Attention is all you need

NIPS 2017

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

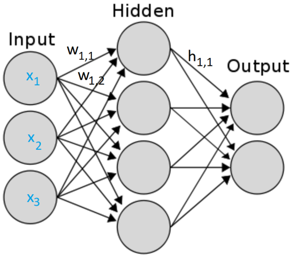
A Landmark Paper

- paper introducing the Transformer architecture
- method proposed for translation but transformer found plenty of applications (BERT, LLMs, Visual Transformers)
- 7th most cited scientific paper*

* Nature, 2025 (https://www.nature.com/articles/d41586-025-01125-9)

Context: Translating Natural Language

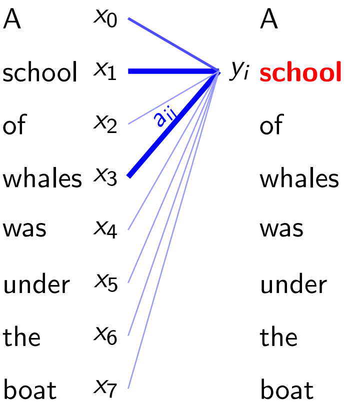
- Generative Task:
  - Input: a sequence of tokens to translate
  - Output: a sequence of token in another language
  - Task: predicting the next token based on the full input and the previous tokens of the output
- Machine Learning:
  - Everything is converted to vector in an embedding space
  - Embeddings are processed by a model with learnable parameters



Challenge: Taking into account the context

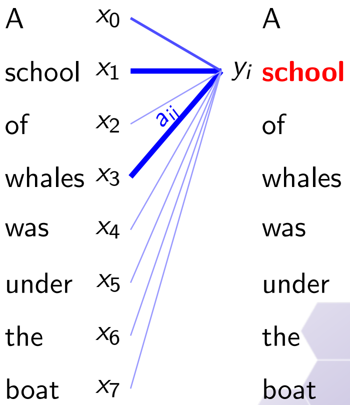
- Example: "A school of whales was under the boat"
- Traduction: Une [?] ...
- The meaning of a word is influenced by other words.
- And the next word depends on what was written before.

Enriching embeddings with attention



Enriching embeddings with attention

- 3 learnable functions: Query, Key and Value
- Attention coefficient from embedding j to embeddingĩ:
  $a_{ij} = \sigma(< Query(x_i).Key(x_j) >)$
- Output at position i:
  $y_i = \sum a_{ij} Value(x_j) = \sum \sigma(< Query(x_i).Key(x_j) >) Value(x_j)$



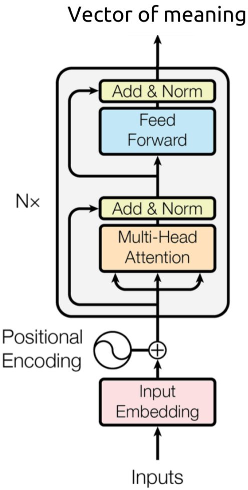
Multi-Head Scaled Dot-Product Attention layer

- Output at position i: $y_i = \sum \sigma(< x_i Q . x_j K >) x_j V$
- Computing all embeddings at once:

$$Y = F(X) = softmax(\frac{XQ(XK)^T}{\sqrt{d_k}})XV$$

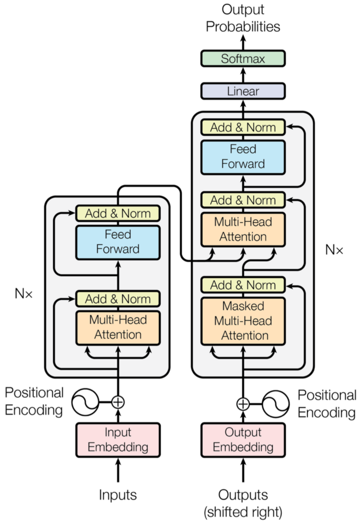
- Multi-Head Attention: $Y = (F_1(X) \oplus ... \oplus F_h(X))W$

The encoder



- **Input Embedding**: converting words to vectors
- **Positional Encoding**: modifying vectors according to the position:
  - $PE_{(pos,2i)} = sin(pos/100000^{2i/d_{model}})$
  - $PE_{(pos,2i+1)} = cos(pos/100000^{2i/d_{model}})$
- **Feed Forward**: linear layer on each embeddings individually

Adding the Decoder



- **Output**: the beginning of the translated sentence
- **Cross Attention**: Query is computed based on output tokens, Key and Value are computed based on input tokens

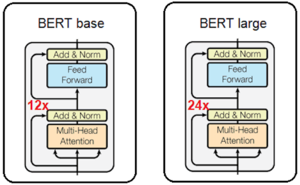
Main strengths

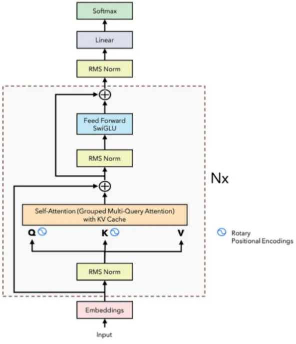
- Allows parallelisation
- Allows long-range dependencies (solves the vanishing gradient problem)

Impressive performances

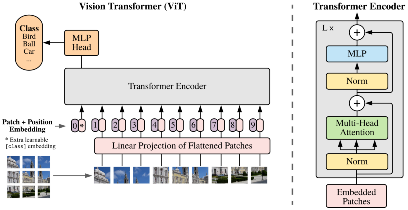
.	EN-GE score	EN-GE Tr.Cost	EN-FR score	EN-FR Tr. Cost
Previous best model	26.36	$7.7*10^{19}$	41.29	$1.2*10^{21}$
Transformer	28.4	$2.3*10^{19}$	41.8	$2.3*10^{19}$

Transformer Architecture in later models





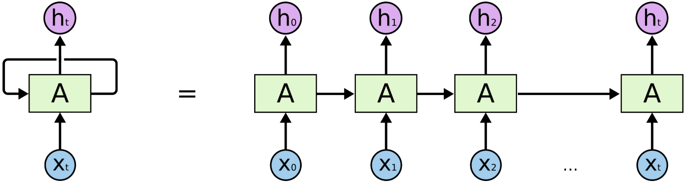
LLaMA



Source: [1]https://sushant-kumar.com/blog/bert, [2]Yumar Jamil youtube, [3]An Image is worth 16*16 words, ICLR 2021

Questions?

(Appendix) Recurrent models: the most common method up to then



Also: Long Short-Term Memory Networks, Gated Recurrent Neural Networks