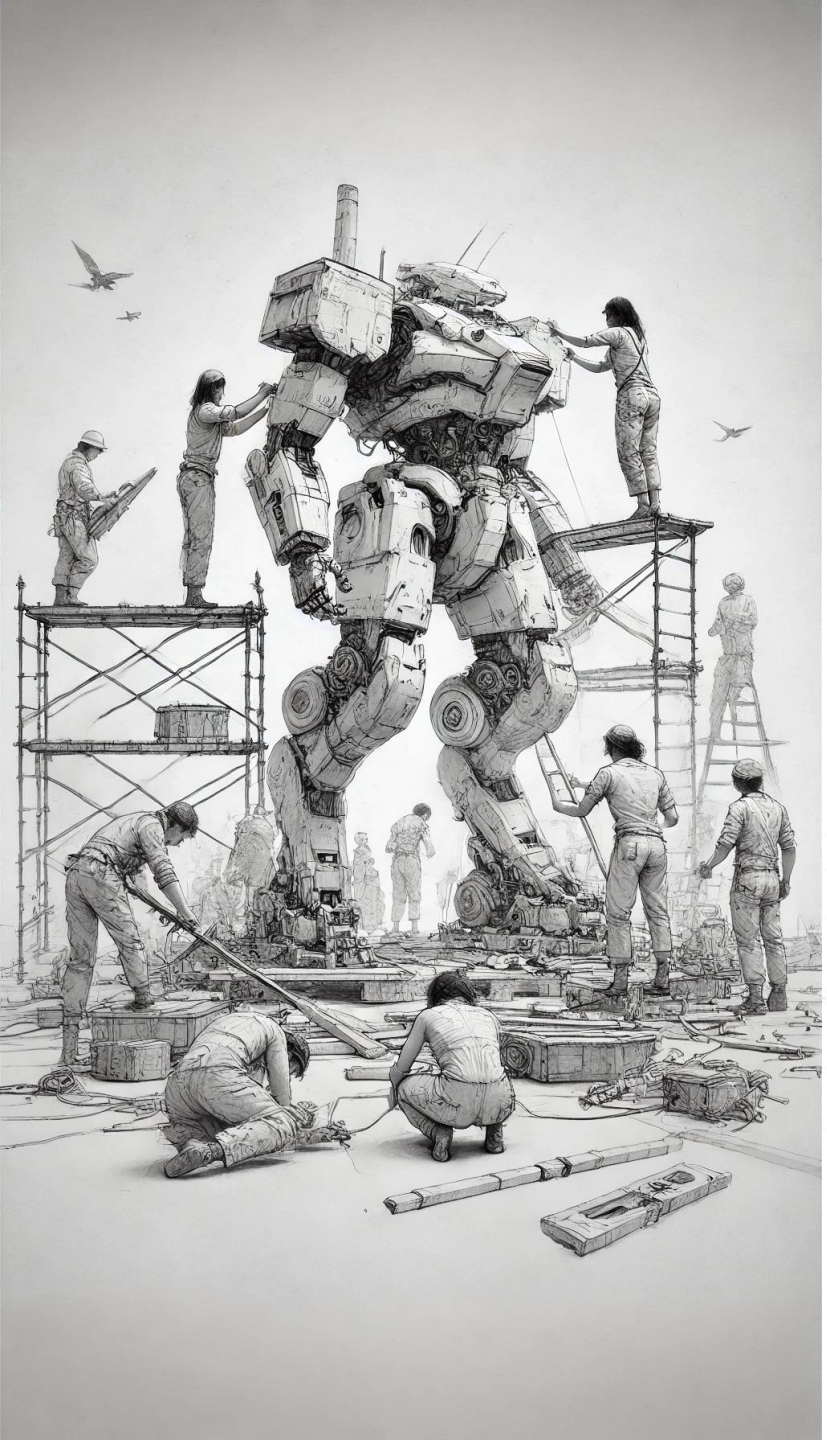


SAISON 24/25

Formation

Introduction au Deep Learning

Séquence n°12

“Attention Is All You Need", quand les Transformers changent la donne !”

FIDLE

Listes de diffusion - Newsletters

FIDLE https://fidle.cnrs.fr/listeinfo

Fidle information list

Agoria http://fidle.cnrs.fr/agoria

AI exchange list

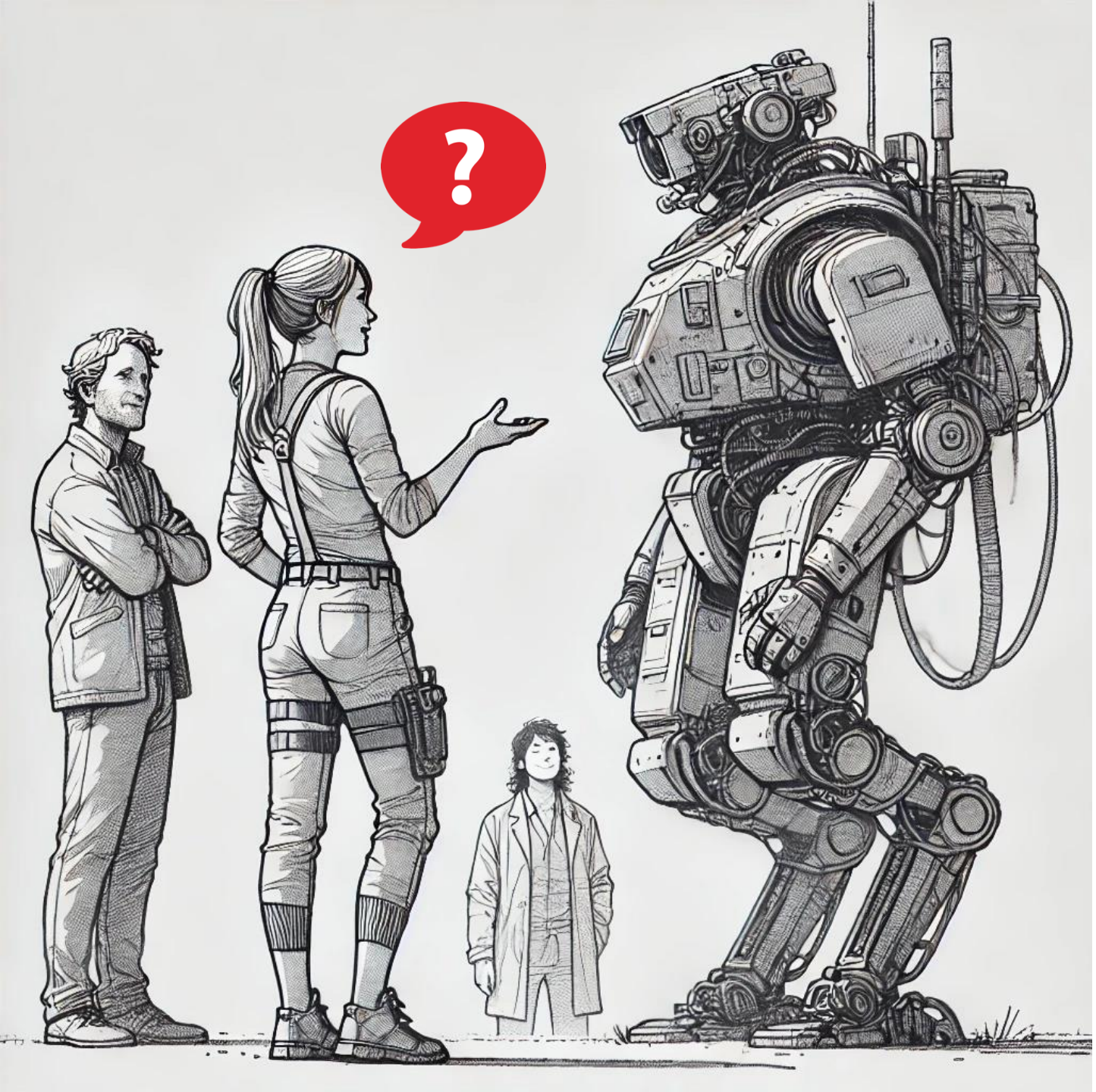
DEVLOG RÉSEAU DU DEVELOPPEMENT LOGICIEL Recherche et Enseignement Supérieur https://listes.services.cnrs.fr/wws/info/devlog

List of ESR* « Software developers » group

GROUPECALCUL https://listes.math.cnrs.fr/wws/info/calcul

List of ESR* « Calcul » group

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations



FIDLE

Pour toutes vos **questions**, n’hésitez pas à utiliser :

- le **chat** pendant le live
- la **liste AGORiA**

Suivez-nous !

FIDLE

youtube.com/@CNRS-FIDLE

linkedin.com/company/fidle-cnrs

twitch.tv/formationfidle_cnrs

FIDLE

Panorama IA

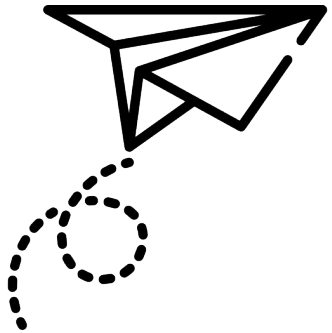
Le magazine de actualité en IA

Vendredi 28 mars, 10h00

http://www.idris.fr/panoramia.html
https://www.youtube.com/@IDRISCNRS

Vos cartes postales

Envoyez vos « cartes postales »
par mail à contact@fidle.cnrs.fr
pour apparaître dans la
prochaine séquence FIDLE



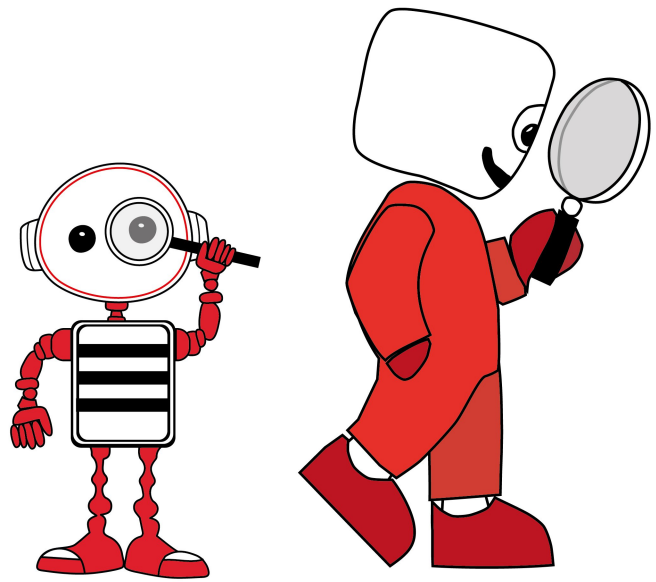
Merci à Gaëtan !

3e Journée

Deep learning pour la Science

FIDLE

cnrs IDRIS



Jeudi 5 Juin 2025
au CNRS à Paris

3 rue Michel-Ange, Paris 16

Gratuit sur inscription :
jdls-2025.sciencesconf.org

cnrs

Cluster IA
Multidisciplinary Institute In Artificial Intelligence

EFELIA

ANITI

DEVLOG RÉSEAU du développement logiciel

UGA Université Grenoble Alpes

GENCI

SIMaP Science et Ingénierie des Matériaux et des Procédés

FIDLE *PRÉSENTE*

PROCÈS DE L'IA

SOYEZ JURÉ.E DANS LE PROCÈS DE L'ANNÉE !

MIAI Cluster IA
Multidisciplinary Institute In Artificial Intelligence

UGA Université Grenoble Alpes

cnrs

JEUDI 10 AVRIL - 18H

MAISON DU TOURISME
GRENOBLE 14 RUE DE LA RÉPUBLIQUE

FIDLE

SAISON 24/25

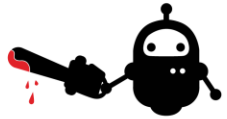
Bases, Concepts et Enjeux

1 ✓

Bases, concepts et histoire...
L'IA, c'est quoi ?

2 ✓

L'IA dans la pratique :-)
Exemples et démos

3 ✓ New !

L'IA est-elle notre amie ?
Le grand procès de l'IA

4 ✓

IA, droit, société et éthique
Vivre avec l'IA

L'IA comme un outil,

5 ✓

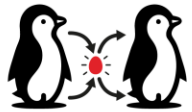
Neural Networks from Zero to Hero
DNN

6 ✓

Convolutional models
CNN

7 ✓

Mathematics, gradients everywhere !

8 ✓

Encoder/Decoder networks
AE/VAE, etc.

9 ✓ New !

Data, embedding and latent spaces
Representing the world

10 ✓

Graph Neural Network
GNN

11 ✓

Learning optimisation
Parameters & metrics

12

«Attention is All You Need»
RNN, Transformers

13 New !

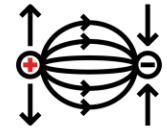
Models that talks !
LLM

14

Models who draw !
Diffusion models

15

Multimodal models
Multimodal LLM

16

Physics-Informed Neural Networks
PINNS

Acteur de l'IA

17 New !

Deep Failed !
The Great AI Blooper Reel

18

Learning optimisation II
Advanced

19 New !

Edge AI / IoT and Inference
Production

12

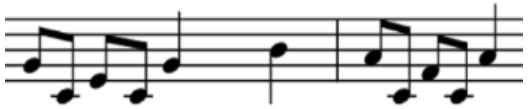
«Attention is All You Need»

RNN, Transformers

1. Prehistory
2. Recurrent Neural Networks
3. The Transformers area
4. Applications
5. And then …

What kind of data ?

Sequential data: « $t \rightarrow x_t \in \mathbb{R}^d$ »

	x_t		y
Speech recognition		→	Welcome to FIDLE
Music generation	/	→	
Image Analysis		→	Building
Activity recognition		→	Hits the ball
DNA analysis	AACTGTTACTGCCAG	→	AACTGTTACTGCCAG
Sentiment analysis	“These slides are cool”	→	
Translation	“Séquence douze”	→	“Sequence twelve”

…

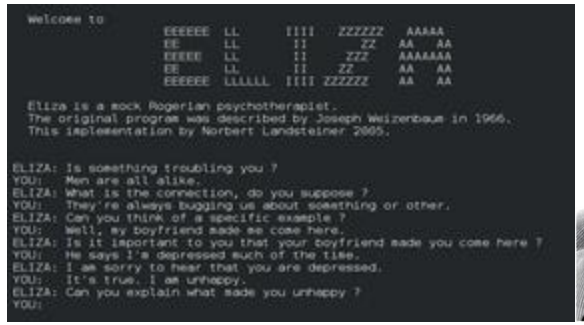
12

«Attention is All You Need»

RNN, Transformers

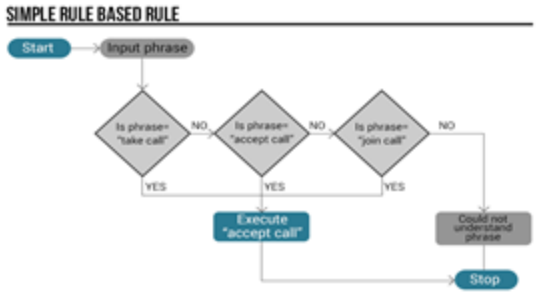
1. **Prehistory**
2. Recurrent Neural Networks
3. The Transformers area
4. Applications
5. And then …

From NLP point of view

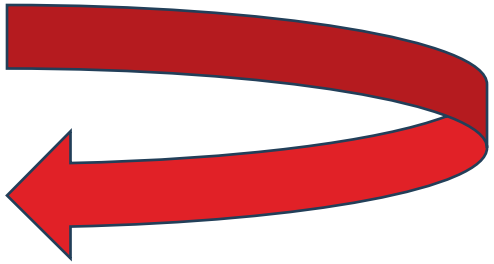


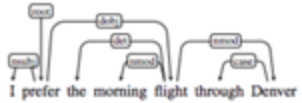
Eliza Chatbot (1966)
J Weizenbaum

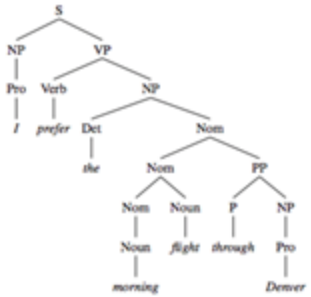
Source :
Wikipedia

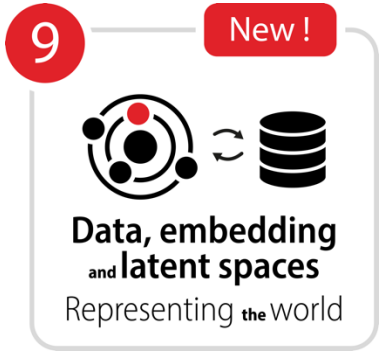


Expert systems,
Regular expressions,
grammars

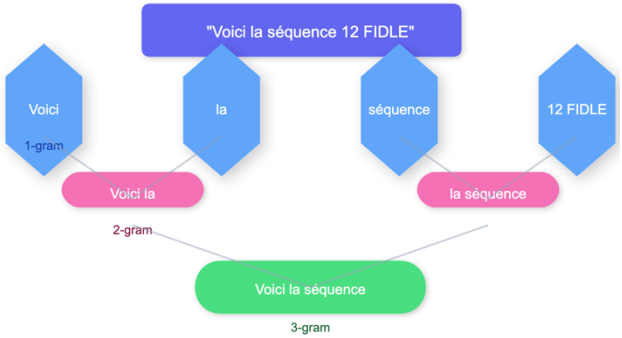








Word2Vec, Glove,…



n-grams models

FIDLE

12

«Attention is All You Need»

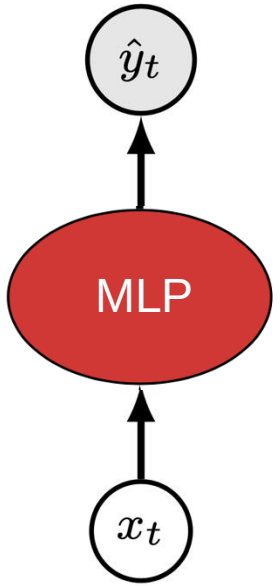
RNN, Transformers

1. Prehistory
2. **Recurrent Neural Networks**
3. The Transformers area
4. Applications
5. And then …

First idea

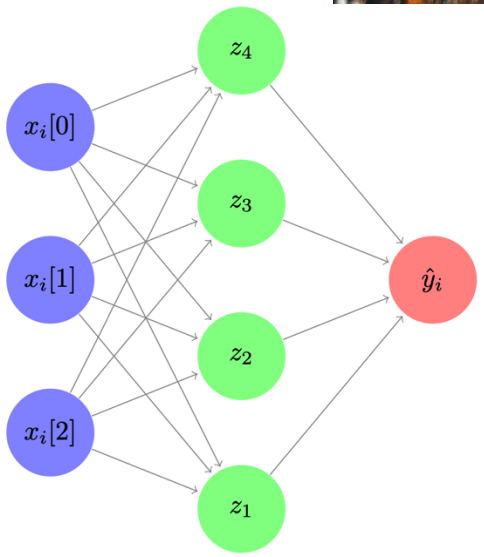
« Paul est basketteur. Il a marqué beaucoup de paniers samedi. »

Objective : predict the next word.



