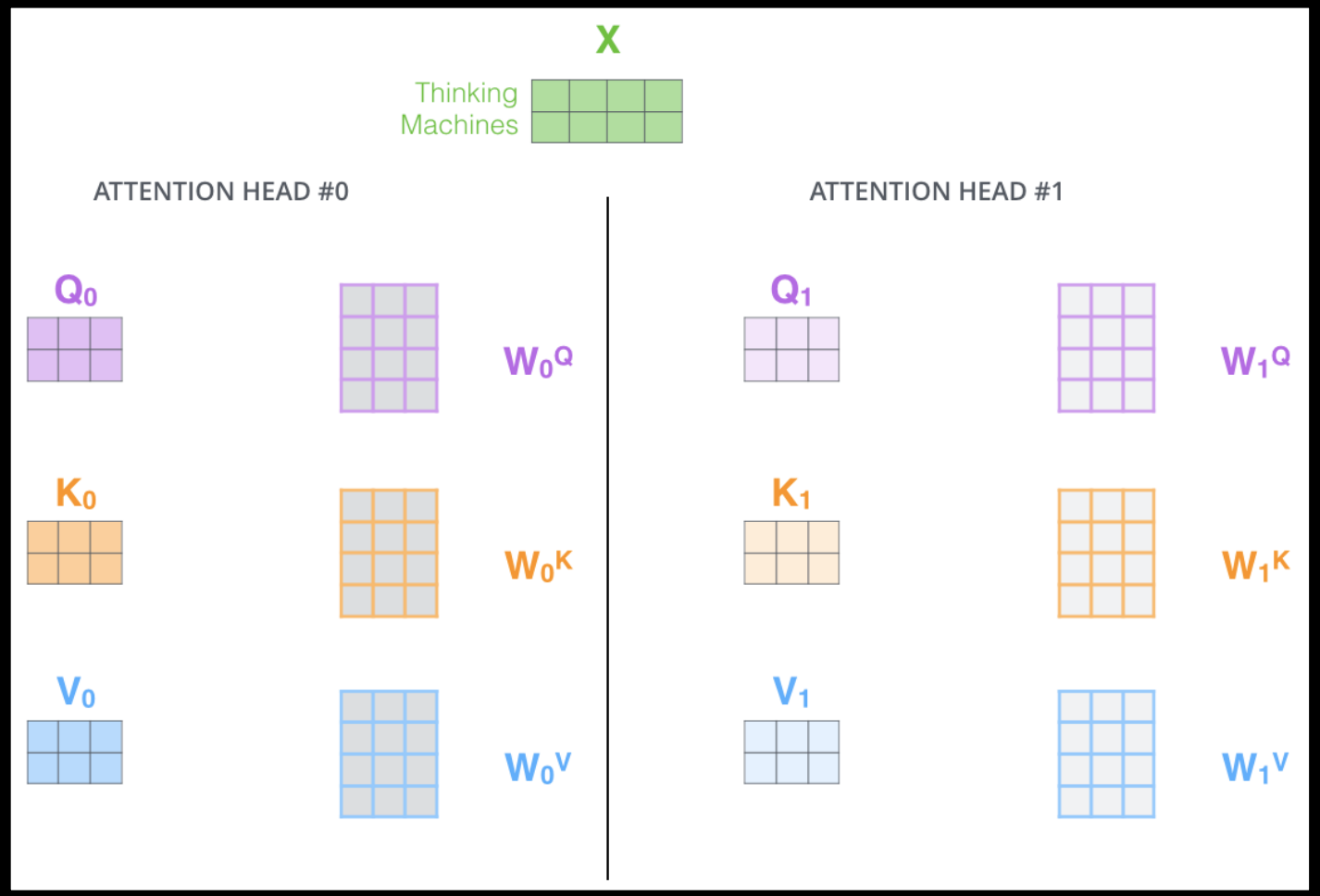
http://jalammar.github.io/illustrated-transformer/

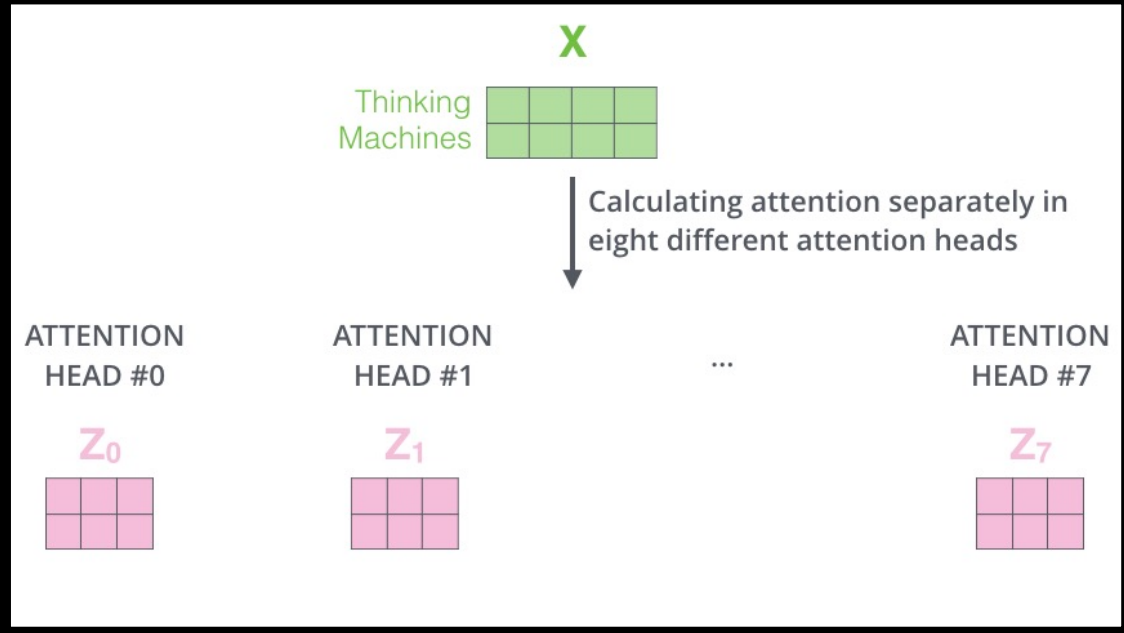
Auto-attention Layer (matrices)



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

Attention Heads

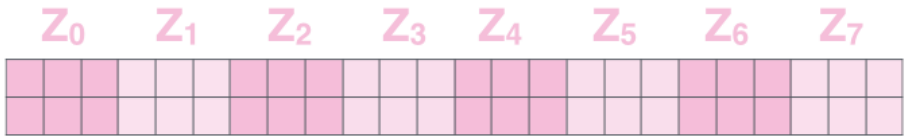




http://jalammar.github.io/illustrated-transformer/

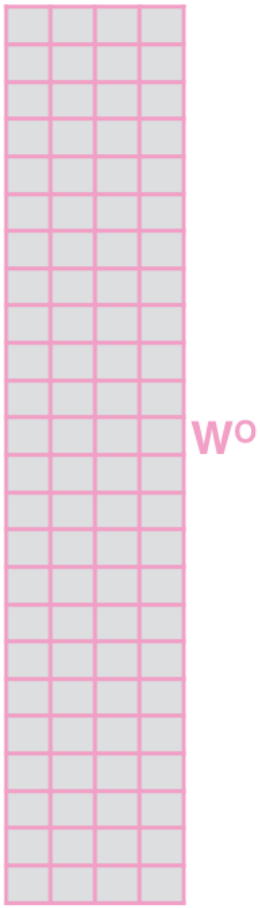
Attention Heads

1) Concatenate all the attention heads



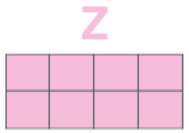
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



