Deep Learning - MAI

Theory - Transformers

Dario Garcia Gasulla
dario.garcia@bsc.es

Disclaimer:

Many of the works this lesson is based on have not been thoroughly replicated yet. Conclusions and interpretations may be unreliable.

Context

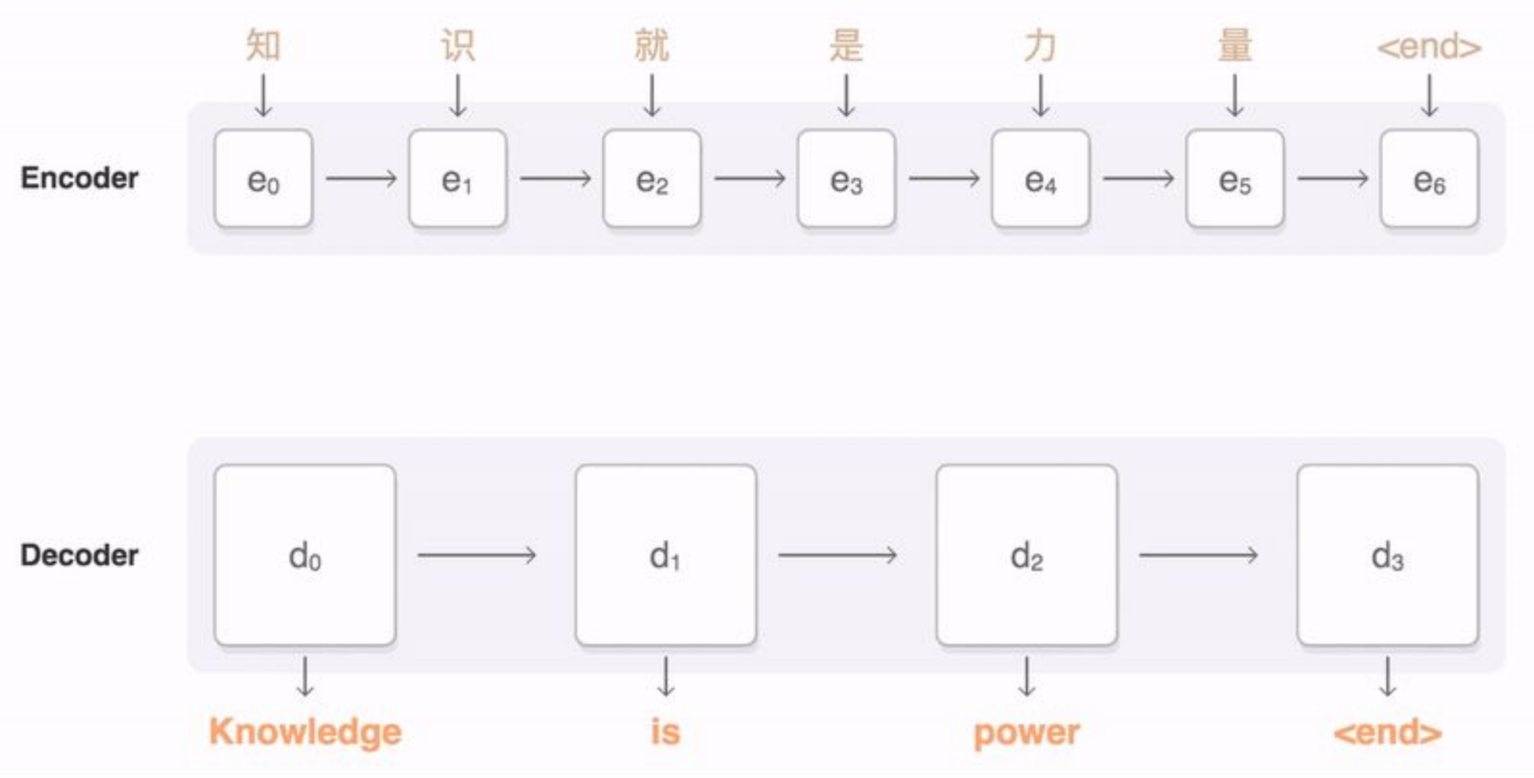
Dario Garcia Gasulla
dario.garcia@bsc.es

From Encoder-Decoder to Attention

- seq2seq limitations
  - Full sentence into a fixed-sized, unique embedding (bottleneck)
  - Different parts of the decoder focus on different parts of the input

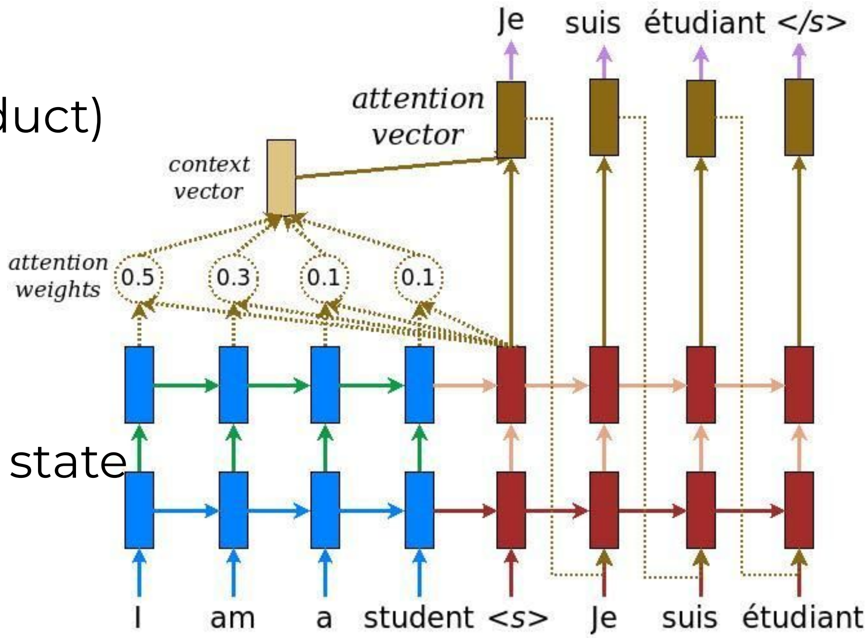
- Solution: Attention
  - Let each decoder step decide which part of the input use

