inventeurs du monde numérique

Introduction to Transformers Neural architecture

Guillermo Andrade B.

/SED/

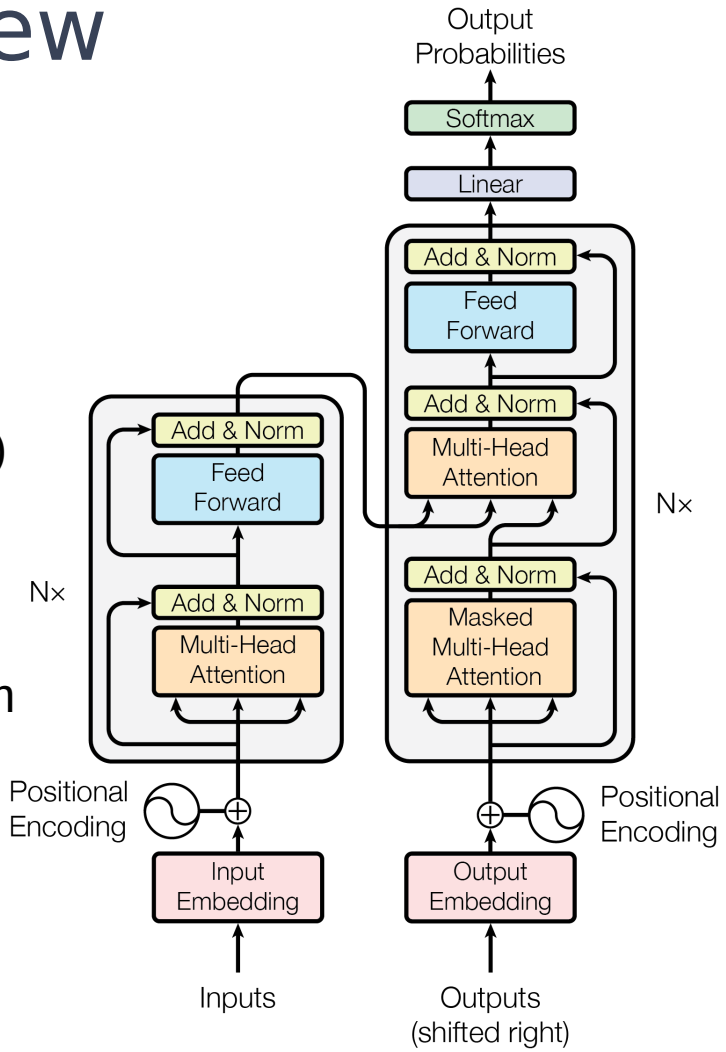
Sommaire

01

Transformers step by step

Transformers overview

- Original Article "*Attention Is All You Need* " - A. Vaswani & Co from Google Brain and Google Research in NIPS 2017 https://dl.acm.org/doi/10.5555/3295222.3295349
- A sequence to sequence processor
- Suited for natural language processing (NLP)
- Well designed for parallel architectures such GPUs
- Outperform classic Recurrent Neural Networks (RNN) models such long short-term memory (LSTM)
- Pre-trained models based on transformers
  - BERT by Google
  - GPT by OpenAI
  - ...



Understanding Transformer

- The Illustrated Transformer (animated graphical explanations) http://jalammar.github.io/illustrated-transformer/
- The Annotated Transformer (annotation in PyTorch notebook) http://nlp.seas.harvard.edu/2018/04/03/attention.html
- Transformer Neural Network Architecture (Description of many variants) https://devopedia.org/transformer-neural-network-architecture

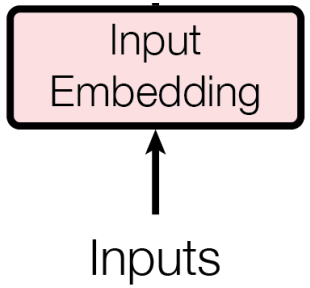
Understanding Transformer

http://jalammar.github.io/images/t/transformer_decoding_1.gif

http://jalammar.github.io/images/t/transformer_decoding_2.gif

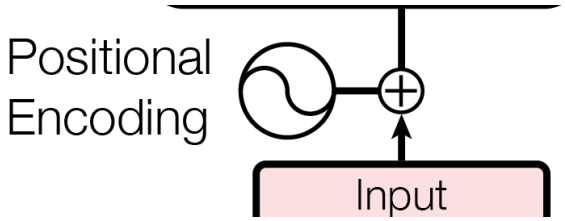
Input embedding
Text to vector

- Maps natural language words to numeric vectors
  - Typically 512 dimension vector
  - Catch context similarity
  - Reduce dimension
- Word2Vec
  - Unsupervised learning algorithm
    - Context | target extracted from text corpus automatically



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Positional Encoding



- Add a vector to embedding input
  - Typically 512 dimension vector
  - Unique vector for every word in sequence
    - PE (pos,2i) = sin(pos/10000 2i/d model)
    - PE (pos,2i+1) = cos(pos/10000 2i/d model)

4 dimensions toy example:

