Seq2Seq Models: *the Transformer, Self-Attention.*

Lorena Méndez

Seq2Seq models are models that map sequences to sequences making minimal assumptions on the sequence structure. The dominant models are based on recurrent or convolutional neural networks that include an encoder and a decoder [1].
Introduced in 2017 by Google, the Transformer is the first model relying entirely on self-attention to handle ordered sequences of data without recurrence or convolution. It is superior in quality while being more parallelizable and requiring significantly less time to train [2].

Despite their flexibility and power, deep neural networks face a challenge dealing with sequences because they require that the dimensionality of the inputs and outputs is known and fixed. This is a significant limitation, since many important tasks such as translation or speech recognition, are best expressed with sequences whose lengths are not known a-priori. To overcome this limitation, Google introduced Seq2Seq models in 2014 [1].

Seq2Seq models are composed of an encoder and a decoder:

- The encoder is a stack of several recurrent units that captures the context of the input sequence in the form of a hidden state vector (encoder vector) and sends it to the decoder.
- The decoder is also a stack of several recurrent units, where each takes a hidden state from the previous unit as an input and produces its own hidden state and an output.

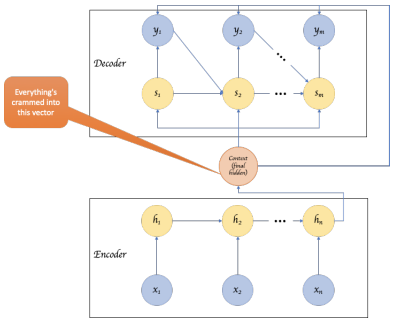


Fig. 1. Seq2Seq model

With this architecture, we can achieve tasks such as translating “What are you doing today?” from English to Chinese (input of 5 words and output of 7 symbols).

But a big drawback is that the longer the input sequence length is, the more difficult is to capture context [3]. To avoid this weakening context issue, Attention was introduced in 2015 [4]. This architecture consumes the hidden state of each element of the input sequence instead of just look at one single context vector as shown in Fig. 2.

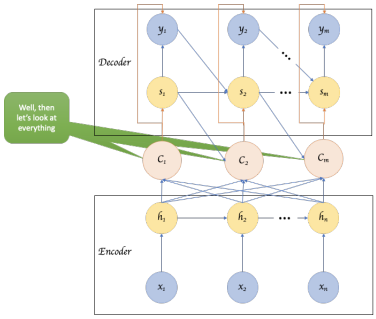


Fig. 2. Attention: using all hidden states, not just the last one

While attention seems to have addressed the limitation of having a single context vector, it has made the model really big, since there are a lot of computations involved when obtaining each context vector for every output step. In addition, these computations cannot be performed in parallel as every step requires information of the previous one. The lack of parallelization was solved when the Transformer was introduced by Google in 2017 [2].

The Transformer

The Transformer is the first model mapping sequences into sequences that relies entirely on self-attention to compute representations of its input and output without using recurrent or convolutional neural networks.

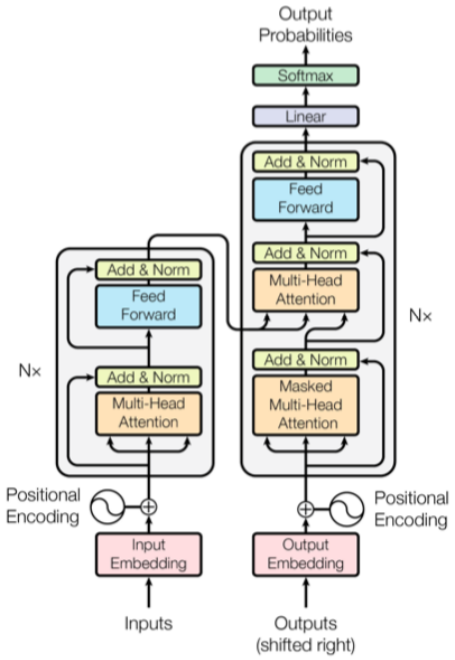


Fig. 3. The Transformer architecture

The Transformer like the Seq2Seq model introduces in 2014, is based in an Encoder/Decoder structure:

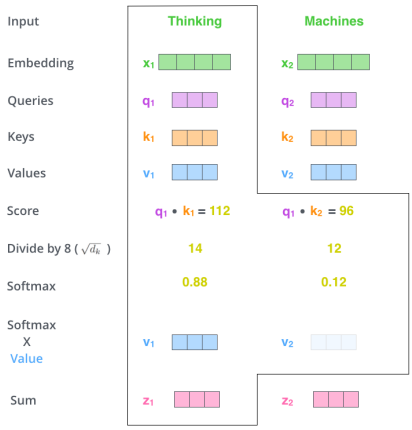
- Encoder: processes the input sequence after the positional encoding. It is composed of a stack of N = 6 identical layers with two sub-layers: a multi-head self-attention mechanism and a simple position-wise fully connected feed-forward network. Around them, there is a residual connection followed by a layer normalization to improve the performance [5].
  The output of the top encoder is transformed into a set of attention vectors. These are used by the decoder to focus on appropriate places in the input sequence.
- Decoder: generates a sequence output. It is composed of a stack of N = 6 identical layers with three sub-layers: two multi-head self-attention mechanisms and a simple position- wise fully connected feed-forward network. Similar to the encoder, residual connections were employed around each of the sub-layers, followed by layer normalization.
  The output of each step is fed into the decoder in the next time step, after being embedded and went through positional encoding. This is repeated until a special symbol is reached indicating the transformer decoder has completed its output.
  The first multi-head self-attention mechanism is modified to ensure that the prediction for each position depends only on the known outputs of the previous positions.

