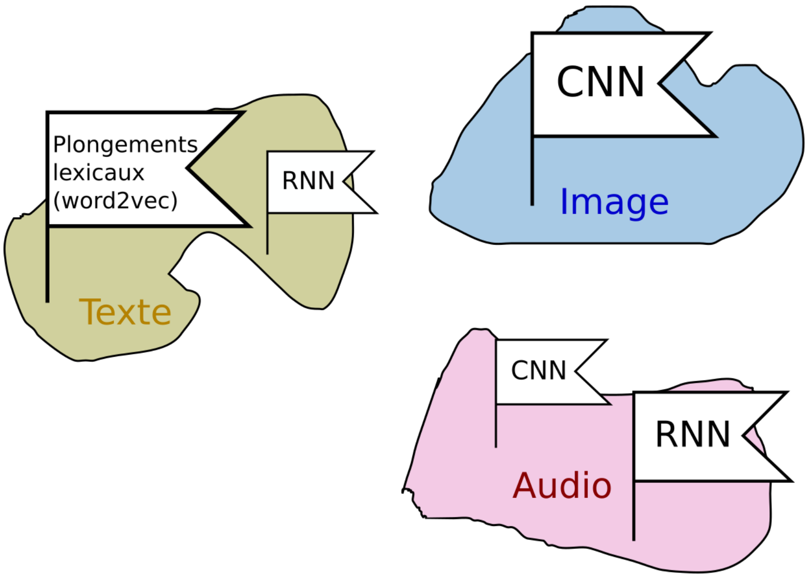
Efficient Transformers

Paul Gay

:231285182275585/photo/1
6454690451457 https://generated.photos/

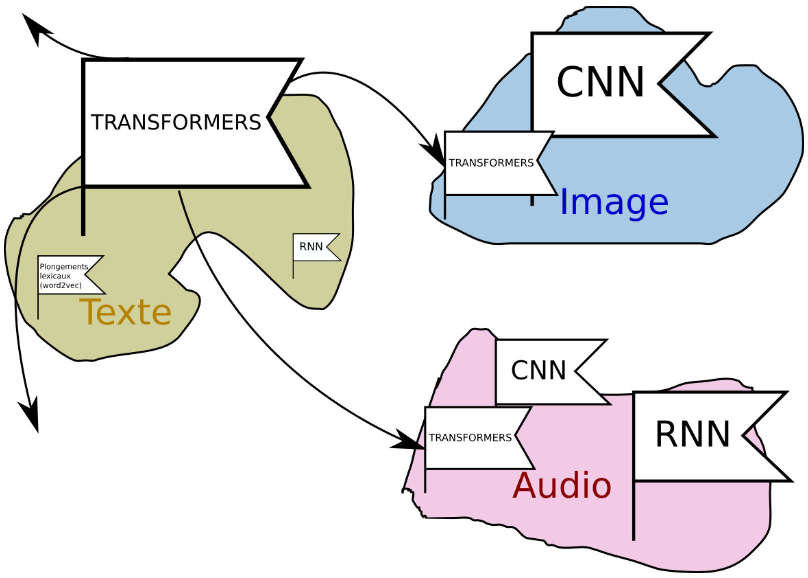
Roughly, how they appear

Before 2018,



Roughly, how they appear

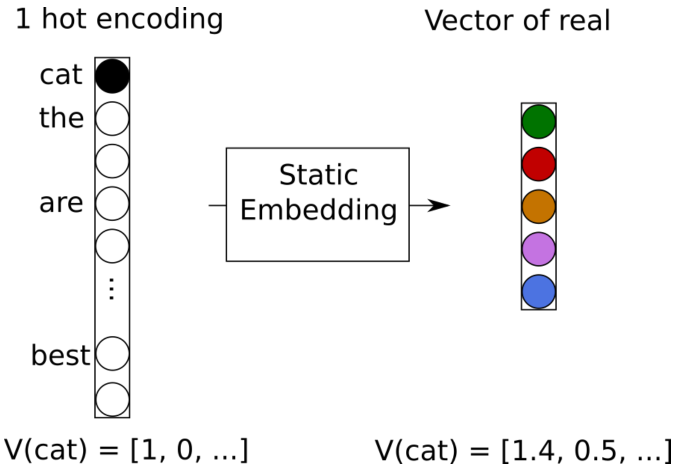
after 2019,



Transformurs

- **Attention**
  - new model for sequential data
- Text generation $P(w_t | w_{t-1})$:
  - predict the next word given the previous ones
- Self supervised loss function
- New state of the art on word embeddings

Non contextual embedding



- Semantic representation to convert a word in vector

$$King \text{ - } man + woman \approx queen$$

Contextual word embedding

- Make the difference between bank and bank?