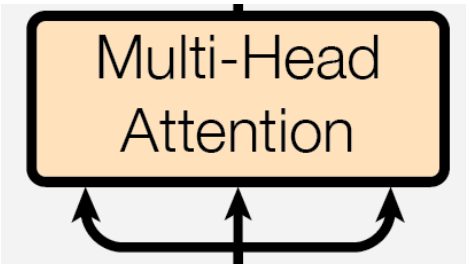
http://jalammar.github.io/illustrated-transformer/

Self-Attention (1/2)

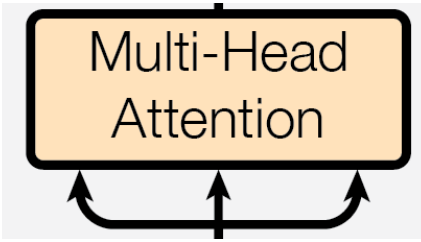
- Multiples operations in parallel (8 heads)
  - For every *i* head:
    - X (matrix of inputs 512 x n words in sentence)
    - Multiply by weight matrices W^Q, W^K, W^V
    - To produce Q,K,V
    - where n x d^k is dimension of K (d^k = 64)

$$Z_i = sofmax\left(\frac{QK^T}{\sqrt{d^k}}\right)V$$

  - Concat the 8 Z_i then multiply by a weight W_0
  - The final output Z as the same dimension that X



Self-Attention (2/2)



1) This is our input sentence*

2) We embed each word*

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

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

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

Thinking
Machines

X

W_0^Q W_0^K W_0^V

Q_0 K_0 V_0

Z_0

W^O

Z

W_1^Q W_1^K W_1^V

Q_1 K_1 V_1

Z_1

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

R

...

W_7^Q W_7^K W_7^V

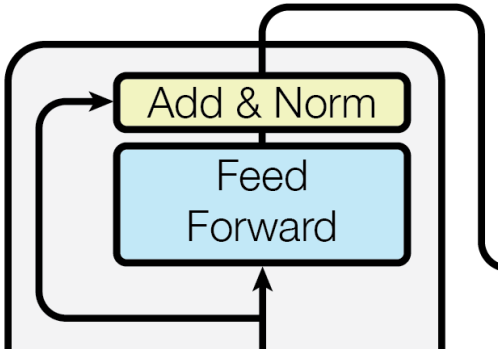
Q_7 K_7 V_7

Z_7

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

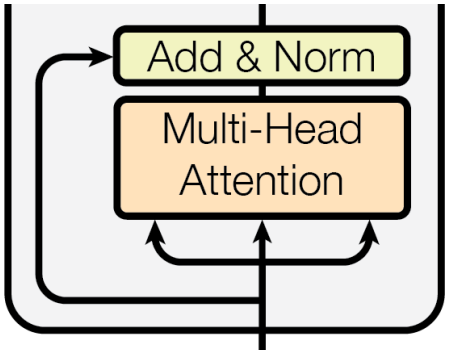
Feed Forward

- 2 layers
- full connected
- ReLU activation



Residual and layer normalization

- The output of Attention Z is added to X from previous stage them normalize the resulting output
  - Preserving positional information across the encoder
  - Help to stabilizes training process



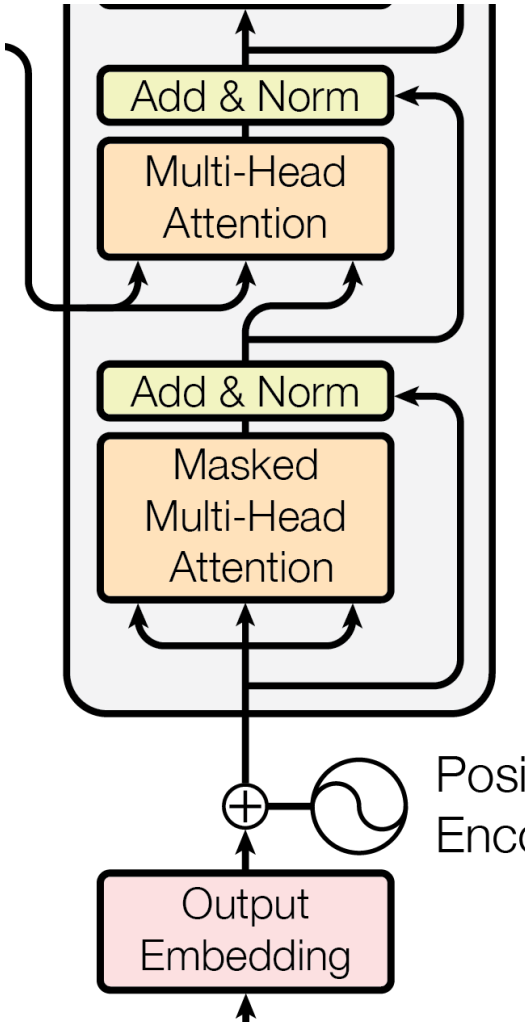
https://www.machinecurve.com/index.php/2020/12/28/introduction-to-transformers-in-machine-learning/

Decoder

- Take K and V from encoder output

- Mask just before SoftMax ensures that the predictions for position *i* can depend only on the known outputs at positions less than *I*