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/

All matrices in one place

1) This is our input sentence*

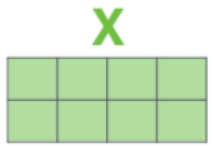
2) We embed each word*

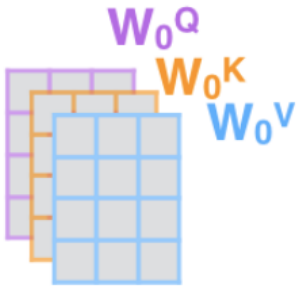
3) Split into 8 heads. We multiply X or R with weight matrices

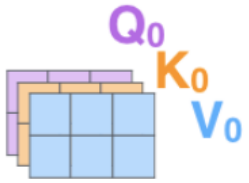
4) Calculate attention using the resulting Q/K/V matrices

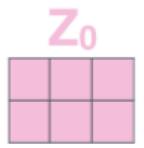
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

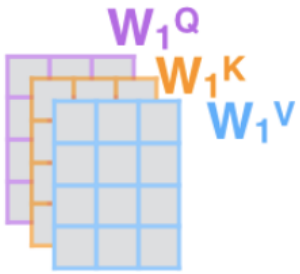
Thinking Machines

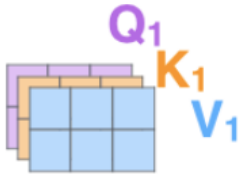


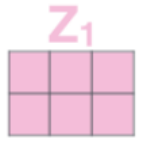


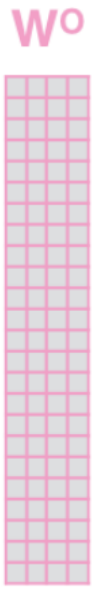


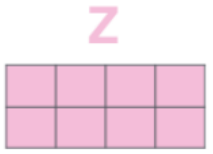




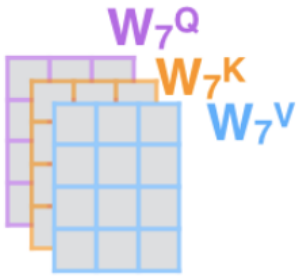


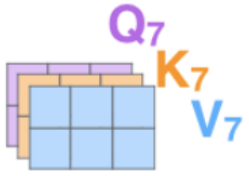


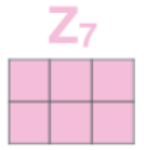




* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

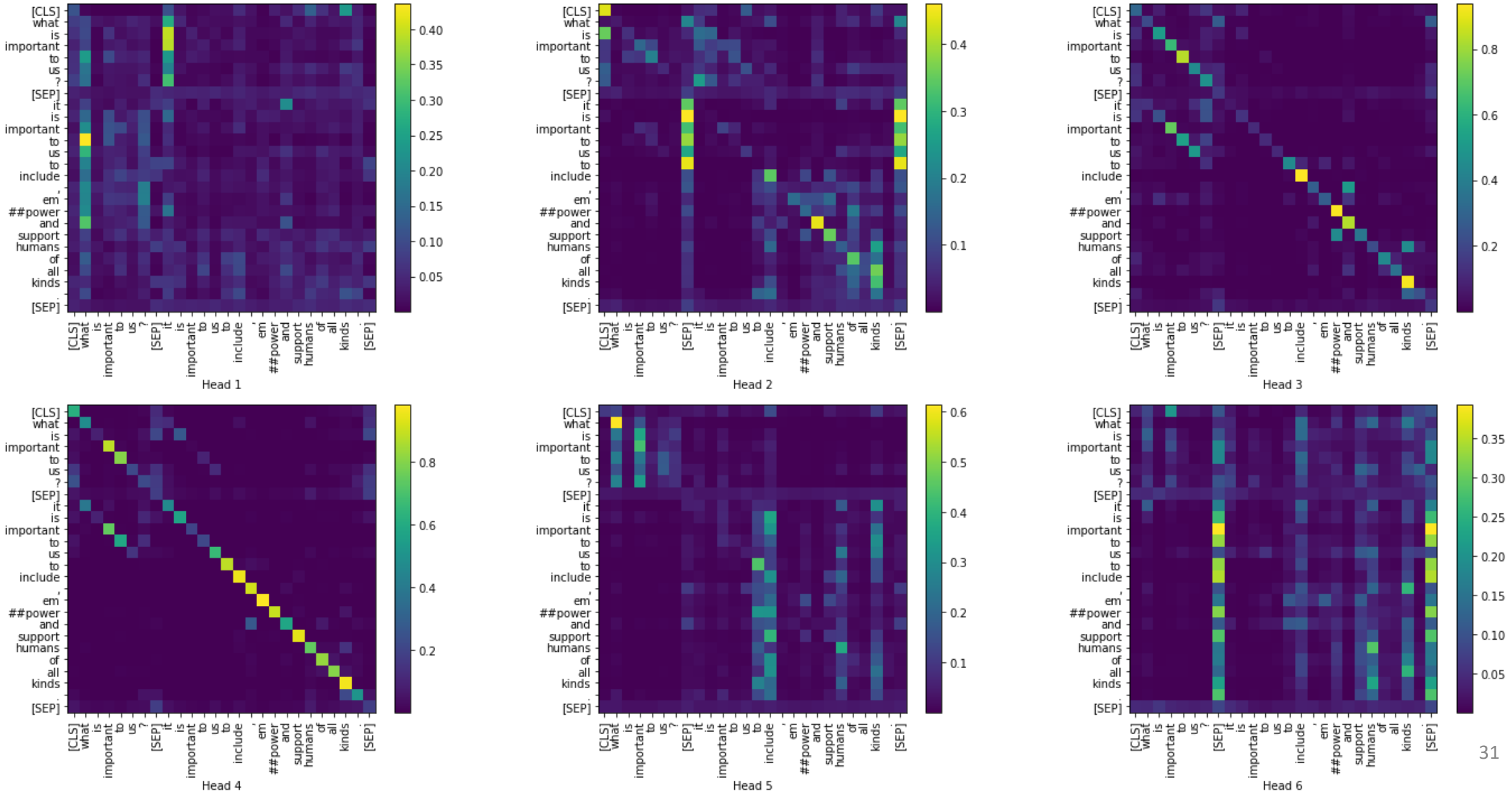




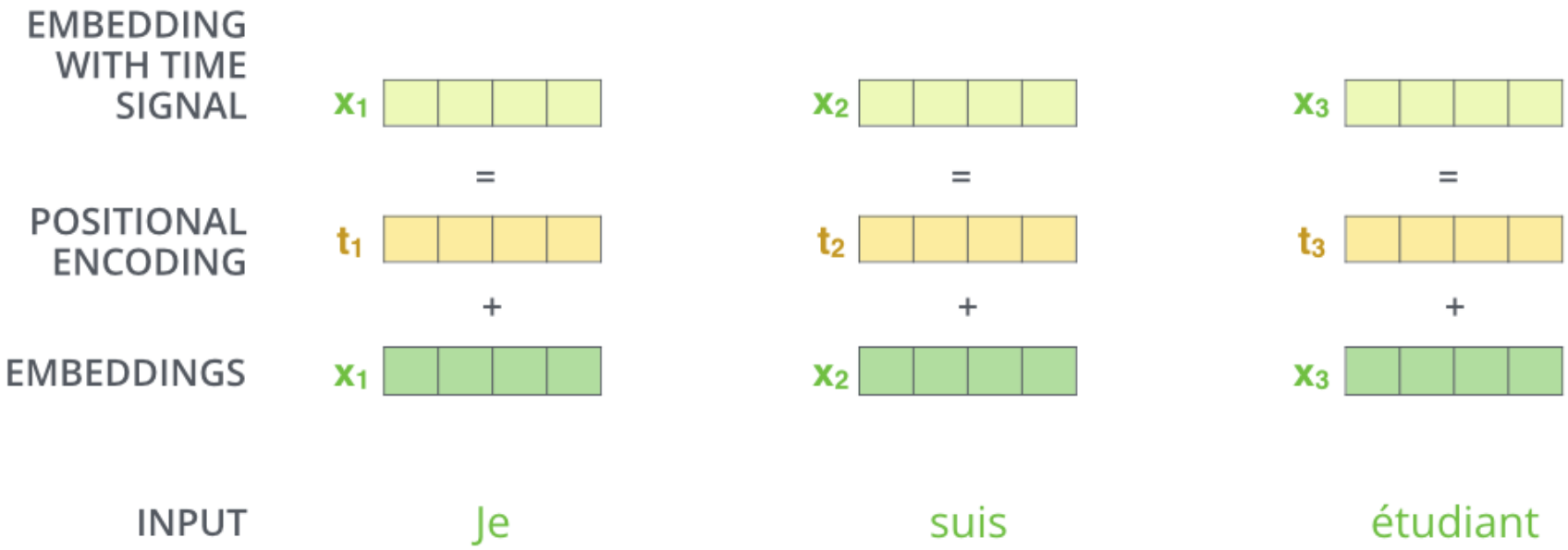


http://jalammar.github.io/illustrated-transformer/