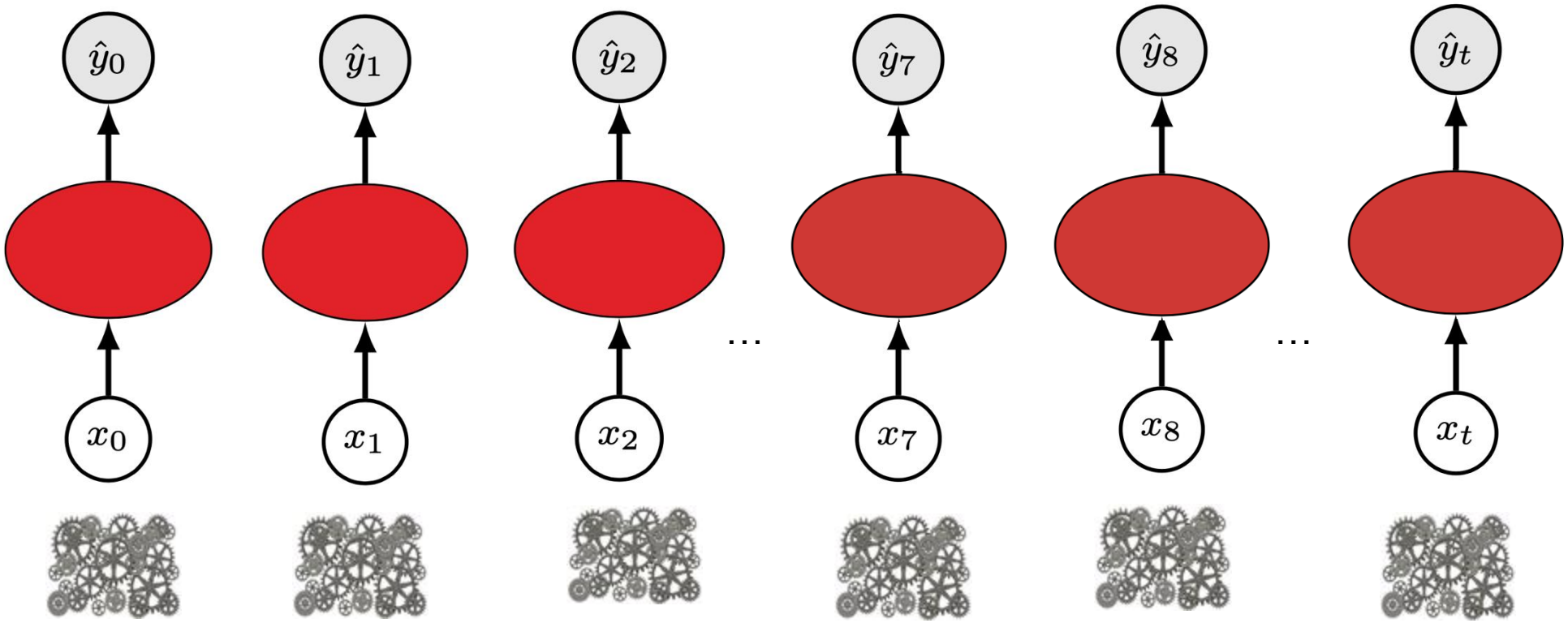
word →  → x_i

Embedding



MLP

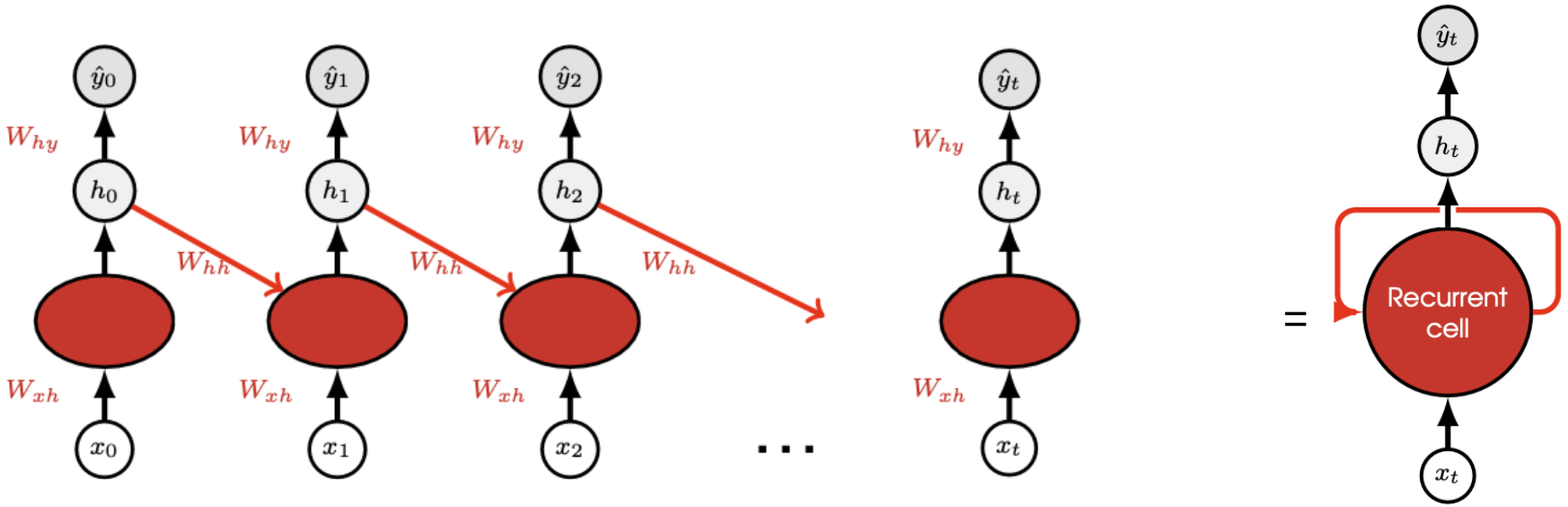
First idea



x_8 depends on x_2 : no possible relation here

Intuition – neurons with recurrence

h_t = memory of the past

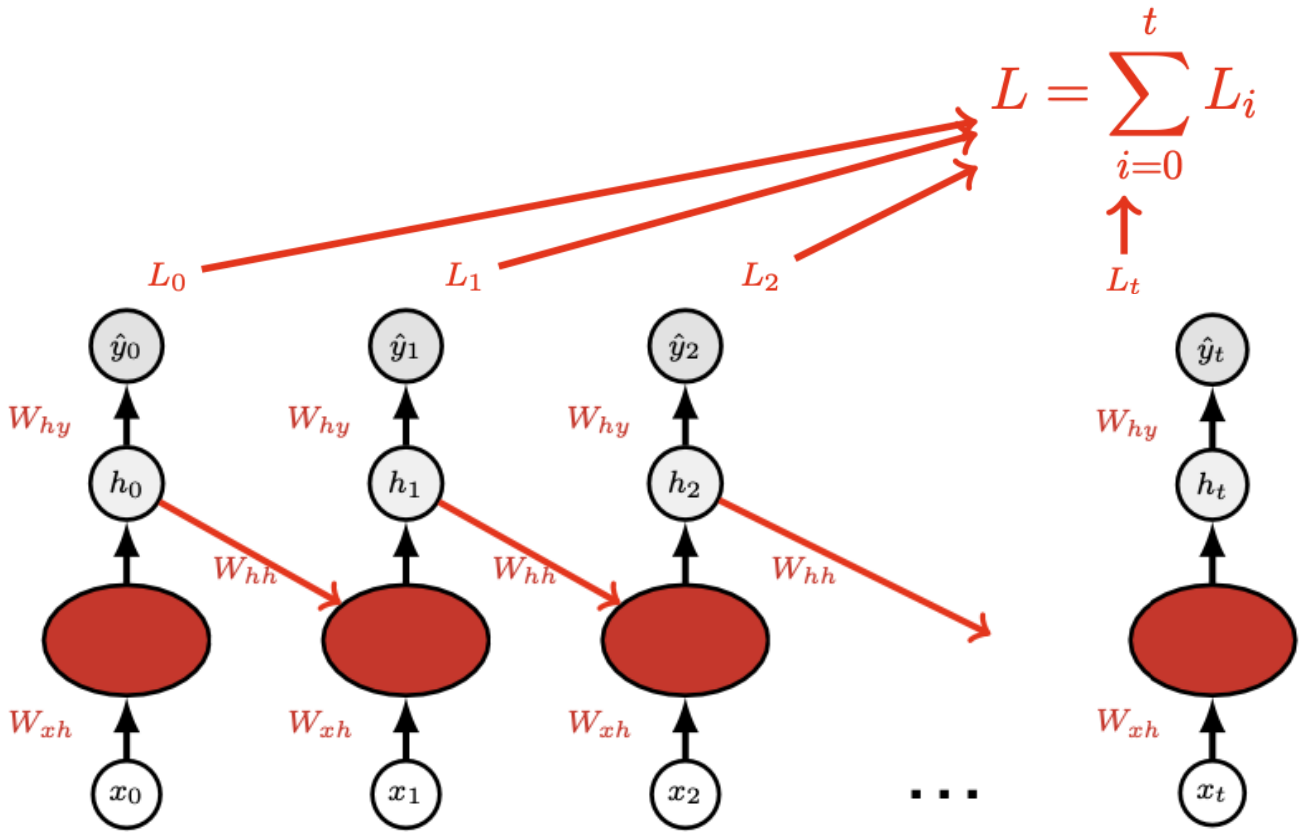


$$h_t = \tanh\left(\boldsymbol{W}_{xh}^T x_t + \boldsymbol{W}_{hh}^T h_{t\_1}\right)$$

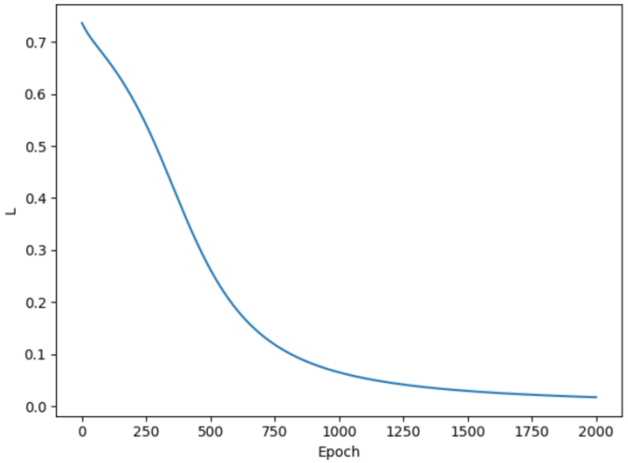
$$\hat{y}_t = \boldsymbol{W}_{hy}^T h_t$$

W shared through time

Recurrent neural Networks – Forward pass



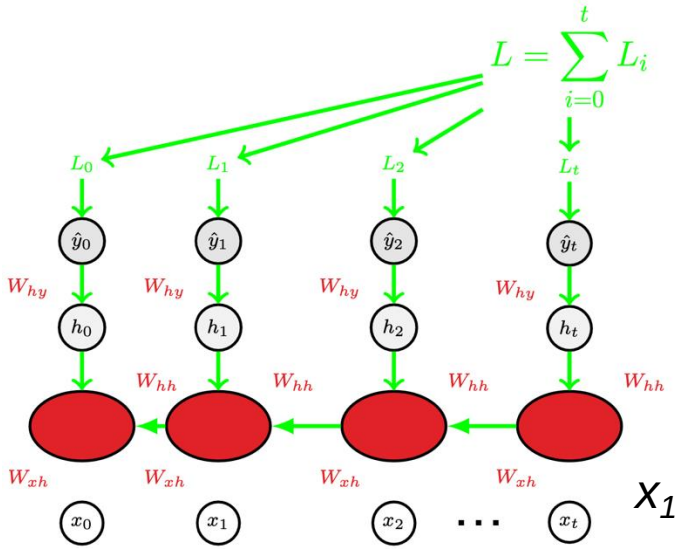
How to optimize L ?



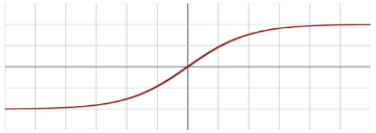
$$h_t = \tanh\left(\boldsymbol{W}_{xh}^T x_t + \boldsymbol{W}_{hh}^T h_{t\_1}\right)$$

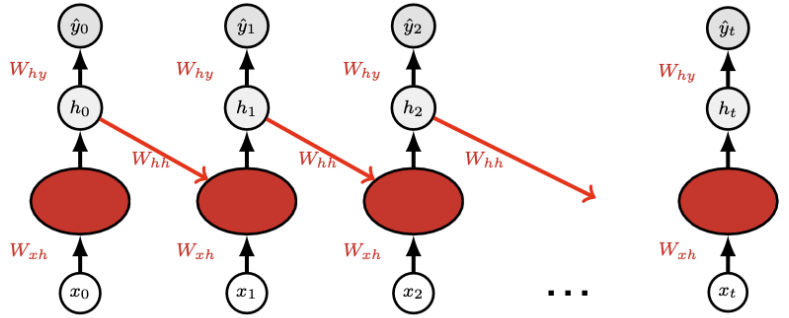
$$\hat{y}_t = \boldsymbol{W}_{hy}^T h_t$$

Recurrent neural Networks – Backward pass



Backpropagation Through Time (BPTT)





x_1 goes *t* times through *tanh*

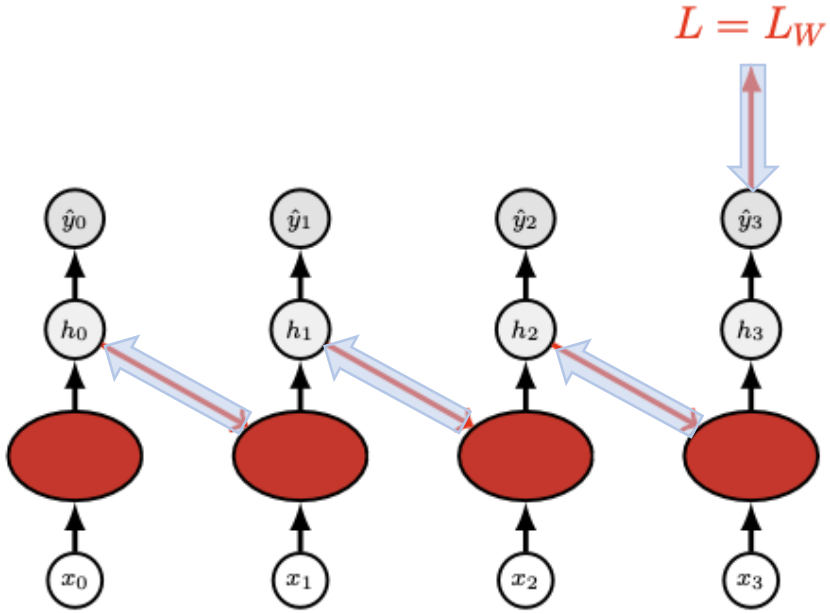
x_1 has less influence than x_t

W is shared : the weight associated with h_1 does not compensate for this memory loss

Paul est basketteur. Il a marqué beaucoup de paniers samedi.

Very Far !!!

Recurrent neural Networks – Problems…



$$\frac{\partial L}{\partial h_0} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}$$

What happens if derivatives are small ?

Vanishing gradient

What happens if derivatives are high ?

Exploding gradient

Recurrent neural Networks – Solutions ?

Exploding gradient → gradient clipping

Vanishing gradient → too difficult for RNN to preserve information over long periods of time

Potential loss of long dependencies

Paul est basketteur. Il a marqué beaucoup de paniers samedi.

Very far !!!

Solutions ?

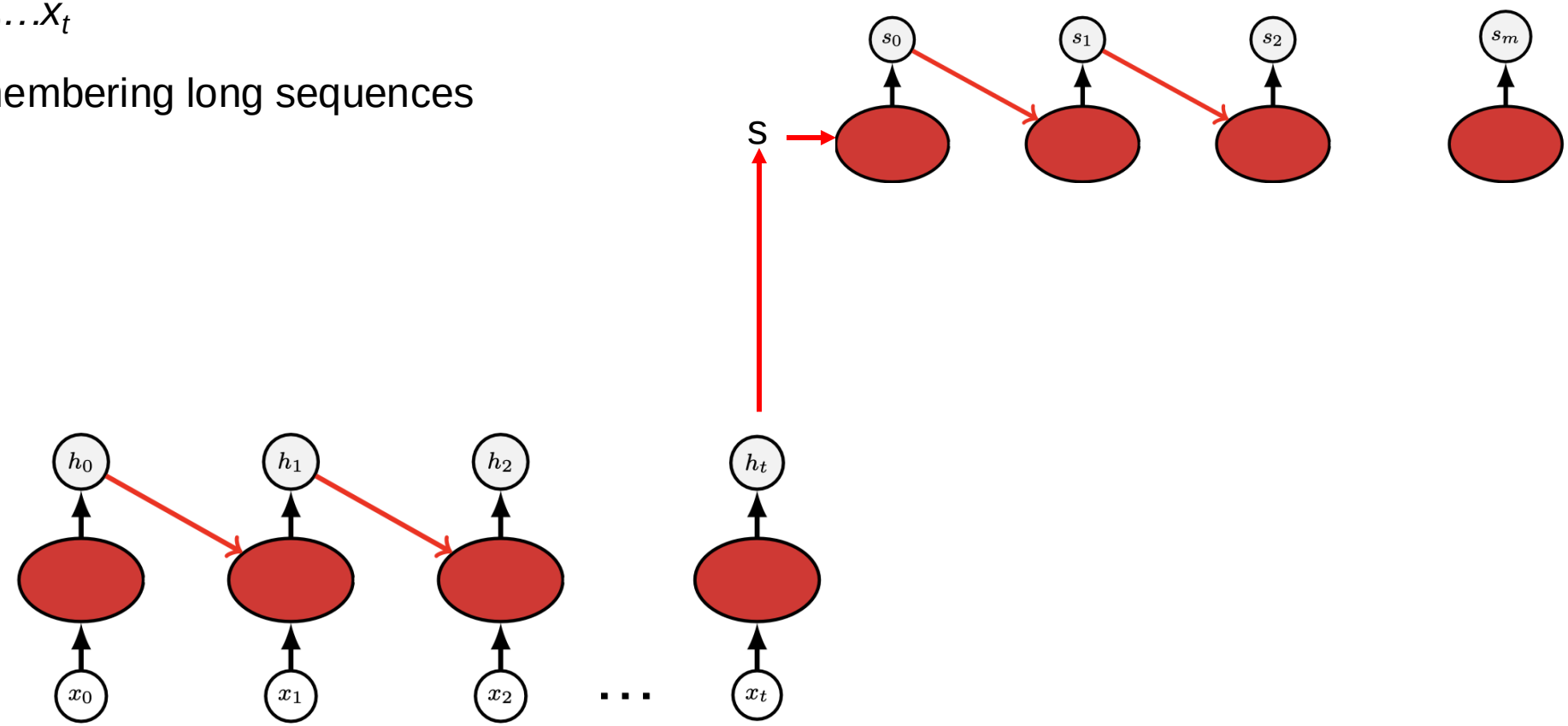
- Separate memory
  - ⇢ LSTM, GRU
- Add direct network connections and/or long-term dependencies
  - ⇢ Skip connexions, attention mecanisms

RNN with attention

Seq2seq models without attention

s: « summary » of $x_0 \ldots x_t$

$s=h_t$ incapable of remembering long sequences



RNN with attention

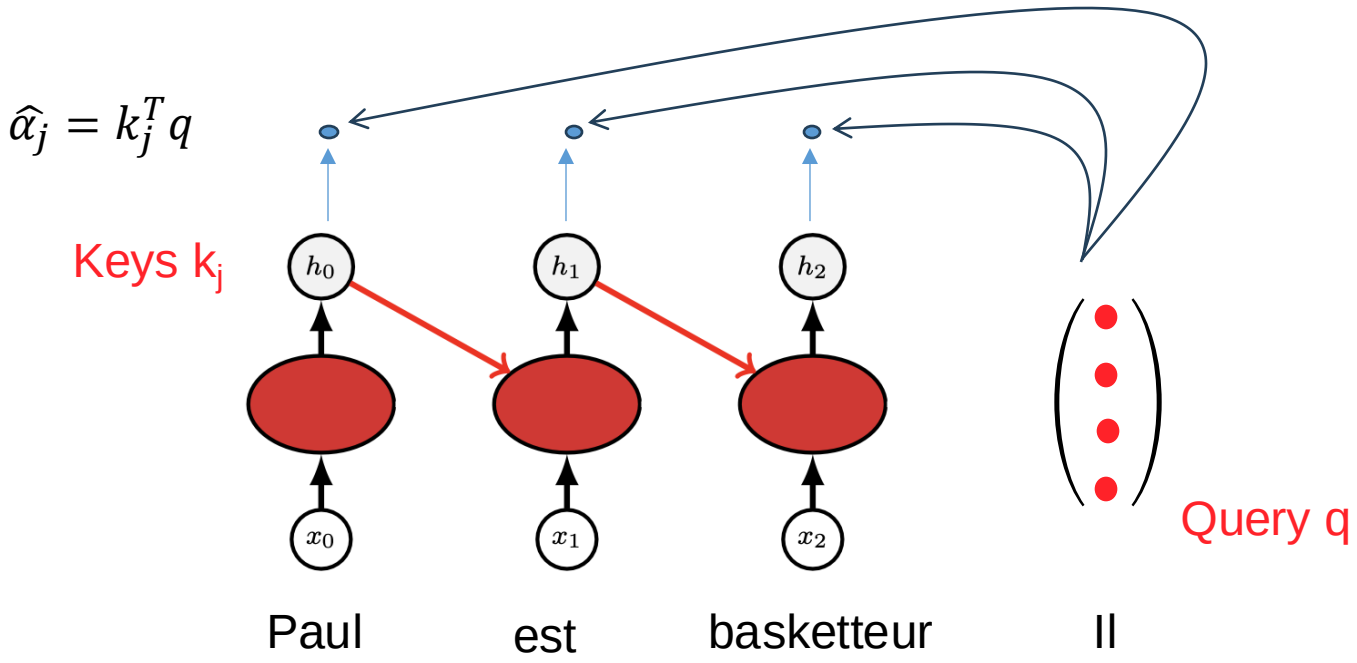
Seq2seq models with attention

Bahdanau et al, 2015

$$\widehat{\alpha_i} = w^T tanh\left(W \begin{bmatrix} k_j \\ q \end{bmatrix}\right) \quad \text{[original]}$$

$$\widehat{\alpha_i} = q^T W k_j \quad \text{[Luong]}$$

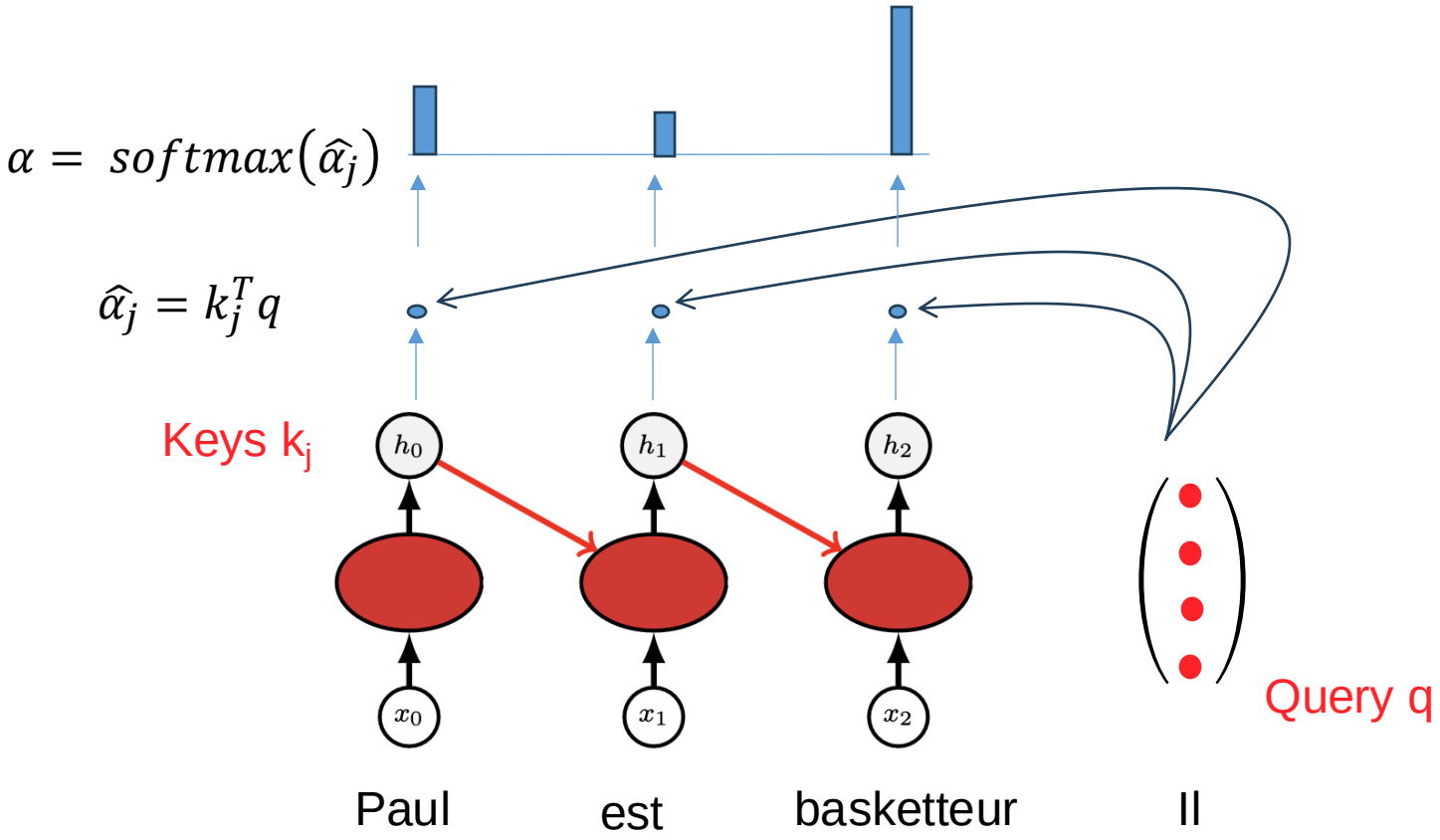
$$\widehat{\alpha_i} = \frac{q^T k_j}{\sqrt{|kj|}} \quad \text{[Vaswani]… soon}$$



