Hands-on Introduction to Deep Learning

Methodology

INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

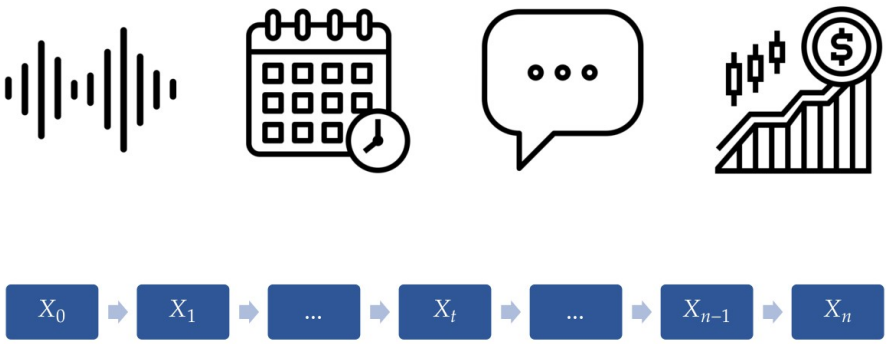
Objectives of this sequence :

- Learn how to approach sequential data
- Learn about RNNs and Transformers to solve sequential data tasks (and more…)

Duration: 1h30m
Document annex : Demo code

Aspects addressed :

- Sequential data definition
- RNN definition
- RNN architectures
- Transformer architecture
- Attention mechanism
- Transformer types
- Transformers in computer vision



Sequential Data

Sequential data are ordered sequences of objects with relationships between these objects.

Many types of sequential data exist:

- Audio
- Text
- Video
- Physical quantities
- …

It is also possible to approach problems where the data are not sequences (image classification) in a sequential form (sequences of image patches).

Warning:
Using this type of data to train machine or deep learning models requires a different pre-processing than with structured data. In particular, the generation of training, validation and test sets with sequential data imposes a well thought-out separation because the sequences cannot be mixed without constraint.

Stock market

	day 1	day 2	day 3
asset 1	9.77	79.94	64.13
asset 2	47.66	74.07	70.90
asset 3	94.25	76.34	99.95
asset 4	41.19	9.99	89.50
asset 5	65.44	63.79	67.14

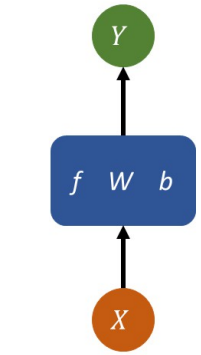
Text

I	am	learning	.
0,83	0,65	-0,90	-0,04
-0,53	0,81	-0,61	-0,12
0,24	-0,14	0,58	0,66
-0,31	0,32	0,37	-0,11
-0,53	0,50	-0,96	0,48
-0,34	-0,85	0,19	-0,78
-0,79	0,53	-0,31	-0,28
-0,23	-0,13	0,33	0,45
0,95	0,53	0,74	-0,24
-0,60	0,04	-0,96	-0,96

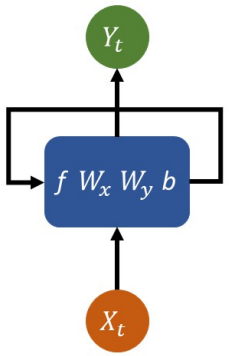
Sequential Data - Examples

Stock market prices can be considered as sequential data because usually stocks follow trends over time. This is also a time series.

Textual data are also considered as sequential data because these are words, but also the position of these words in the text, which will give meaning to a sentence.



$$Y = f(W.X + b)$$



$$Y_t = f(W_x.X_t + W_y Y_{t-1} + b)$$

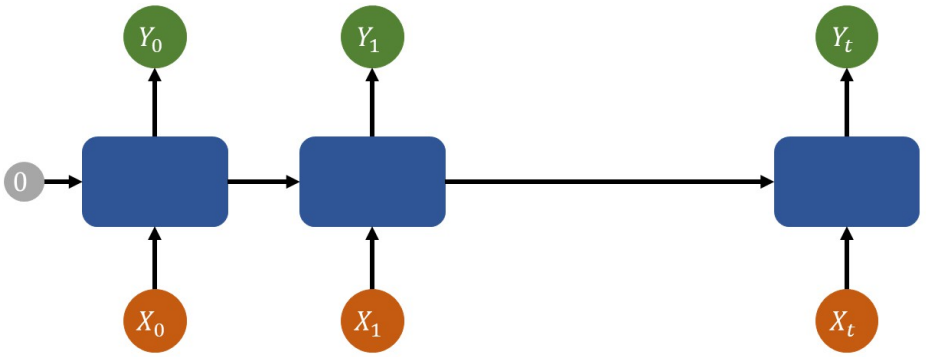
Simple recurrent neuron

A classic neuron, used in dense layers, can be transformed to take into account the sequential aspect.