Attention Heads

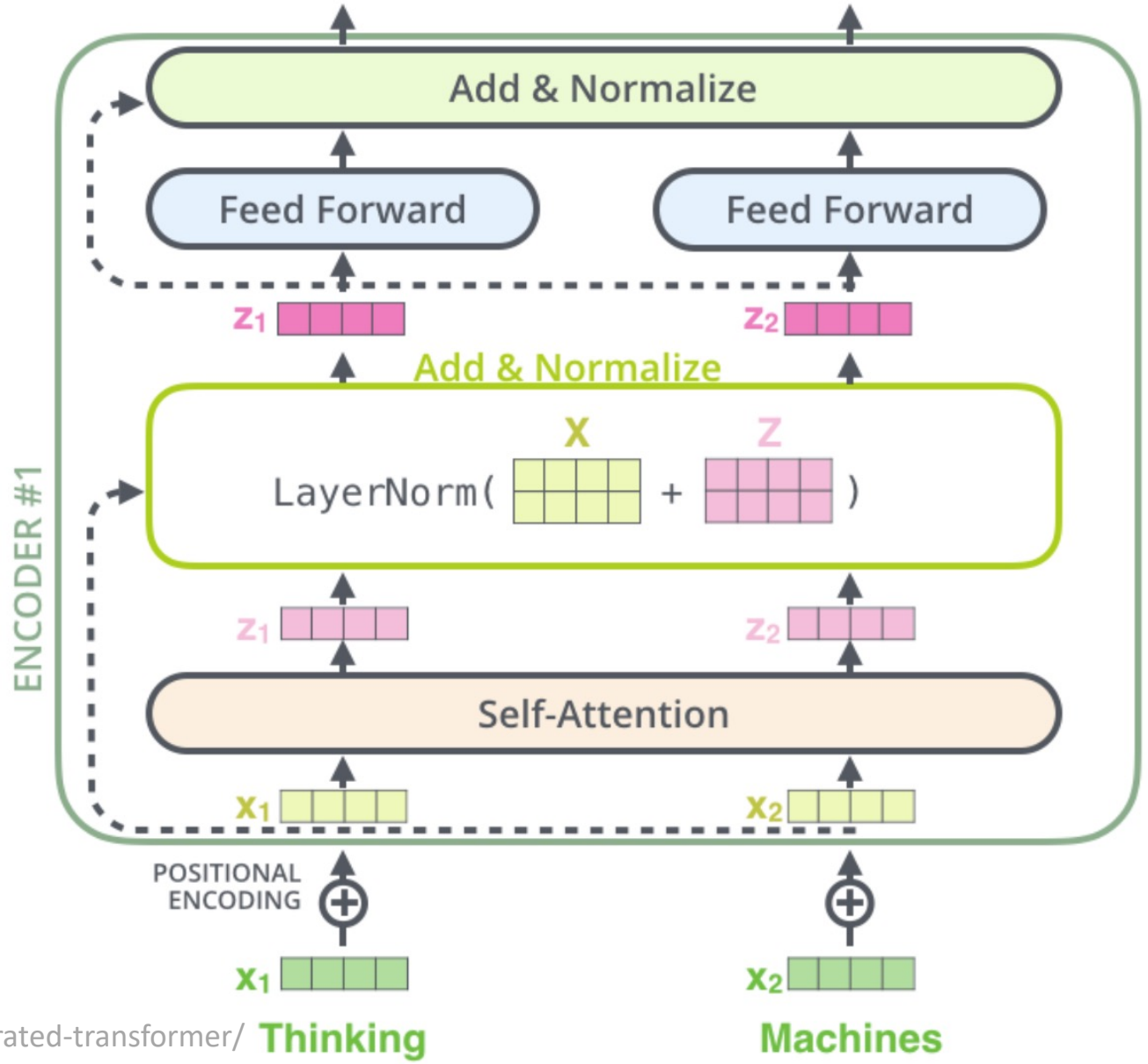


Positional Embeddings



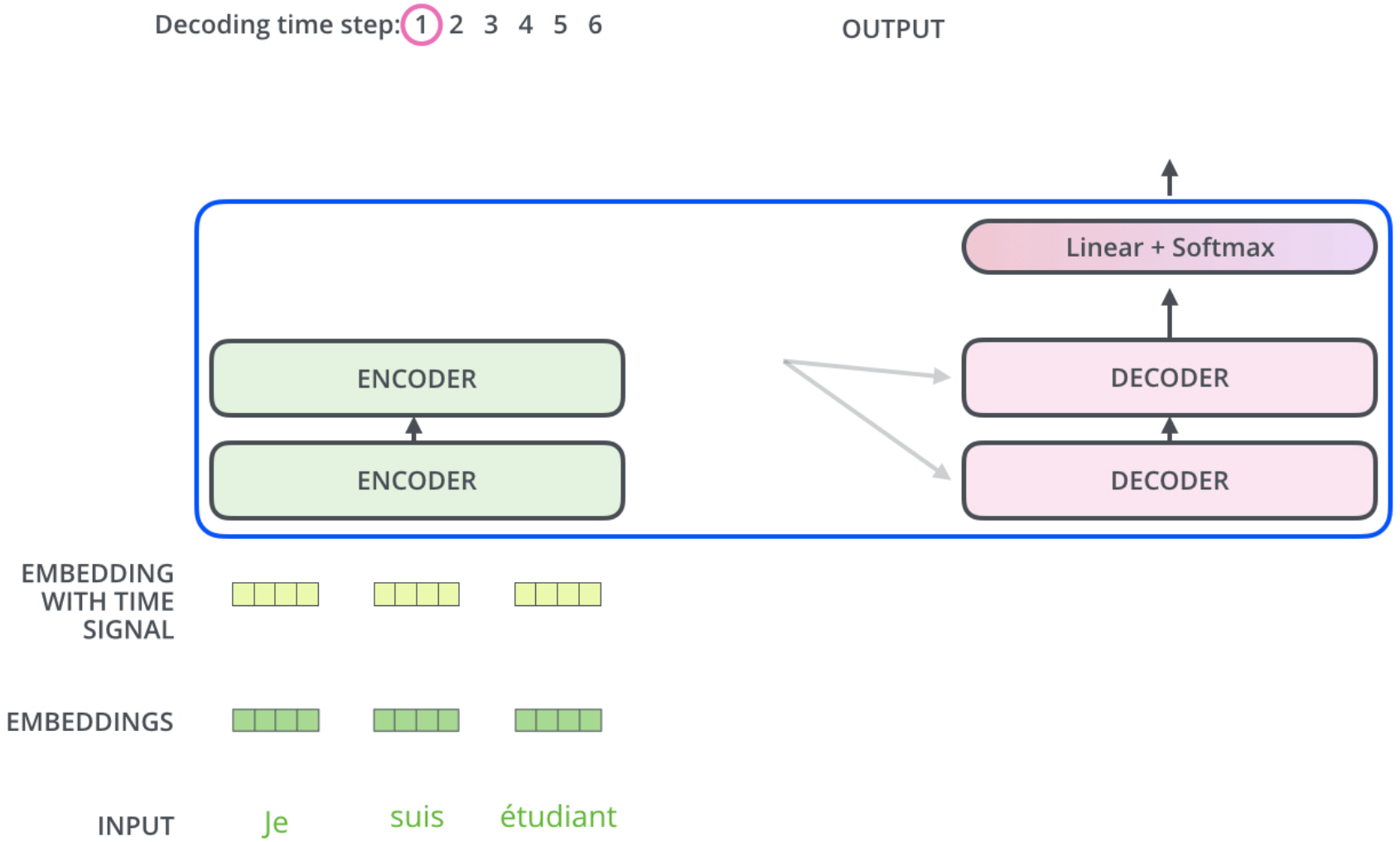
http://jalammar.github.io/illustrated-transformer/

The residuals



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

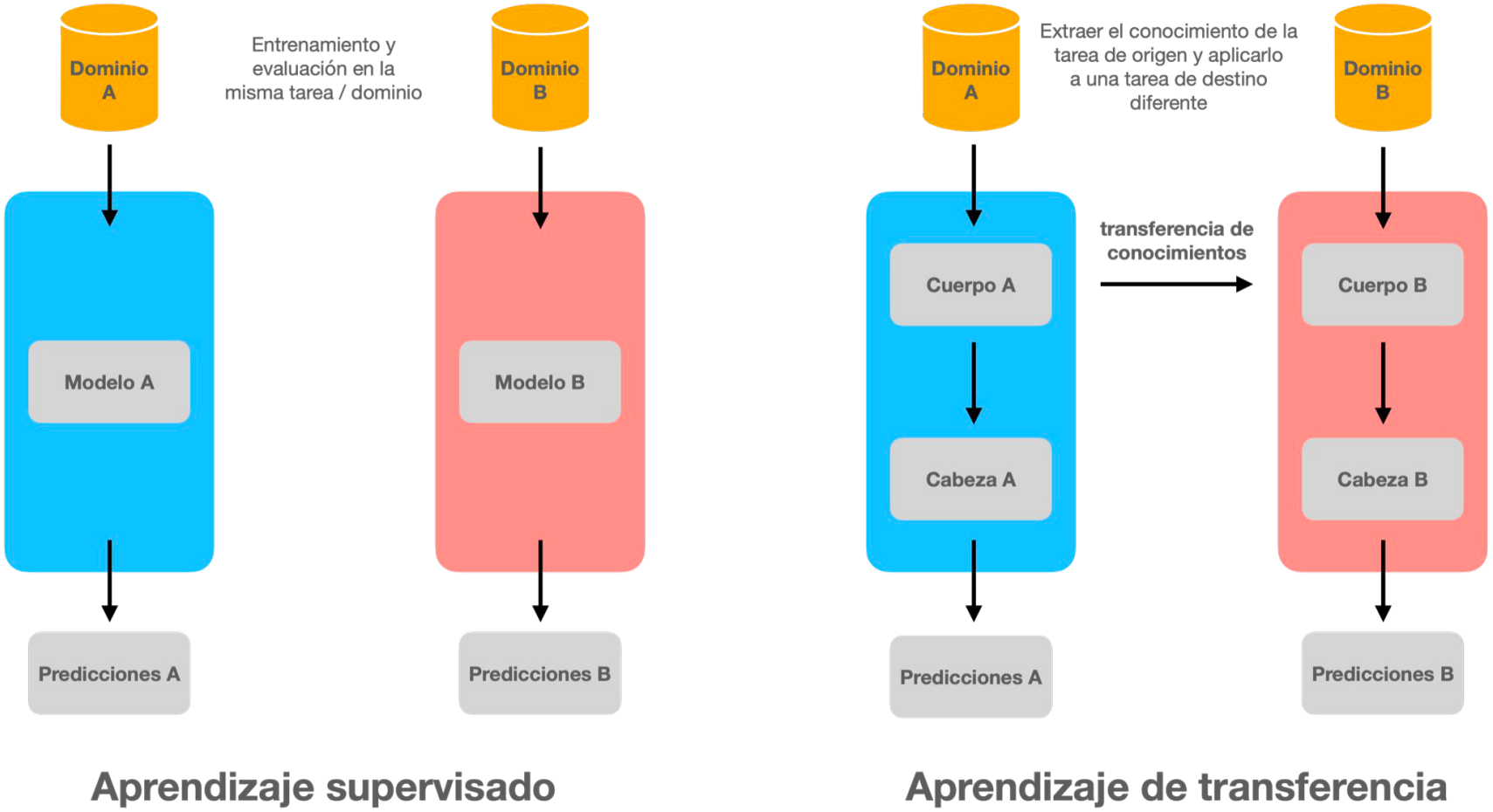
Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

Transfer Learning

Transfer Learning

It is a way to leverage a model trained for one task, in another task that is unlikely to have as much data available.