RNN with attention

Seq2seq models with attention

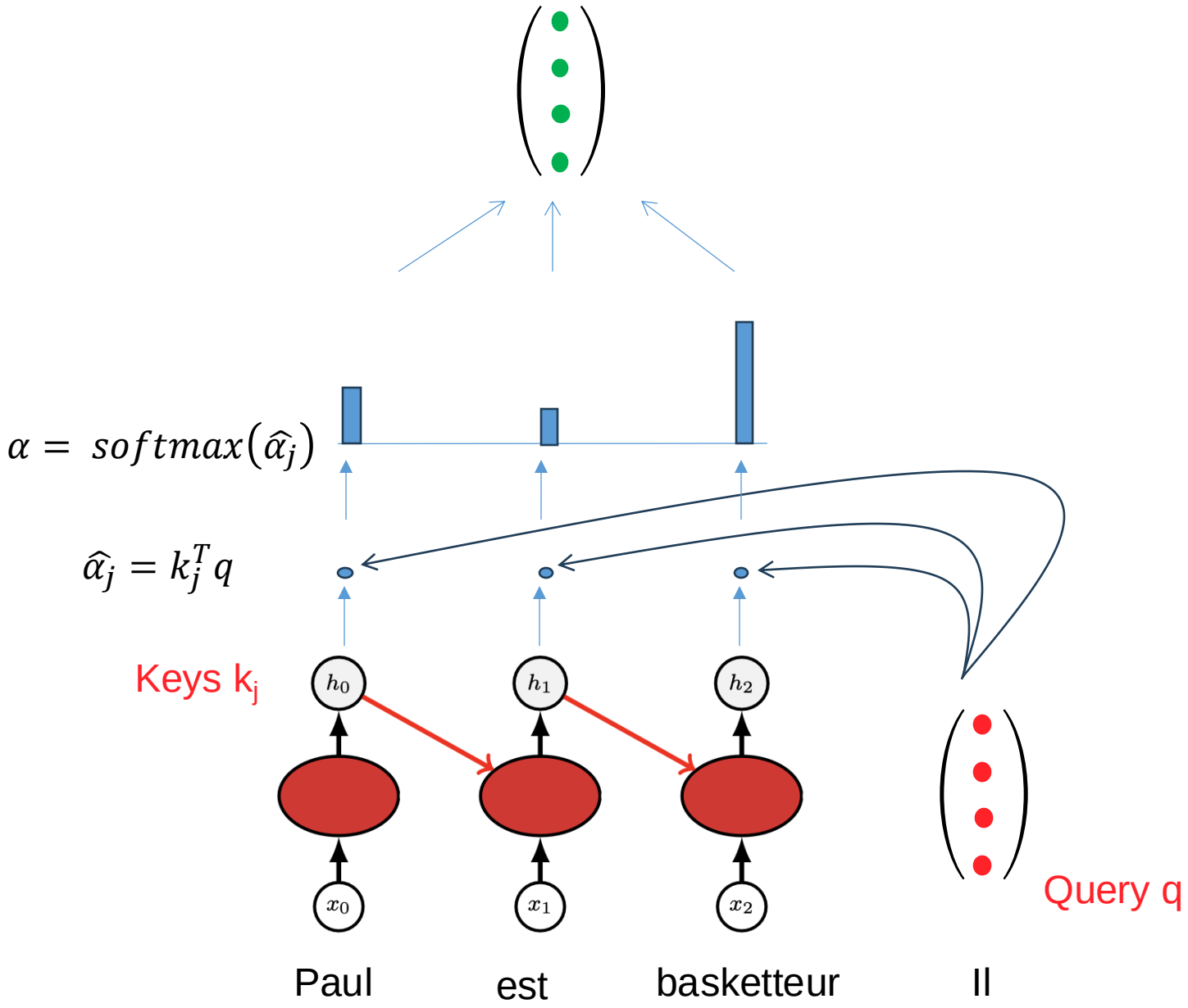
Bahdanau et al, 2015



RNN with attention

Seq2seq models with attention

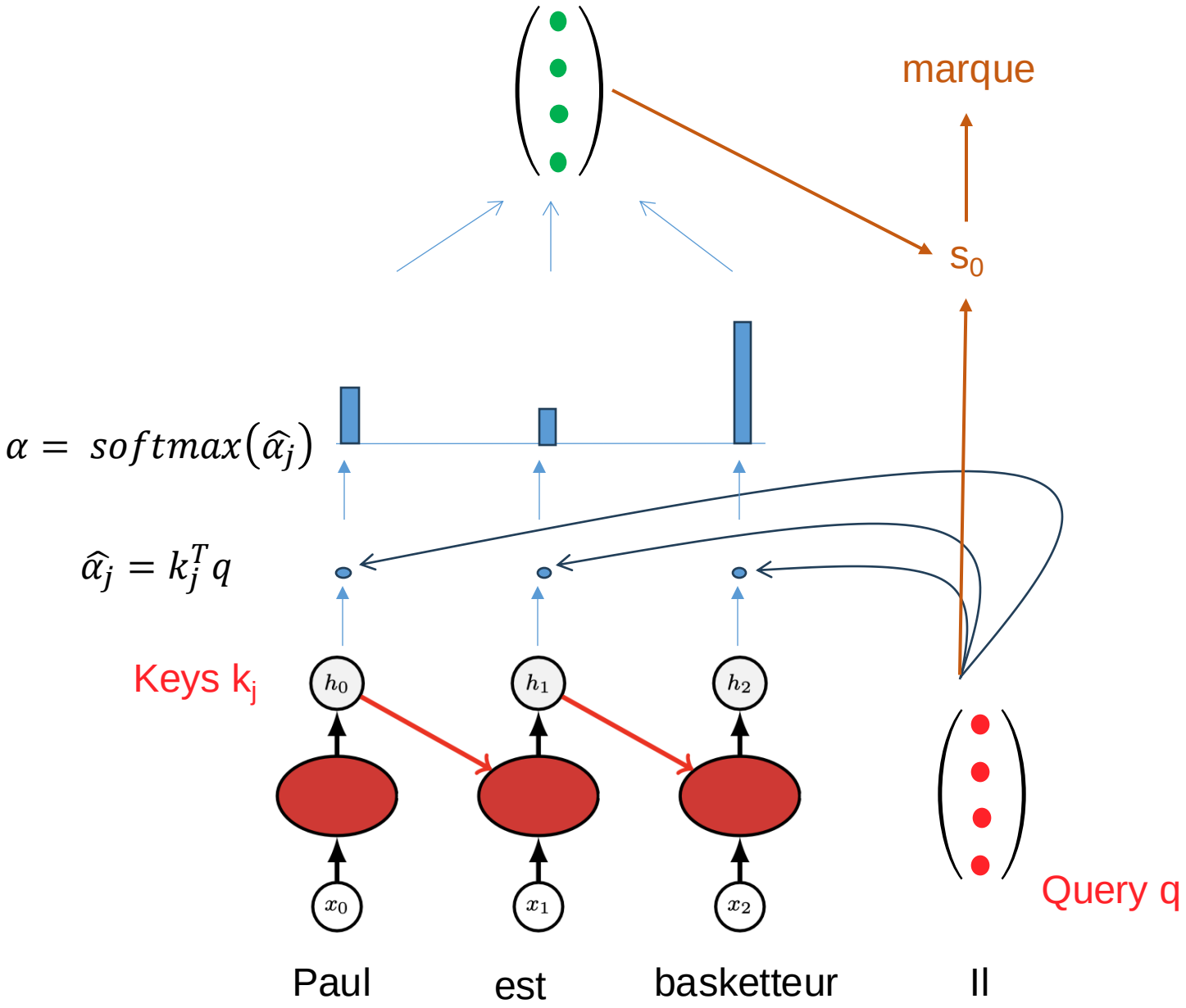
Bahdanau et al, 2015



RNN with attention

Seq2seq models with attention

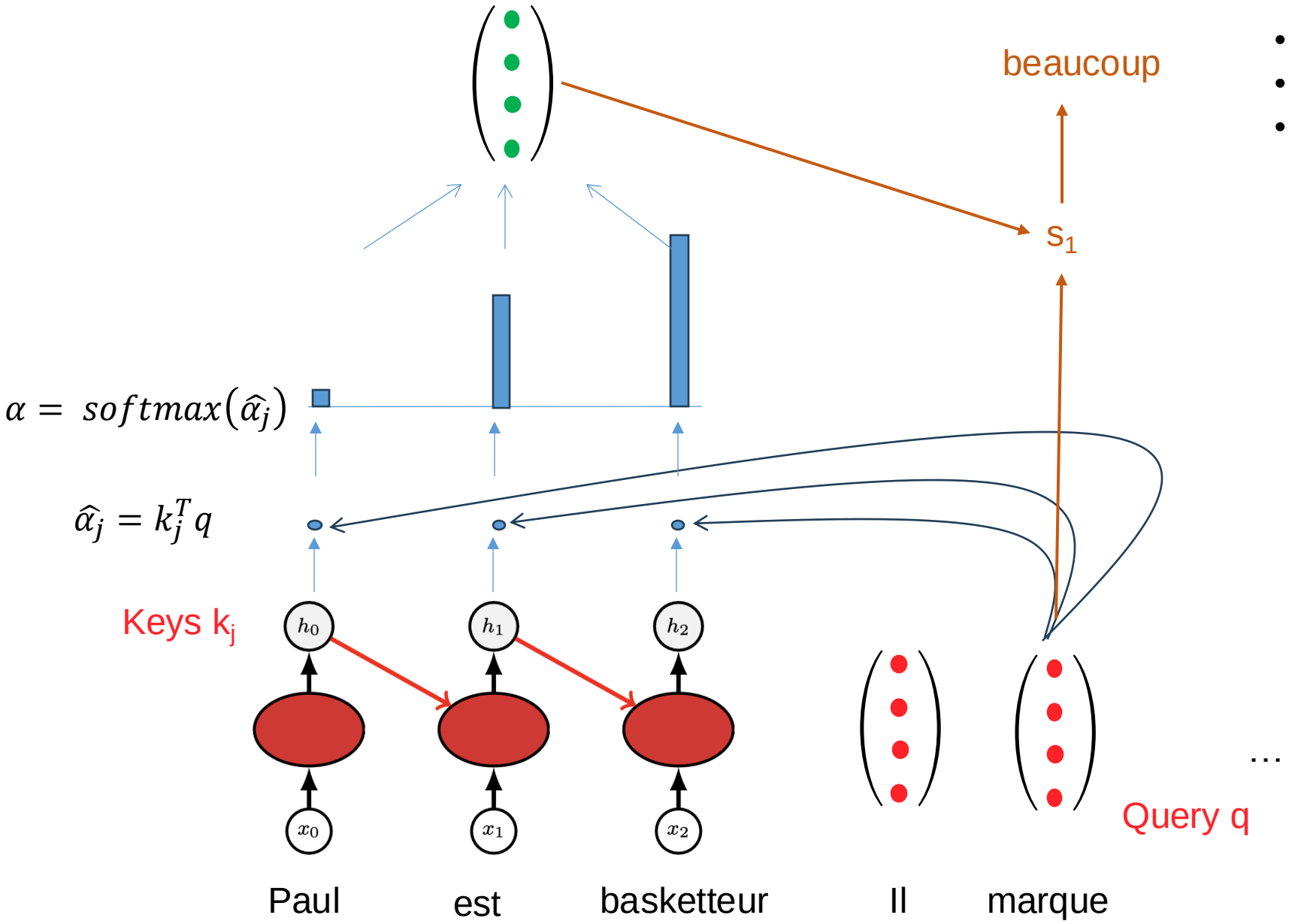
Bahdanau et al, 2015



RNN with attention

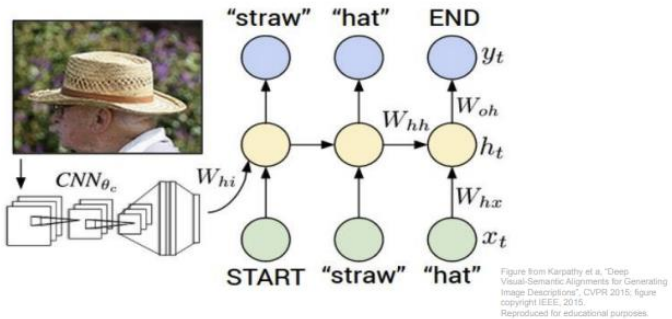
Seq2seq models with attention

Bahdanau et al, 2015

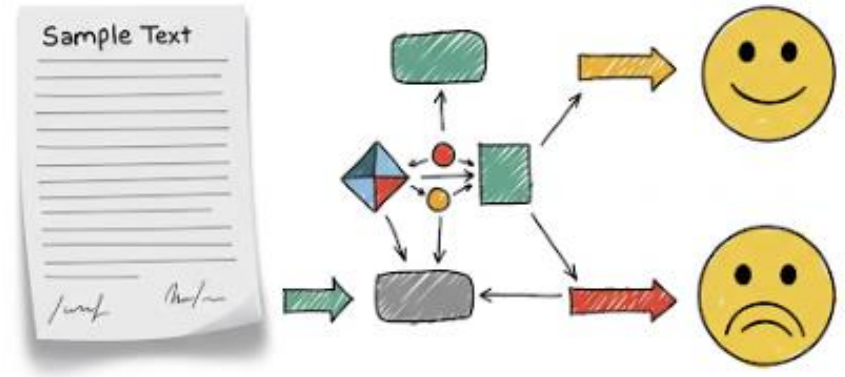


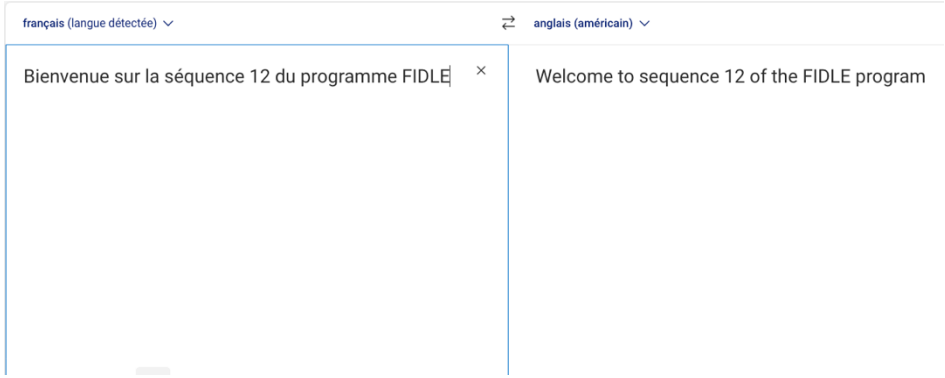
- Solve the bottleneck problem
- Helps with vanishing gradients
- Provides some interpretability

Recurrent Neural Networks – Architectures



One-to-many

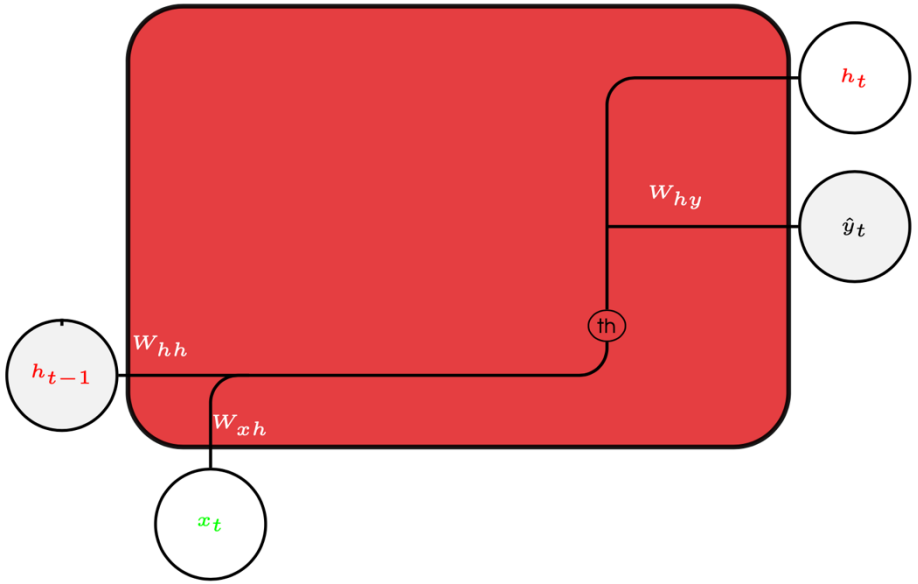




Many-to-many

Question break

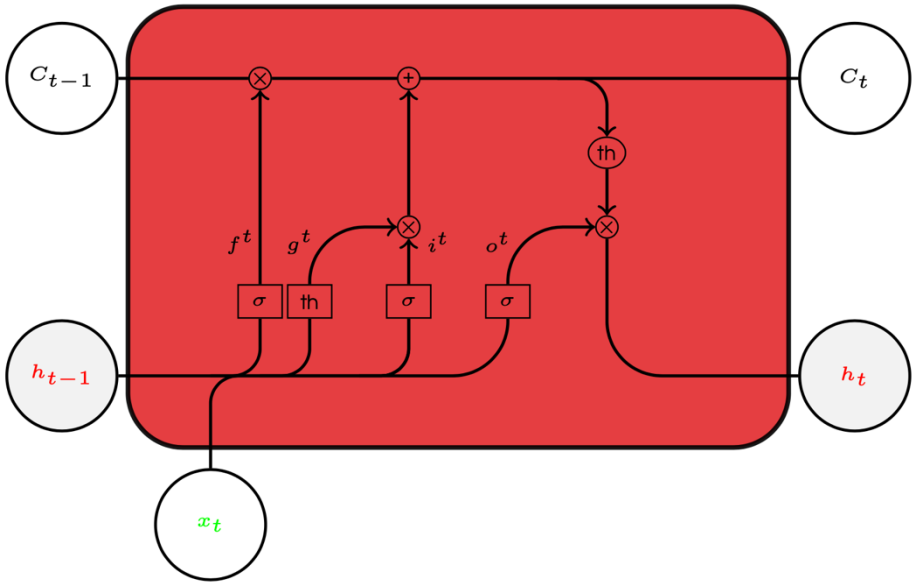
Recurrent Neural Networks– Another way to see



$$h_t = \tanh\left(\boldsymbol{W}_{xh}^T x_t + \boldsymbol{W}_{hh}^T h_{t\_1}\right)$$

$$\hat{y}_t = \boldsymbol{W}_{hy}^T h_t$$

Long Short Term Memory (LSTM)



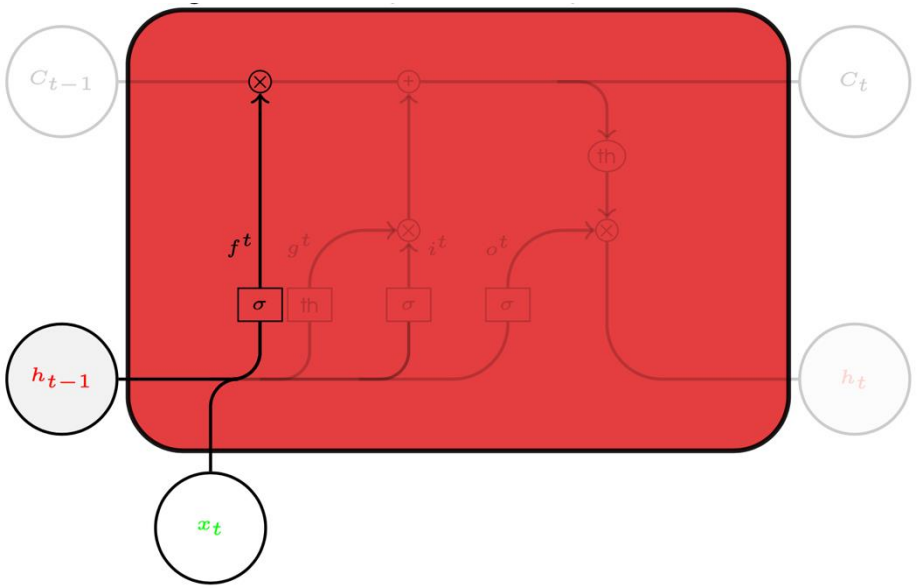
➔ **Key concept:** persistent state C_t, representation of the past

➔ **Gate mechanisms**: tracking information over time, even for multiple time steps

➔ Provide a quantity of information in [0,1] to the persistent state

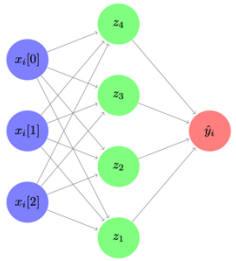
1. When I read x_t, do I still need to memorize the information I've retained so far?
2. What new information does x_t give me?
3. How do I update the persistent state?

LSTM – Forget gate



What quantity of information to omit from C_t ?

$$f_t = \sigma\left(\boldsymbol{W_f}[h_{t-1}, x_t] + b_f\right)$$



MLP

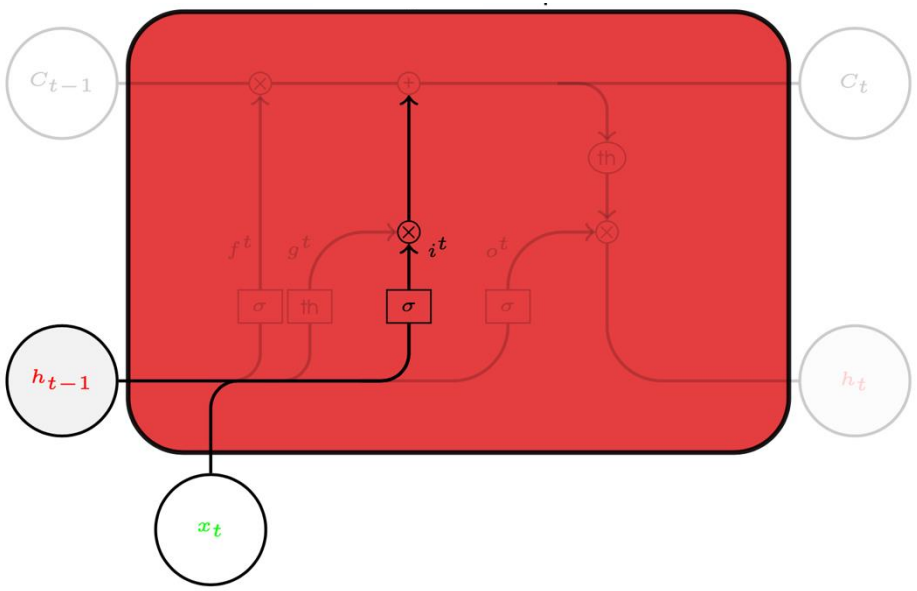
Paul est basketteur. Il a marqué beaucoup de paniers samedi. Jeanne aime plutôt le handball.

f_t=0

C_t contains the subject's gender (Paul)

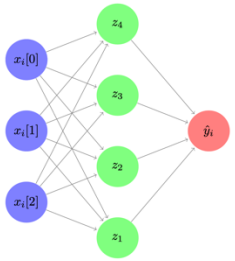
C_t new subject -> forgetting the previous gender

LSTM – Input gate



1. What new quantity of information to store in C_t ?

$$i_t = \sigma(\boldsymbol{W_i}[h_{t-1}, x_t] + b_i)$$



2. Which candidates to add C_t ?

$$g_t = \tanh(\boldsymbol{W_c}[h_{t-1}, x_t] + b_c)$$

3. Update the persistent state

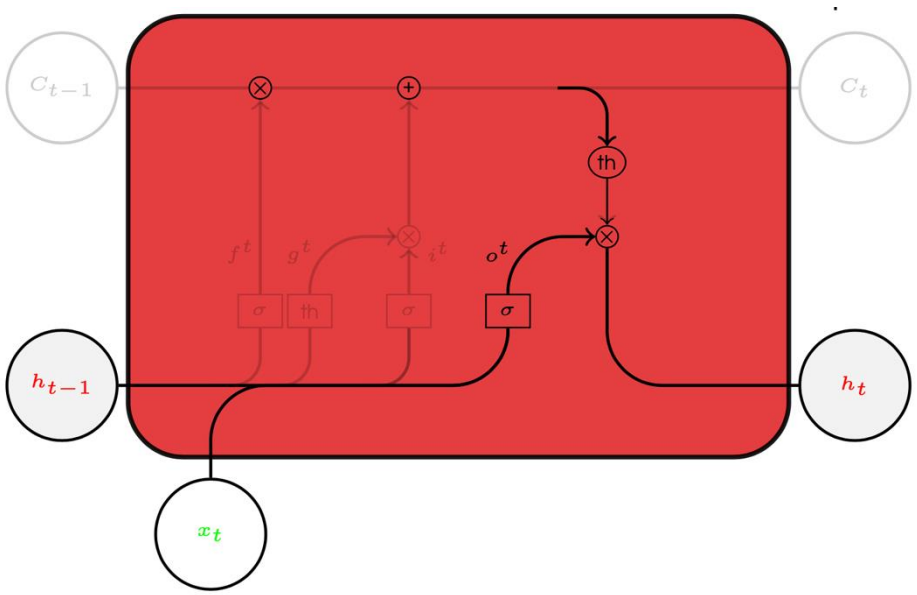
$$C_t = f_t C_{t-1} + i_t g_t$$

Paul est basketteur. Il a marqué beaucoup de paniers samedi. Jeanne aime plutôt le handball.