The output of a neuron is returned as input for the next iteration so we implement a form of recurrence in the neuron.

The neuron, therefore, takes as input:

- An input data (x) at a time t
- The model output at time t-1



Recurrent neuron unfolded

You can unfold the previous representation.

It is thus possible to form a recurrent layer.

The first cell takes as input:

- The first element of the sequence
- A null vector

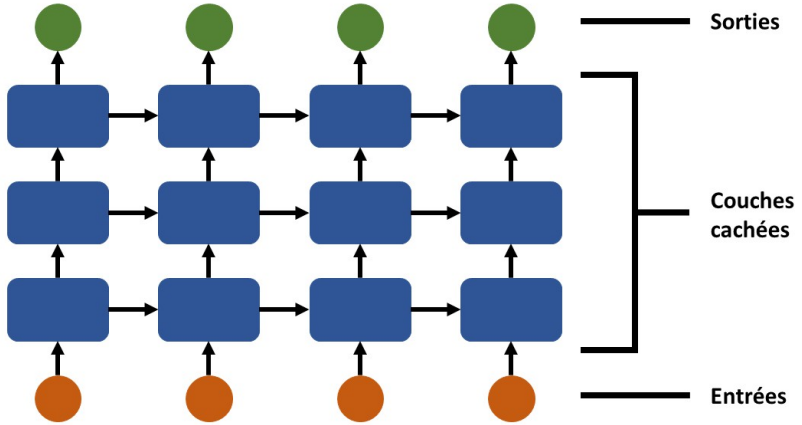
The following cells have as input:

- The corresponding sequence data
- The output of the previous cell

Each cell has, therefore, two inputs which are:

- Concatenated
- Multiplied by the weights
- Passed to the activation function

In the case of “Vanilla” RNN, the activation function is a hyperbolic tangent.



Recurrent Neural Network

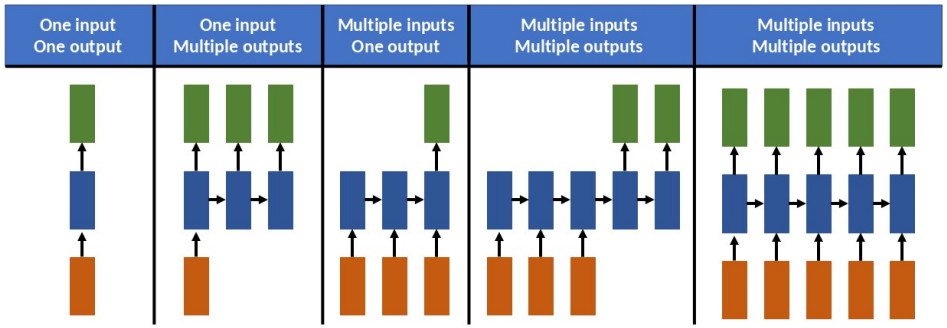
As with classic neurons, it possible to use recurrent neurons to form hidden layers.

Although the model is unfolded in the graphic, it is not possible to calculate the outputs simultaneously because there is a time dependence.

Several techniques are available for the backpropagation. For example, it is possible to backpropagate the gradients partially over part of the sequence by defining an appropriate loss function.

Limits:

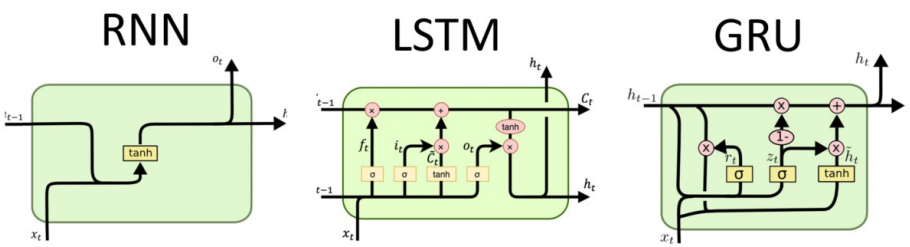
- Slow in training and inference
- Weak parallelization
- Vanishing or exploding gradients
- Short memory



A flexible model type

Depending on the number of inputs and outputs, there are multiple uses:

- Multiple to single: Sentiment classification
- Single to Multiple: Image Annotation
- Multiple to multiple: Translation
- Multiple to Multiple: Series Prediction



