Attention Is All You Need

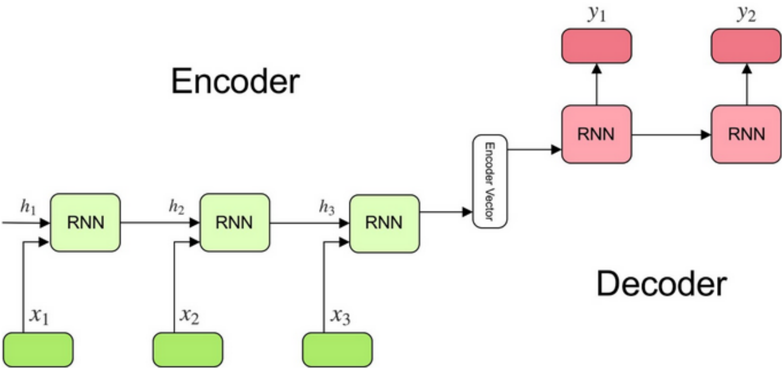
A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin

Google Brain, Google Research, University of Toronto

NIPS 2017

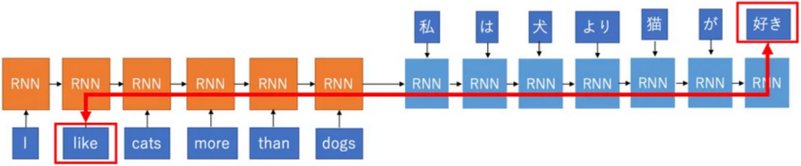
ML Reading Group on June 16, 2020

Sequence-to-Sequence models



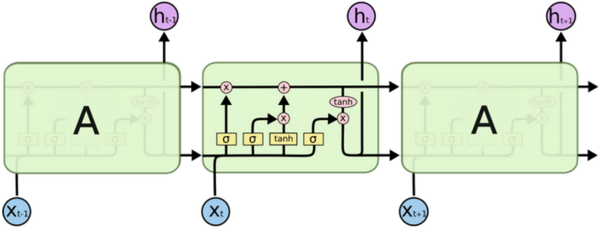
- most popular models
- language translation

RNNs



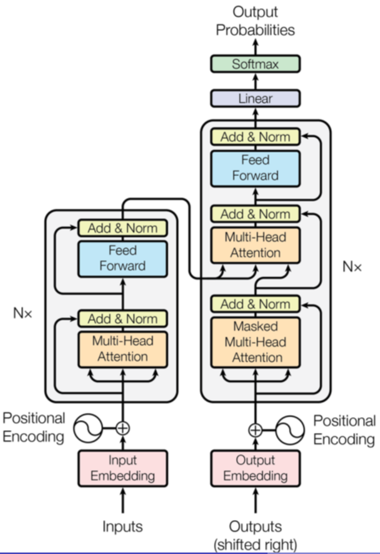
- feed-forward NNs rolled out through time (back-propagation through time)
- slow to train
- cannot deal with long sequences: vanishing/exploding gradients

LSTMs



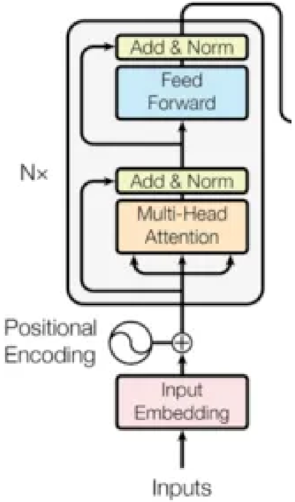
- Hochreiter and Schmidhuber (1997)
- path to allow past information to skip processing and move to next
- forget, input, output gating: better coping with longer sequences
- slower to train than RNNs: data passed sequentially, no parallelization

The Transformer model



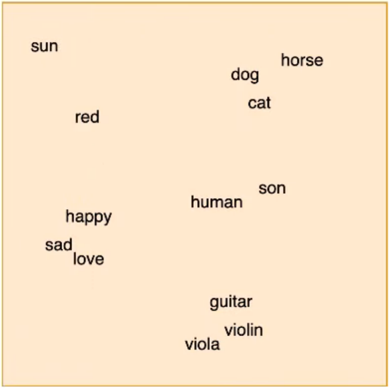
- same encoder-decoder model as RNNs
- the input sequence passed in parallel
- no concept of time-step (like RNNs that construct hidden state)