https://somosnlp.org/nlp-de-cero-a-cien/sesion-04

Transfer Learning

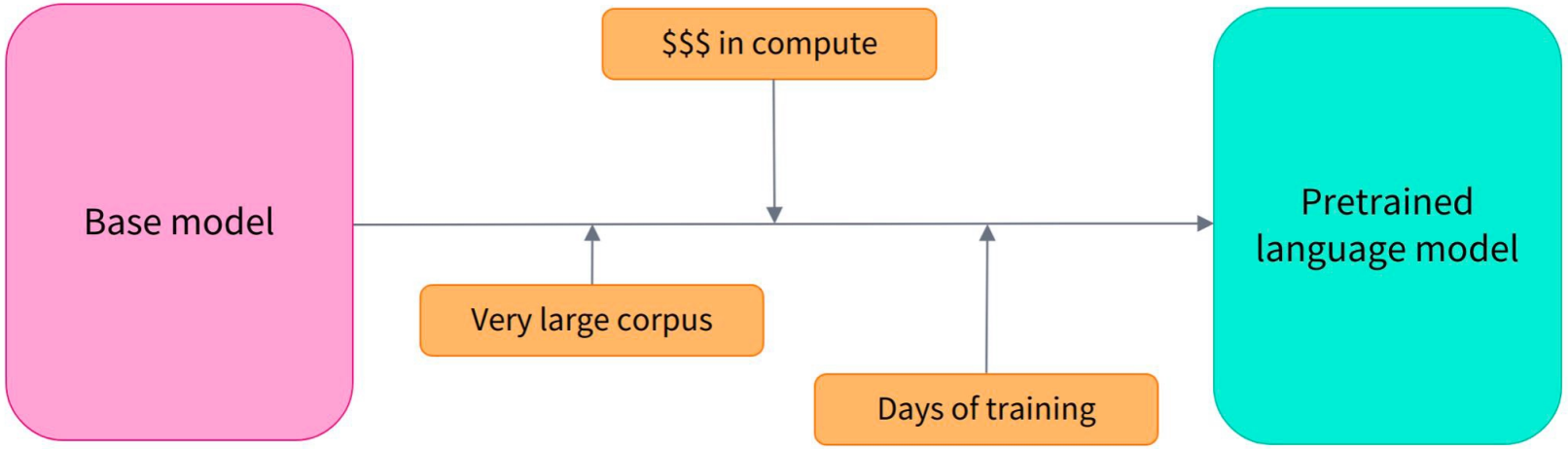
A pre-trained model can be exploided in the following ways:

- **Feature extraction**. The model is used to obtain a representation of the data and the data is used in a model to solve a specific task.

- **Fine tuning**. The weights of the pre-trained model are adjusted to fit the specific task. Typically a very small learning rate and only a few epochs are used.

Pretraining

- In this step we train a model from scratch
- It requires a lot of data and a lot of computation
- After training, the "weights" are reused in various tasks (text classification etc.).



https://somosnlp.org/nlp-de-cero-a-cien/sesion-04

Self-Supervised Learning

- Some pre-trained language models were not created to solve a particular task, but are intended to create a **model** to be reused in other tasks.

- In PLN, pretext tasks are commonly used to train such models.

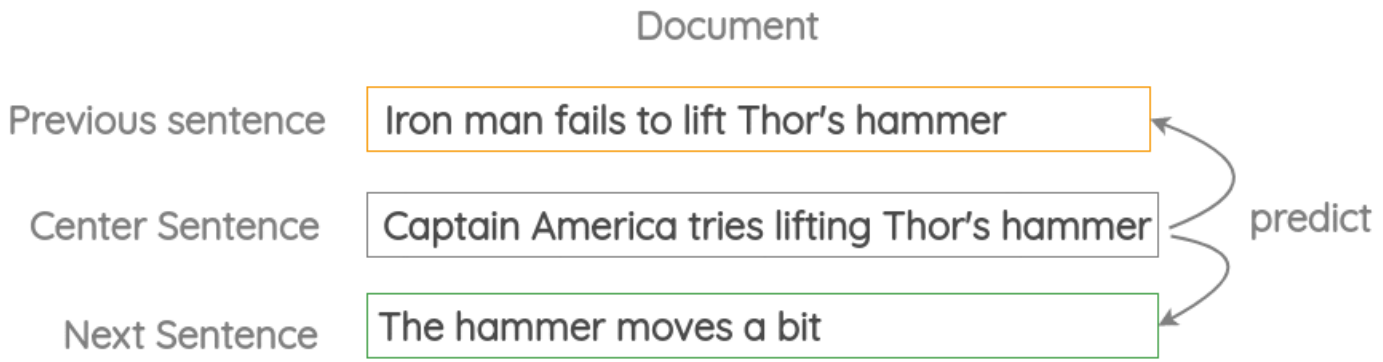
- These tasks are of the self-supervised type, since they seek to take advantage of the large amount of existing textual data without the need for manual labeling.