I catch a fish from the river bank

And I attacked a bank...

- Context : depends on surrounding words
  - **Difference** w.r.t. word2vec which are dictionnaries
- This is what transformers can do

In practice

These models have enabled the learning on huge corpora and set a new state of the art in NLP.

- Many models availble for different languages and domains
- Hugging Face Library

In practice

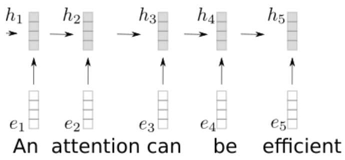
These models have enabled the learning on huge corpora and set a new state of the art in NLP.

- Many models availble for different languages and domains
- Hugging Face Library
- Come to see them at IAPau4!

Live conference on Friday : 03/12/2021

Attention mechanism

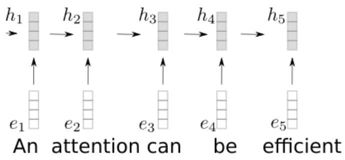
Learn on a sequence

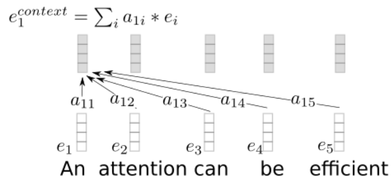


- Words are given one by one to the RNN to refine its latent state h_i
- Compute the state h_i requires to first compute the previous states $h_{1:i-1}$

Attention mechanism

RNN vs attention

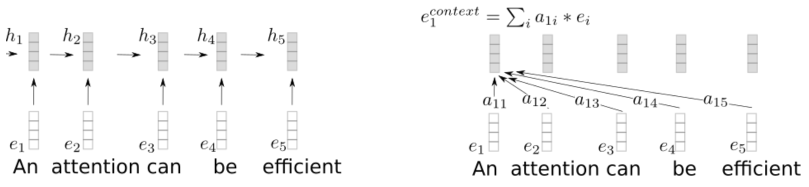




- Contextual representation $e_i^{context}$ is a weighted sum of the input vectors
- a_{ij} coefficients are attention weights

Attention mechanism

RNN vs attention



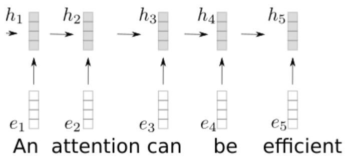
- Contextual representation $e_i^{context}$ is a weighted sum of the input vectors
- a_{ij} coefficients are attention weights

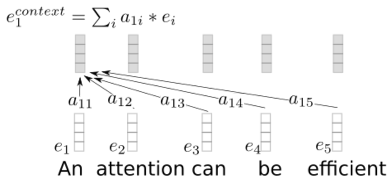
Commonly used in deep learning

Bahdanau et al. NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. ICML, 2015

Attention mechanism

RNN vs attention





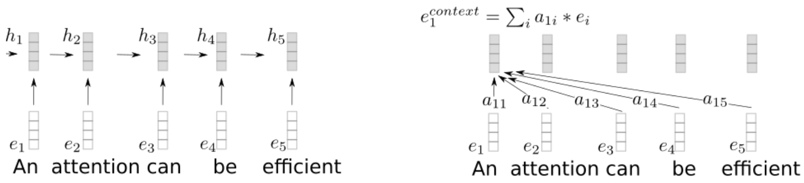
Advantage 1: information propagation

- Direct access to each word

=> Lower risk of forgetting than RNN

Attention mechanism

RNN vs attention

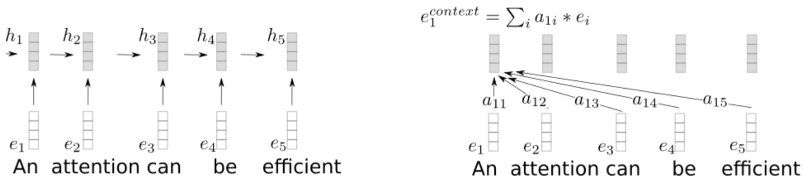


Advantage 2: Parallelisation

- The $e_i^{context}$ can be computed in parallel
  - Recall : with RNN, we would have need previous states
  - Faster learning enables bigger models