The Encoder



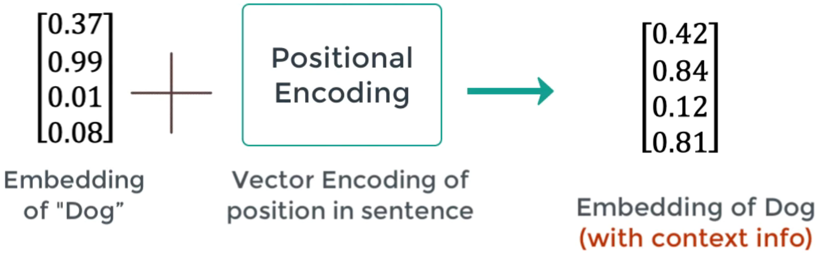
- Multi-Head Attention block
- Feed-Forward layer
- residual connection: learn the difference, supress vanishing gradient problem
- layer normalization

Problem with the embedding space



- words $\rightarrow$ vectors, closer to each other for similar words
- words in different sentences have different meanings
- positional encoder breaks the equality of different embeddings

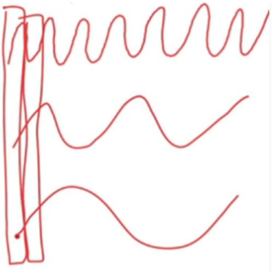
Positional Encoder



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

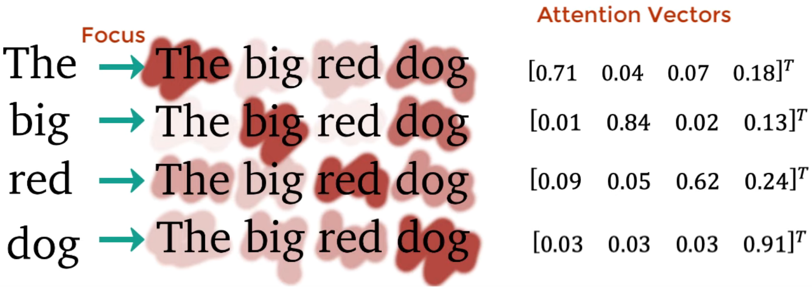
- vector containing context based of position of word in sentence

Positional Encoder functions



- some kind of continuous 'binary' encoding of position
- comparing positional encodings gives the distance between words
- wavelenghts of geometric progression between 2π and $10000 \cdot 2\pi$

Attention



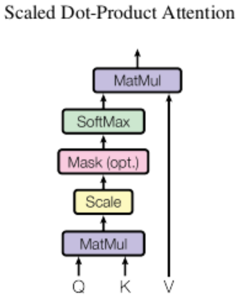
- (self) attention
- for every word, it computes the contextual relationships with other words in sentence

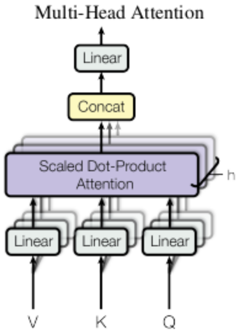
How Attention is computed: Scaled Dot-Product Attention

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d^k}}\right)V$$

- dot-product determines cosine similarity between Key and Query
- as dot product tends to grow with dimensionality of the query and key, so scaling prevents explosion to huge values
- softmax determines a more like a 'one-hot' encoding, and basically a component from V gets selected
- some kind of indexing scheme

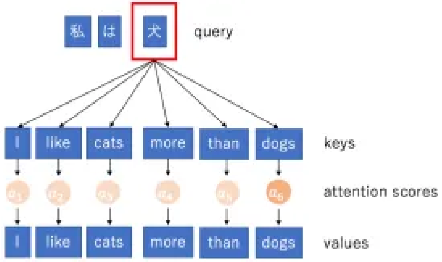
Multi-Head Attention





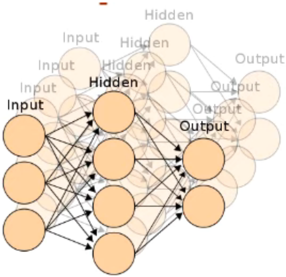
- computes the relevance of a set of Values based on Keys and Queries
- attention is used as a way for the model to focus on relevant information
- computes multiple attention vectors for every word (multi-head)