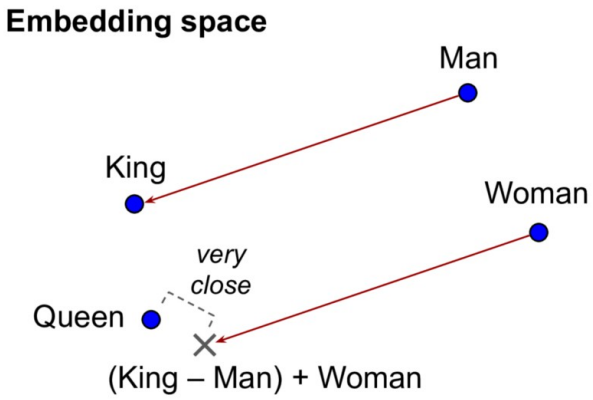
Tembhurne, Jitendra V., and Tausif Diwan. « Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks. » *Multimedia Tools and Applications* 80.5 (2021) : 6871-6910.

Simple RNN vs LSTM vs GRU

There are different types of cells (neurons for vanilla RNN) with their advantages and disadvantages.

LSTM (Long-Short Term Memory) has two outputs to simulate two memories: a short-term memory and a long-term memory. It has several "gate" mechanisms that allow several operations such as resetting the long-term memory (reset) if it's not relevant anymore.

GRU (Gated Recurrent Unit) offers several mechanisms similar to the LSTM (such as the reset gate) but with a single output. It is less accurate (results) than the LSTM but much more efficient (calculation, memory, etc.).



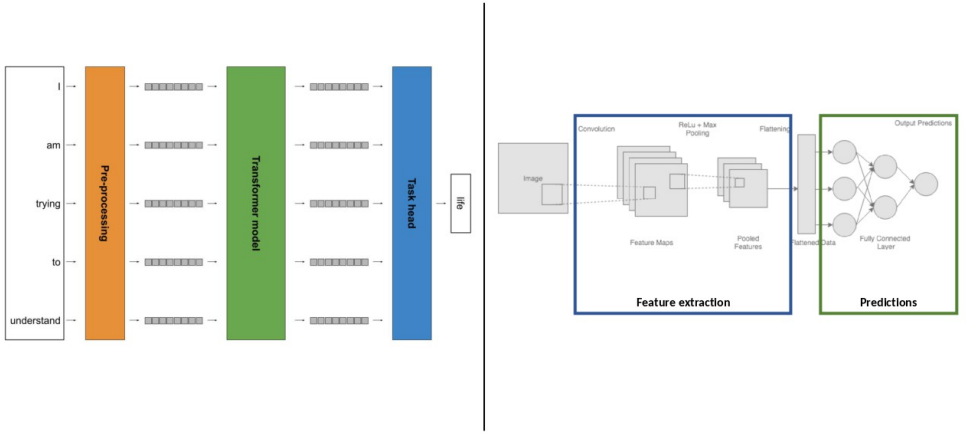
Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2019.

Sequences in NLP

The Transformers which we are going to study can be applied to almost any field. However, we will study the theory with Natural Language Processing examples.

To fully understand what follows, you must first understand how the semantics of words is represented in NLP. Each word is represented by a vector which mathematically represents the meaning of the word. The texts studied are therefore sequences of vectors, thus a matrix.

Operations are possible on these vectors to make sense of them (see slide).

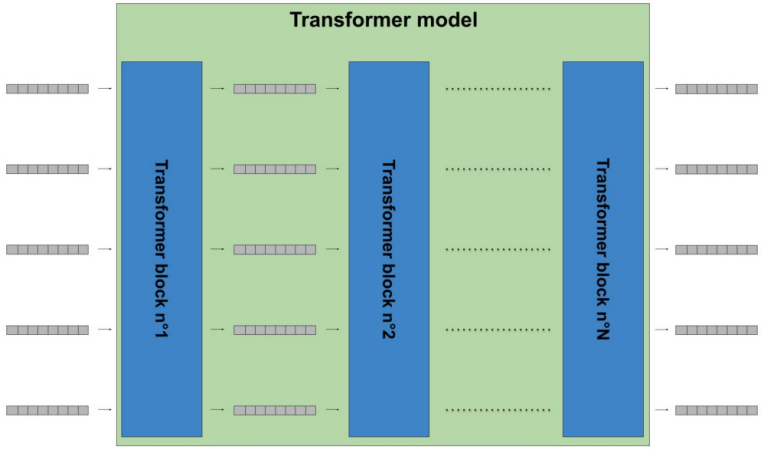


Transformer in a system

Transformers can be integrated into an AI system in the same way that we integrate convolutional layers into these systems.

For convolutional models, we first preprocess the input (often an image) to make it usable by convolutional layers. Then the input goes through the convolutional layers to extract meaningful features (for a neural network). And finally, these features go through a final dense layer (or several) to obtain the desired output for the task.