Attention mechanism

RNN vs attention



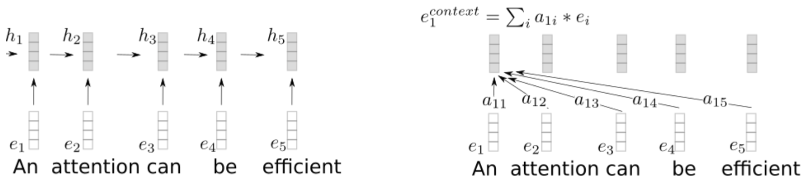
Drawback 1: Word sequence order

- The computation does not take into account the word position

We lost this information in $\sum_i a_{ji} e_i$

Attention mechanism.

RNN vs attention

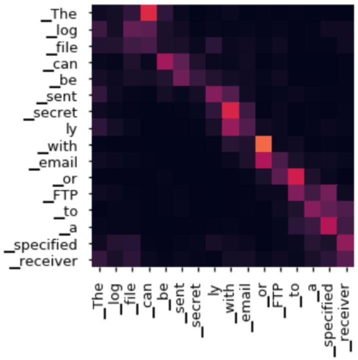


Drawback 2: Quadratic memory cost!

- We have to handle the matrix of the a_{ij} of size $n_{seq} \times n_{seq}$
- Usually, sequences are limited to $n_{seq} = 512$ ou 1024

Model interpretation

- Visualise the attention weights a_{ij} indicates what the model considers important or ignore

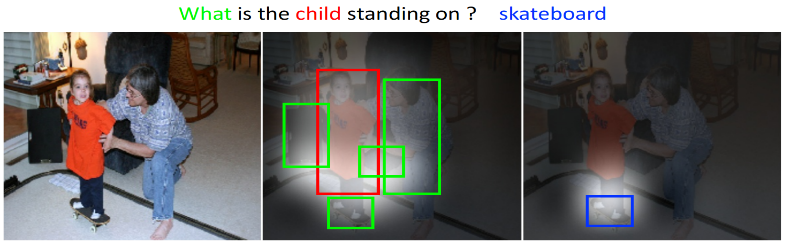


OpenNMT: Open-Source Toolkit for Neural Machine Translation

On this example, "*the*" depends heavily on "*can*".

Model interpretation

- Visualise the attention weights a_{ij} indicates what the model considers important or ignore



Quickly adopted in Vision, audio, text, graph, to increase model expression power.

Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. CVPR 2015

Auto-attention

Transfomers are models based on attention

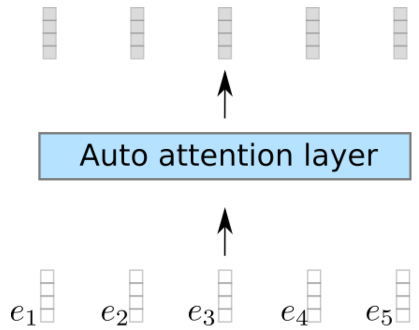
- First proposed for automatic translation

Vaswani et al. Attention is all you need. NIPS, 2017

- And then to create static embeddings (BERT)

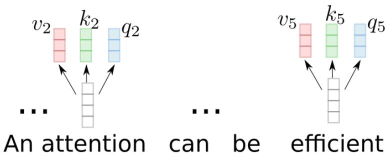
Delvin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805, 2018

Auto-attention



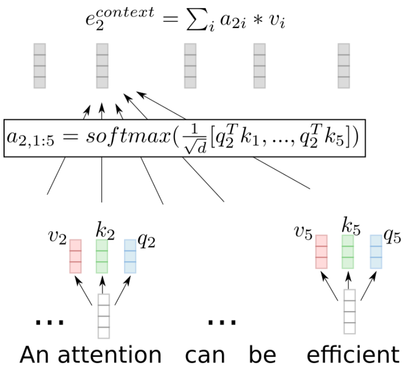
- Goal : to transform a sequence of vectors in a more meaningful sequence
- Main transformer building block, like convolutional layer for CNN