C_t contains the subject's gender (Paul)

g_t : new subject

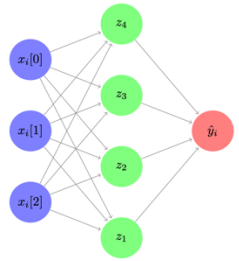
LSTM – Output gate



What is the influence of C_t / output h_t to provide ?

$$o_t = \sigma(\boldsymbol{W_o}[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \tanh(C_t)$$



MLP

Paul est basketteur. Il a marqué beaucoup de paniers samedi. Jeanne aime plutôt le handball.

h_t : singular/plural

LSTM – Several architectures

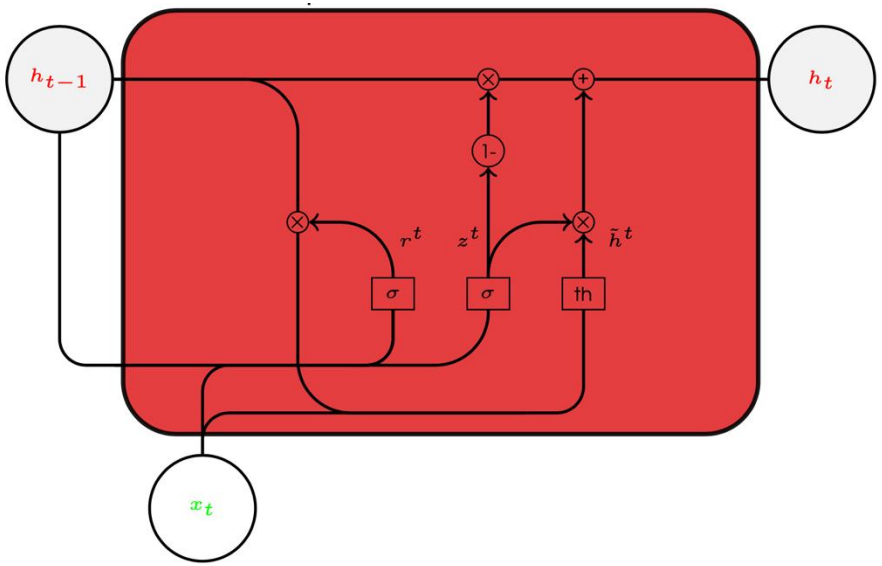
- LSTM with peephole connections
  Gates monitor the persistent stage

$$g_t = \sigma\left(\boldsymbol{W_g}[C_{t-1}, h_{t-1}, x_t] + b_g\right)$$

LSTM – Several architectures

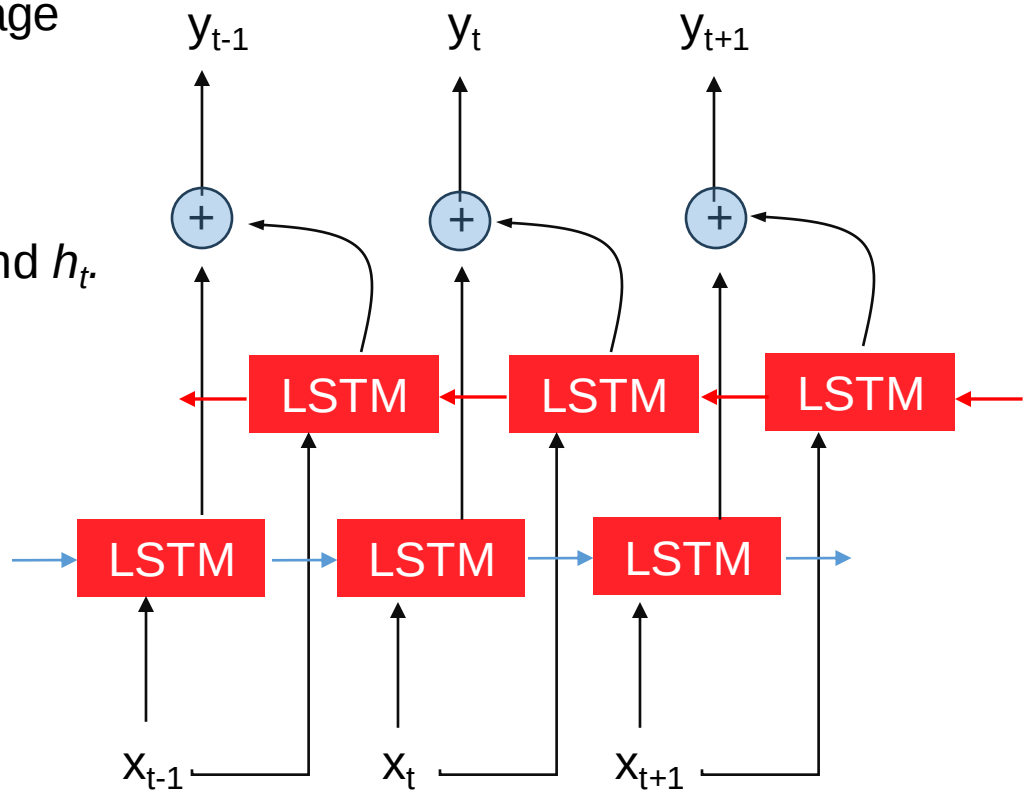
- LSTM with peephole connections
  Gates monitor the persistent stage

- GRU : (i_t f_t) → z_t / fusion *of* C_t and h_t.



LSTM – Several architectures

- LSTM with peephole connections
  Gates monitor the persistent stage
- GRU : (i_t f_t) → z_t / fusion *of* C_t and h_t.
- Bi-LSTM
- …



From sequentiality to global attention

Challenges of RNN/LSTM

- **Sequential processing:** word-by-word analysis
  → limitation in capturing long-term dependencies
  → slows down learning and inference
- **Computation time**: sequential processing preventing parallelization
- **Memory bottleneck**: even with LSTM, difficulties in maintaining context over very long sequences

The Transformer revolution

- **Global vision** : attention mechanism allowing analysis of all relationships simultaneously
- **Massive parallelization**: processing all words in parallel
  → speed gains
- **Rich contextualization**: each word is directly connected to all others
  → better understanding of context

Transition from "word-by-word" analysis to "all words connected" understanding

Question break

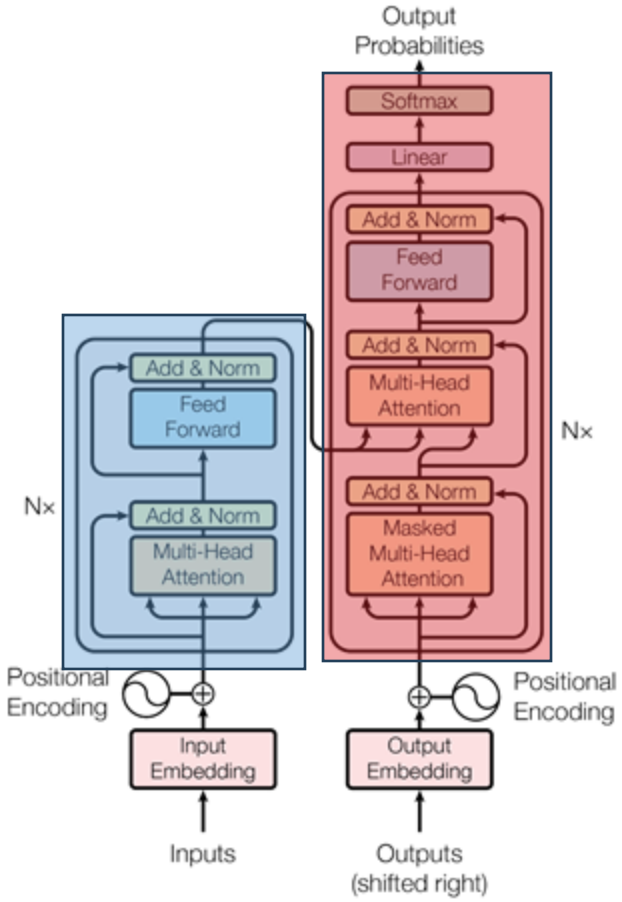
12

«Attention is All You Need»

RNN, Transformers

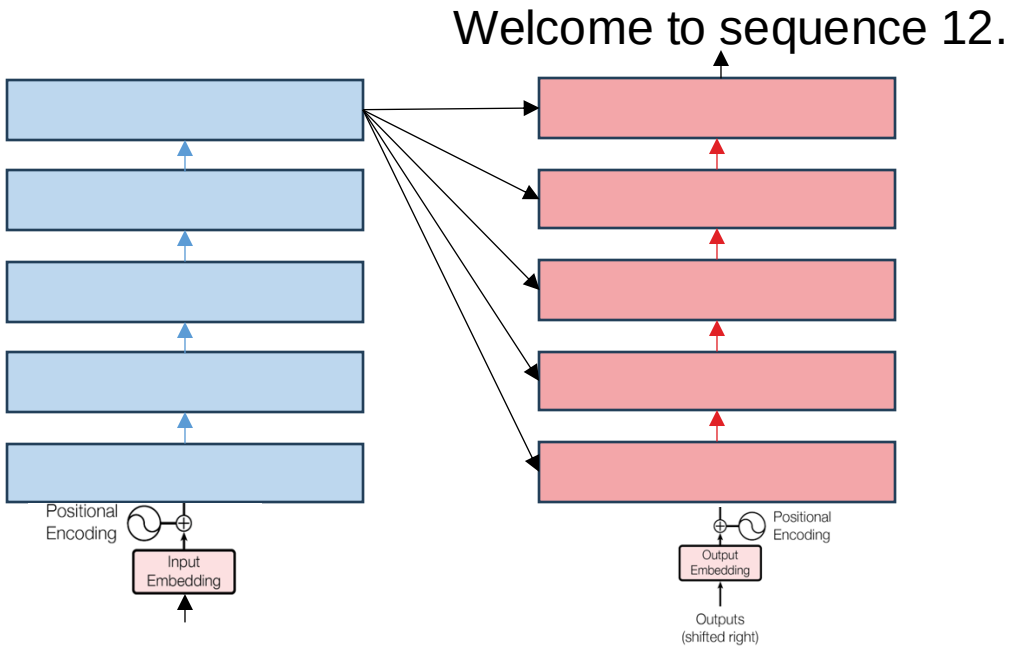
1. Prehistory
2. Recurrent Neural Networks
3. **The Transformers area**
4. Applications
5. And then …

Global architecture



Encoder : Read and globally encodes the input

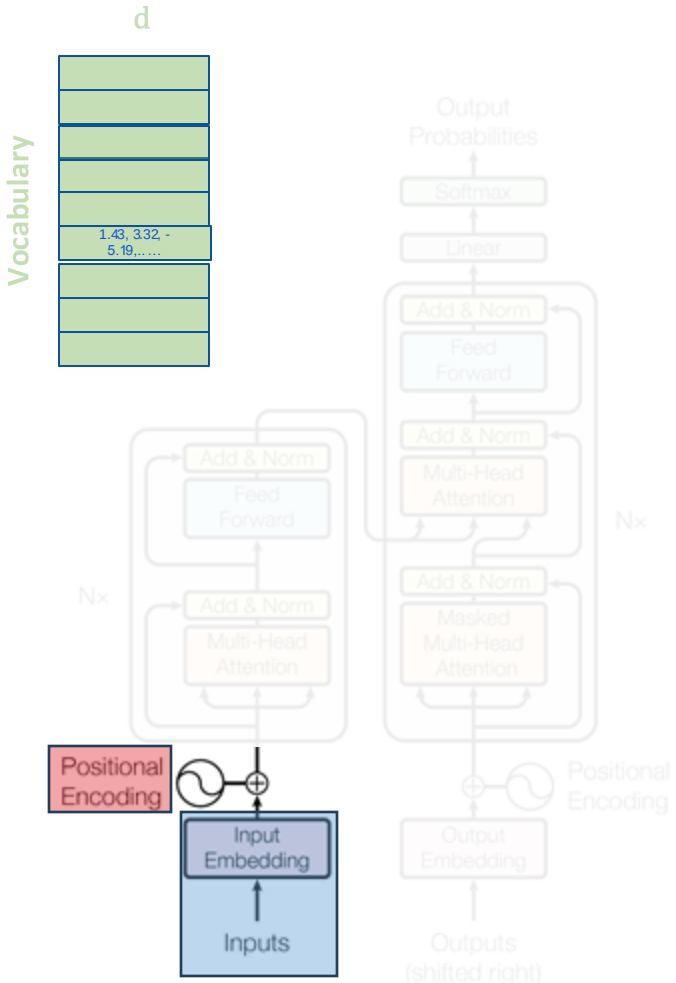
Decoder : generates the output



Bienvenue à la séquence 12.

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Encoder



Word embedding

1- word → tokens " Paul est basketteur" ⇢ [Paul_] [est_] [basket] [teur]

2- tokens → indexes in the vocabulary [Paul_] [est_] [basket] [teur] ⇢ [3 433 1231 87]

Positional Encoding

Language is not permutation invariant

"Jean pense que Pierre a tort" vs "Pierre pense que Jean a tort"

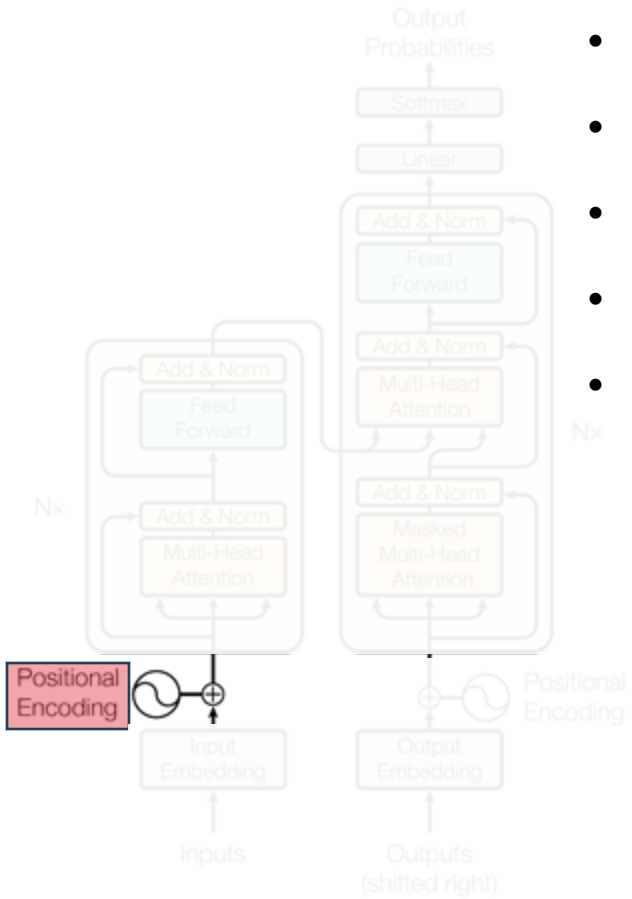
- Must encode word position.

Tokenizer Positional encoding

word j → [3 433 1231 87] + ? $= x_j \in \mathbb{R}^d$

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Encoder



Contraints ?

- Unique encoding ✔
- Simple relationship between the encoding of two known positions ✔
- Generalization to sequences of any size ✔
- Generalization to any dimension ✘
- Deterministic encoding ✔

Solutions

[Paul_] [est_] [basket] [teur]

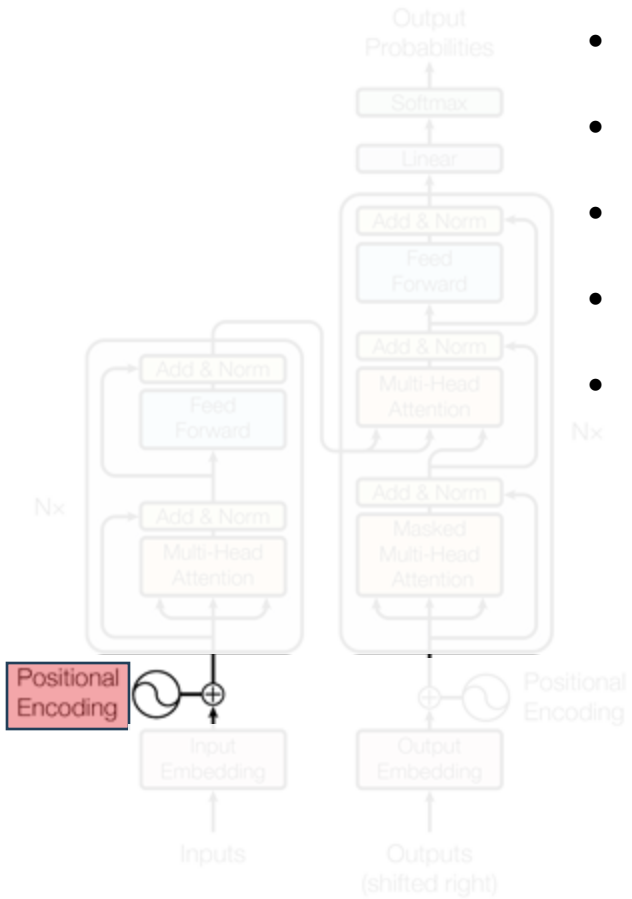
1 2 3 4

$$\begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix} \in \mathbb{R}^d$$

Coding amplitude

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Encoder



Contraints ?

- Unique encoding ✔
- Simple relationship between the encoding of two known positions ✘
- Generalization to sequences of any size ✔
- Generalization to any dimension ✘
- Deterministic encoding ✔

Solutions

[Paul_] [est_] [basket] [teur]

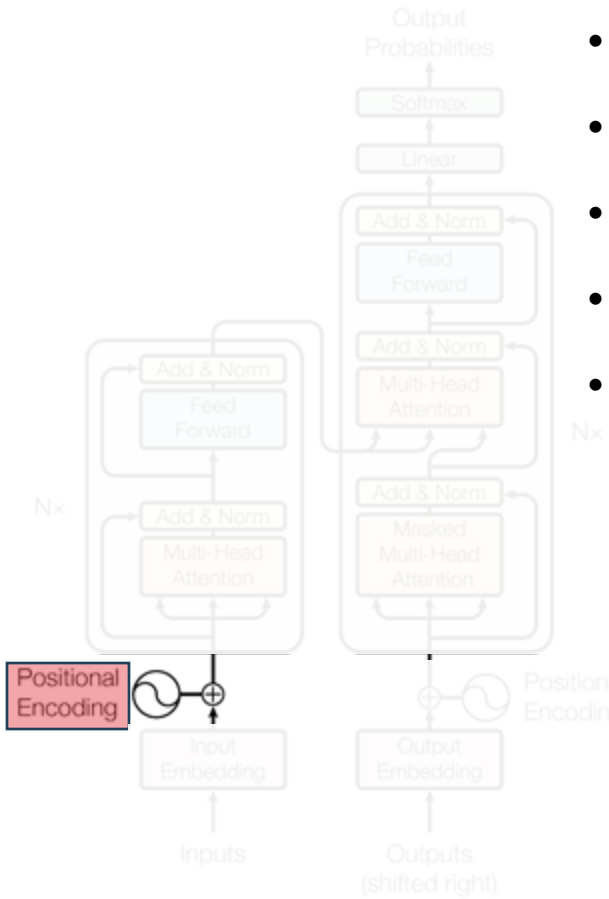
1 2 3 4

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^d$$

Hard to optimize

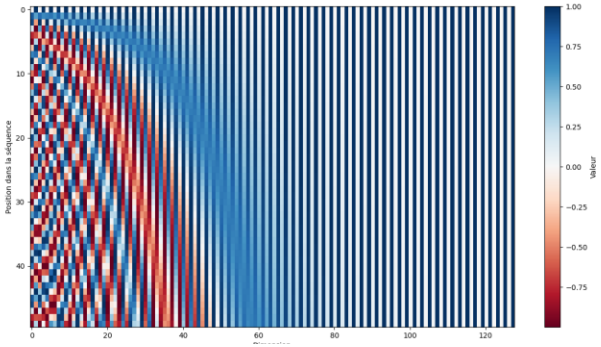
Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Encoder



Contraints ?