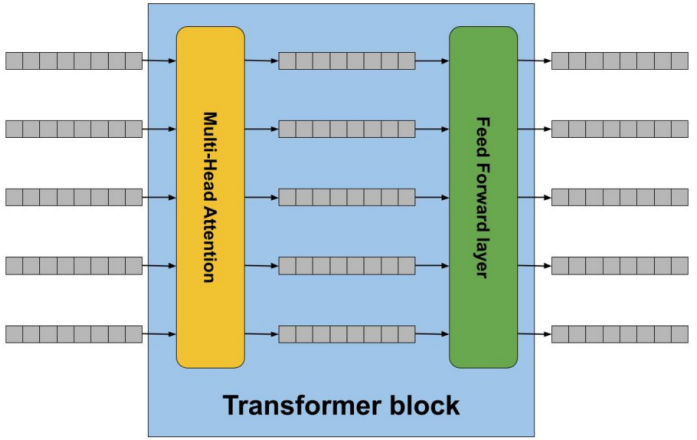
For Transformers, it's the same, except it's not convolution layers that will extract the features but the Transformer itself.



Transformer architecture (1)

We can first see the architecture of Transformers as a series of "Transformer Blocks".

These blocks take as input a matrix of the size defined by the vector input sequence and the output is a matrix of the same size.

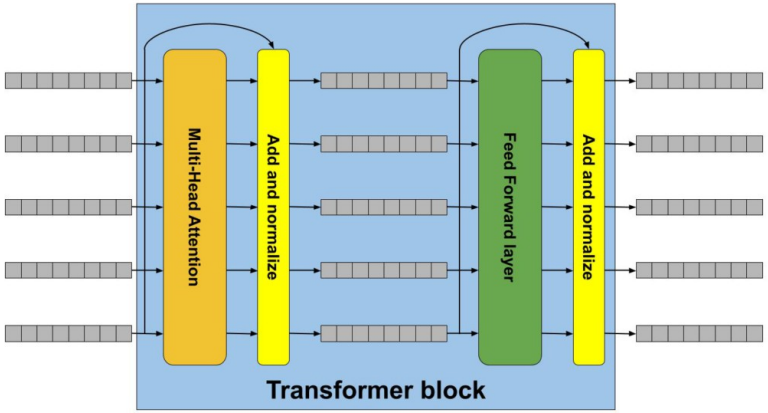


Transformer architecture (2)

The Transformer Blocks consist of two main parts: a Multi-Head Attention and an MLP (Feed Forward Layer) block.

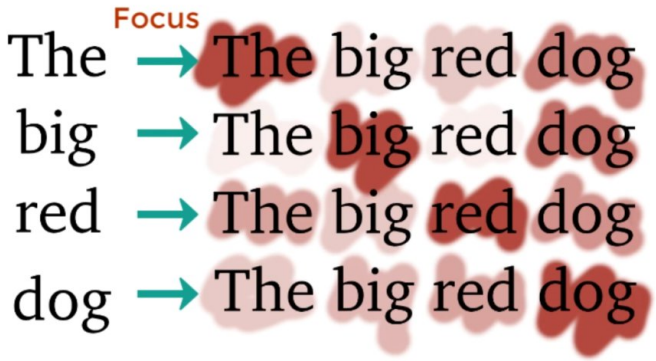
The Attention is the most important part of the Transformer because this is what makes the model a Transformer. It takes a matrix as input and usually outputs a matrix of the same size. We will see a little later what happens inside.

The MLP part is simply composed of a few dense layers of classical neural networks.



Transformer architecture (3)

The “Transformer Block” is more complex than what we have previously seen. It has some operations that can make training more stable and efficient. It all depends on the version.

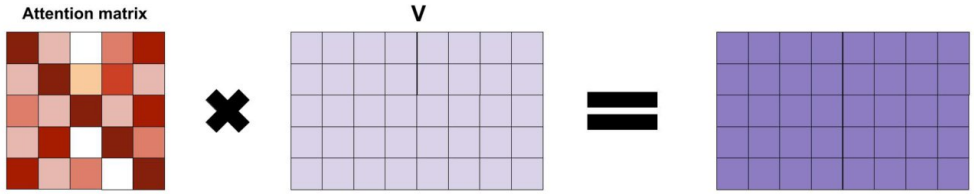


Transformer Neural Networks - EXPLAINED! (Attention is all you need) : https://www.youtube.com/watch?v=TQQlZhbC5ps

Intuition behind the Attention mechanism (1)

The goal of the Attention mechanism is to transform each vector of the sequence so that it contains the information of all the vectors by mixing them with a certain weighting.