Auto-attention



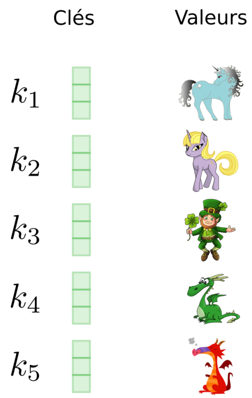
- Each input vector e_i is transformed :
  - $v_i = Ve_i$ // Value
  - $q_i = Qe_i$ // Query
  - $k_i = Ke_i$ // Key
- with V, Q et K learned matrices

Auto-attention



- Attention weights computed from key and request vectors
- Final output $e^{context}$: weighted sum => information selection

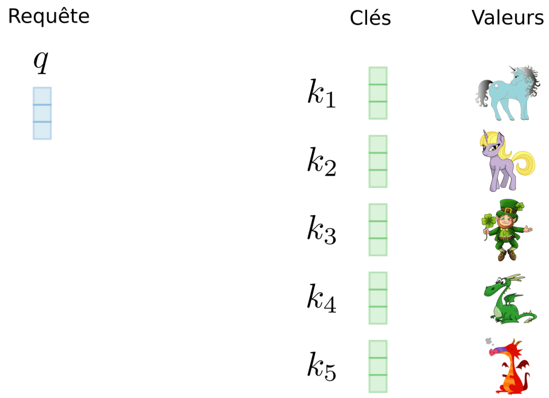
"Key"", "Value" et "Query"?



Term originated from information retrieval

- Let's assume a database containing different objects
- Each object is described by a key vector

"Key"", "Value" et "Query"?



- Let's do a query :

query = "I want a blue unicorn"

- We transform this query into a vector

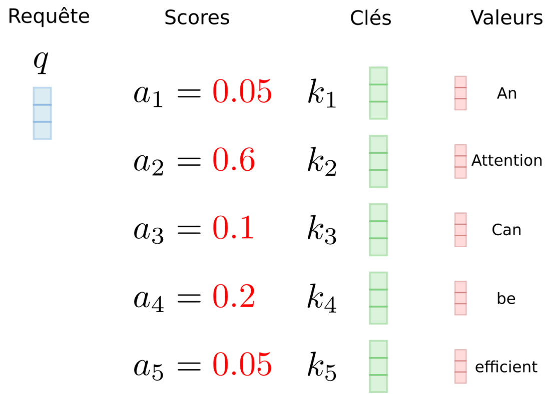
q = Embedding(query)

"Key"", "Value" et "Query"?

Query	Scores	Keys	Value
q	$q^T k_1 = 1.4$	k_1	
	$q^T k_2 = 1.1$	k_2	
	$q^T k_3 = 0.3$	k_3	
	$q^T k_4 = 0.6$	k_4	
	$q^T k_5 = 0.7$	k_5	

- Compute a score between the query and each key vector
- Here, the returned object will be the *1* (maximal score)

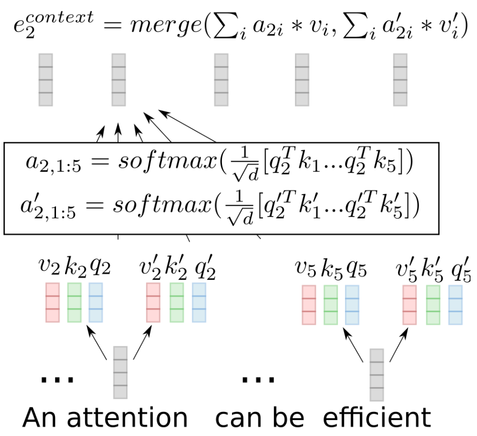
Auto-attention



$$[a_1, \dots, a_5] = softmax(\frac{1}{\sqrt{d}}[q^T k_1, \dots, q^T k_5])$$

- With Bert, values, queries and keys correspond to word tokens
- weighted sum rather than 1 best selection