Pretext Tasks in NLP

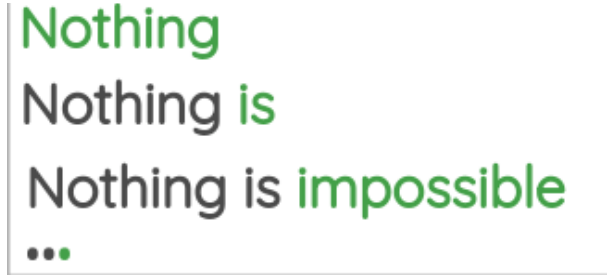
- Prediction of a central word

- Context prediction

- Neighbor Sentence Prediction

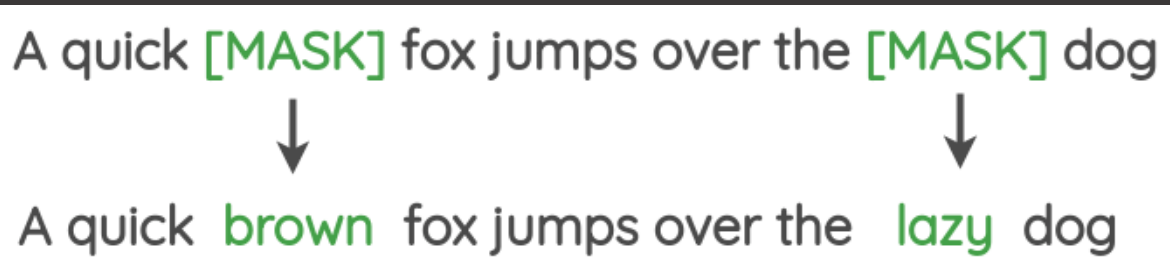


- Language modeling

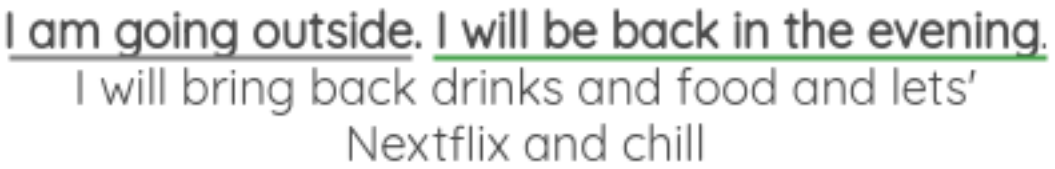


Pretext Tasks in NLP

- Masked language modeling

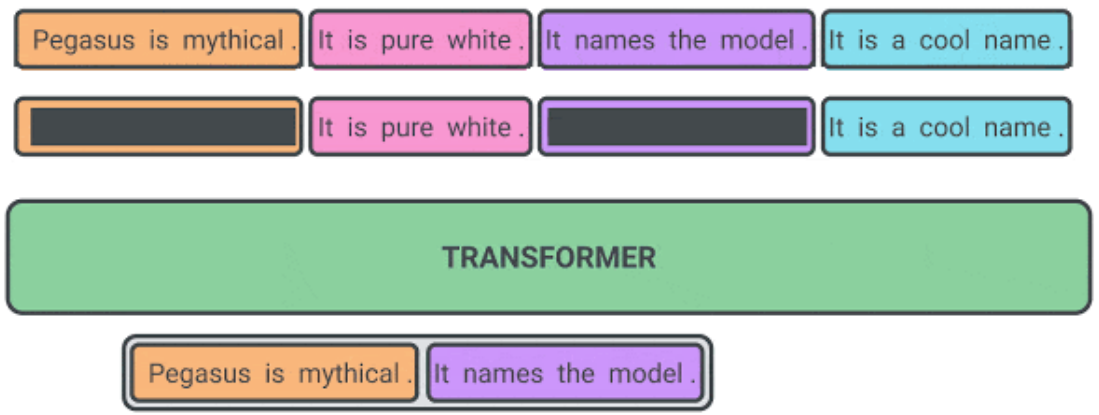


- Next sentence prediction



- Sentence order prediction

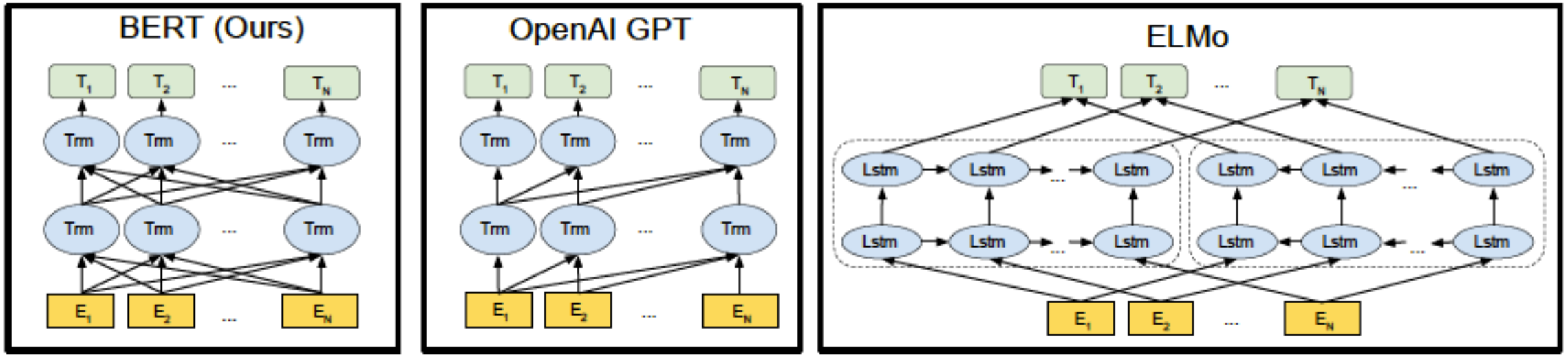
- Sentence prediction



BERT

Bidirectional Encoder Representations from Transformers
Devlin et al 2019

https://www.aclweb.org/anthology/N19-1423/



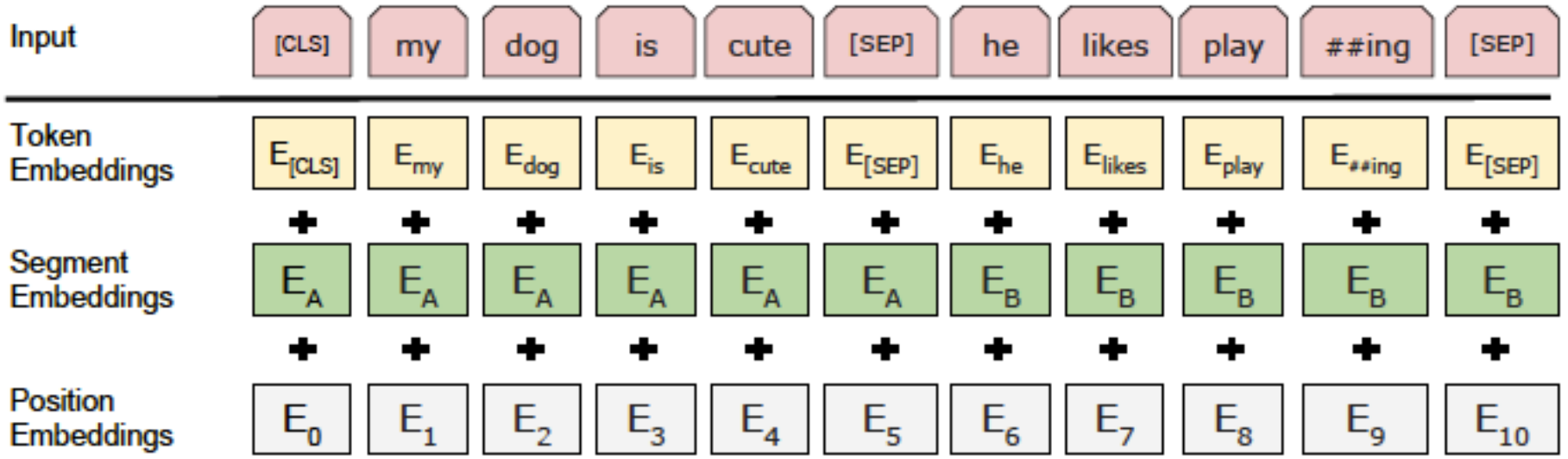
BERT - Arquitectura

BERT consists of the Transformer architecture encoder.
It has stacked encoder blocks.

The article proposes two versions of the model:

- Base
  - 12 layers
  - 12 heads
  - 768 Hidden States Dimension
  - 110 Million of parameters
- Large
  - 24 layers
  - 16 heads
  - 1024 Hidden States Dimension
  - 340 Million of parameters

BERT - Input


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

BERT - Pretraining

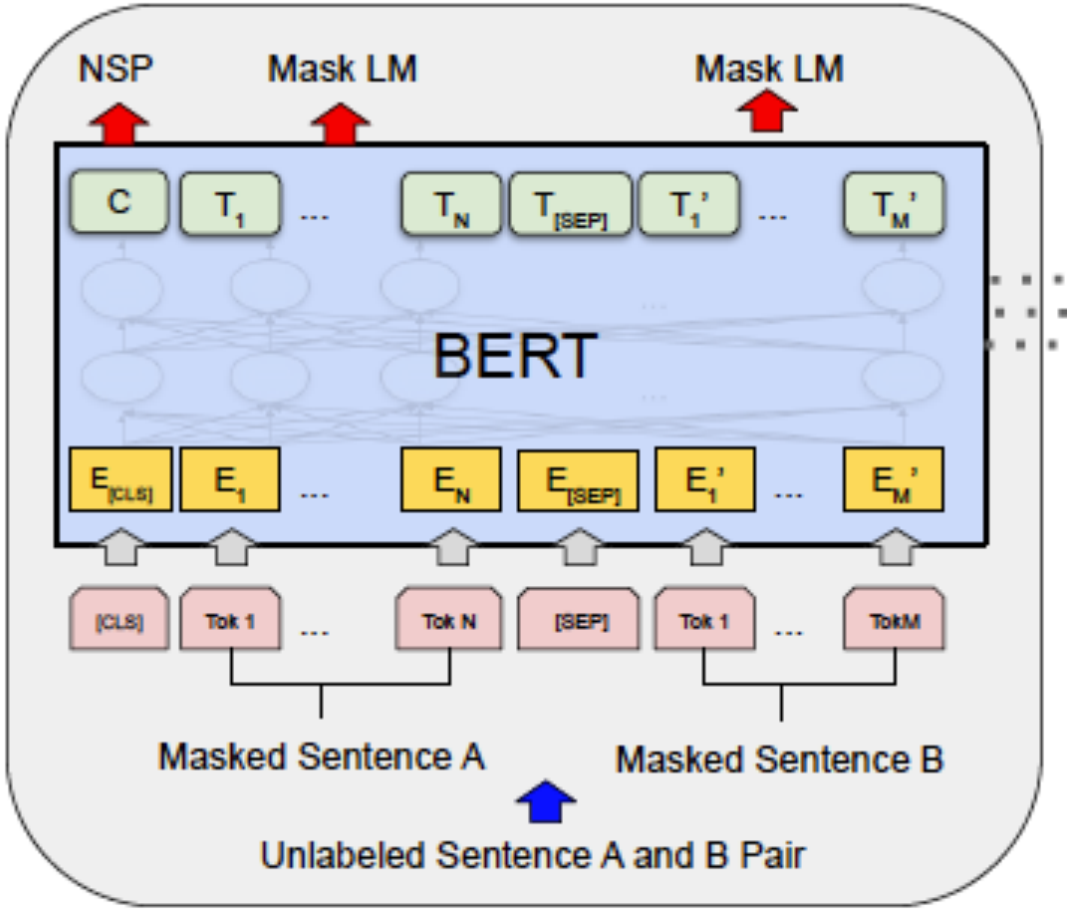
BERT used two pretext tasks for its pre-training.