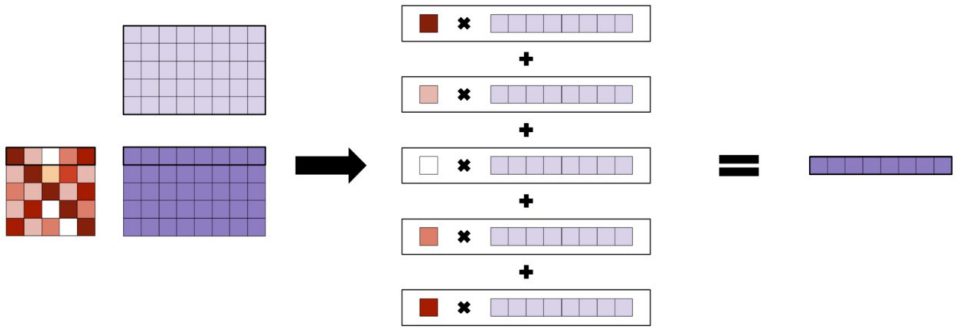
If we take the example of NLP, initially each vector contains the information of the word it represents. But after applying Attention to the sequence, each vector will have the information of the whole text (with a weighting that follows a certain logic).



Intuition behind the Attention mechanism (2)

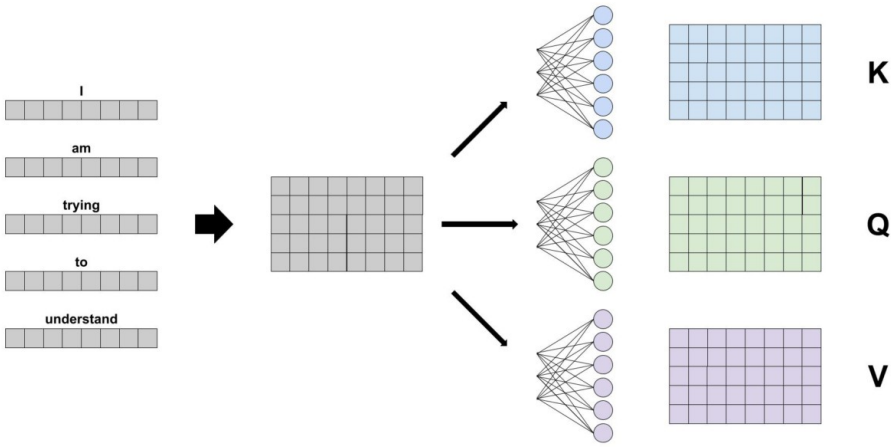
Transformation of each vector is done by multiplying the matrix that represents the sequence by the Attention matrix.

The Attention matrix contains the weights which will be used to weight each vector for their transformation. For each sequence, an Attention matrix is computed in a Transformer during the training and inference.



Intuition behind the Attention mechanism (3)

Here we can see that the first vector of the sequence will turn into a weighted sum of the whole sequence.



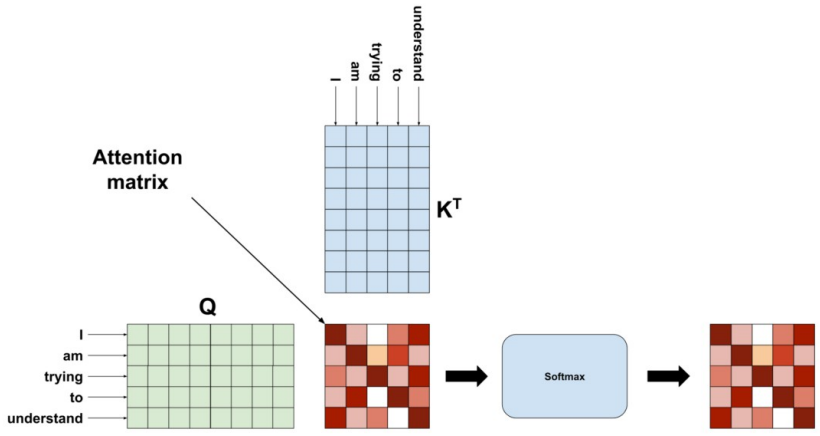
Attention mechanism (1)

We will now see the complete process of the Attention mechanism.

The first part consists of doing three linear transformations, in parallel, to each vector of the sequence in order to obtain 3 new matrices (K, Q, V) which still represent each word of the text but in a different embedding space.

This transformation is important for what follows because it is what will allow a coherent calculation of the Attention matrix.

These are the three neural networks of the linear transformation that are trained during gradient descent.