- Unique encoding ✔
- Simple relationship between the encoding of two known positions ✔
- Generalization to sequences of any size ✔
- Generalization to any dimension ✔
- Deterministic encoding ✔



Solutions

[Paul_] [est_] [basket] [teur]

pos : 1 2 3 4

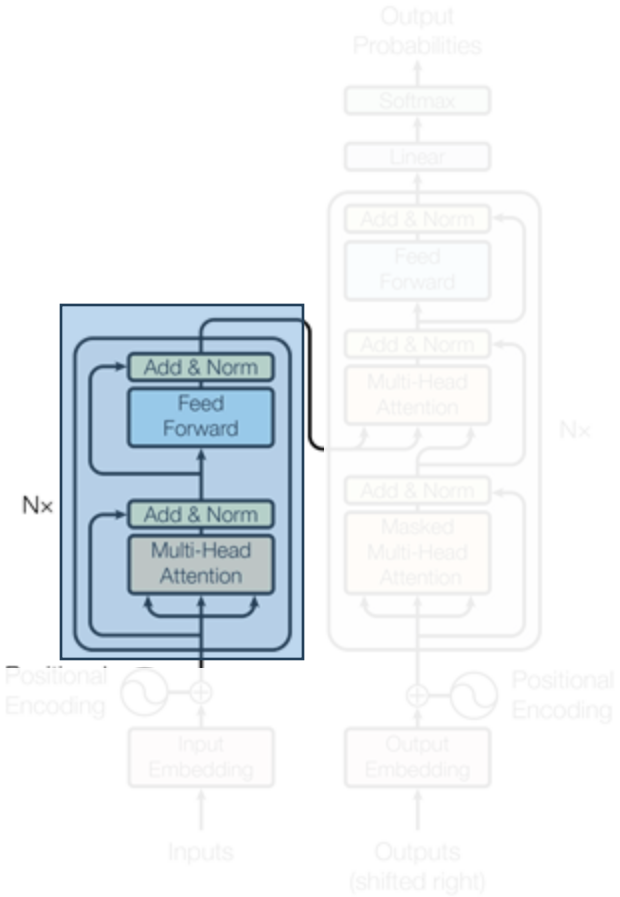
$\forall\, i \in [1, d]$

$PE(pos,2i)=\sin(\frac{pos}{10000^{2i/d}})$

$PE(pos,2i+1)=\cos(\frac{pos}{10000^{2i/d}})$

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Encoder



Encoder : Read and globally encodes the input

- Self attention mechanism

  ⇢Capture relations between tokens

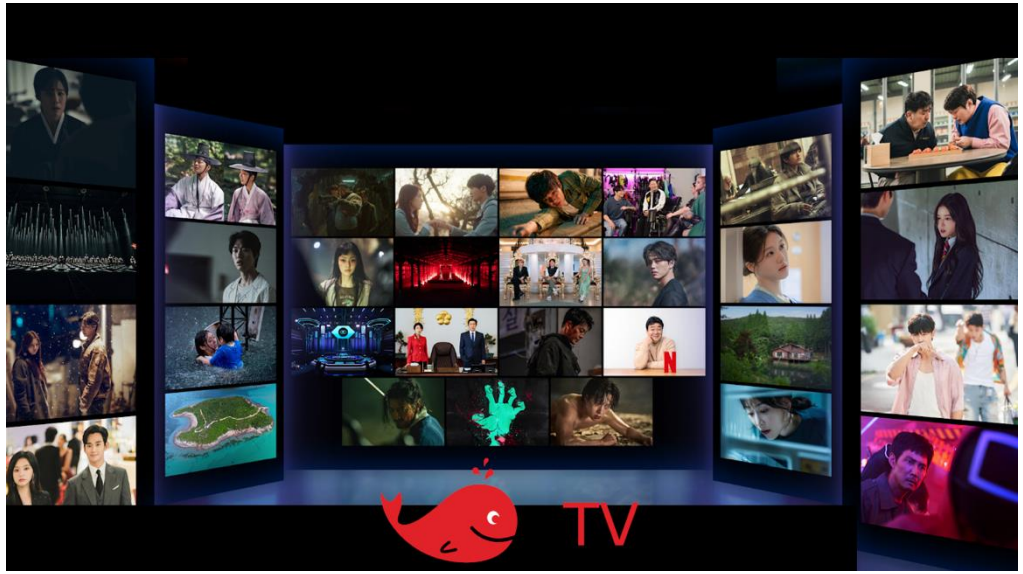
- Use of a MultiLayer Perceptron

  ⇢Introduce non-linearities

- Use of tricks (residual connections, normalization)

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Key, Query, Value



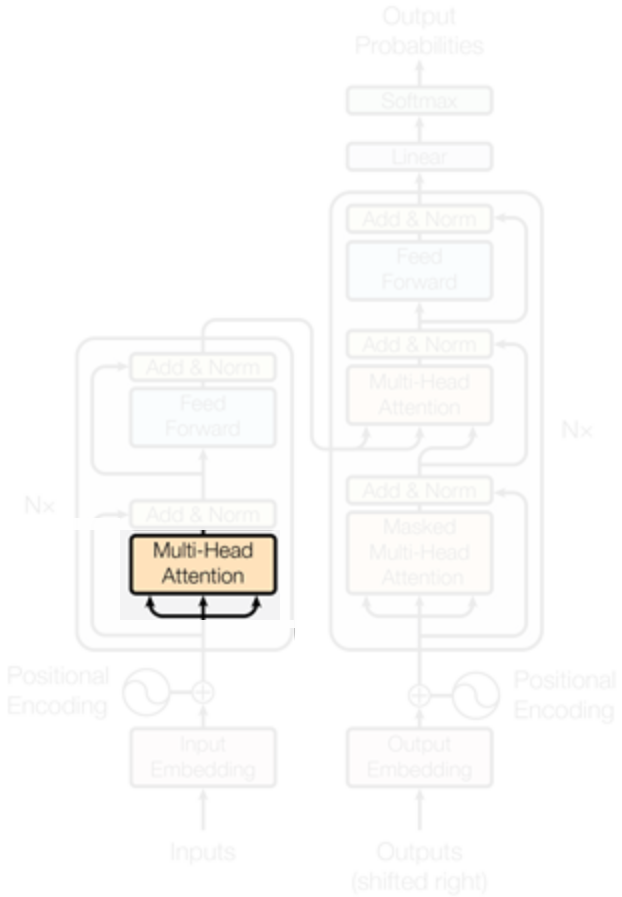
Value $movie = \begin{bmatrix} genre \\ release\ date \\ duration \\ \dots \end{bmatrix}$ Key

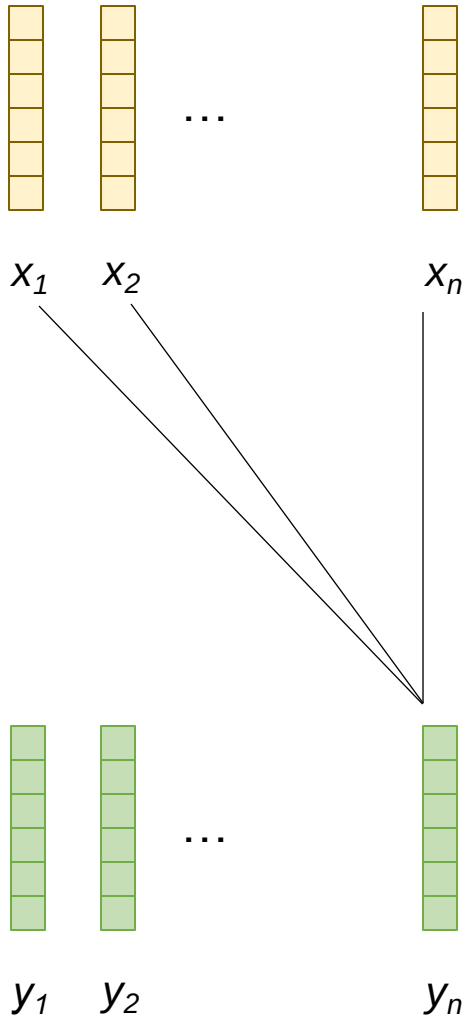
Query

$\begin{bmatrix} comedy \\ 2023 \\ 2h \\ \dots \end{bmatrix}$

I'd watch a film…

Self Attention mecanism





« Paul est basketteur. Il a marqué beaucoup de paniers samedi. »

$$y_i = \sum_{k=1}^{n} a_{ik} x_k$$

$$\begin{cases} a_{ik} \geq 0 \\ \sum_{k=1}^{n} a_{ik} = 1 \end{cases}$$

a_{ik}?

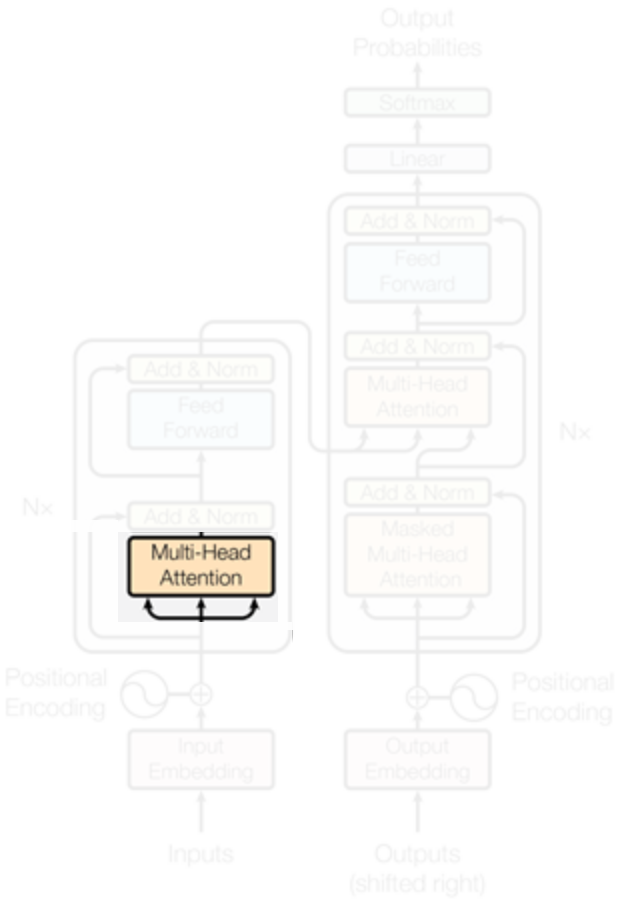
$x_k^T x_i$: to which extent token embedded in x_k has importance for token embedded in x_i

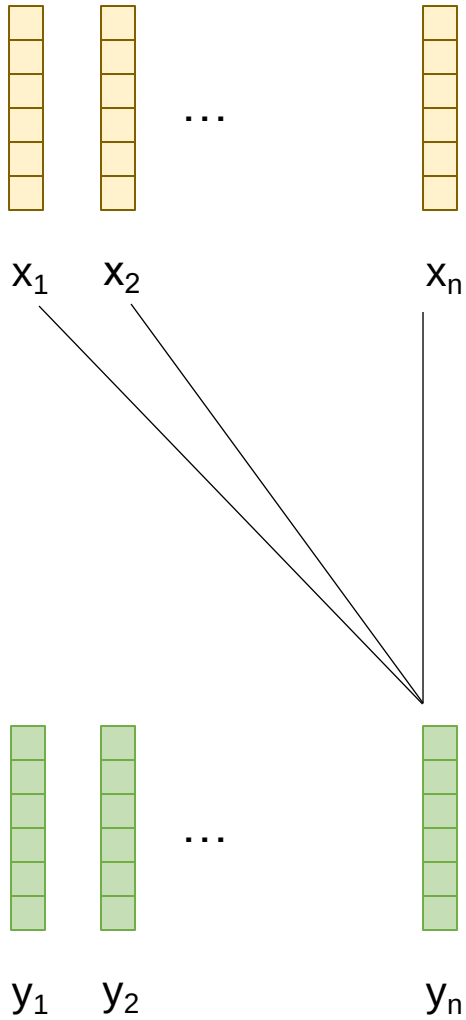
x_i = value = key
x_k = query

$$a_{ik} = \frac{e^{x_k^T x_i}}{\sum_{j=1}^{n} x_j^T x_i}$$

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Self Attention mecanism





$X\ n \times d$

$$y_i = \sum_{k=1}^{n} a_{ik} x_k$$

Self Attention

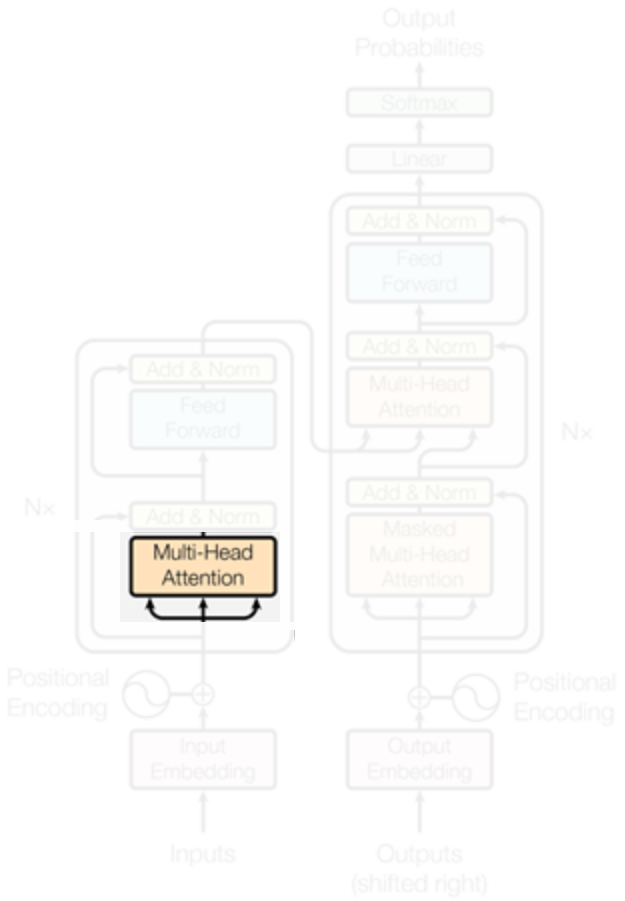
$$Y = softmax[XX^T]X$$

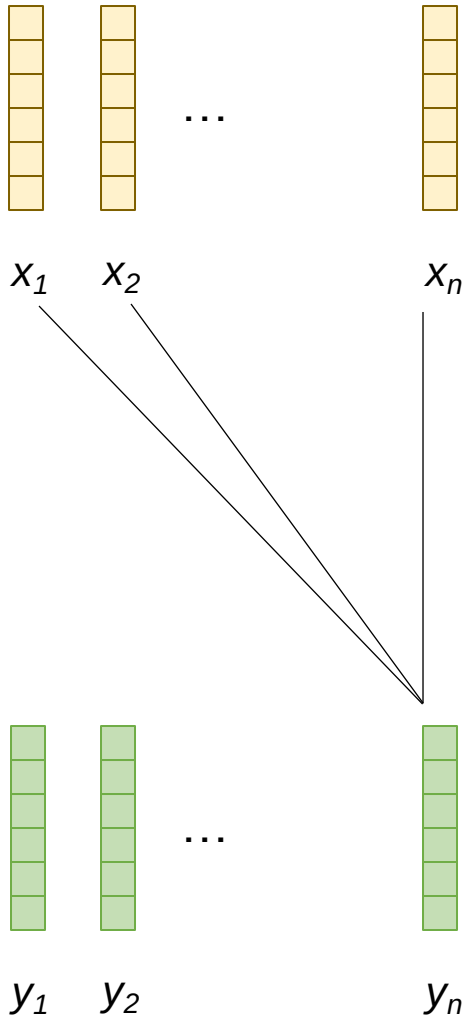
No learning !!

$Y\ n \times d$

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Self Attention mecanism

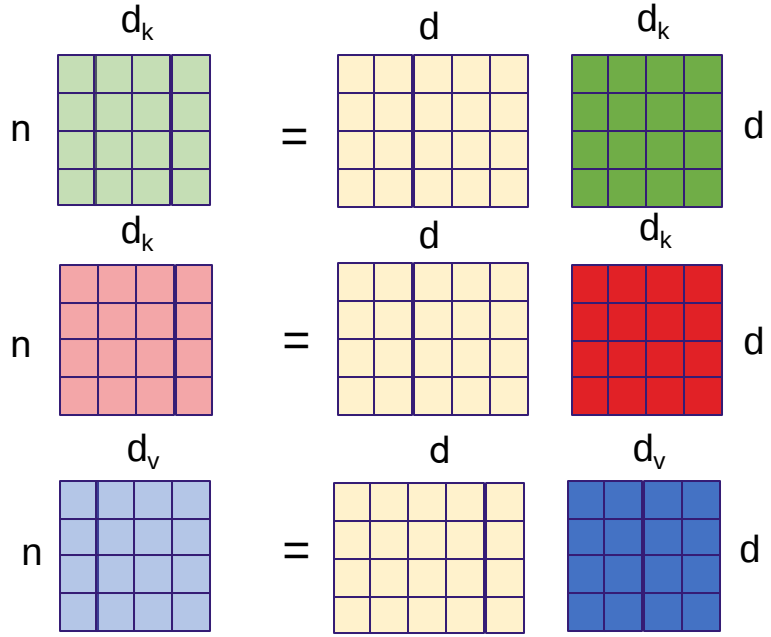




$$Q = XW_q$$

$$K = XW_k$$

$$V = XW_v$$



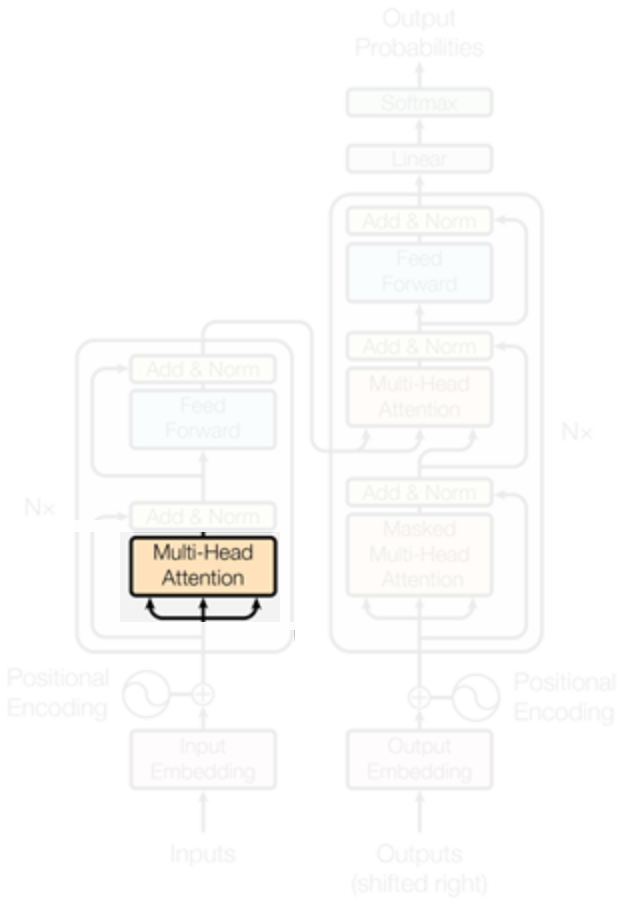
$$Y = softmax[QK^T]V$$

$$Y = softmax\left[\frac{QK^T}{\sqrt{d_k}}\right]V$$

Attention head

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Self Attention mecanism



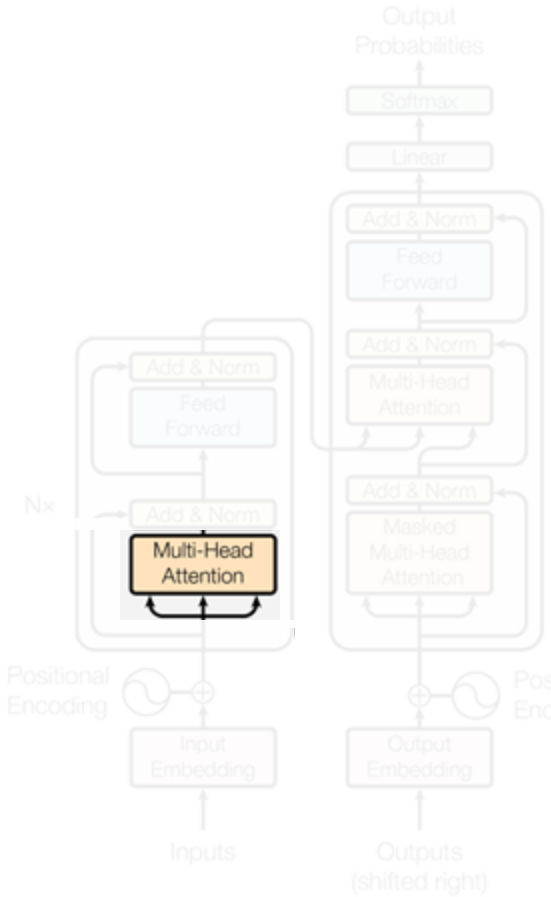
Attention head: capture dependencies between the x_i's

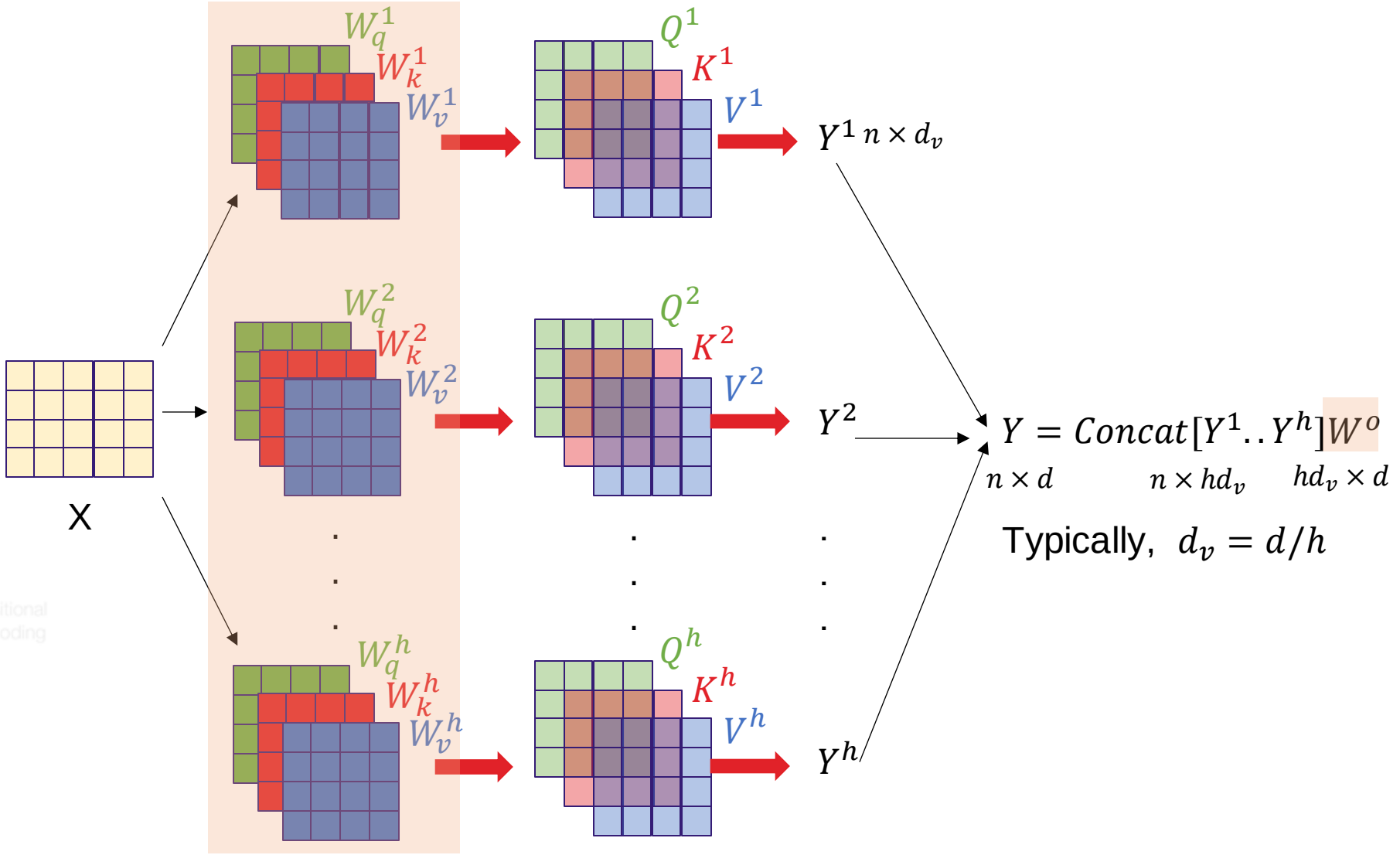
There may exist several types of dependencies