Multi-Head Attention (2)



- query is the word being decoded
- keys and values are the source sentence
- attention score represents the relevance (large for word 'dog')

Feed-forward NN



- looks for features into attention vectors
- non-linear mapping between input and output space, with ReLU in between:

  $FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$

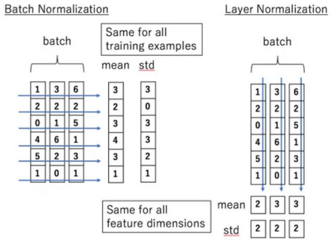
Layer Normalization

Batch normalization

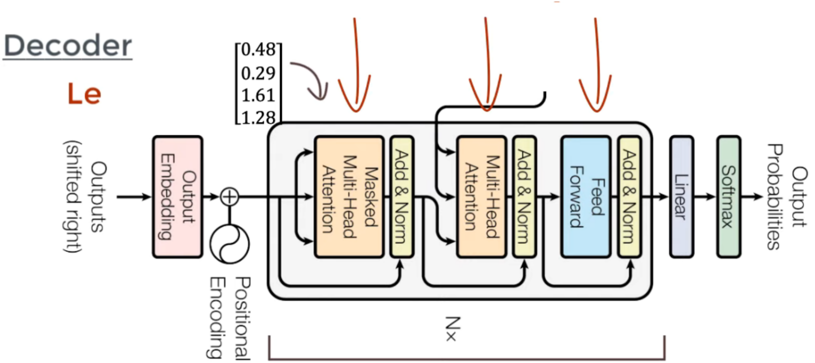
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_{ij} - \mu_j)^2$$
$$\hat{x_{ij}} = \frac{x_{ij}-\mu_j}{\sqrt{\sigma_j^2+\epsilon}}$$

Layer normalization:

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$$
$$\sigma_i^2 = \frac{1}{m}\sum_{j=1}^{m}(x_{ij} - \mu_i)^2$$
$$\hat{x_{ij}} = \frac{x_{ij}-\mu_i}{\sqrt{\sigma_i^2+\epsilon}}$$

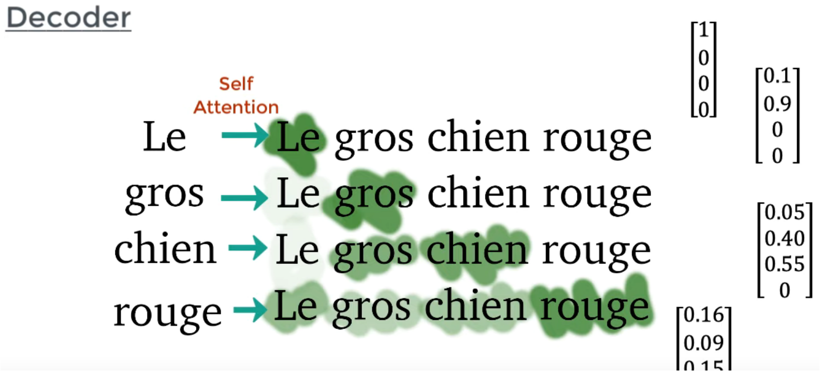


- BN: statistics computed across the batch, the same in batch
- LN: computed across features, independent of other samples
- smoothens the landscape loss; arbitrary batchsize



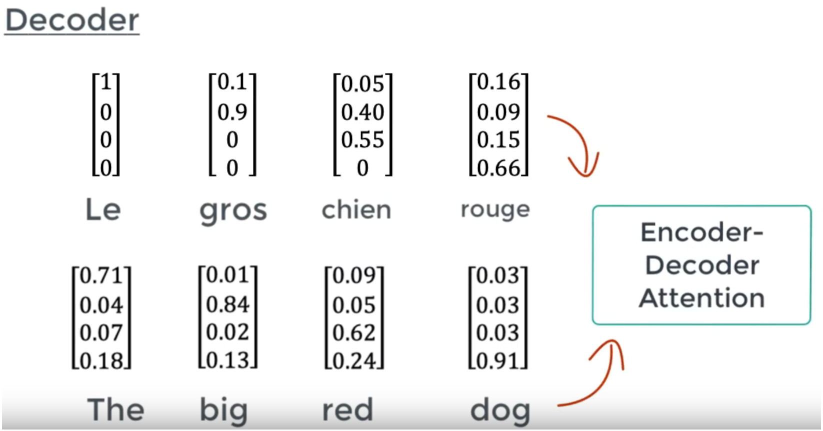
- positional encoding in the second language is performed
- three components similar to the encoder block