Attention mechanism (2)

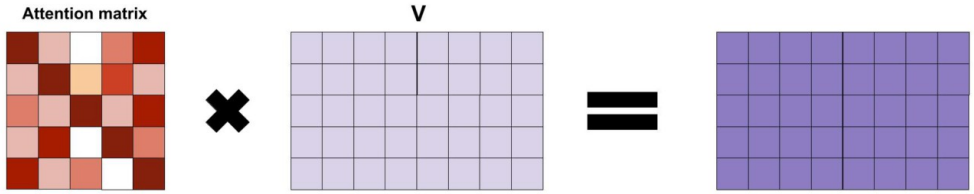
The Attention matrix is calculated from the matrix Q and K. We must multiply the matrix Q with the transpose of the matrix K to get it.

Multiplying the two matrices makes it possible to obtain the dot product of all the vectors of each sequence ($\mathbf{M_{i,j} = Q_{i,:}.K_{:,j}}$).

The weights of the Attention mechanism will, therefore, be a function of the angular distance of each of the row vectors of Q with the column vectors of KT.

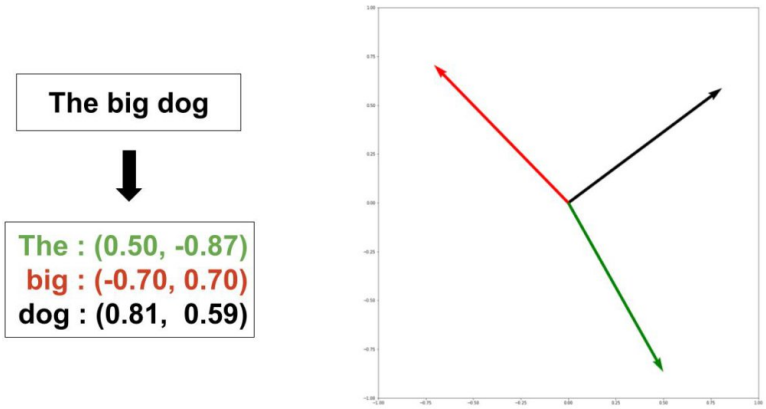
This is why we also speak of dot product Attention when it comes to the Attention mechanism of the Transformers.

After the matrix multiplication, we apply a softmax function to each row of the Attention matrix in order to obtain weights between 0 and 1.



Attention mechanism (3)

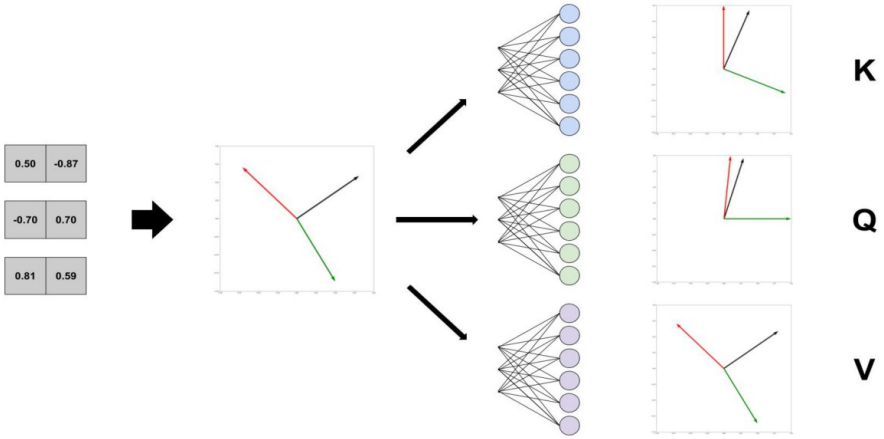
As soon as we have obtained the Attention matrix, we can multiply it by the V matrix, as we have seen previously (intuition of Attention).



Attention mechanism - Example (1)

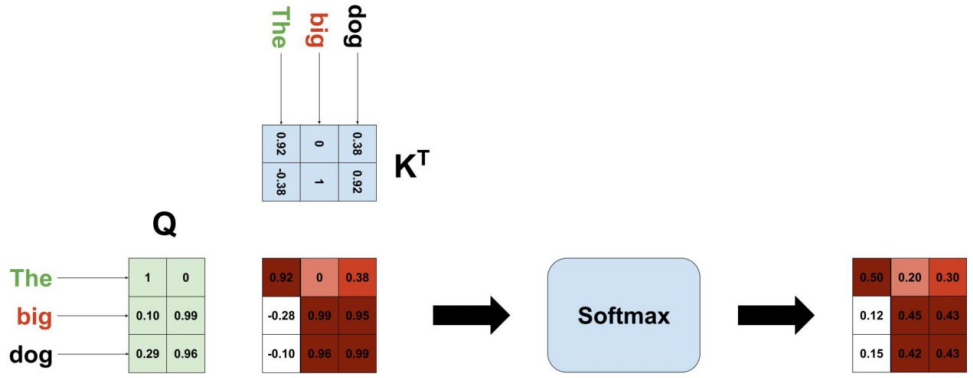
We will take a very simplified example to understand the theory of Attention. Let us consider the phrase "The big dog" which is represented in a 2-dimensional space.