Attention overview



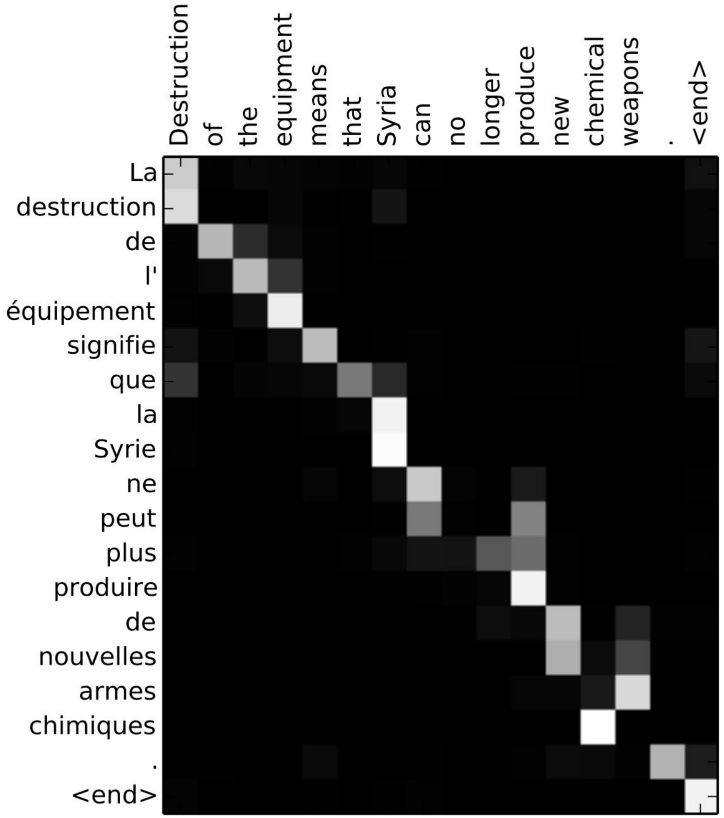
Seq2seq with attention

- All encoder hidden states as decoder input
- Decoder states…
  - Score prev. hidden states (dot product)
  - Weight by their own probabilities (multuply by softmax value)
  - Sum to make the **context vec**.
  - Concatenate with hidden decoder state
  - Output and fed to next step



Why seq2seq with attention

- Enables one different context for each decoding step
  - No fix-sized bottleneck
- Provides shortcuts (better gradient flows)
- More fine-grained -> better interpretability



Attention to Transformers

Dario Garcia Gasulla
dario.garcia@bsc.es

The limits of RNNs

- The main challenges of RNNs
  - Distances (long, short or both?)
  - Directionality (data accessibility)
  - Lack of focus specificity (all look the same)
  - Poor parallelization
- How can we solve that?
  - As long as we work with **sequences**, is tough
    - Memory is hard to implement
    - Computational dependencies by sequential design

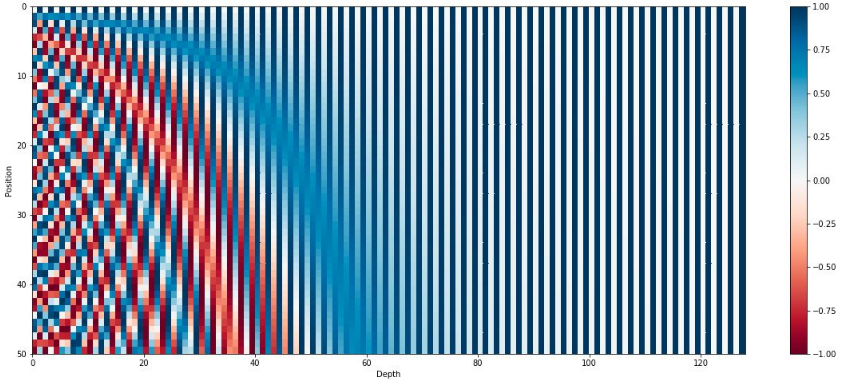
The Attention revolution

- What if we *get rid of the sequence*? What if attention is enough?
  - No more sequences, no more memory, no more dependencies
  - Meet the Transformers
- Closer to *fully connected* than RNNs.
- All tokens processed concurrently (instead of recurrently)
  - Inputs are *sets* instead of sequences
  - Self-attention for focus

Transformers and ~~Order~~ Position

- We need to keep some notion of position
  - Add order information on the input token embedding
  - Token representation changes with position
- *Positional encoding* through Sinusoidal functions
  - Add the position vector to each embedding
  - Provides consistent distances in the embedding space
    - Regardless of sequence length
    - Bounded range of values
    - Deterministic

50 positions
128 embedding
[-1,1] range



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

[54,55]

How basic attention works

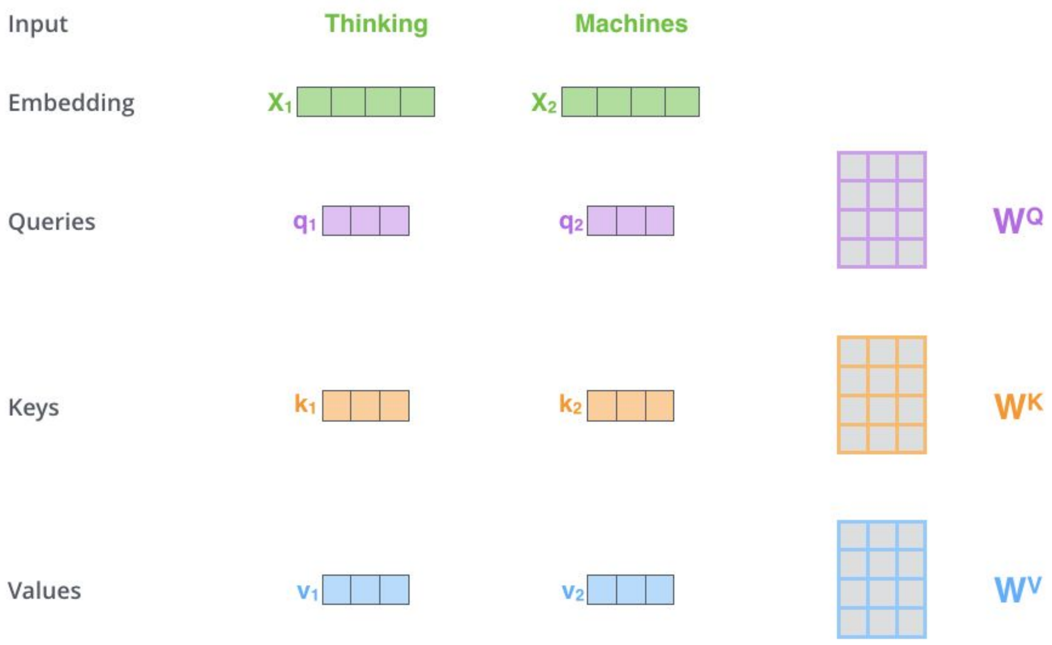
- Pseudo-limited connectivity (learnt sparsity, dense computation)
- Every input token has its own embedding
- All tokens stacked (e.g., word embeddings) are the input
- Length of token is arbitrary (e.g., 512)
- Number of tokens defined by dataset