The final Linear Softmax layers turn the vector of floats, that is given as an output of the decoder, into a word.

The use of multi-head self-attention makes the Transformer superior in quality while being more parallelizable and requiring significantly less time to train.

Self-Attention

Self-Attention is an attention mechanism relating different positions of a sequence in order to compute a representation of the sequence itself. This is how it works:



1. **Creates three vectors from each of the encoder's input vectors** called query, key and value vector. These vectors are created by multiplying the embedding by three matrices obtained during the training process.
2. **Calculates a score.** The score determines how much focus to place on other parts of the input sentence as we encode a word at a certain position.
3. **Divide the scores by the square root of the dimension of the key vectors** to have more stable gradients.
4. **Pass the result through a softmax operation.** The softmax determines how much each word will be expressed at this position. Clearly, the word at this position will have the highest softmax score, but sometimes its useful to attend to another word that is relevant to the current word.
5. **Multiply each value vector by the softmax score.** The intuition here is to keep intact the values of the word(s) we want to focus on and drown out irrelevant words.
6. **Sum up the weighted value vectors.** This produces the output of the self-attention layer at this position, which will be sent through a feed forward neural network (when using the transformer).

To calculate Self-Attention using matrices: first, pack the embeddings into a matrix (X) and multiply it by the three matrices (W^Q, W^K, W^V) obtained during the training process to get the query (Q), key (K) and value (V) matrices.

$$Q = X \times W^Q$$

$$K = X \times W^K$$

$$V = X \times W^V$$

Steps 2-6 can be summarized by this formula:

$$Z = softmax(\frac{Q \times K^T}{\sqrt{d_k}}) \times V$$

Multi-head self-attention

Multi-head self-attention consists in performing self-attention several times (8 in the Transformer) with different weight matrices and concatenate all the outputs to obtain a matrix that captures the information of all attention heads. This improves the attention layer in two ways:

1. Expands the model's ability to focus on different positions. For example, if we are translating a sentence like "The animal did not cross the street because it was too tired", we would want to know which word "it" refers to.
2. It gives attention layer multiple "representation subspaces", since we have multiple sets of query-key-value matrices.

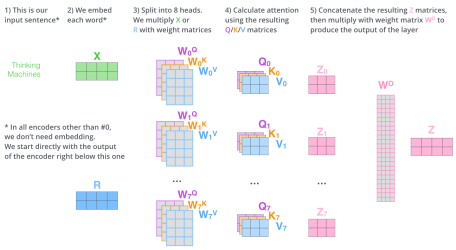


Fig. 4. Multihead Attention.

For example, in the sentence "The animal didn't cross the street because it was too tired", the different attention heads are focusing in different parts of the sentence as we encode the word "it":

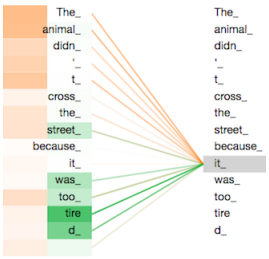


Fig. 5. Multihead Attention. (2 heads visualization)

One attention head is focusing most on "the animal", while another is focusing on "tired". This means, the model's representation of the word "it" refers to "animal" and "tired".

With this architecture and relying on multi-head self-attention, it is shown that the Transformer outperforms both recurrent and convolutional models on academic English to German and English to French translation benchmarks [2]. On top of higher translation quality, the Transformer requires less computation to train and is a much better fit for modern machine learning hardware, speeding up training by up to an order of magnitude.

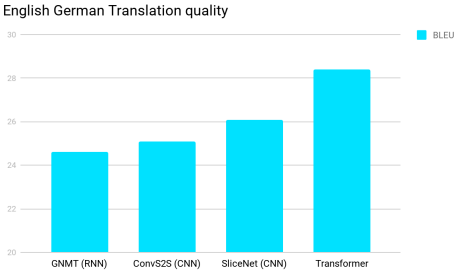


Fig. 6. BLEU scores - English German translation quality

References

1. Sutskever, Ilya; et al.: 'Sequence to sequence learning with neural networks', *Advances in neural information processing systems*, 2014, p. 5255-5262
2. Vaswani, Ashish; et al.: 'Attention Is All You Need', *Advances in Neural Information Processing Systems*, 2017, p. 5998-6008
3. Koehn, Philipp and Knowles, Rebecca: 'Six Challenges for Neural Machine Translation', *Proceedings of the First Workshop on Neural Machine Translation*, 2017, p. 28–39
4. Bahdanau, Dzmitry; et al.: 'Neural Machine Translation by Jointly Learning to Align and Translate', 2014
5. He, Kaiming; et al.: 'Deep Residual Learning for Image Recognition', 2015