The vectors are quite distant from each other because their meanings are also distant.



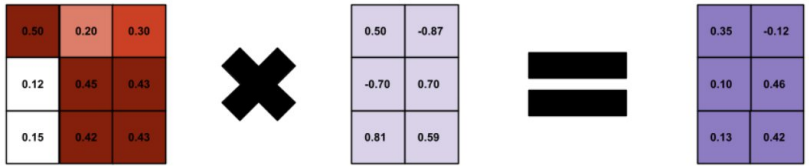
Attention mechanism - Example (2)

The linear transformations will bring the vectors representing “big” and “dog” closer together so that the weights of the Attention matrix linking them are larger than the others.



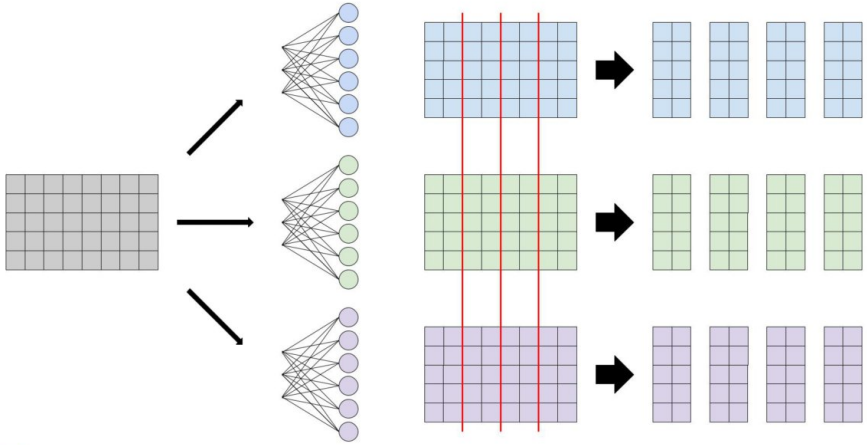
Attention mechanism - Example (3)

We see that the preceding transformation made it possible to obtain stronger links between “big” and “dog”. After applying Attention, their respective vectors will have more information from each other and less information from “The”.



Attention mechanism - Example (4)

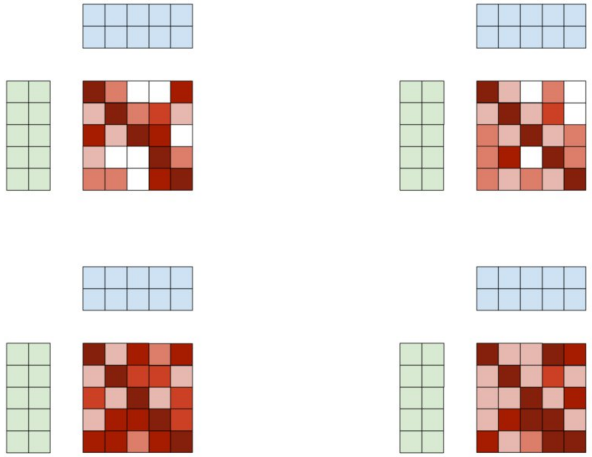
We finally apply the attention calculated previously to the matrix V.



Multi-Head Attention (1)

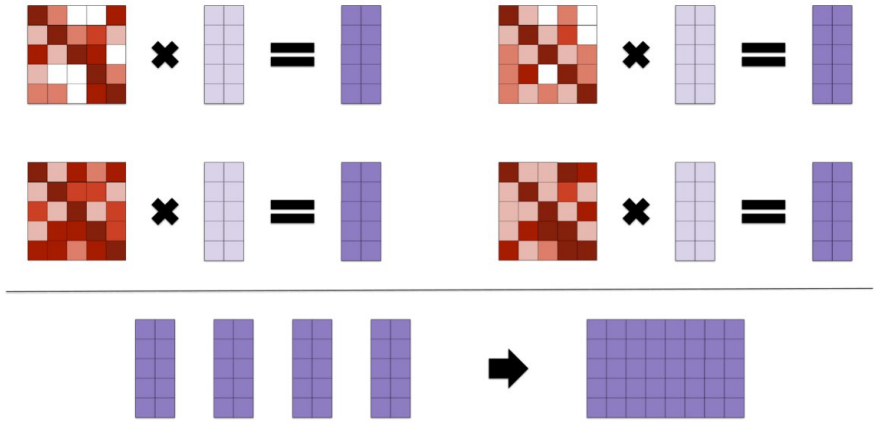
In most Transformers, it is not single Attention that is applied but Multi-Head Attention.