Why attention works

- What should be computed together with input X?
- Learn and use a ‘mask’
  - **Query** for what you want to match (current token)
  - **Keys** to match the query with (other tokens)
  - **Value** to be returned (relevance between both)
- Let’s do it weightedly, through matrix multiply
  - No dependencies. Parallelism!

Basic attention

- Three weight matrices (**Q**,**K**,**V**) learnt
- Dot product from input embedding of token *X* and **Q**,**K**,**V** matrices
  - **Q**,**K**,**V** vectors for token *X*
  - Typically smaller dimensionality than token



Basic attention

- Attention of token *X* on token *Y* (all with all):
  - Dot product between **Q** vector of *X* and **K** vector of *Y*
  - Stabilize gradients (square root of vector length)
  - Normalize (apply softmax)
  - Multiply by **V** vector of *Y* (weighting Y by relevance of Y w.r.t X)
  - Sum over all *Y* -> output for *X*

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

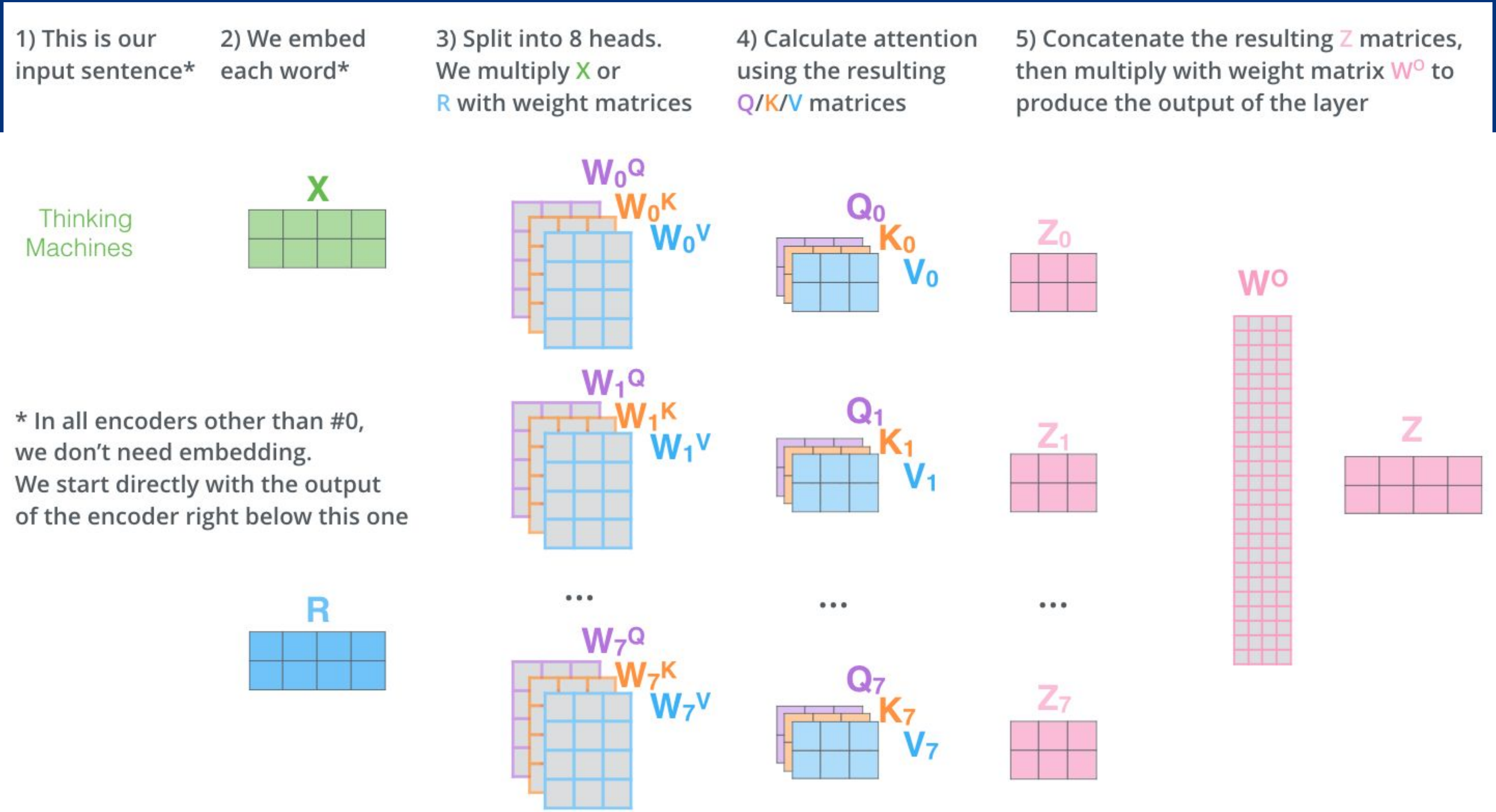
Multiple Embedding Spaces

- Multi-headed attention
- Learn different sets of Q,K,V matrices
- Each provides a different view on the data (enforceable on att. weights)
- On output
  - Concat all output embeddings in feature dim.
  - Multiply by another learnt matrix to fit dim.
- Attention heads can be computed in parallel