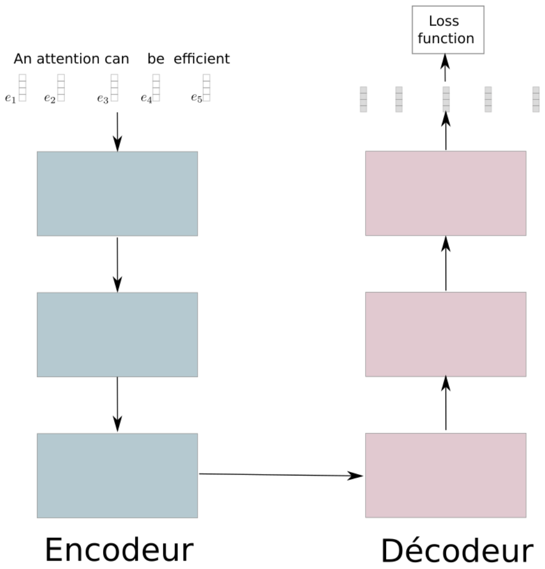
Multiple heads



- Heads : triplets of key/value/query (≈ 10)
- Implicit speicialisation of different concepts
  - subject/verb, punctuation,...
- Head fusion with a fully connected layer (or other)

Encoder-Decoder Architecture

Elmo, BERT (encoder), GPT (decoder)



Input encoding

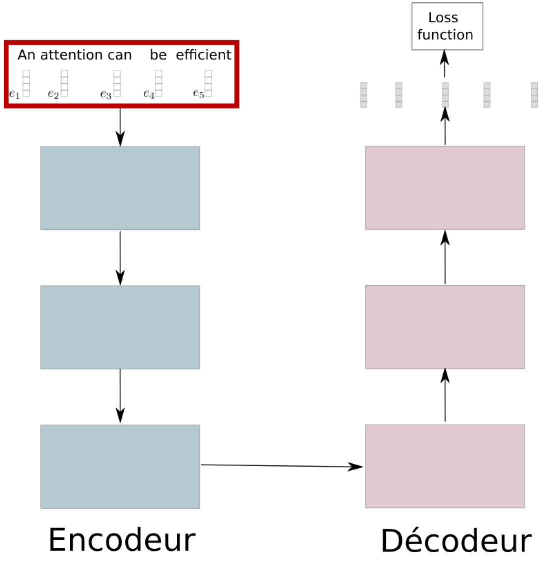
- Out of word vocabulary

Sub token division

undo => un - do

- Byte pair encoding (BPE)