- Masked language modeling.
- Next sentence prediction.

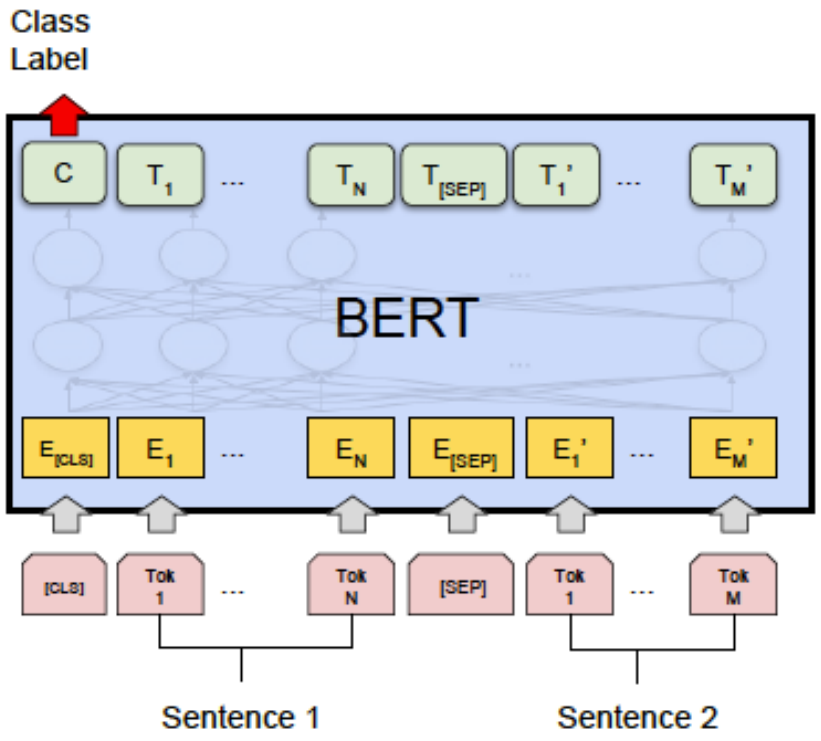
Data

- Wikipedia 2.5 billion words.
- BookCorpus 800 million words.

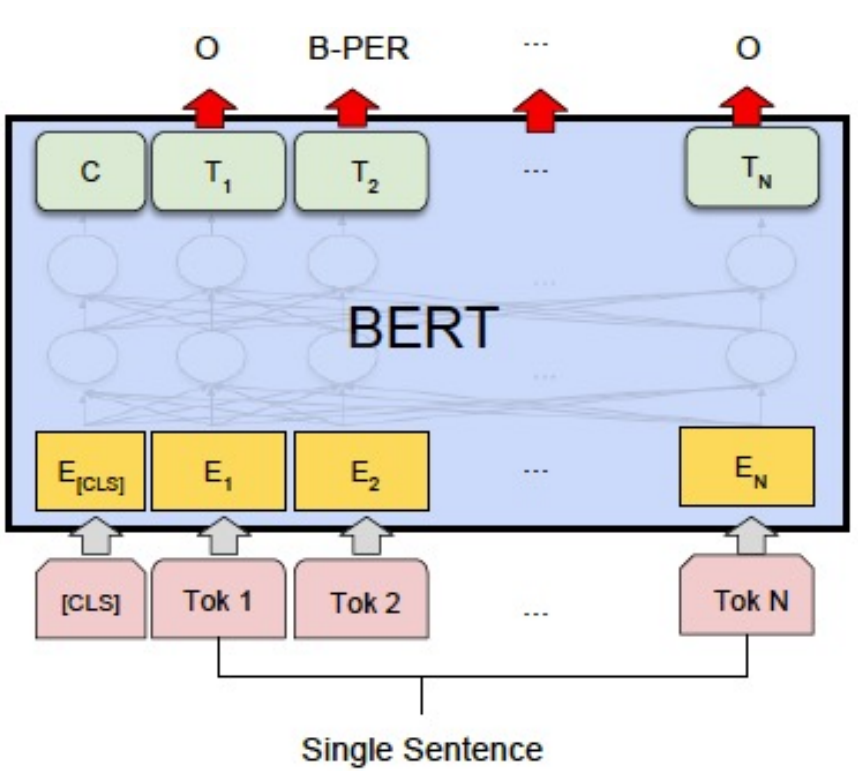
Training for 4 days.