Computing in Parallel

- Attention relates inputs at arbitrary distance within **constant** num. ops
  - Close or far away, it’s the same
  - Fully-connected style
- ByteNet does so within a **logarithmic** num. ops (dilated convolutions)
- Convs s2s does so within a **linear** num. ops
- Retaining memory is more complicated as this grows

[59,60]

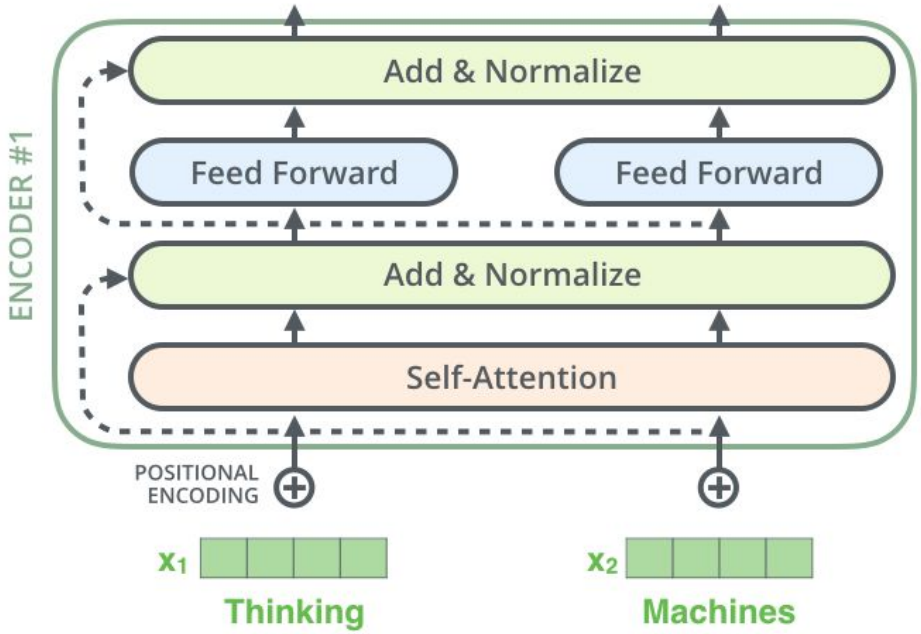


1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

[57]

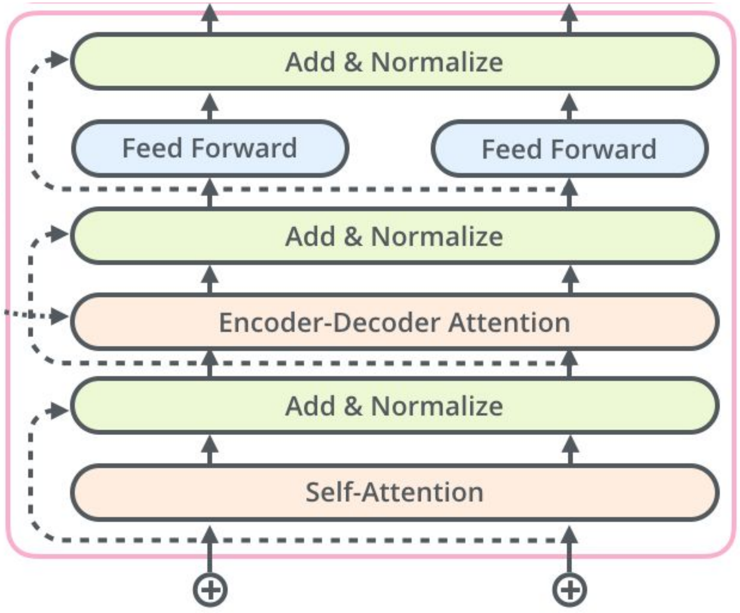
The Encoder block

- Self-Attention + Feed Forward
  - Each token follows its own path
- Both with
  - Residual connection
    - To self-attend or not
  - Layer normalization
    - Sample-wise layer-wide mean and var.
- Stack several of these blocks



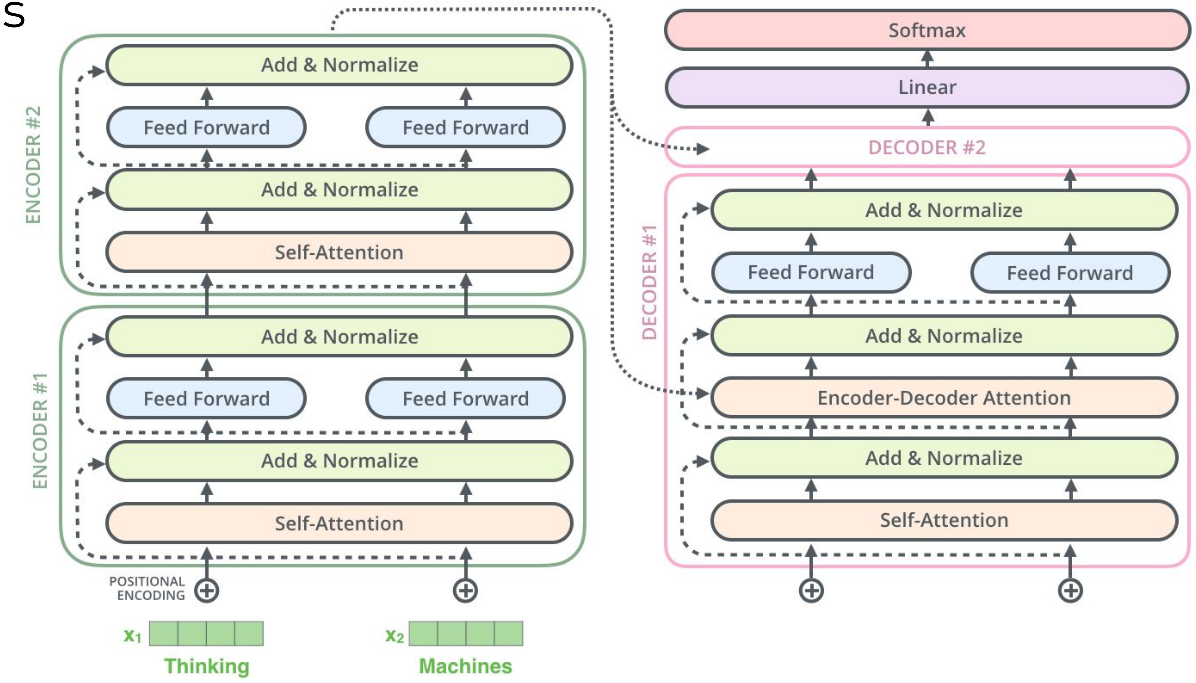
The Decoder block

- Same components as encoder
  - Self-Attention in the past only (mask out previous tokens)
  - Encoder-Decoder attention (**K** & **V** from encoder, **Q** from prev dec.)
  - Feed Forward, Residual & Norm
- Input: Special token, then previous token (also with pos. encoding)