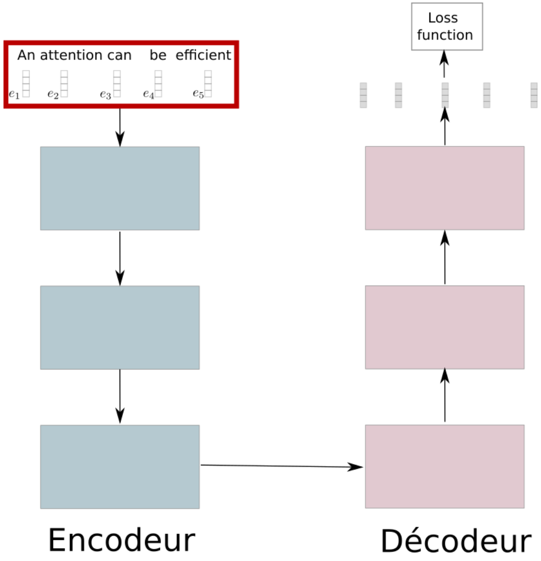
Detection of recurrent patterns



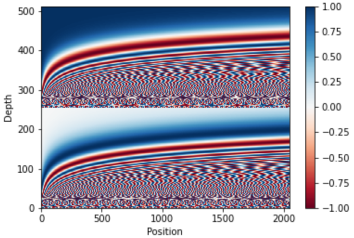
Position encoding

Adding a encoding p

- input = BPE + p
- p encode the position
- cosinus, sinus function



Position encoding



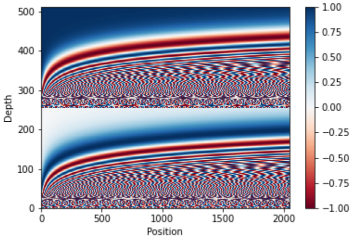
- Encoding of word position

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

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

with pos the position, i the dimension index, and d the embedding dimension

Position encoding

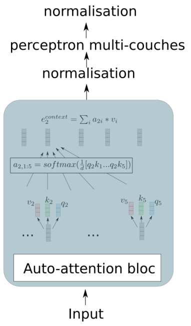


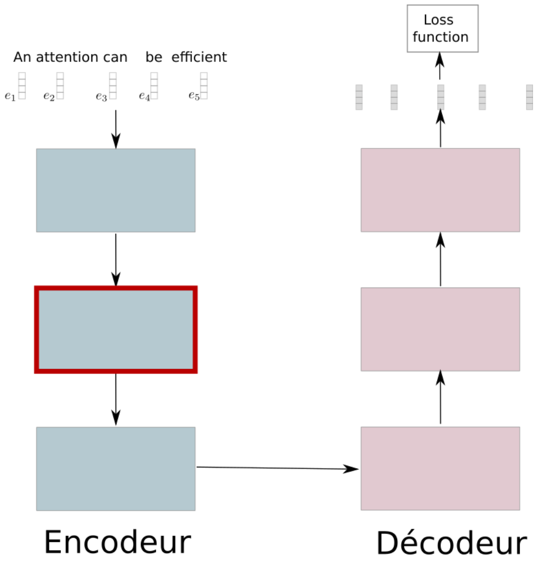
- Relative position can be computed with a linear function
  - in other words the model could compute distances between words

Encoding block

- In a block :

auto-attention layer

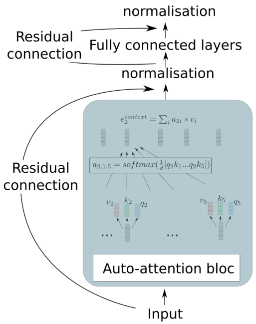


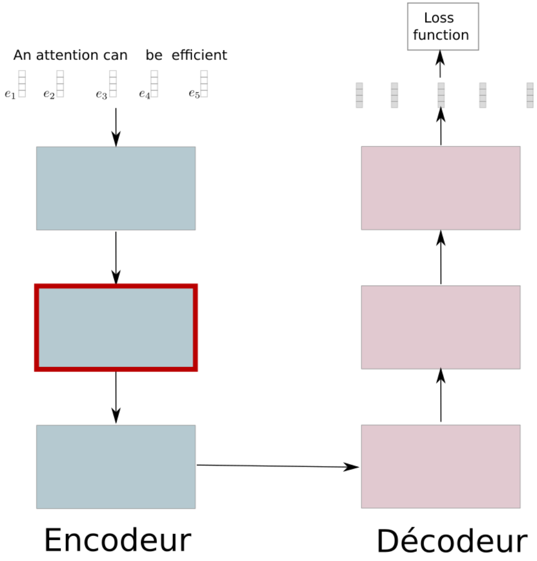


Encoding block

- In a block :

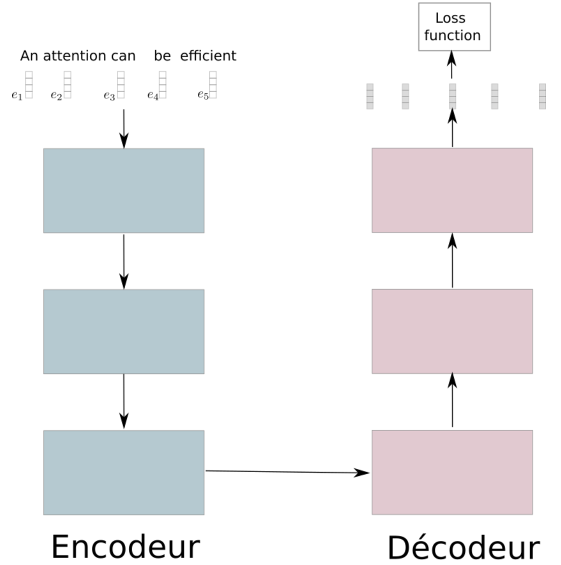
Residual connection





Encoding block

- Decoder similar to the encoder
- Only has access to previous tokens



Learning

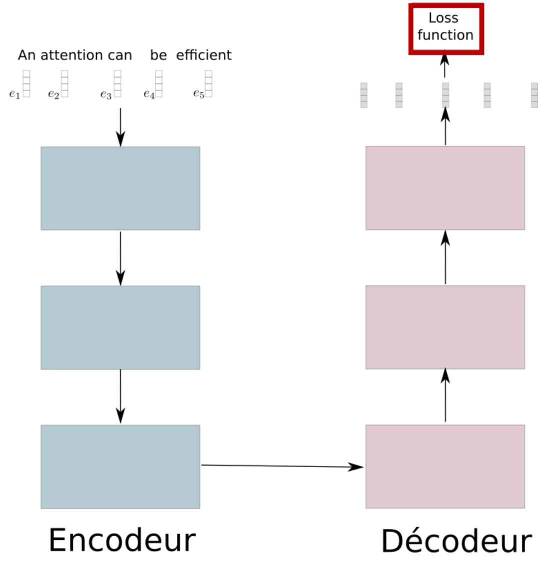
Auto supervision :