Block 1: Masked Multi-Head Attention

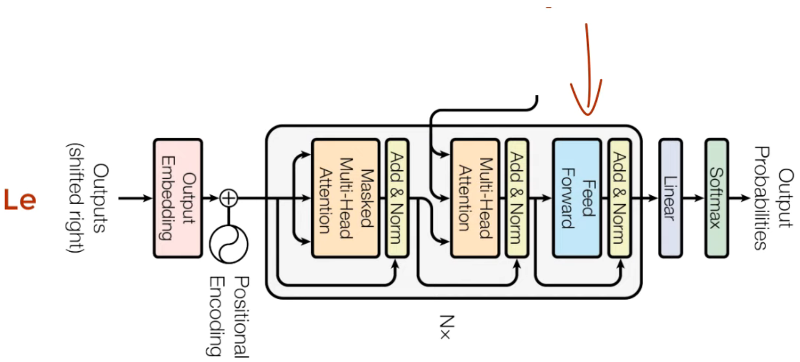


- generates attention vectors for every word to construct context
- generates only attention for previous words (no learning otherwise)

Block 2: Multi-Head Attention

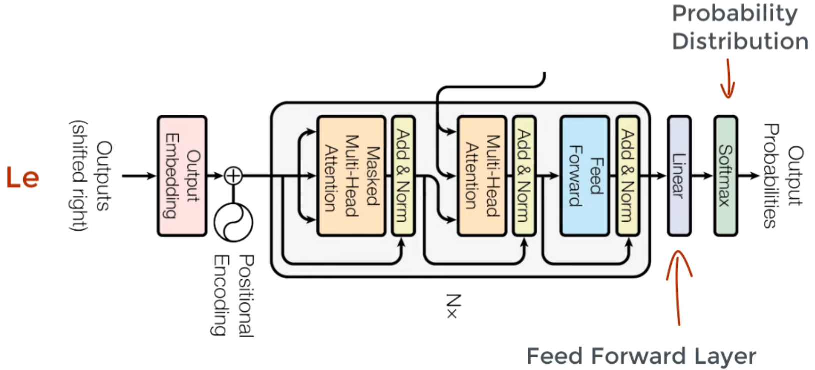


- gets positional-encoding vectors and vectors from the encoder
- output: attention vectors for every word in both languages
- generates similar attention vectors for both languages



- reduces dimensionality of the feature vector and computes features

Output



- feed-forward layer maps to the number of words in second language
- softmax transforms output to a probability distribution
- final word is the one given by the highest probability

Training procedure

- standard WMT 2014 English-German dataset, 4.5 million sentence pairs
- byte-paired encoding (37000 tokens vocabulary)
- 25000 tokens per batch
- 8 NVIDIA P100 GPUs, 12 hours for base model, 3.5 days large model (1024 as dimension, $d_{ff} = 4096$, heads=16)
- ADAM optimizer with varying LR
- residual dropout regularization

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

- outperforms previous SoA by more than 2.0 BLEU
- even base model surpasses all previously published models at a fraction of training cost of any previous model
- base model uses averaged model of last 5 checkpoints (20 checkpoints for big)

Conclusion

- first sequence transducer model based entirely on attention
- recurrent layers replaced with multi-headed self-attention
- significantly faster than traditional RNNs
- paper's code: `https://github.com/tensorflow/tensor2tensor`

Bibliography

1. https://arxiv.org/abs/1706.03762
2. http://mlexplained.com/2018/01/13/
   weight-normalization-and-layer-normalization-explained-no
3. https://www.youtube.com/watch?v=TQQlZhbC5ps
4. https://www.youtube.com/watch?v=iDulhoQ2pro
5. http://mlexplained.com/2018/01/13/
   weight-normalization-and-layer-normalization-explained-no
6. https://towardsdatascience.com/
   understanding-encoder-decoder-sequence-to-sequence-model-
7. http:
   //colah.github.io/posts/2015-08-Understanding-LSTMs/