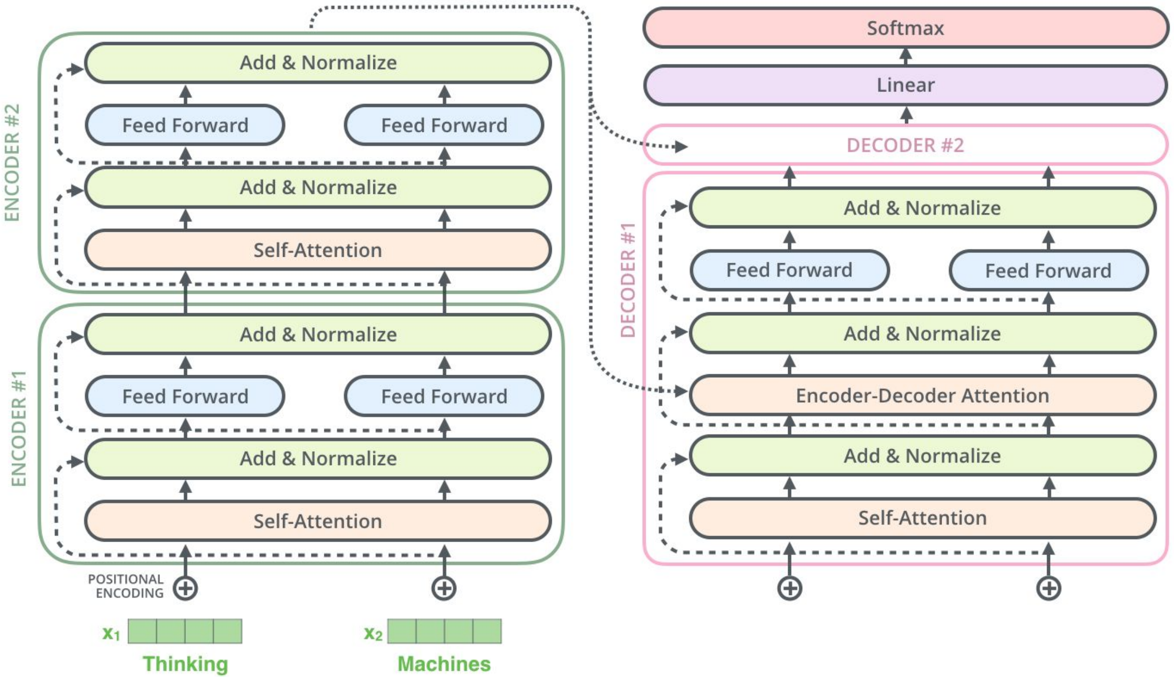
From input to output

- Linear layer for logits (dictionary length!)
- Softmax for probabilities



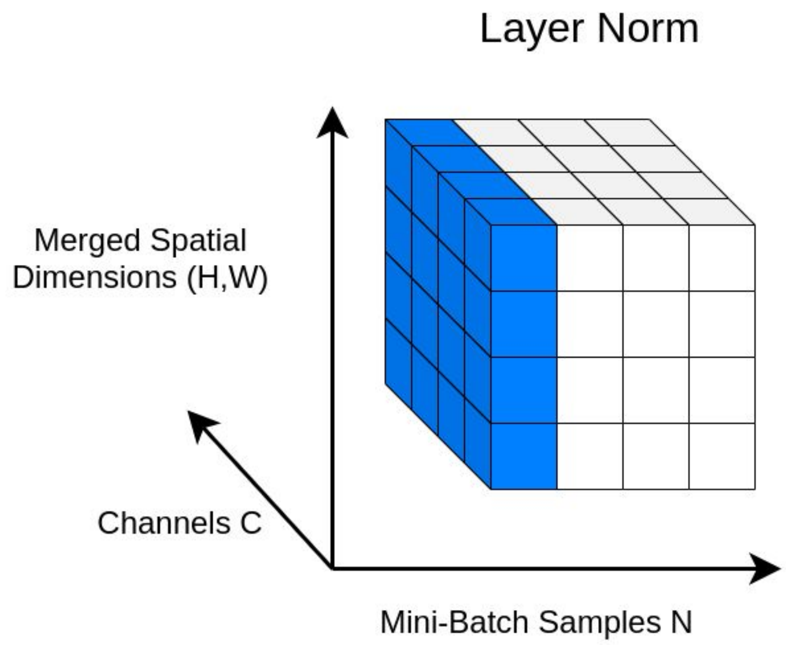
General Transformers

- Without positional encoding, a transformer is a fully connected NN with focus



Layer Normalization

- Normalize sample-wise
- Compute mean and std-dev across spatial dimensions (1 for sequences) and channels



Loss & Training

- A transformer outputs a vector of probabilities a number of times (?)
  - Cross entropy loss against golden probabilities*
- Batch training requires padding
- As with RNNs, decoders use
  - Greedy search (explore one path only)
  - Beam search (explore *n* branches on each step)

