MO434 - Deep Learning
Transformers

Alexandre Xavier Falcão

Institute of Computing - UNICAMP

afalcao@ic.unicamp.br

- RNNs and LSTMs are widely used in sequential tasks, such as next word prediction, machine translation, and text generation.

- RNNs and LSTMs are widely used in sequential tasks, such as next word prediction, machine translation, and text generation.

- However, they cannot capture long-term dependency.

- RNNs and LSTMs are widely used in sequential tasks, such as next word prediction, machine translation, and text generation.

- However, they cannot capture long-term dependency.

- Transformers appeared to overcome that limitation by getting rid of recurrence and adopting self-attention [1].

- RNNs and LSTMs are widely used in sequential tasks, such as next word prediction, machine translation, and text generation.
- However, they cannot capture long-term dependency.
- Transformers appeared to overcome that limitation by getting rid of recurrence and adopting self-attention [1].
- A transformer is a deep architecture to solve sequence-to-sequence tasks.

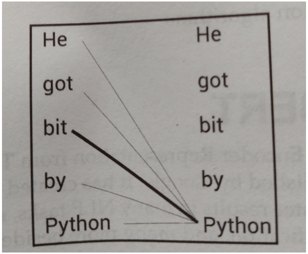
- RNNs and LSTMs are widely used in sequential tasks, such as next word prediction, machine translation, and text generation.
- However, they cannot capture long-term dependency.
- Transformers appeared to overcome that limitation by getting rid of recurrence and adopting self-attention [1].
- A transformer is a deep architecture to solve sequence-to-sequence tasks.
- Let's understand how a transformer works with a language translation task.

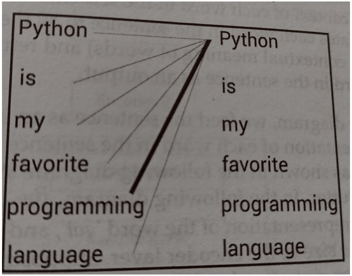
Agenda

- The encoder of a transformer.
  - Positional encoding.
  - Self-attention mechanism.
  - Other operations.
- The decoder of a transformer.
  - Masked multi-head attention.
  - Other operations.

- The encoder of a transformer is a context-based embedding model, differently from word2vec which is a context-free embedding model.

- The difference is that the former correlates each word of a sentence with the others (self-attention), generating a different representation when they have distinct meanings.

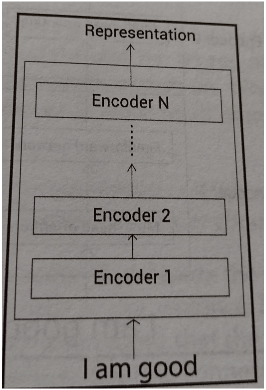


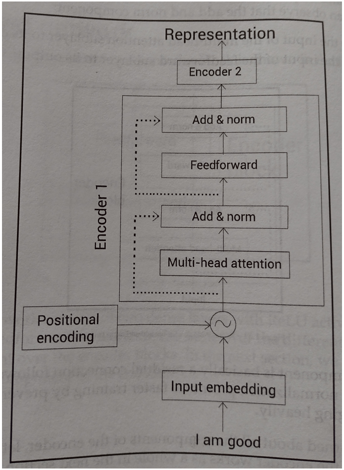


Figures from Getting Started with Google BERT from now on.

The encoder of a transformer

A transformer may have multiple encoders (left), being the configuration of each encoder as shown on the right.





- An input sentence is always converted into a sequence of token ids (by a tokenizer) when inserted into the model.

- By training, a transformer learns an input embedding for each word in a sentence, forming an input matrix $\mathbf{X}$.

- However, before the self-attention mechanism, it is important to encode the position of each word in the sentence. This requires a positional encoding matrix $\mathbf{P}$ such that $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{P}$.

$$P = \begin{array}{r} \text{I} \\ \text{am} \\ \text{good} \end{array} \left[\begin{array}{cccc} \sin(pos) & \cos(pos) & \sin\left(\frac{pos}{100}\right) & \cos\left(\frac{pos}{100}\right) \\ \sin(pos) & \cos(pos) & \sin\left(\frac{pos}{100}\right) & \cos\left(\frac{pos}{100}\right) \\ \sin(pos) & \cos(pos) & \sin\left(\frac{pos}{100}\right) & \cos\left(\frac{pos}{100}\right) \end{array} \right]$$

where pos is the position of the word in the sentence: 0 for 'I', 1 for 'am' and 2 for 'good'.

The encoder of a transformer

The self-attention mechanism requires three matrices, $\mathbf{Q}$ (query), $\mathbf{K}$ (key) and $\mathbf{V}$ (value), such that $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K$ and $\mathbf{V} = \mathbf{X}\mathbf{W}^V$ with the weight matrices $\mathbf{W}^*$ learned by training.