⇢ Multi-head : parallelized attention heads, with independent W_q, W_k, W_v

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

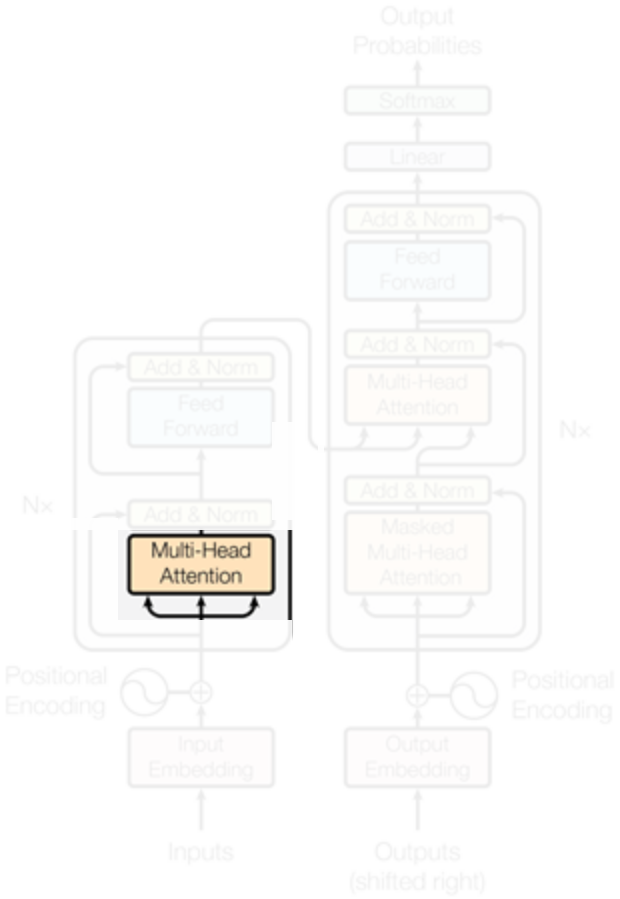
Self Attention mecanism

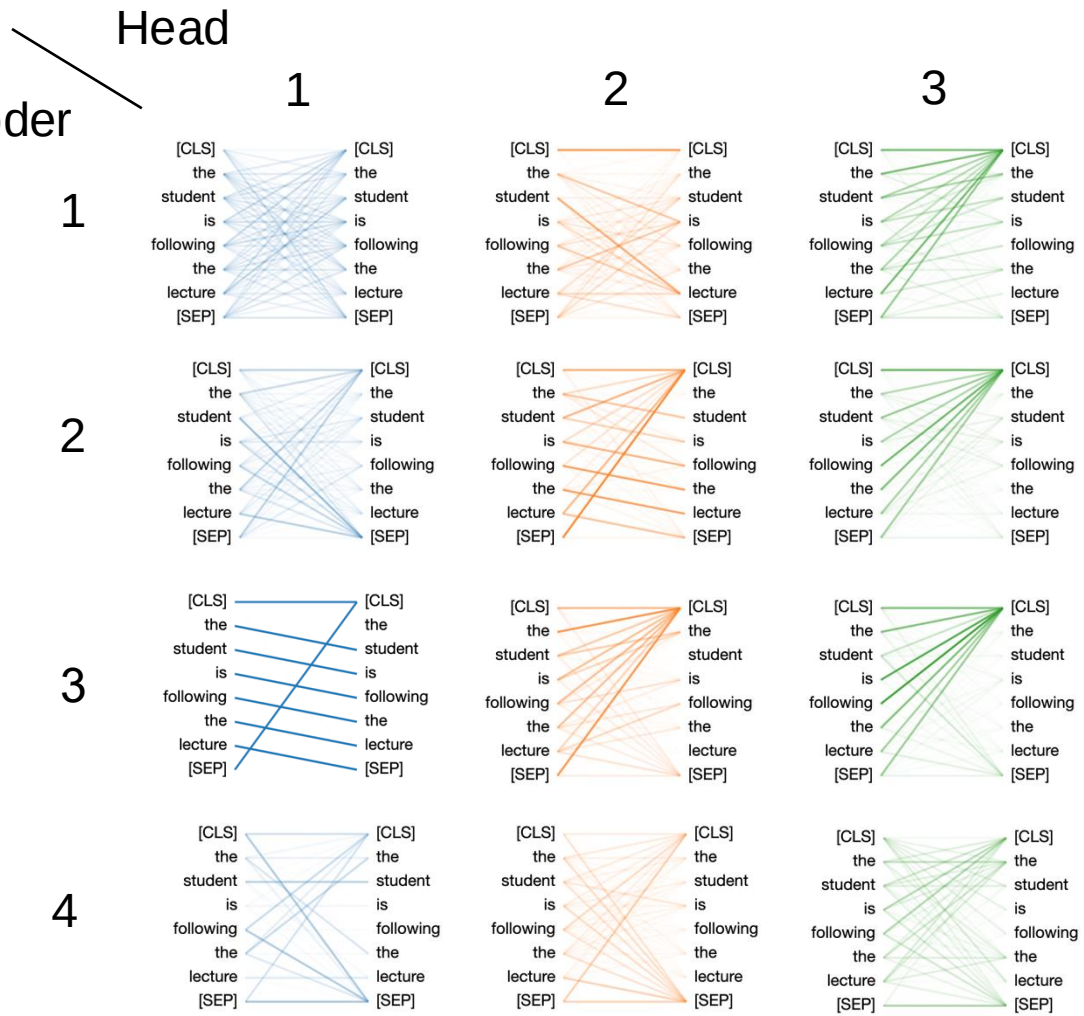


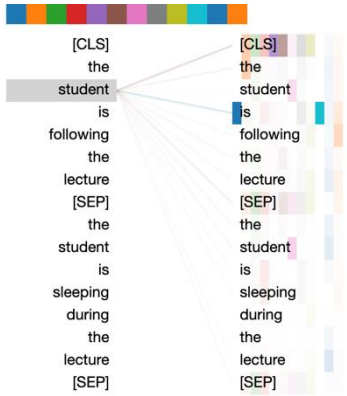


Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Self Attention mecanism



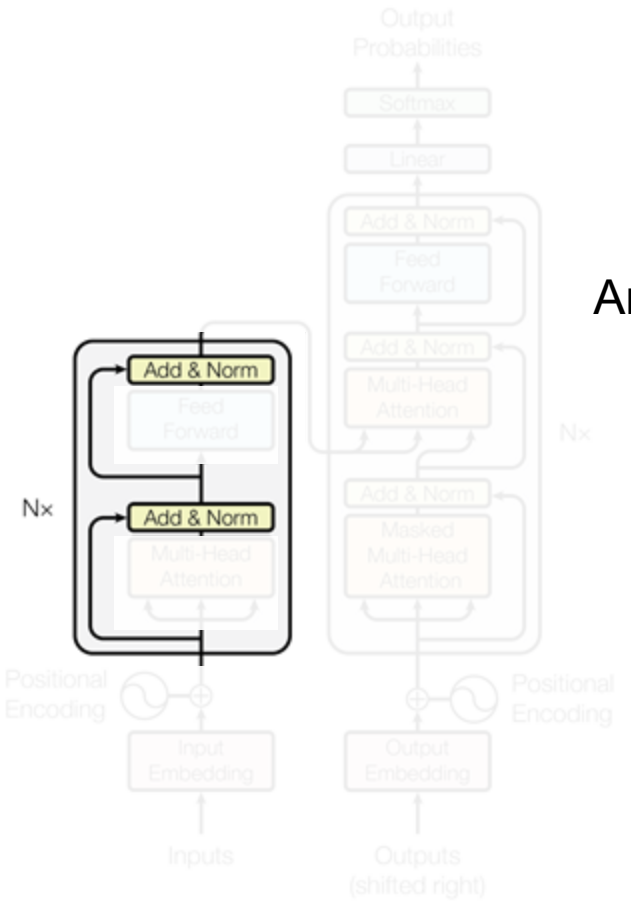




from BertViz

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

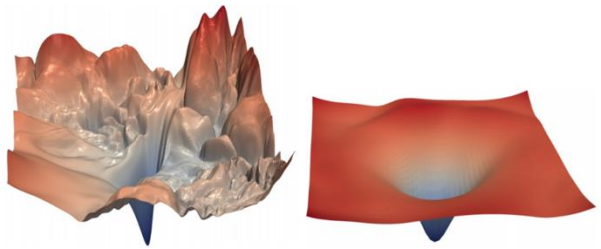
Tricks



Residual connections / Layer normalization

Improve training efficiency

$$Z = LayerNorm(Y + X)$$



An attention head…

Allows to capture token dependencies

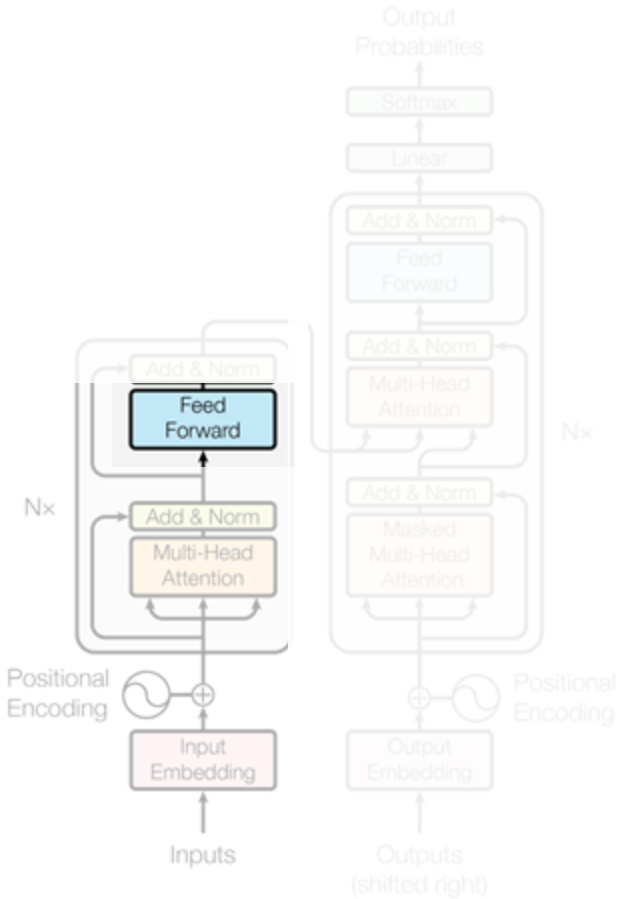
Compute linear combinations of *X* rows

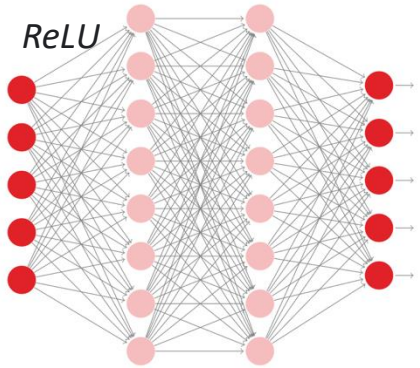
⇢ Limited expressivity !

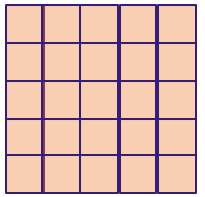
⇢ Post processing of outputs using a MLP

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Multilayer Perceptron







MLP(Z)

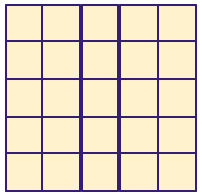
+

Transformer layer

$$\tilde{X} = LayerNorm[PMC(Z) + Z]$$

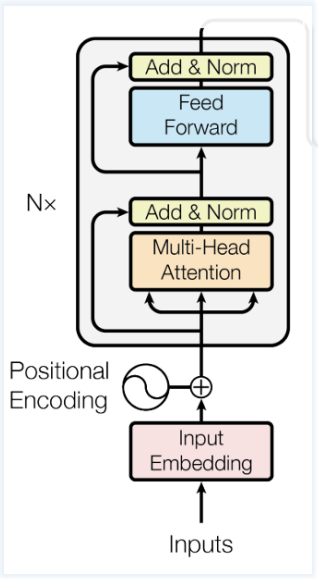
Z

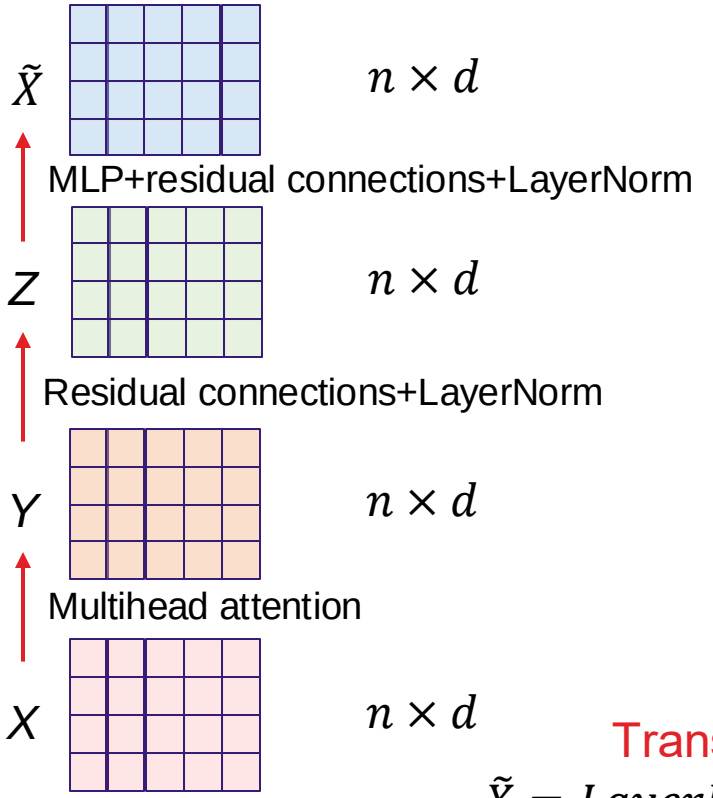
Z
$n \times d$

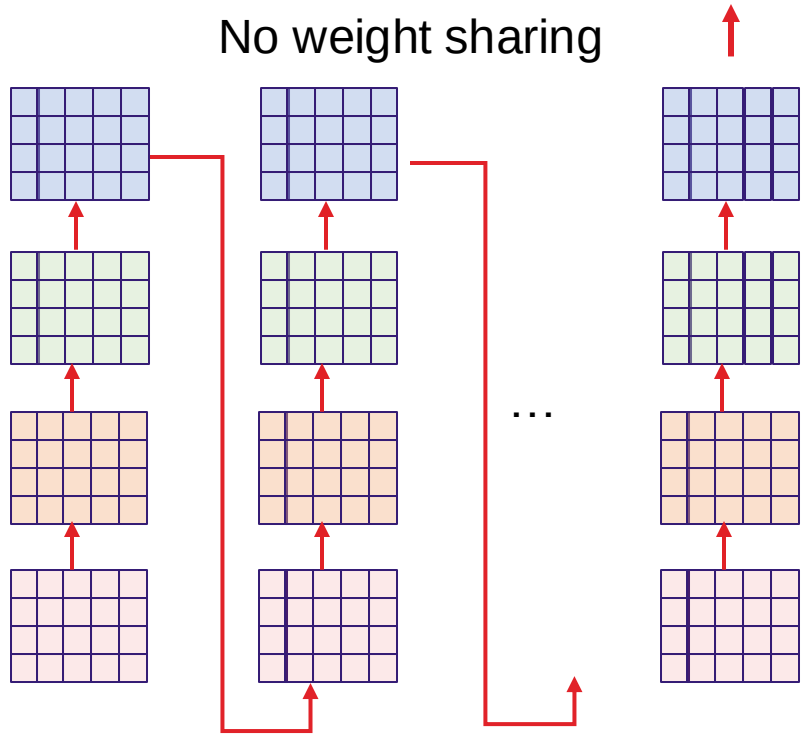


Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Summary…







Transformer layer

$$\tilde{X} = LayerNorm[PMC(Z) + Z]$$

$$Z = LayerNorm(Y + X)$$

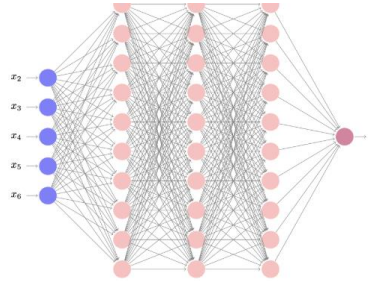
$$Y = Concat[Y^1..Y^h]W^o$$

$$Y^i = softmax\left[\frac{Q^i K^{iT}}{\sqrt{d_k}}\right]V^i$$

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Summary…

https://vbarra.github.io/DLbook/transformers.html



Introduction

- Ressources PyTorch
- Les réseaux de neurones
- Diapositives

Modèles classiques

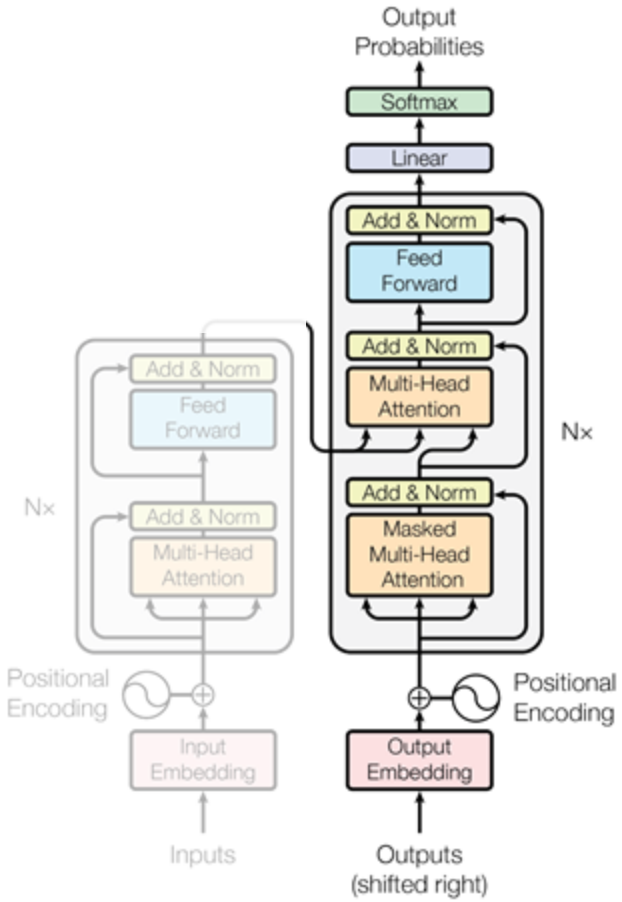
- Perceptron
- Perceptrons multicouches
- Réseaux convolutifs
- Auto-encodeurs
- Réseaux récurrents
- Utilisation de réseaux existants
- Transformers
- Graph Neural Networks

On implémente le mécanisme d'auto-attention multiple (multihead attention)

```python
class MultiHeadAttention(nn.Module):
    def __init__(self, d, H):
        super(MultiHeadAttention, self).__init__()
        assert d % H == 0

        self.d = d
        self.H = H
        self.d_q = d // H

        self.W_q = nn.Linear(d, d)
        self.W_k = nn.Linear(d, d)
        self.W_v = nn.Linear(d, d)
        self.W_o = nn.Linear(d, d)

    # Calcul des attentions
    def Ah(self, Q, K, V, mask=None):
        attn_scores = torch.matmul(Q, K.transpose(-2, -1)) / math.sqrt(self.d_q)
        # Cas de l'auto-attention masquée
        if mask is not None:
            attn_scores = attn_scores.masked_fill(mask == 0, -1e10)
        attn_probs = torch.softmax(attn_scores, dim=-1)
        output = torch.matmul(attn_probs, V)
        return output

    # Redimensionne le tenseur d'entrée pour les têtes
    def tetes(self, x):
        batch_size, N, d = x.size()
        return x.view(batch_size, N, self.H, self.d_q).transpose(1, 2)

    # Combine les attentions de toutes les têtes
    def concateneAttention(self, x):
        batch_size, _, N, d_q = x.size()
        return x.transpose(1, 2).contiguous().view(batch_size, N, self.d)

    # Calcul l'auto-attention multiple.
```

```python
class Transformer(nn.Module):
    def __init__(self, taille_voc_source, taille_voc_cible, d, H, nb_layers, dh,
        super(Transformer, self).__init__()
        self.encoder_embedding = nn.Embedding(taille_voc_source, d)
        self.decoder_embedding = nn.Embedding(taille_voc_cible, d)
        self.positional_encoding = PositionalEncoding(d, max_N)

        self.encoder_layers = nn.ModuleList([EncoderLayer(d, H, dh, dropout) for
        self.decoder_layers = nn.ModuleList([DecoderLayer(d, H, dh, dropout) for

        self.fc = nn.Linear(d, taille_voc_cible)
        self.dropout = nn.Dropout(dropout)

    def generate_mask(self, src, cible):
        src_mask = (src != 0).unsqueeze(1).unsqueeze(2)
        cible_mask = (cible != 0).unsqueeze(1).unsqueeze(3)
        N = cible.size(1)
        nopeak_mask = (1 - torch.triu(torch.ones(1, N, N), diagonal=1)).bool()
        cible_mask = cible_mask & nopeak_mask
        return src_mask, cible_mask

    def forward(self, src, cible):
        src_mask, cible_mask = self.generate_mask(src, cible)
        src_embedded = self.dropout(self.positional_encoding(self.encoder_embedc
        cible_embedded = self.dropout(self.positional_encoding(self.decoder_embe

        enc_output = src_embedded
        for enc_layer in self.encoder_layers:
            enc_output = enc_layer(enc_output, src_mask)

        dec_output = cible_embedded
        for dec_layer in self.decoder_layers:
            dec_output = dec_layer(dec_output, enc_output, src_mask, cible_mask)

        output = self.fc(dec_output)
        return output
```

Decoder – Global view



Role: generate an output sequence from the encoded representation of the input sequence.