Transformer details

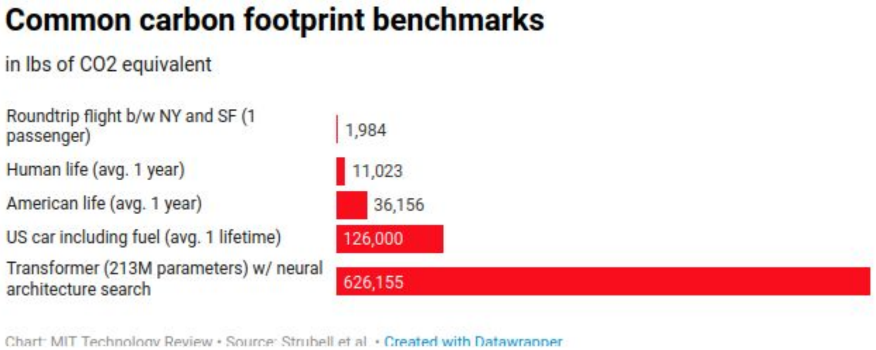
- In the original paper
  - Adam optimizer. Warm-up round and then decay
  - Dropout on residual connections, embeddings sums and pos. enc.
  - Label smoothing

Limitations of Transformers

- Reduced resolution (averaging attention)
  - Multi-head to circumvent
- Sequence length
  - All tokens must be computed concurrently (for context)
- Computational cost / Complexity
  - All relations are learnt (quadratic self-attention complexity). No limited connectivity by design.

A serious issue

- Transformers are efficient, but costly
  - Worthy trade-off?
  - Measuring efficiency
- Interpretability (too many heads)
- Google ethical crisis (Gebru, Bengio, ...)
  - Stochastic parrots
- Interpretability (too many heads) & Bias (too many data)



[63,64,73,74,75]

Fancy Transformers

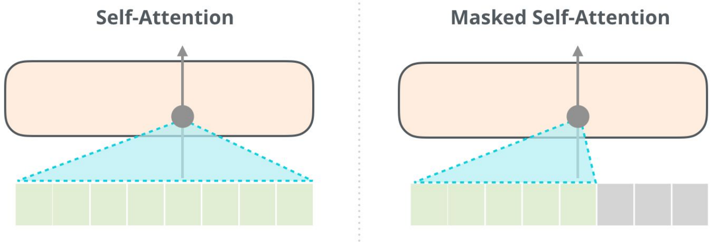
Beyond Encoder-Decoder

- Encoder-Decoder was inherited from RNN times
- Transformers (aka self-attention) is beyond that
- What works:
  - Pre-train heavy (as in Google-level, Millions of $)
  - Fine-tune for everything
- The story goes: GPT - BERT - GPT2 - GPT3 - ....
- Tell me how do you pre-train and...

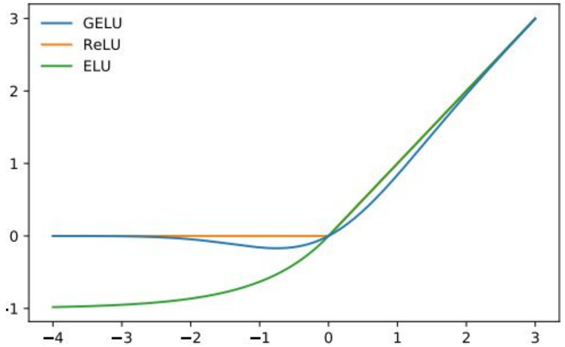
[68,69,76]

The two (main) sides

- Encoder only (BERT)
  - Bidirectional Transformer
  - Gain context (classification↑)
- Decoder only (GPT family)
  - Left to Right Transformer
  - Gain auto-regression (generation↑)



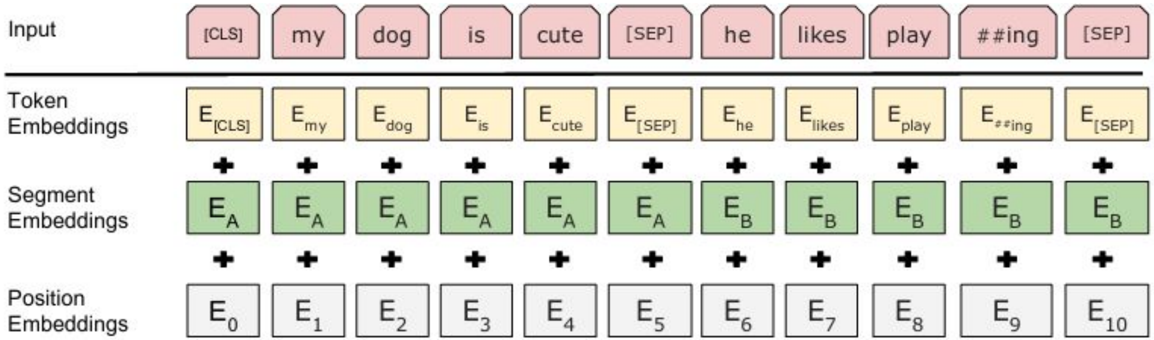
- GELU instead of ReLU
  - Gaussian Error Linear Unit



[68,69,76]

Famous Transformers: BERT

- For text generation: Encoder only
- Special token to separate sentences, and embedded id (+pos. enc.)
- Train two tasks concurrently
  - Masked LM: Mask 15% of tokens, and try to predict them
  - NSP (Sentence prediction): Is the follow up sentence correct?