- At every step, list of predicted embedding words are passed at input

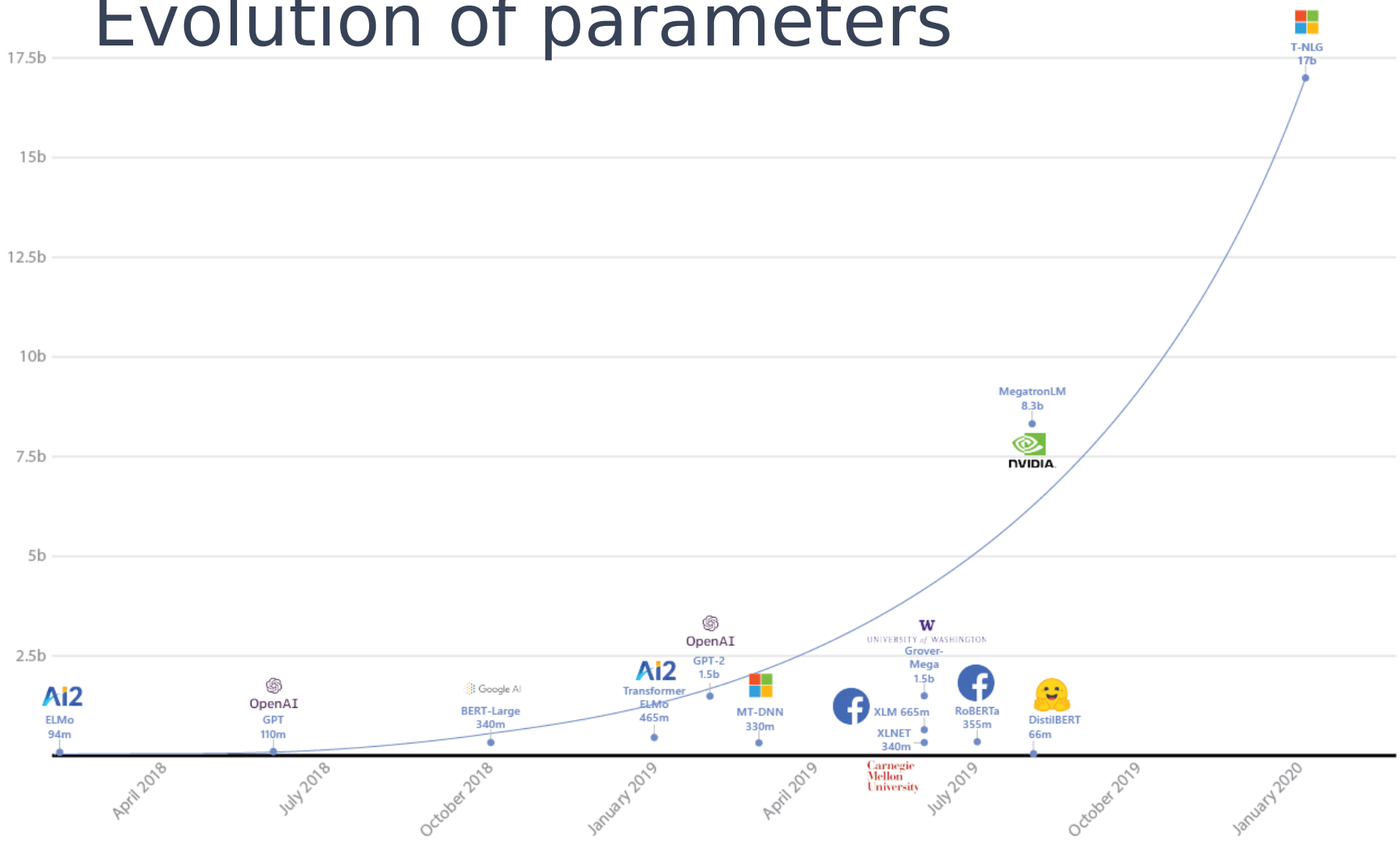


http://nlp.seas.harvard.edu/2018/04/03/attention.html#decoder

02

Material implementations

Evolution of parameters



https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

inventeurs du monde numérique

Megatron-LM from NVIDIA

- 8.3 billion parameter (Fit in 8 V100)
  - 32 attention heads
  - 96 hidden size/head
  - 72 layers
- 15.1 PetaFLOPs sustained
  - 32 DGX-2H servers
    - 512 Tesla V100 SXM3 32GB GPUs.
- 300 GB/sec bandwidth between GPUs inside a server
  - NVSwitch
- 100 GB/sec of interconnect bandwidth between servers
  - 8 InfiniBand adapters per server
- 2 days training

https://arxiv.org/pdf/2104.04473.pdf

Turing-NLG model from Microsoft

- 17 billion parameters
  - 78 Transformer layers
  - hidden size of 4256
  - 28 attention heads
- NVIDIA DGX-2
  - InfiniBand connections
  - 256 NVIDIA GPU
  - NVIDIA Megatron-LM framework

https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

Merci !

inventeurs du monde numérique