The self-attention mechanism requires three matrices, $\mathbf{Q}$ (query), $\mathbf{K}$ (key) and $\mathbf{V}$ (value), such that $\mathbf{Q} = \mathbf{X}\mathbf{W}^{Q}$, $\mathbf{K} = \mathbf{X}\mathbf{W}^{K}$ and $\mathbf{V} = \mathbf{X}\mathbf{W}^{V}$ with the weight matrices $\mathbf{W}^{*}$ learned by training.

A self-attention matrix $\mathbf{Z}$ with a new embedding for each word is then defined by

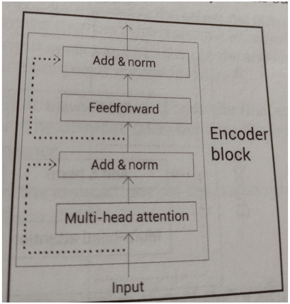
$$\mathbf{Z} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{t}}{\sqrt{d}}\right)\mathbf{V},$$

where d is the embedding dimension of each word. Matrix $\mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{t}}{\sqrt{d}}\right)$ contains the correlations between each pair of words in the sentence. Matrix V essentially adapts the correlation matrix to different tasks.

The encoder of a transformer

- To treat possible ambiguities, we usually use multiple attention heads and the resulting self-attention matrices are concatenated and multiplied by another weight matrix to create the final **Z**.

- The encoder block also contains a feedforward layer with two dense layers with RELU activation, additive skip connections and normalization tto speed up convergency.

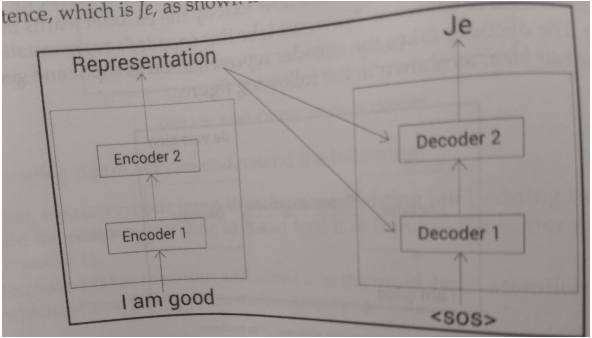


How to extract the word embeddings from an encoder?

If you want to use an encoder to simply extract word embeddings for classical operations, such as matching, clustering or classification, the following notebook shows how to ▸ (REPRESENT.).

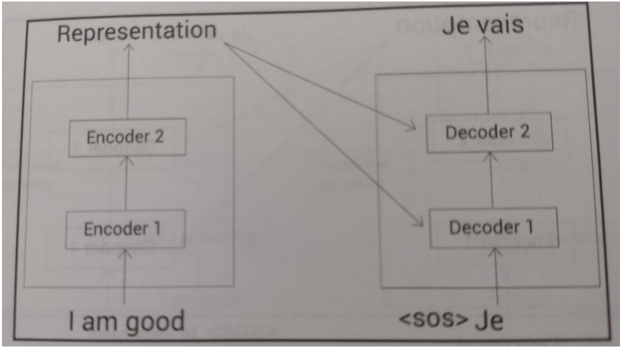
The decoder of a transformer

Similarly to the encoder, a transformer usually has a stack of decoders. At each step t, the output of step $t-1$ is used as input, being $< sos >$ and $< eos >$ the start-of-sentence and end-of-sentence tags.

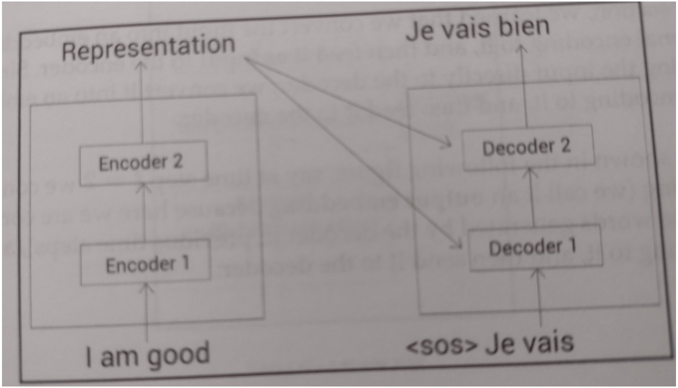


The decoder of a transformer

Similarly to the encoder, a transformer usually has a stack of decoders. At each step t, the output of step $t-1$ is used as input, being $< sos >$ and $< eos >$ the start-of-sentence and end-of-sentence tags.