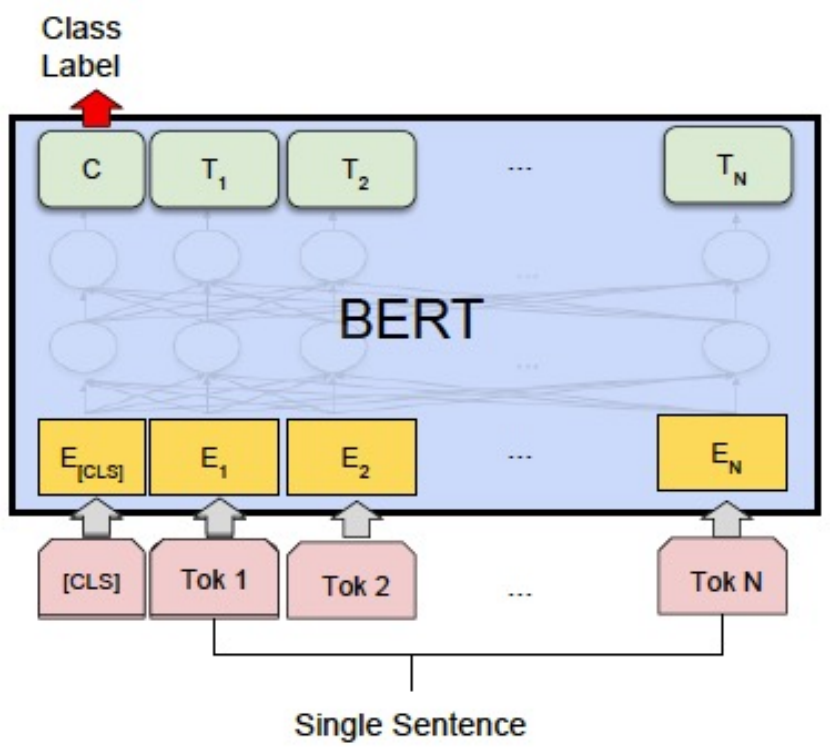
BERT – Fine-tuning



Classification of a pair of sentences



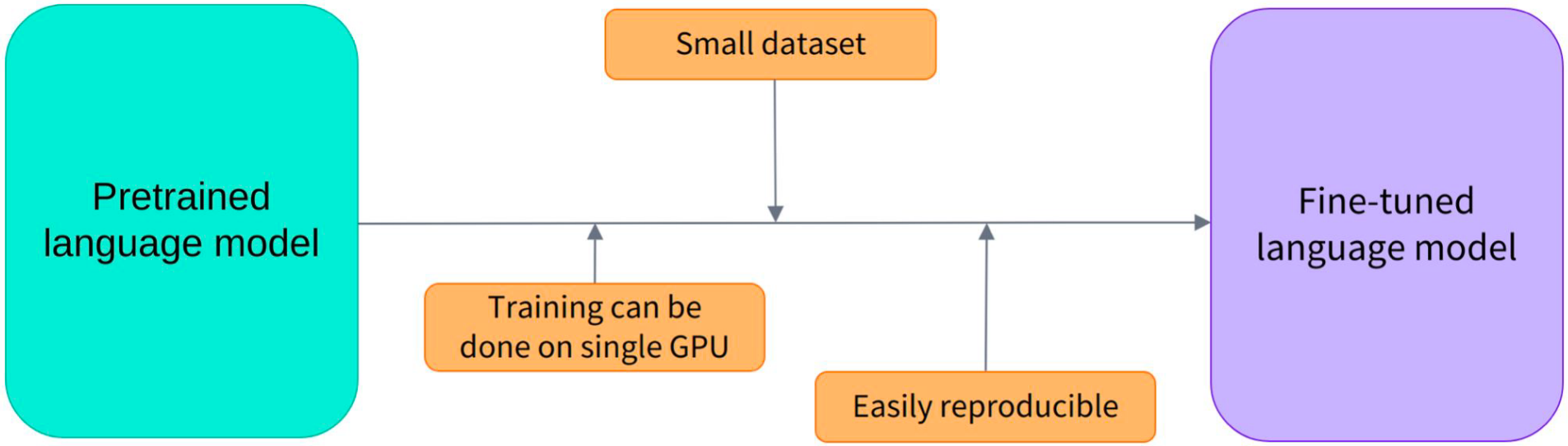
Sequence labeling



Classification of a sentence

Fine-tuning

- Use the weights of the pre-trained model as "body".
- Adding a task-specific "head" (e.g. fully connected network + softmax)
- Fine-tune head weights with supervised training



https://somosnlp.org/nlp-de-cero-a-cien/sesion-04

Youtube videos
BooksCorpus
SWAG,IMDB,Twitter
Knowledge Graph
Monolingual Corpora (104 languages)
Biomedical corpus
Corpora/Data
Clinical notes/EHR
Hierarchical diagnostic codes
Scientific publications
English Wikipedia
Graph Neural Net
SciBERT
ERNIE(1)
M-BERT
Pre-trained models
ClinicalBERT
ERNIE(2)
G-BERT
BioBERT
VideoBERT
BERT
TransBERT
Generic & Domain specific NLP tasks (e.g. NER)
DocBERT
Relation classification
PatentBERT
Code Switching (e.g. English/Hindi mix sentences)
Prediction tasks (e.g. Hospital readmission)
Fine-tuned models
Video captioning (classification)
Classification tasks (e.g. medication recommendation)

https://towardsdatascience.com/a-review-of-bert-based-models-4ffdc0f15d58

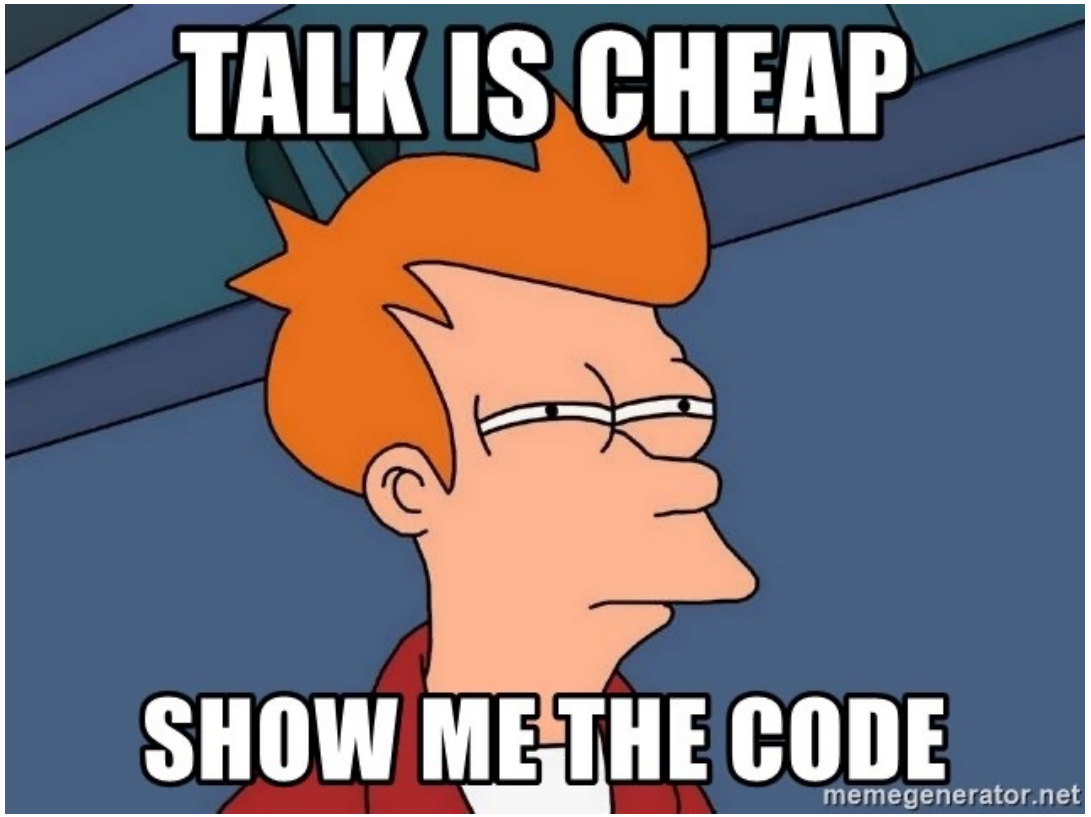
OtherTransformers

BERT is not the only pre-trained model based on Transformer. There are many variants that try to improve some aspects such as performance and resource consumption.

- **XLNet**. Improved pre-training methodology, more data and more computational power. Report better results than BERT.
- **RoBERTa**. The developers of this model considered BERT to be "under-trained". They adjusted the hyperparameters, eliminated the next-sentence prediction task. They used more text for training.
- **DistilBERT**. It learns avern approximate version of BERT, maintains 97% of its performance using half the parameters. For training it uses a technique known as distillation.
- **GPT-2**. Originally GPT used only the Transformer architecture **decoder**, so it is unidirectional. For training, 40GB of text was used. GPT-2 model now bidirectional, it is trained on next word prediction on 40 terabytes of text.

Hand on the code!

We will use the Huggingface library to implement a sentiment analysis model on the TASS 2020 corpus:
https://huggingface.co/docs/transformers/index



Jupyter notebook