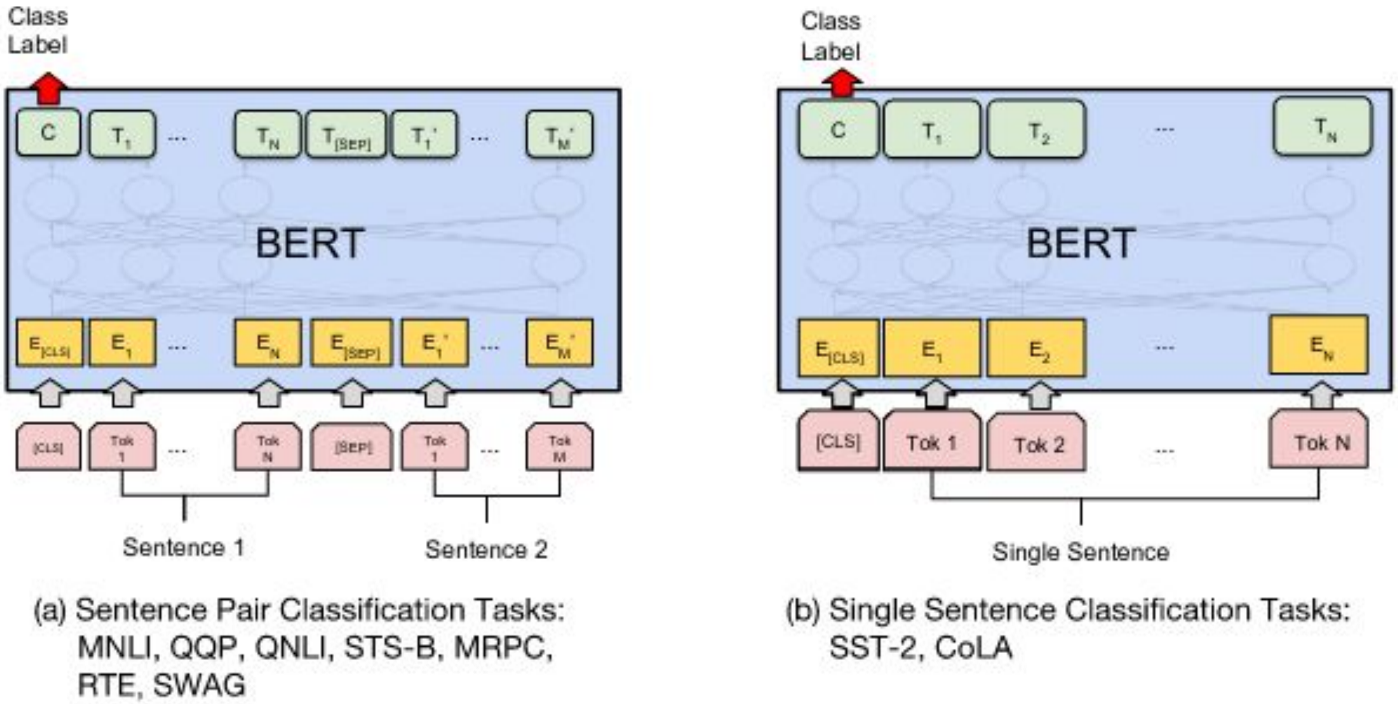
[65,66]

Famous Transformers: BERT

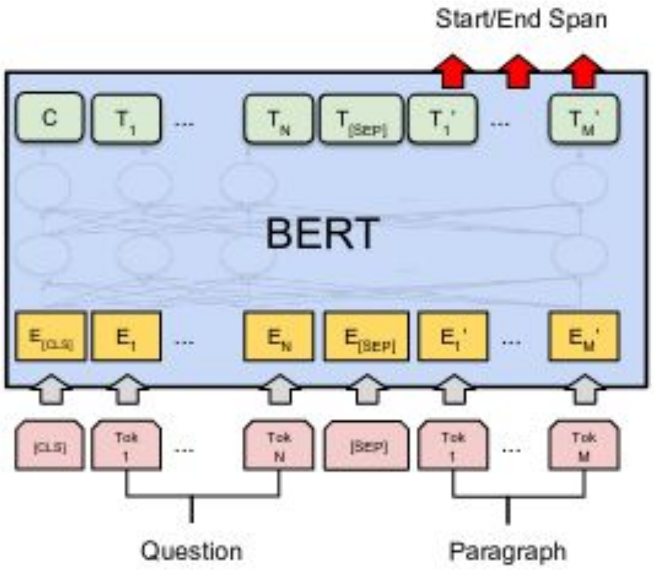
- Pre-train (bulk text) + fine-tuning (paraphrasing, QA, classification, ...)
- BERT-base:
  - 6 blocks, 12 attention heads, 110M params (4 TPUs 4 days)
- BERT-large
  - 12 blocks, 16 attention heads, 340M params (16 TPUs 4 days)
- Fine-tuning: 1 TPU 1 hour

[65,66]

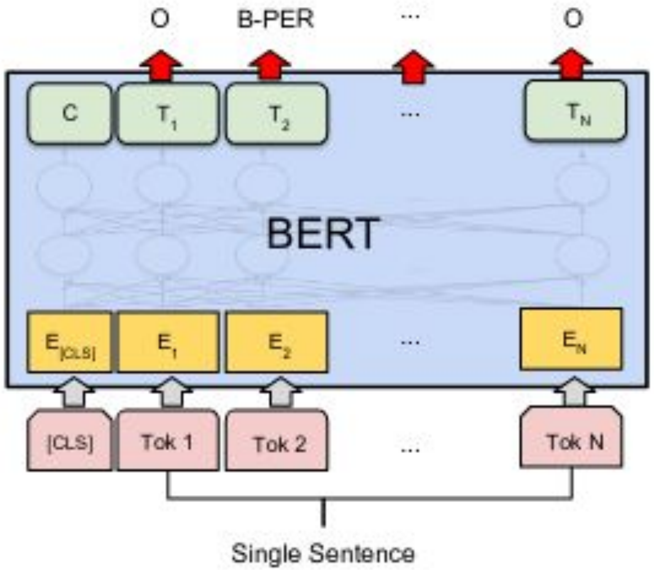
Famous Transformers: BERT



Famous Transformers: BERT



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

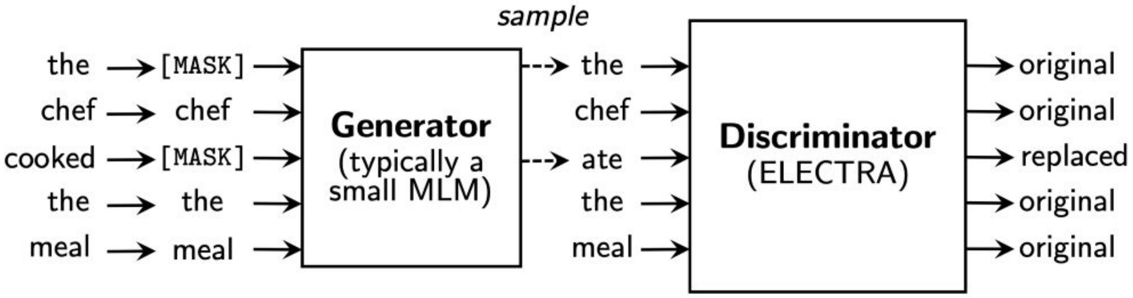
Famous Transformers: GPT

- GPT
  - Pretrain + fine-tune (117 M params)
- GPT2
  - More data, 48 blocks, zero-shot task/transfer (1,500 M params)
  - 1024 tokens
- GPT3 (& DALL-E)
  - More data, 96 blocks, 96 heads, (175 B params)
  - 2048 tokens