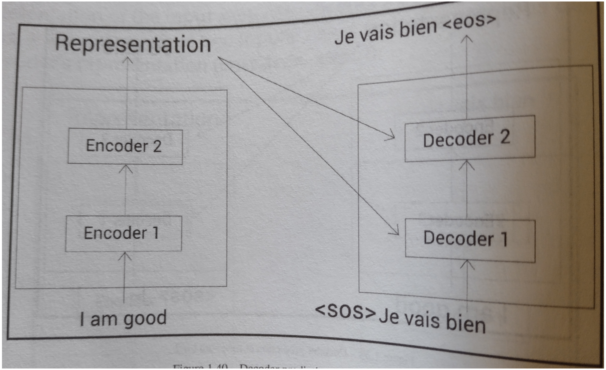


Similarly to the encoder, a transformer usually has a stack of decoders. At each step t, the output of step $t-1$ is used as input, being $< sos >$ and $< eos >$ the start-of-sentence and end-of-sentence tags.

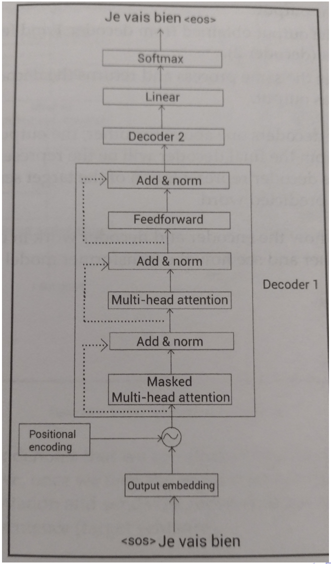


The decoder of a transformer

Similarly to the encoder, a transformer usually has a stack of decoders. At each step t, the output of step $t-1$ is used as input, being $< sos >$ and $< eos >$ the start-of-sentence and end-of-sentence tags.



Differences lie on both multi-head attention sublayers.



The decoder of a transformer

- A self-attention matrix of the entire input sentence $< sos >$ **Je vais bien** can be computed at each head, but it has to simulate all four steps: $< sos >$, $< sos >$ **Je**, $< sos >$ **Je vais**, and $< sos >$ **Je vais bien**.

The decoder of a transformer

- A self-attention matrix of the entire input sentence $< sos >$ **Je vais bien** can be computed at each head, but it has to simulate all four steps: $< sos >$, $< sos >$ **Je**, $< sos >$ **Je vais**, and $< sos >$ **Je vais bien**.

- Therefore, the elements to the right of each word can be masked by $-\infty$ in each given self-attention matrix $\mathbf{Z}$.

	<SOS>	Je	vais	bien
<SOS>	9.125	$-\infty$	$-\infty$	$-\infty$
Je	5.0	12.37	$-\infty$	$-\infty$
vais	7.25	5.0	10.37	$-\infty$
bien	1.5	1.37	1.87	10.0

The decoder of a transformer

- Again, the matrices of each head are concatenated and multiplied by a weight matrix to obtain a final matrix.

The decoder of a transformer

- Again, the matrices of each head are concatenated and multiplied by a weight matrix to obtain a final matrix.

- Now, let $\mathbf{M}$ be the output of the add&norm sublayer after masked multi-head attention.

The decoder of a transformer

- Again, the matrices of each head are concatenated and multiplied by a weight matrix to obtain a final matrix.
- Now, let $\mathbf{M}$ be the output of the add&norm sublayer after masked multi-head attention.
- The subsequent multi-head attention must be able to use $\mathbf{M}$ and the output matrix $\mathbf{R}$ from the encoder.

The decoder of a transformer

- Again, the matrices of each head are concatenated and multiplied by a weight matrix to obtain a final matrix.
- Now, let $\mathbf{M}$ be the output of the add&norm sublayer after masked multi-head attention.
- The subsequent multi-head attention must be able to use $\mathbf{M}$ and the output matrix $\mathbf{R}$ from the encoder.
- At each given head, the query, key and value matrices are $\mathbf{Q} = \mathbf{MW}^Q$, $\mathbf{K} = \mathbf{RW}^K$, and $\mathbf{V} = \mathbf{RW}^V$.

- Again, the matrices of each head are concatenated and multiplied by a weight matrix to obtain a final matrix.

- Now, let $\mathbf{M}$ be the output of the add&norm sublayer after masked multi-head attention.

- The subsequent multi-head attention must be able to use $\mathbf{M}$ and the output matrix $\mathbf{R}$ from the encoder.

- At each given head, the query, key and value matrices are $\mathbf{Q} = \mathbf{M}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{R}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{R}\mathbf{W}^V$.

- As described earlier, they are used to obtain one self-attention matrix per head, which contains the similarities between input and target words.

Training can use cross entropy since the decoder generates a probability distribution of the words in a vocabulary.

Training can use cross entropy since the decoder generates a probability distribution of the words in a vocabulary.

Let's see how to use transformers (i.e., the BERT model) from hugging face for several applications (TRANSFORMERS).

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin.
Attention is all you need, 2017.