- Predict a mask

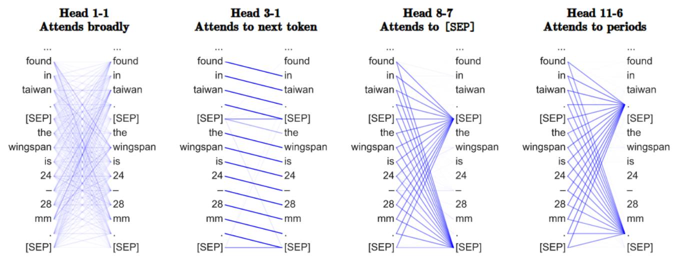
  An attention can be **????**

- next sentence prediction

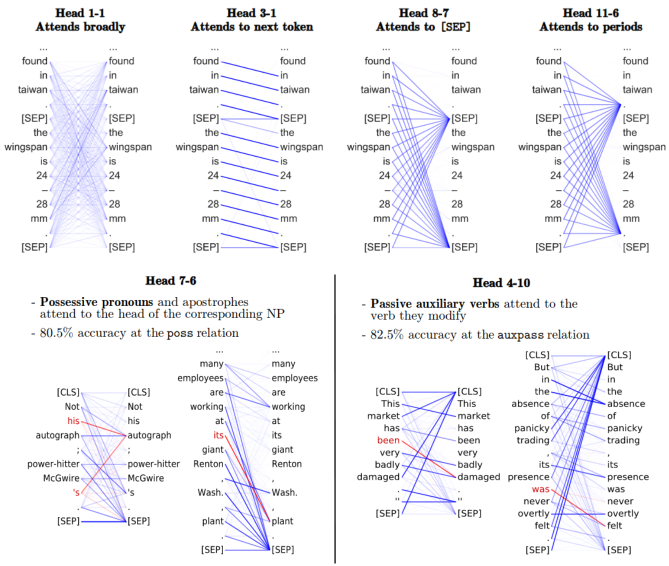
- and so on



Model analysis



Model analysis



Clark et al. What Does BERT Look At? An Analysis of BERT’s Attention. ACL 2019

A success

- Many application in NLP
    - Adding a simple classifier + Fine tuning
    - État de l'art en classification, Q&A, traduction et bien d'autres
- Integration in Google in 2019
- Text generation with GPT

https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3

Bertology

Many models have been proposed

- CamemBERT
- BART
- Electra
- DistillBERT
- **GPT** ...

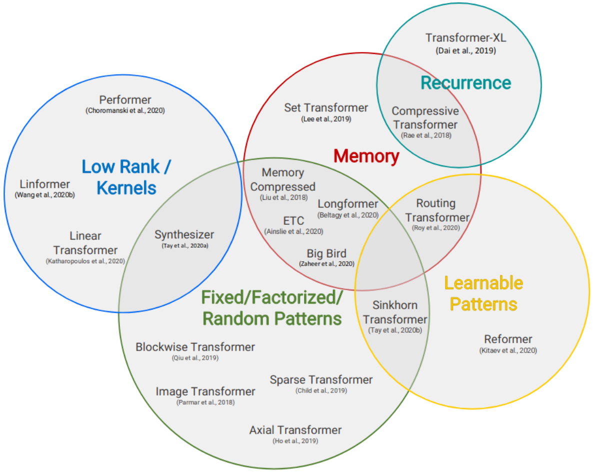
Qiu et al. Pre-trained Models for Natural Language Processing: A Survey.
arxiv:2003.08271, 2020

An other type of machine learning

Learning requires dozens even hundreds of gpus

	#Params (millions)	**# Taille du corpus**
Resnet	60	100M images
Bert	340	3000M tokens
GPT	115	800M tokens
GPT-2	1500	10B tokens
GPT-3	175000	300B tokens

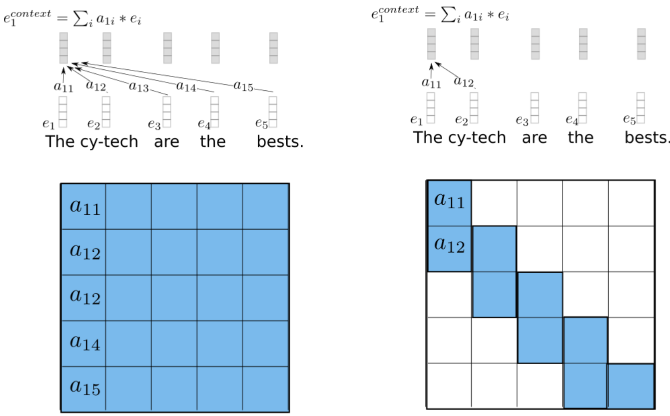
How can we make them lighter ?