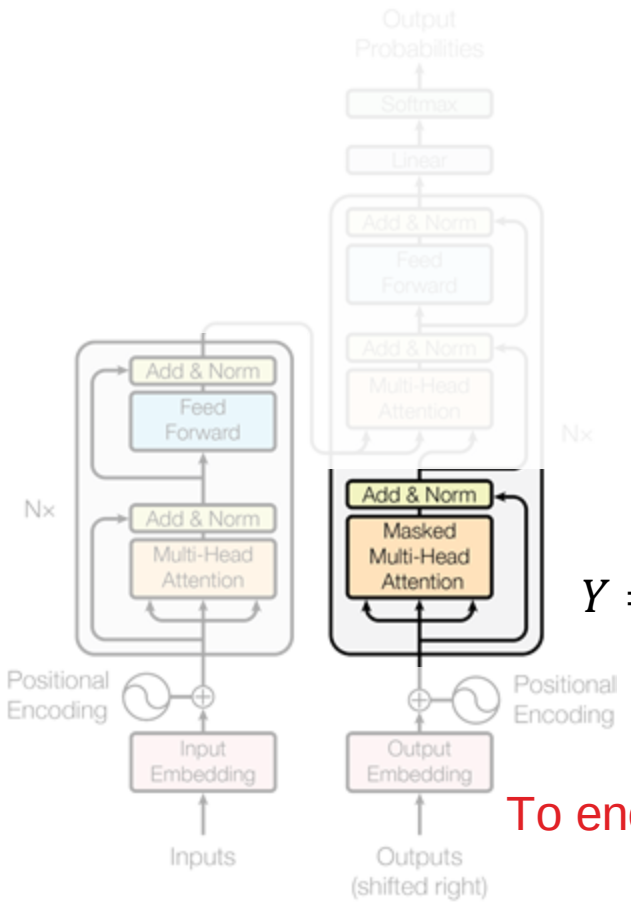
Works in an auto-regressive way: generates one token at a time using the previous tokens

Same components as the encoder but…

- Masked Self-attention
- Encoder-decoder attention

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Decoder – Masked self attention

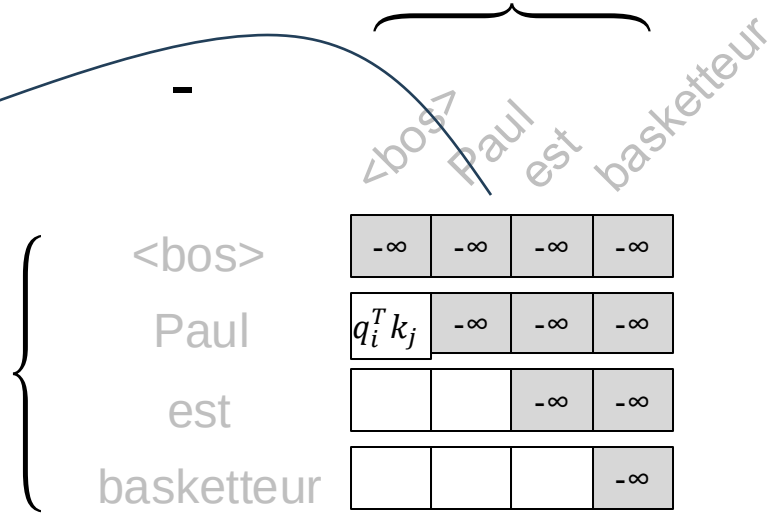


- **Aim** :
  - Consider previous tokens to generate the next one
  - Prevent access to future tokens (masking).
- **How does it work ?** :
  - Compute attention scores between all tokens in the sequence.
  - Apply a $d \times d$ mask

.. we forbid the grey ones.

$$Y = softmax\left[\frac{QK^T}{\sqrt{d_k}}\right]V$$

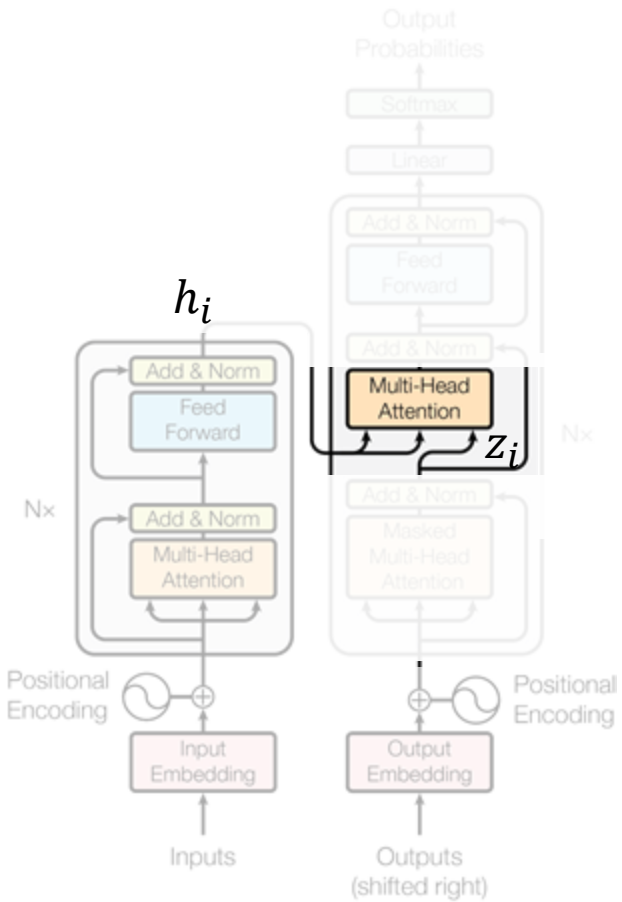
-



To encode these words..

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Decoder – Cross attention



•**Aim** : combine information from the encoder with tokens generated by the decoder

•**How does it work ?** :

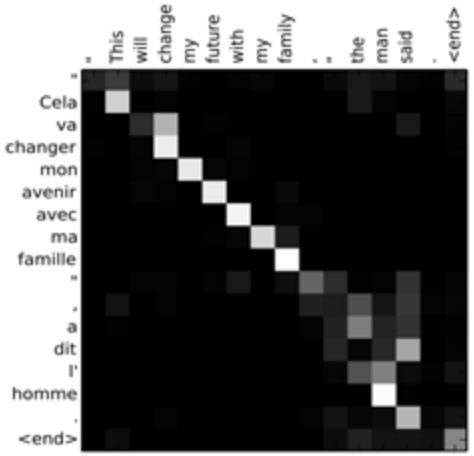
- keys and values : from the encoder's output

$$k_i = Kh_i\ , v_i = Vh_i$$

•Queries: from the output of the masked self attention

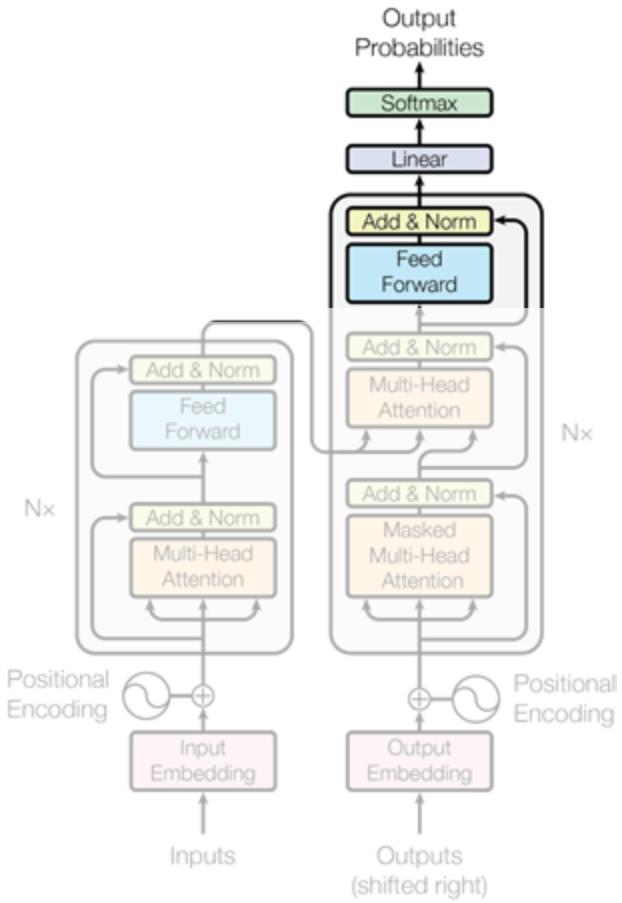
$$q_i = Qz_i$$

•Attention scores between Q and K



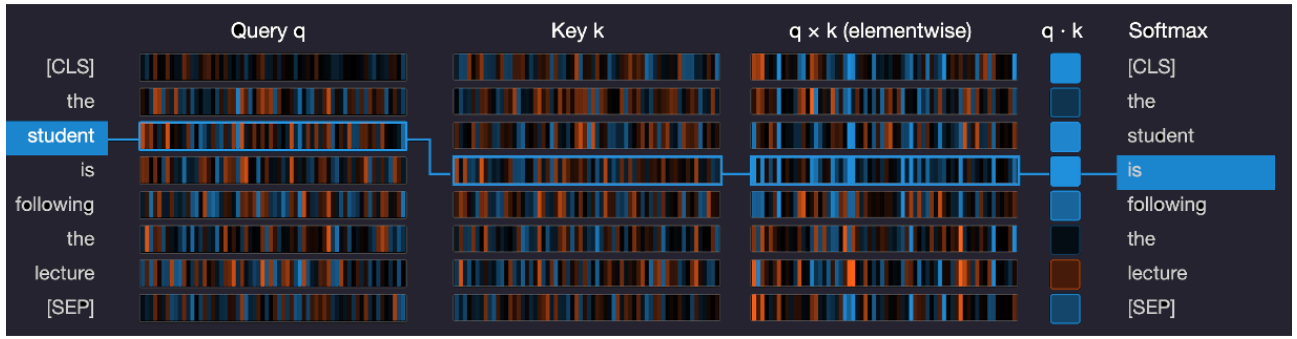
Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Decoder – MLP and output



All set to predict a probability distribution over the entire vocabulary

- **MLP**: nonlinearity
- **Linear layer**: embedding in $\mathbb{R}^d$
- **Softmax**: probability distribution over the tokens



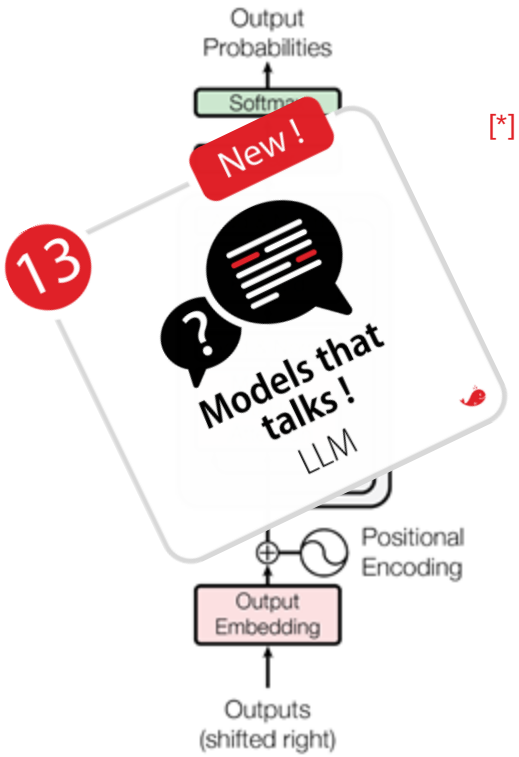
From BertViz

Source "Attention Is All You Need", Vaswani *et al.*, NIPS 2017

Classical architectures

Decoder-only
GPT

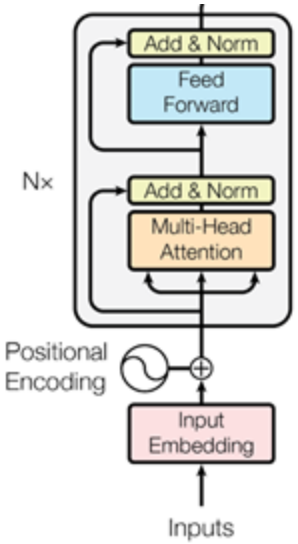
[basketteur_]



[Paul_] [est_]

Encoder-only
BERT

[*] [*] [basketteur_] [*] [*] [marqué_] [*] [*] [paniers]



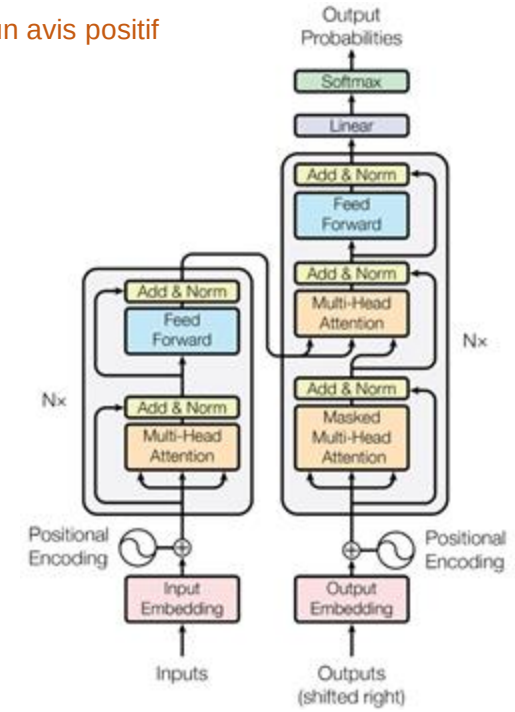
[Paul_] [est_] **[MASK]** [Il_] [a_]
[MASK] [beaucoup_] [de_] [**MASK**]

Encoder-Decoder
T5

Bonjour à tous.

Vous assistez à la séquence 12 de FIDLE.

C'est un avis positif

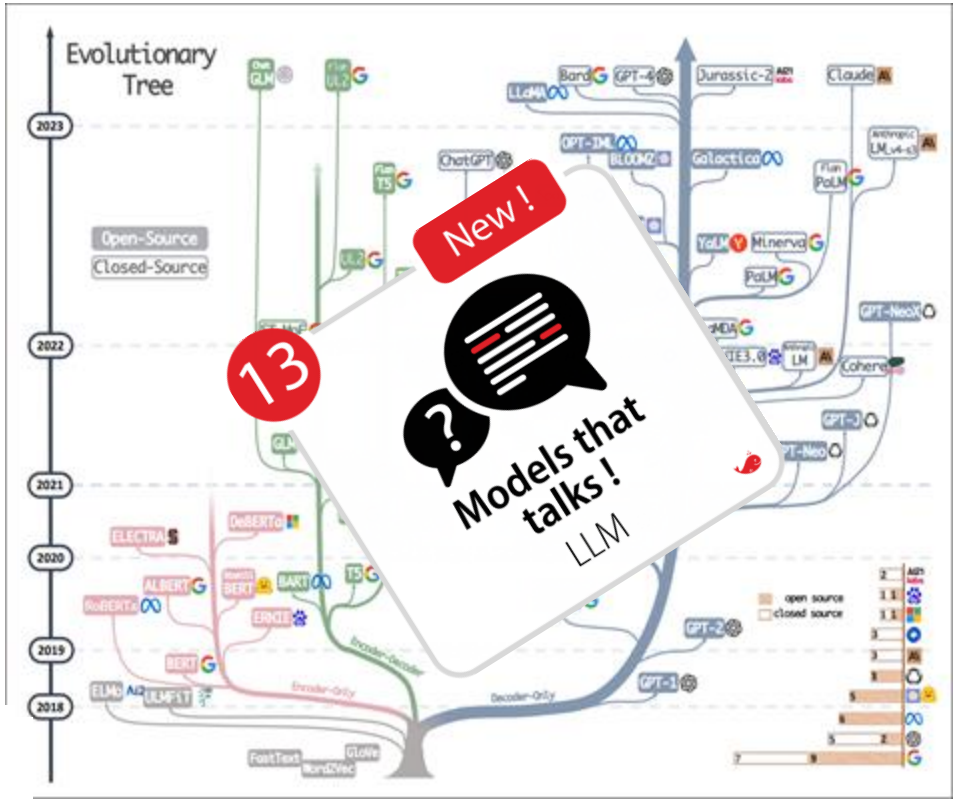


Traduire : FR-EN: Hi everybody.

Résumer: Aujourd'hui a lieu la séquence 12 du programme ...

QA : Est-ce un avis négatif ? Ce cours est génial !

All you need is… attention



https://arxiv.org/abs/2304.13712

The BEATLES

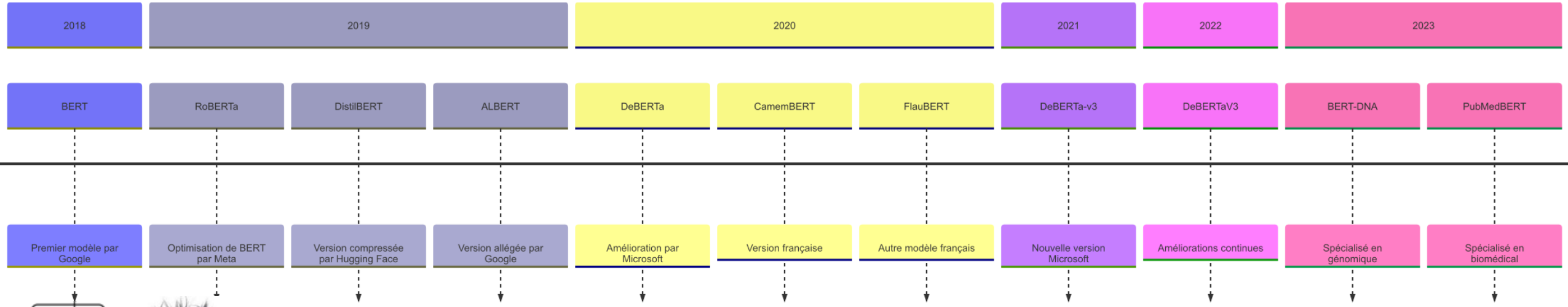
All you need is Attention

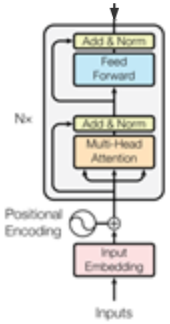
Bidirectional Encoder Representations from Transformers

Devlin et al, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Proc. NAACL-HLT, 2019

key: learn representations based on **bidirectional context**

We went to the river bank.
I need to go to bank to make a deposit.





bookcorpus
/bookcorpus

WIKIPEDIA
The Free Encyclopedia

BERT base, N=12
110M parameters

BERT large, N=24
340M parameters

Pretraining, two objectives:

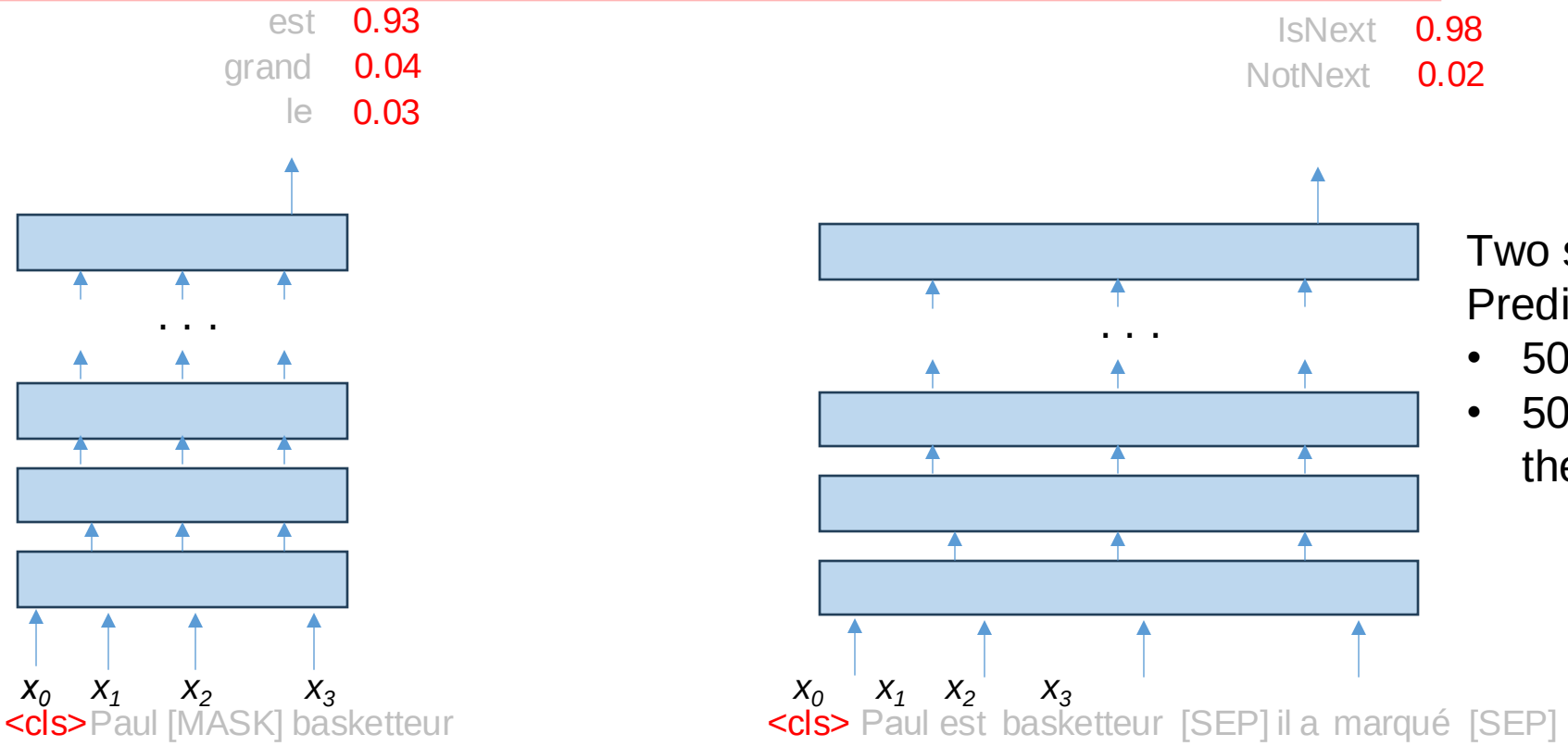
- MLM (Masked Language Model)
- NSP (Next Sentence Prediction)

Fine tuning using transfer learning

BERT- pretraining

MLM (Masked Language Model) **NSP** (Next Sentence Prediction)

- Trained together
- <cls> for NSP
- Other tokens for MLM



Two sentences S_1, S_2 fed in at a time.
Predict if S_2 follows S_1

- 50% of the time S_2 randomly selected
- 50% of the time, truly follows S_1 in the corpus

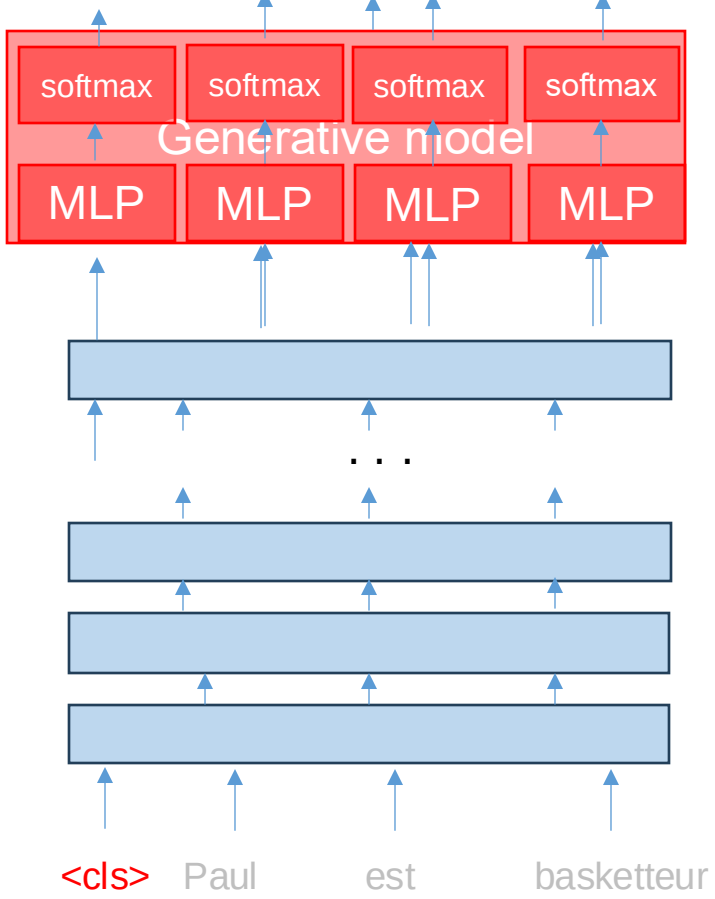
15% of all input words randomly masked.