The principle is very similar to classical Attention, the difference being that we will slice the matrices Q, K and V into n parts (also called head) and reproduce the calculations of classical Attention n times (in parallel).



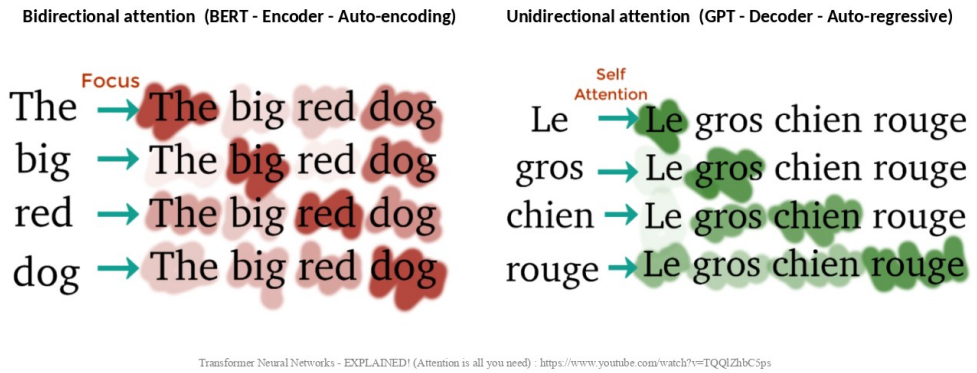
Multi-Head Attention (2)

The advantage of Multi-Head Attention is that we will produce several Attention matrices which makes it possible to have several links between the different elements of the text.



Multi-Head Attention (3)

After applying all calculations to the different heads, the results of each head are concatenated in order to obtain a matrix of the same dimension as that of the input.

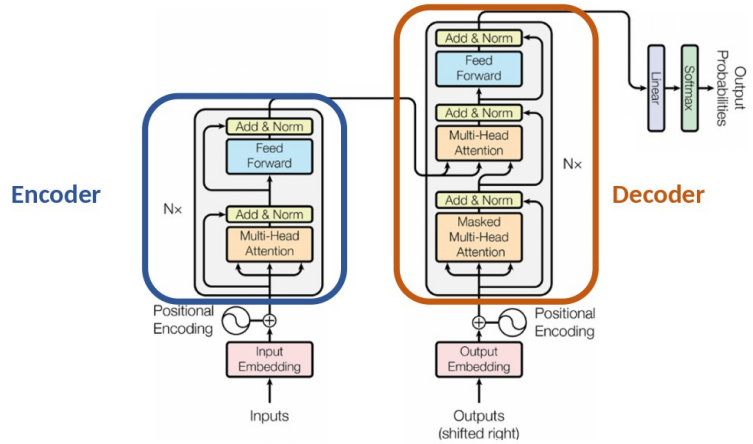


Transformer types (1)

There are several variants of the Transformer, the best known being the auto-regressive model.

The only difference between this model and the one we saw previously (Auto-encoding) is that we will apply unidirectional Attention rather than bidirectional.

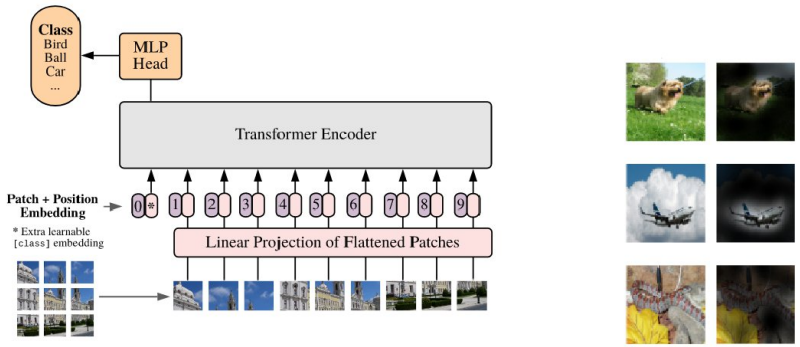
The advantage of this Attention is that the model is forced to use the preceding elements for the Attention calculation. This means that the model will be better in some tasks (such as generation).



Transformer types (2)

Another variant of Transformers is the encoder-decoder. (Although this was the first Transformer, it is a little less popular than the two versions we looked at previously).

It combines the two previously seen versions and can be very effective in multi-tasking.



Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

Vision Transformers

We can apply what we have seen previously to other fields. The Vision Transformer is a good example of the application of Attention to computer vision.

The principle is simple: We slice the image into patches (part of an image) and we flatten each patch into a vector and this will give us a sequence of vectors.

This vector sequence is then treated as the vector sequence in NLP.