Tay et al. Efficient Transformers: A Survey. arxiv:2009.06732 2020

Problem with attention

- Most works focus on reducing the quadratic complexity of attention.



Full attention　　Sparse attention

Limit the field of view of attention:

- Fixed or Learned patterns (Block wise, Strided,...)
- Use one token as a Global memory
- attend to random keys
- Or a combination of methods:

Zaheer et al. Big Bird: Transformers for Longer Sequences. NIPS 2020

Performers

Replace the attention score:

$$a_{i,1:L} = softmax(\frac{1}{\sqrt{d}}[q_i^T k_1, \ldots, q_i^T k_L])$$

Performers

Replace the attention score:

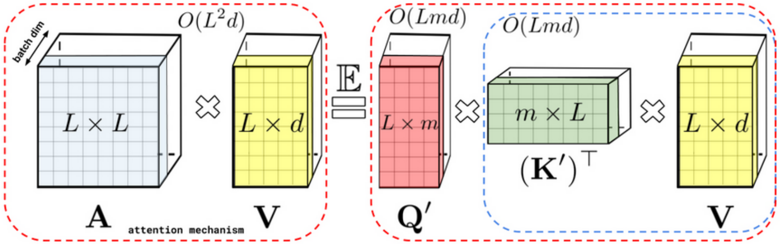
$a_{i,1:L} = softmax(\frac{1}{\sqrt{d}}[q_i^T k_1, \ldots, q_i^T k_L])$

... by using random features

$a_{i,j} = q_i'. k_j' = \phi(q_i). \phi(k_j)$

$$\phi(X) = \frac{c}{\sqrt{(M)}} f(WX + b)$$

Performers



This allows for linear Complexity with respect to sequence length.

Public benchmarks

Comparison of transformers along 5 tasks:

- LONG LISTOPS

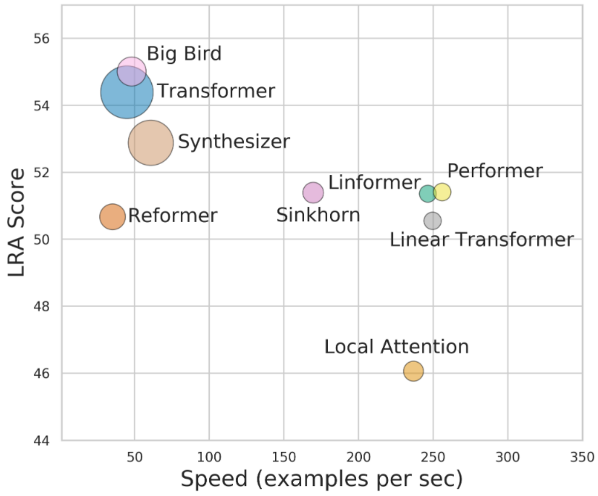
INPUT: [MAX 4 3 [MIN 2 3] 1 0 [MEDIAN 1 5 8 9, 2]] OUTPUT: 5

- TEXT CLASSIFICATION
- DOCUMENT RETRIEVAL
- IMAGE CLASSIFICATION
- PATHFINDER

Tay et al. LONG RANGE ARENA: A BENCHMARK FOR EFFICIENT TRANSFORMERS. arXiv:2011.04006. 2021

Benchmarks results

Comparison of transformers with LRA score : Integral over the 5 tasks



Ressources

- blog post on the mechanism of transformer

Blogs de Jay Alammar

http://vandergoten.ai/2018-09-18-attention-is-all-you-need/

- Implementations

https://blog.floydhub.com/the-transformer-in-pytorch/

https://huggingface.co

- The annotated transformer "Attention is all you need" with code

http://nlp.seas.harvard.edu/2018/04/03/attention.html

Thanks for your attention