- 80% [MASK]
- 10% revert back to the same word
- 10% corrupted as wrong word

Tokenizer
+
Positional encoding
+
Segment encoding

word $\longrightarrow$ x_j

BERT- Fine tuning



Text classification: only the first output position used

Token classification

Text to image synthesis

Trained model

Question break

FIDLE

12

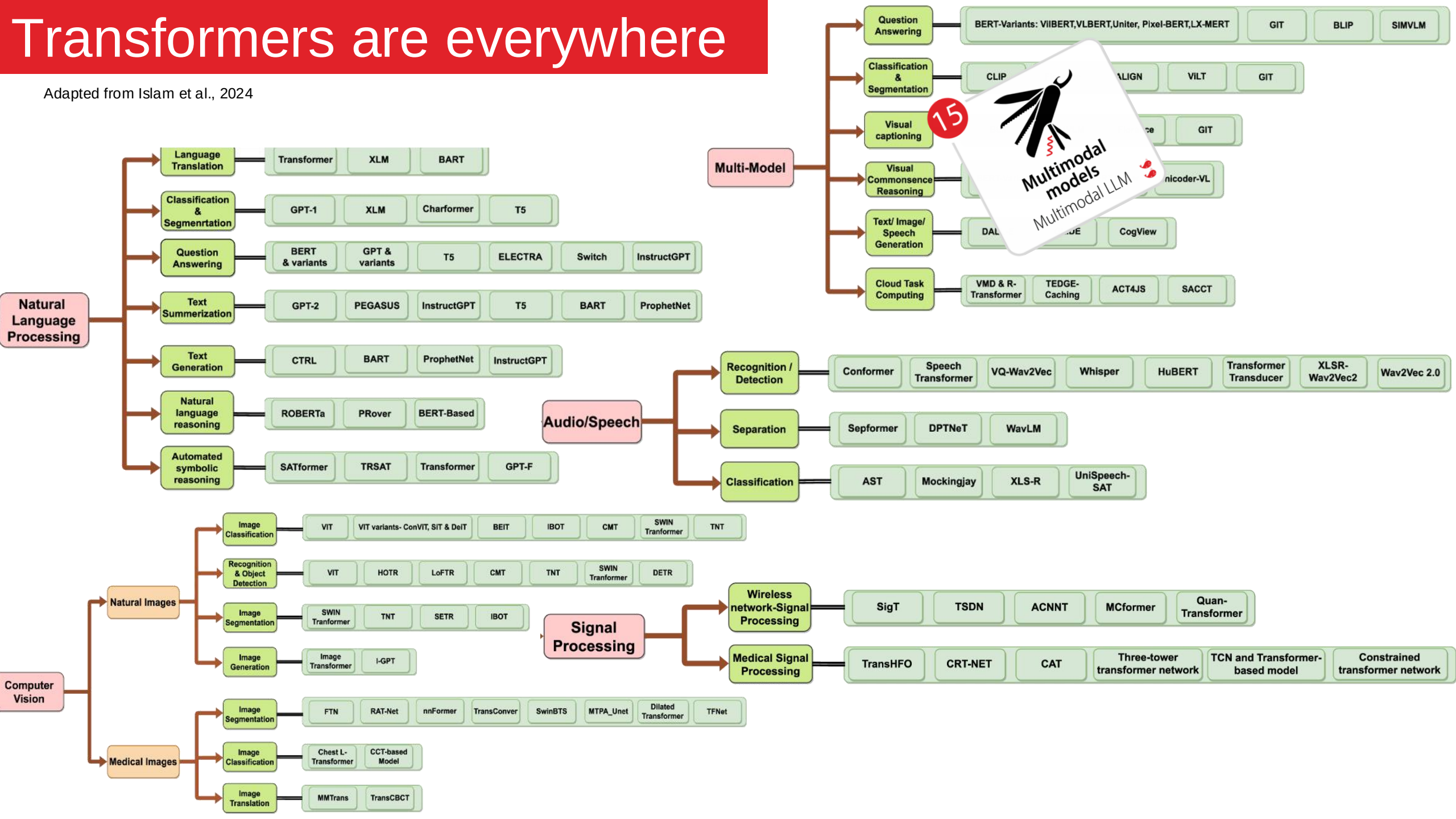
«Attention is All You Need»

RNN, Transformers

1. Prehistory
2. Recurrent Neural Networks
3. The Transformers area
4. **Applications**
5. And then …

Transformers are everywhere

Adapted from Islam et al., 2024



Computer Vision

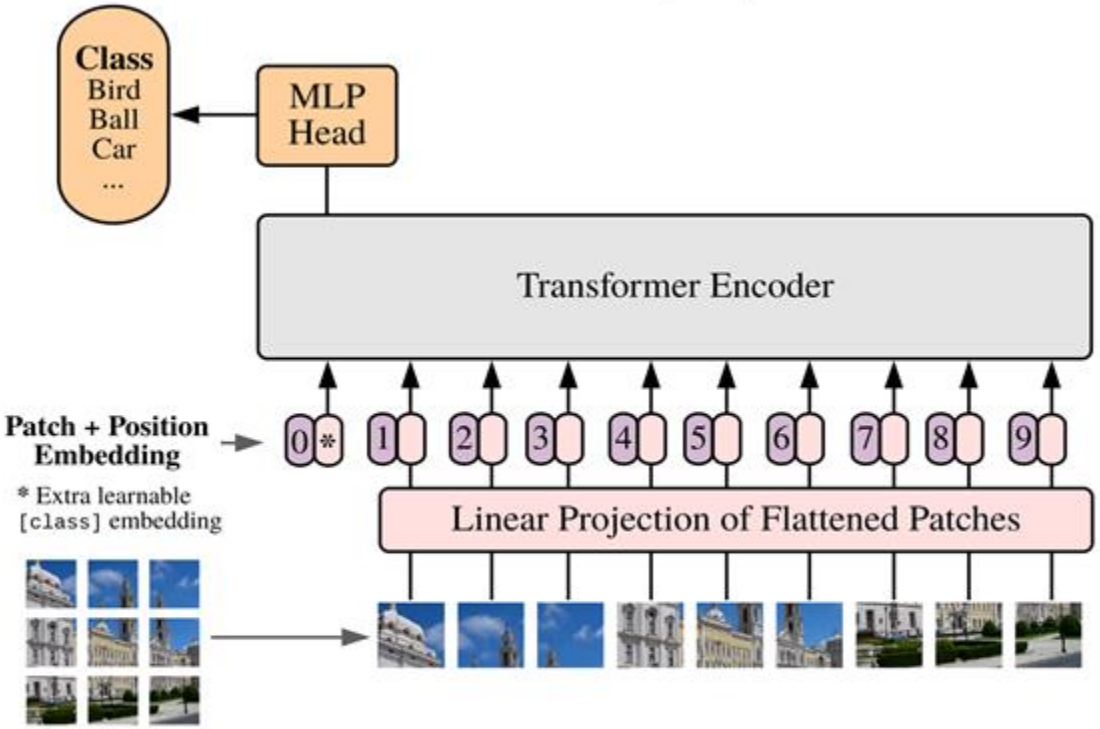
Many prior works attempted to introduce self-attention at the pixel level.

For 224px², that's 50k sequence length, too much!

Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020

Breakthrough:

- **"Tokenize" the image** by **cutting it into patches** of 16x16 pixels,
- Process each patch as a token, e.g. embedding it into input space.



Vision Transformer (ViT)

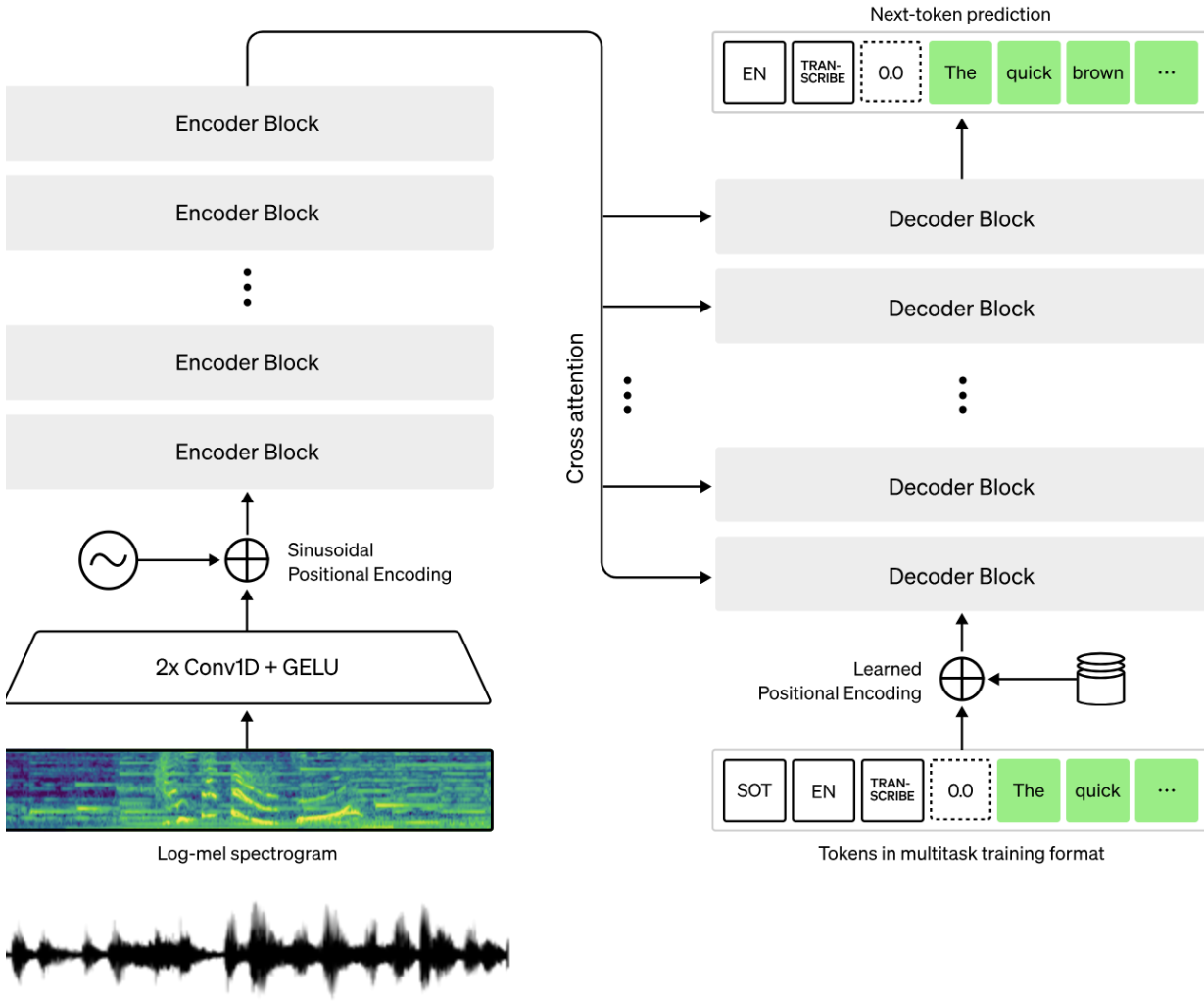
Speech Recognition

Gulati et al., Conformer: Convolution-augmented Transformer for Speech Recognition, 2020

Radford et al., Robust Speech Recognition via Large-Scale Weak Supervision, 2022

≈ Computer Vision.

Images ↔ spectrograms



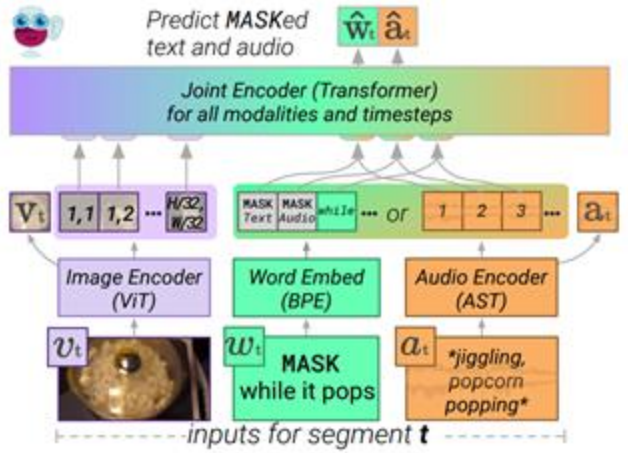
Exists as encoder-decoder variant, or as encoder-only variant with CTC loss.

And many more…

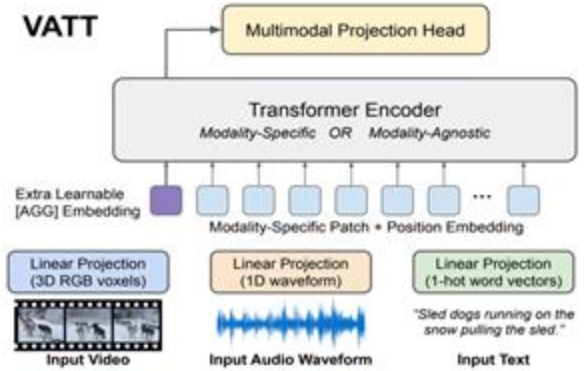
A Transformer can ingest any data that can be tokenized

(Multimodal) data → token(s) (adapter) → Transformer → output

MERLOT. Source: Zellers et al. 2021



VATT. Source: Akbari et al., , 2021



LIMoE. Source: Mustafa et al., 2022

12

«Attention is All You Need»

RNN, Transformers

1. Prehistory
2. Recurrent Neural Networks
3. The Transformers area
4. Applications
5. **And then …**

What need to be fixed ?

FIDLE

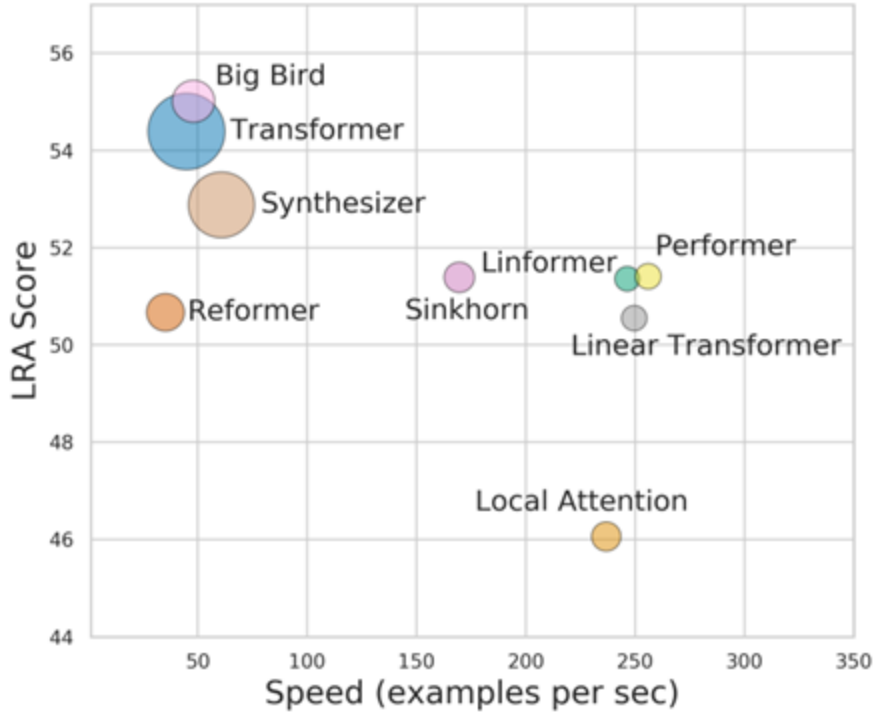
All pairs of interactions → **Quadratic compute** in self-attention: ***O(n²d)***

For recurrent models, it only grew linearly!

We'd like to use large *n*:

- Whole articles or books
- Full video movies
- High resolution images

Many *O(n)* approximations to the full self-attention have been proposed in the past two years.



« Surprisingly, we find that most modifications do not meaningfully improve performance »

Do Transformer Modifications Transfer Across Implementations and Applications?

Sharan Narang* Hyung Won Chung Yi Tay William Fedus

Thibault Fevry† Michael Matena† Karishma Malkan† Noah Fiedel

Noam Shazeer Zhenzhong Lan† Yanqi Zhou Wei Li

Nan Ding Jake Marcus Adam Roberts Colin Raffel†

New perspectives ?

New mechanisms to replace attention in Transformers:
Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

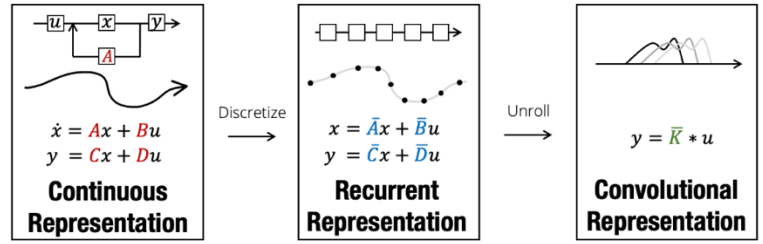
New perspectives ?

New mechanisms to replace attention in Transformers:

Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

State Spaces Models (SSM):

- A class of models inspired by control theory and dynamical systems.
- Represent sequences as state transitions, capturing dependencies over time.
- Better suited for continuous data (e.g., audio, video, time series).
- Reduced memory and computational requirements compared to Transformers



Source: Gu et al, 2022

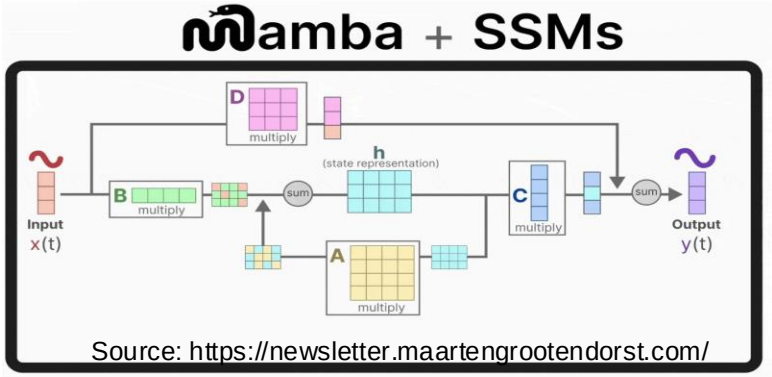
New perspectives ?

New mechanisms to replace attention in Transformers:

Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

State Spaces Models (SSM):

- A class of models inspired by control theory and dynamical systems.
- Represent sequences as state transitions, capturing dependencies over time.
- Better suited for continuous data (e.g., audio, video, time series).
- Reduced memory and computational requirements compared to Transformers
- Example: MAMBA



Source: https://newsletter.maartengrootendorst.com/

New perspectives ?

New mechanisms to replace attention in Transformers:

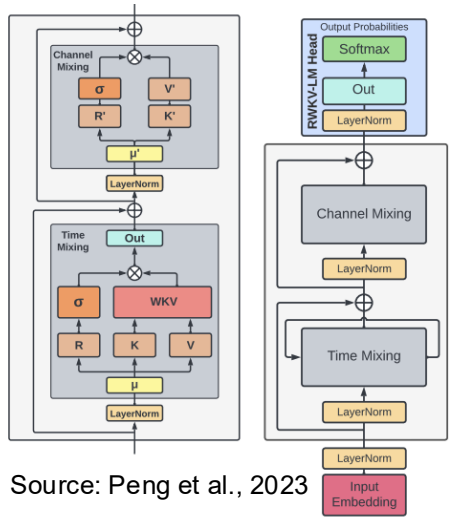
Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

State Spaces Models (SSM):

- A class of models inspired by control theory and dynamical systems.
- Represent sequences as state transitions, capturing dependencies over time.
- Better suited for continuous data (e.g., audio, video, time series).
- Reduced memory and computational requirements compared to Transformers
- Example: MAMBA

Receptance Weighted Key Value (RWKV):

- Efficient parallelizable training of transformers
- Efficient inference of RNNs



Source: Peng et al., 2023

New perspectives ?

New mechanisms to replace attention in Transformers:

Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

State Spaces Models (SSM):

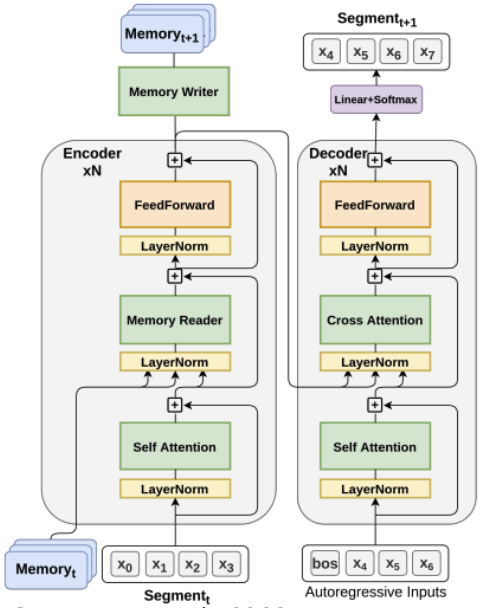
- A class of models inspired by control theory and dynamical systems.
- Represent sequences as state transitions, capturing dependencies over time.
- Better suited for continuous data (e.g., audio, video, time series).
- Reduced memory and computational requirements compared to Transformers
- Example: MAMBA

Receptance Weighted Key Value (RWKV):

- Efficient parallelizable training of transformers
- Efficient inference of RNNs

Memory-Augmented Models

- Utilizes an external dynamic memory to encode and retrieve past information
- Scalable to very long sequences



Source: Wu et al., 2022

New perspectives ?

New mechanisms to replace attention in Transformers:

Gated recurrences / convolutions, data-dependent linear attention, sparse attentions,…

State Spaces Models (SSM):

- A class of models inspired by control theory and dynamical systems.
- Represent sequences as state transitions, capturing dependencies over time.
- Better suited for continuous data (e.g., audio, video, time series).
- Reduced memory and computational requirements compared to Transformers
- Example: MAMBA

Receptance Weighted Key Value (RWKV):

- Efficient parallelizable training of transformers
- Efficient inference of RNNs

Memory-Augmented Models

- Utilizes an external dynamic memory to encode and retrieve past information
- Scalable to very long sequences

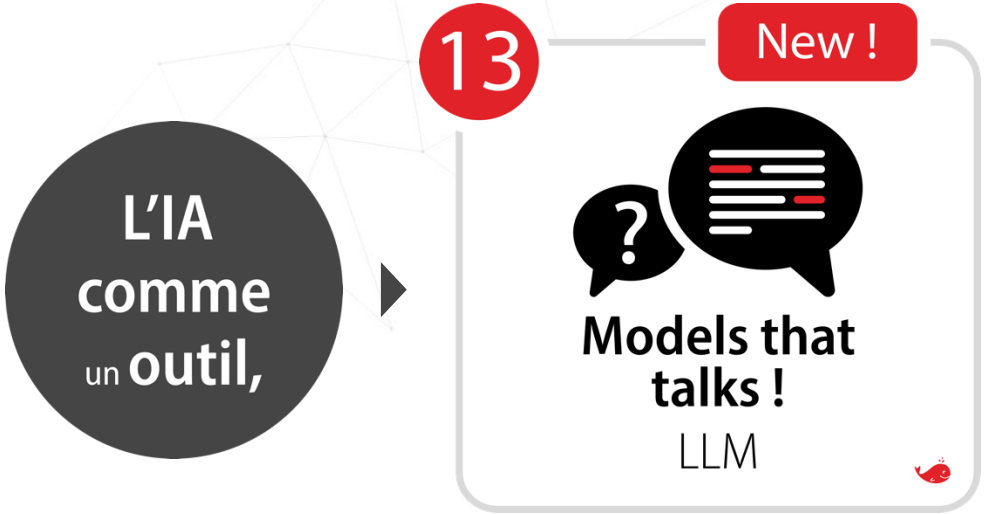
But Today it seems that Transformers are still obtaining the state of the art results un many domains

Thanks for your… Attention

FIDLE

Jeudi 20 Mars 2025, 14h

Prochaine séquence :



cnrs UGA Université Grenoble Alpes MIAI Cluster IA EFELIA

GENCI ANITI DEVLOG RÉSEAU GROUPE CALCUL Resinfo cnrs IDRIS SIMaP