[67,78]

Pre-training Transformers like GANs

- Masked Language Model (BERT)
  - Limited token efficiency
  - Differences between train/test
- Electra
  - Generator / Discriminator scheme (keep the later)
  - Validate each token
  - Full token efficiency
  - Faster (12x)



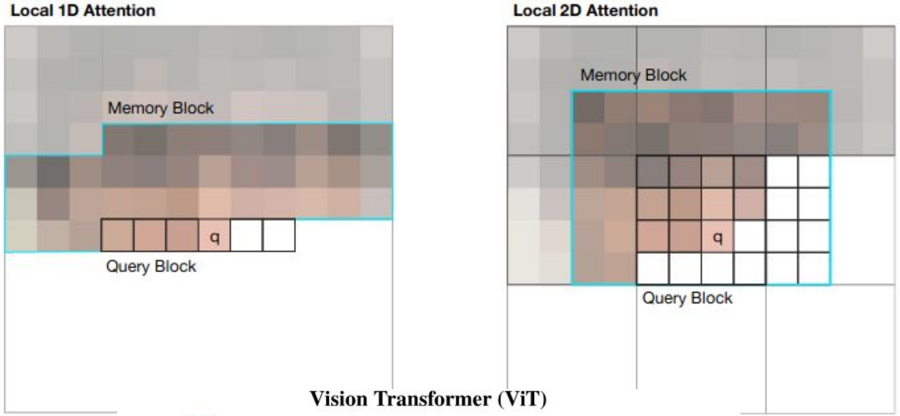
Vision Transformers (ViTs)

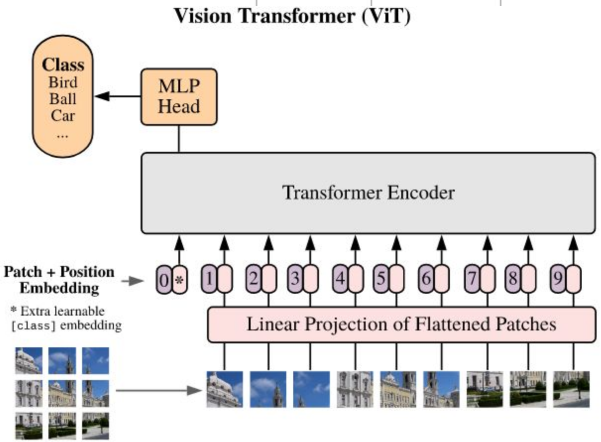
- Lack inductive biases implicit in CNNs
  - Translation invariance (weight sharing)
  - Locality (limited connectivity)
- **These can be learnt from enough data** (14M - 300M samples)
  - Mitigable by knowledge distillation - soft labels - noisy student (?)
- Each pixel attending to each other pixel is unfeasible
  - Several local self-attention mechanisms are being proposed

[70,72]

Vision Transformers (ViTs)

- Doing CNNs with Transformers
  - Self-attention limited spatially
  - Images flattened to 1D
  - Positional encodings
  - Attention bottlenecks
  - Autoencoders





[70,71,77,78]

So what are Transformers?

- Great models for processing data which can be represented as a set of independent numerical features
  - More powerful and smarter version of FFN nets
  - If computation and data availability allows!
- Capable of including location info through Positional Encodings
- Can be good for sequences (the shorter the better). Not for streams, recursion and hierarchies.
- The biggest hammer out there right now

[78,81,82]

References

References

[53] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).

[54] https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

[55] https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/

[56] https://theaisummer.com/transformer/

[57] http://jalammar.github.io/illustrated-transformer/

[58] https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

References

[59] http://nlp.seas.harvard.edu/2018/04/03/attention.html
[60] Kalchbrenner, Nal, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. "Neural machine translation in linear time." arXiv preprint arXiv:1610.10099 (2016).
[61] http://vandergoten.ai/2018-09-18-attention-is-all-you-need/
[62] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.
[63] Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. arXiv preprint arXiv:1906.02243.
[64]
https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

References

[65] https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c
[66] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
[67] https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2
[68] https://medium.com/@shoray.goel/gelu-gaussian-error-linear-unit-4ec59fb2e47c
[69] http://jalammar.github.io/illustrated-gpt2/
[70] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

References

[71] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., & Tran, D. (2018, July). Image transformer. In International Conference on Machine Learning (pp. 4055-4064). PMLR.
[72] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2020). Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877.
[73] Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?🦜. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (pp. 610-623).
[74] Dong, Y., Cordonnier, J. B., & Loukas, A. (2021). Attention is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth. arXiv preprint arXiv:2103.03404.

References

[75] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. Journal of Machine Learning Research 21, 248(2020), 1–43. http://jmlr.org/papers/v21/20-312.html
[76] https://hannes-stark.com/assets/transformer_survey.pdf
[77] Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., & Carreira, J. (2021). Perceiver: General Perception with Iterative Attention. arXiv preprint arXiv:2103.03206.
[78] https://www.youtube.com/watch?v=PtdpWC7Sr98
[79] https://medium.com/dair-ai/bert-is-extremely-inefficient-this-is-how-to-solve-it-688b09350f10
[80] Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555.

References

[81] Hahn, M. Theoretical limitations of self-attention in neural sequence models.Trans. Assoc.Comput. Linguistics, 8:156–171, 2020.URL https://transacl.org/ojs/index.php/tacl/article/view/1815.

[82] Tran, K. M., Bisazza, A., and Monz, C. The importance of being recurrent for modeling hierarchical structure. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pp. 4731–4736. URL https://www.aclweb.org/anthology/D18-1503/.

Dario Garcia-Gasulla (BSC)

dario.garcia@bsc.es

HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE