Introduction to Deep Learning

The transformers (freely inspired by http://jalammar.github.io/illustrated-transformer/)

J. Rynkiewicz

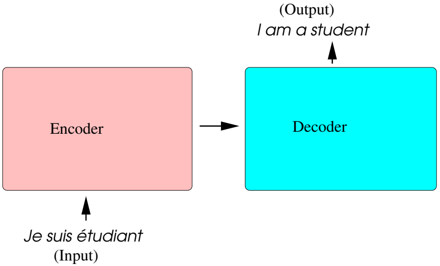
Université Paris 1

2022

The transformers

Transformer neural networks aim to predict a sequence of variable length according to another sequence of variable length. The principle is to take into account the context of the observations to predict (concept of attention) :

- Let us write :
  - $(X_1, \cdots, X_{T_X})$, the explanatory sequence.
  - $(Y_1, \cdots, Y_{T_Y})$ the sequence to be predicted. Note that T_Y is also to be predicted.
  - θ the model's parameter vector.
- This formalism is suitable for chatbots or machine translation systems.
- In general, the architecture of this neural network is composed of an "encoder" and a "decoder" :

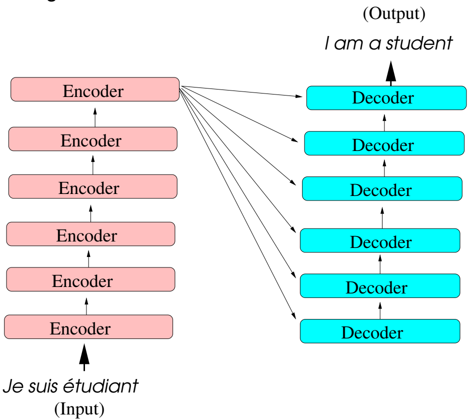


Tokenizer

- To tokenize a text consists in breaking it down into words or sub-words. They are then encoded in numbers (ids).
- The three most commonly used techniques are : Byte-Pair Encoding (BPE), WordPiece and SentencePiece.
- Splitting a text only in characters does not allow to obtain results close to the state of the art.
- But splitting the text according to the spaces is not satisfactory. It is necessary to differentiate the punctuation and to make possible the construction of new words.
- So we mix the coding of words and characters : sub-words.
- The idea of the sub-words is to keep the most used words whole, but to cut out the rare words or words transformed by the grammar (adverb etc...)
- Subword tokenization keeps the vocabulary size reasonable (typically 50000 tokens).
- The best sets of sub-words are built by keeping the most frequent sub-words (BPE) or those that maximize the likelihood of the text (WordPiece).
- SentencePiece tokenizes the texts by considering spaces as characters, then uses BPE or Wordpiece.

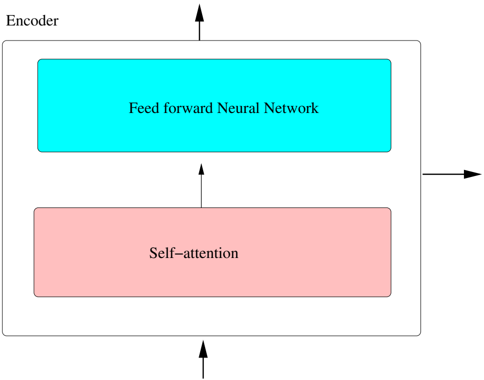
Encoder, decoder

The encoder is a stack of N small encoders, the decoder a stack of N small decoders. In the original article $N = 6$.



Small encoder

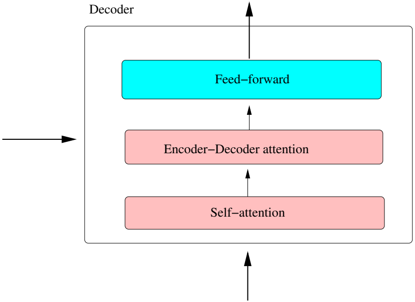
- The small encoders have the same architecture (but they don't share their parameters).



- The input of the small encoder first goes through a Self-attention layer (described later).
- The output of the small encoder passes, before, through a feed-forward network whose architecture is identical for all encoders.

Small decoder

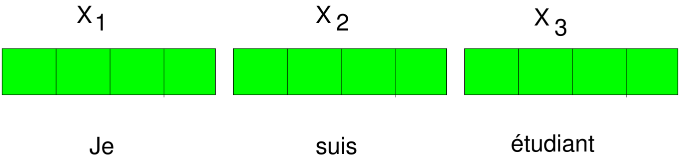
- The small decoders all have the same architecture (but they don't share their parameters).



- The input of the small decoder first goes through a layer of Self-attention (described later).
- There is an intermediate layer to focus on the input sequence and its transformation by the encoder.
- The output of the small encoder passes, before, through a feed-forward network whose architecture is identical for all decoders.

Word representation

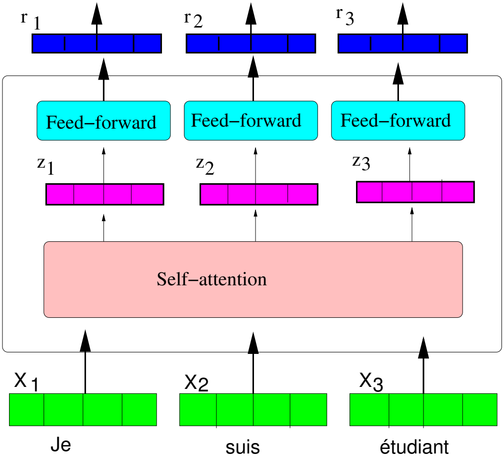
- We start by embedding the words in a continuous space ($\mathbb{R}^{512}$ in the original article).



- For the illustration, the embedding is in $\mathbb{R}^4$.
- The embedding takes place only for the input of the first small encoder.
- All other small encoders receive the output of the previous small encoder (of the same size as the embedding).
- After , the vectors pass through the two layers of the first small encoder.

Propagation through the first small encoder

- The small encoder receives a list of vectors and returns a list of vectors of the same size.
- The vectors first pass through the Self-attention layer, then through the Feed-forward networks.

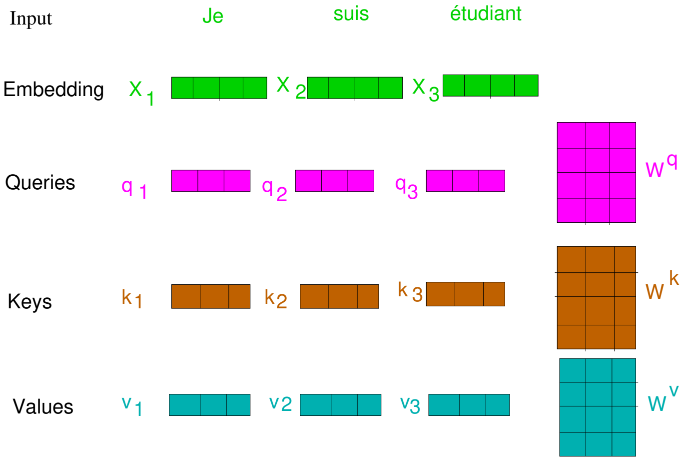


Attention Mechanism

- Attention allows us to take into account the context of a word.
- To translate the sentence "The sheep did not cross the street because it was too tired”
- Ẁhat does it refer to “it” in the texte “the sheep” or the “street" ?
- This is an easy question for a human being, but difficult for a machine.
- The machine must therefore estimate whether the word “it” is more related to the word "sheep" or the word "street".
- The layer of “Self-attention” of transformers proposes a method to allow this estimation.

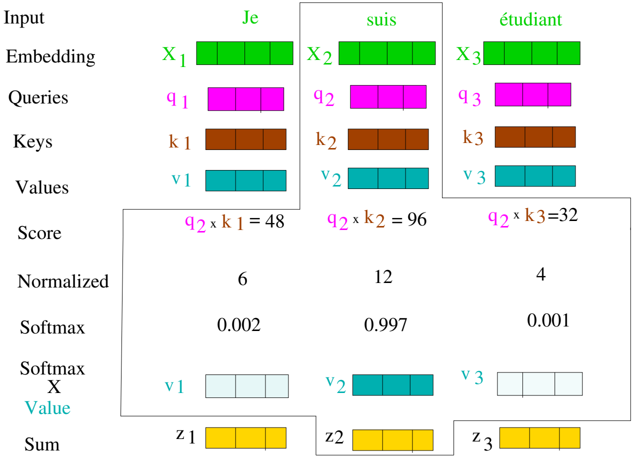
Detail of the "Self-attention" layer (1)

- We start by defining three vectors : The query vector, the key vector and the value vector.
- These vectors are created by multiplying the "Embedding" by a weight matrix (of dimension 64x512 in the original article).



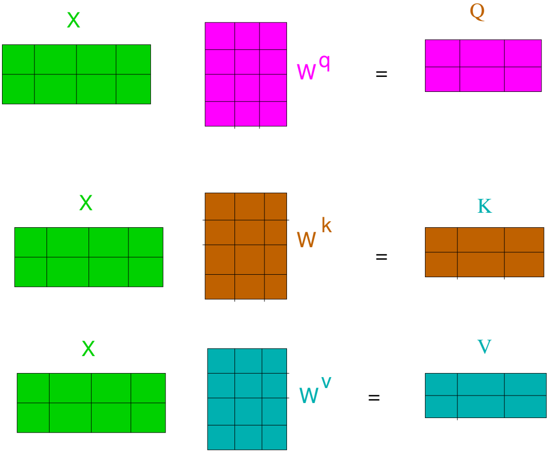
Detail of the "Self-attention" layer (2)

- The embedding is transformed into a query, a key and a value and then each value is weighted by a softmax of the score induced by all keys.
- The weighted sum of the values is the "z" output of the attention layer.



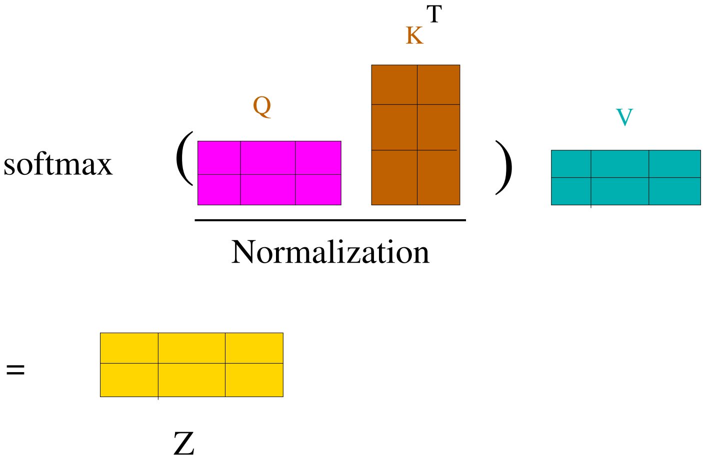
Matrix computation for attention (1)

- The matrix form allows to parallelize the calculations.



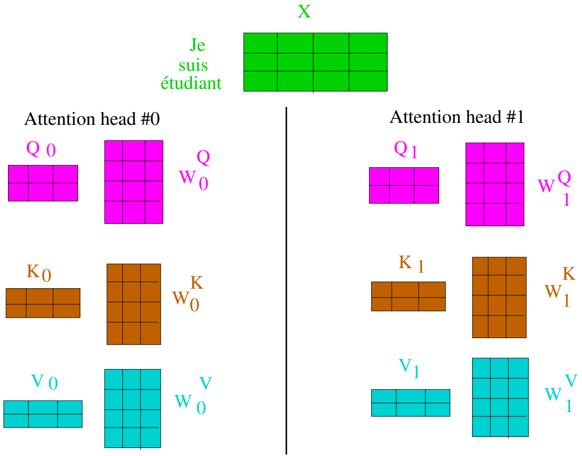
Matrix computation for attention (2)

- The detailed computation of attention can be summarized by the following matrix equation :



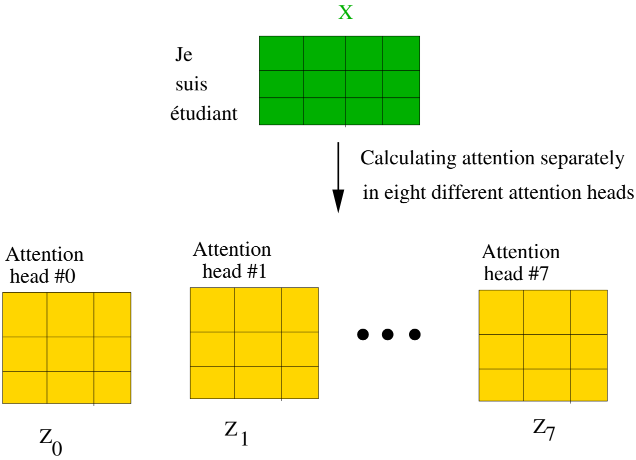
"Multi-headed" attention layer

- In the original article, there are several layers of attention in parallel.
- This allows the model to have a larger context representation space.



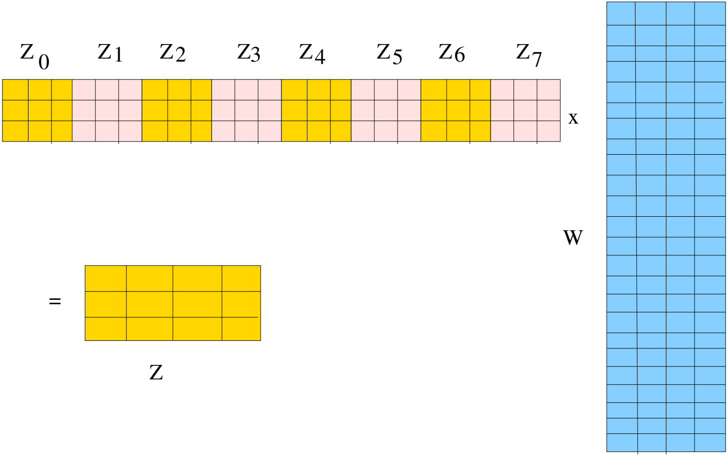
Computations of the “Multi-headed” attention layer

- Each layer of attention has different weights.
- In the original paper, the authors compute 8 matrices Z.



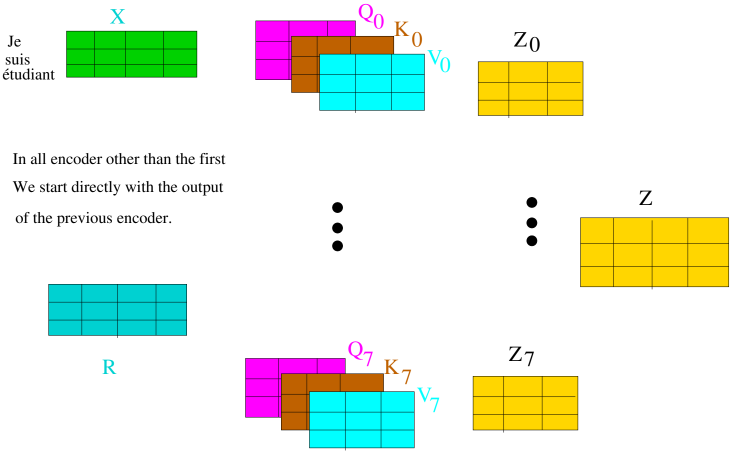
Final attention computation

- To obtain the final attention, we concatenate the outputs of the attention layers.
- Then, we multiply this vector by a weight matrix W which will be estimated.



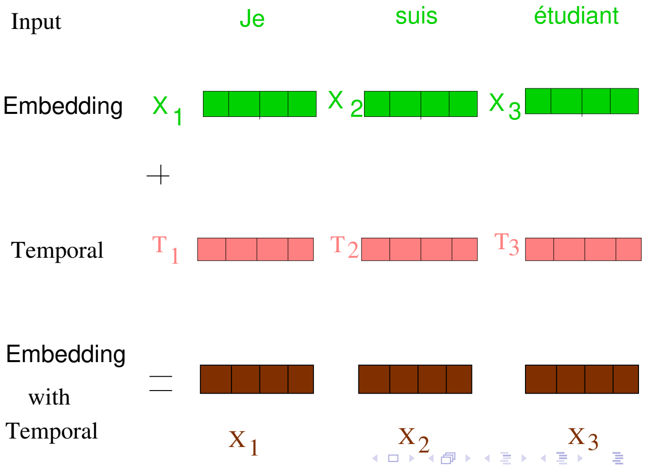
To summarize the multiheaded attention layer

- The final attention vector Z will be obtained by the computations of the previous slides.



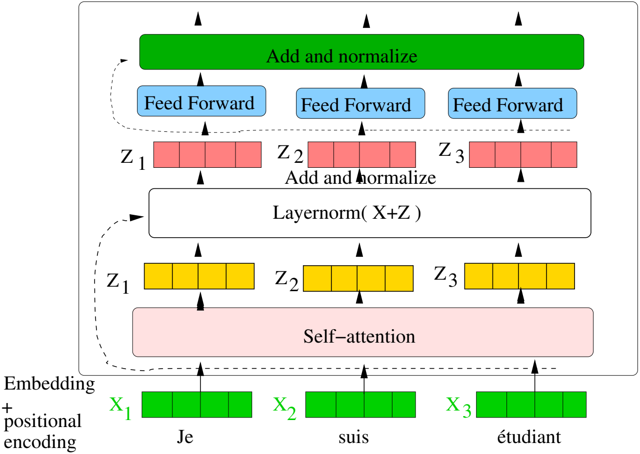
Encoding of the position to represent the order

- We want to be able to take into account the order of the words in the sentence.
- For this we will embed a periodic signal (a cosine) in a space of the same size as the one where we plunged the words.
- Then we will add the two embeddings.



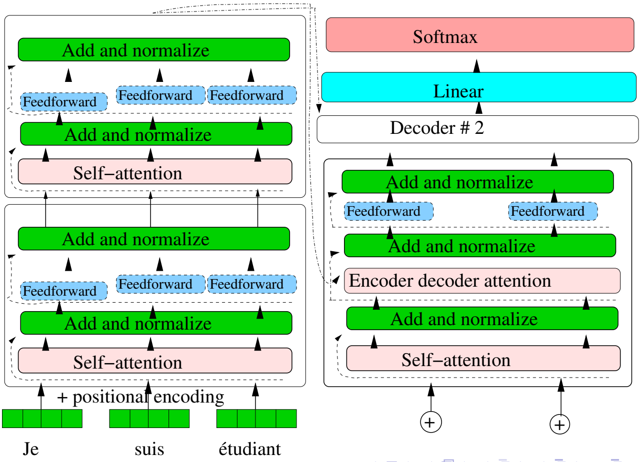
Residual connections

- Each sub-layer has a residual connection (copy of the input).
- This improves the transmission of information and the computation of the gradient.



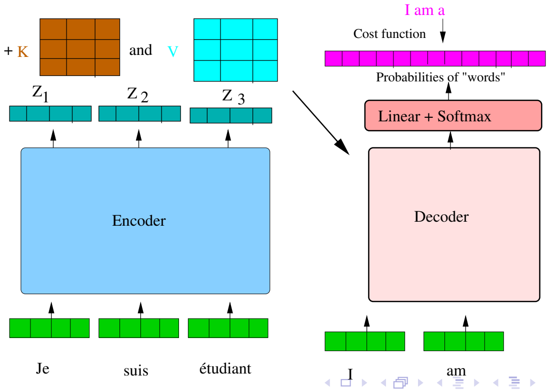
Link between encoder and decoder

- We represent here an encoder made of two small encoders, idem for the decoder.
- The decoder architecture has an additional layer of attention to account for the encoder output.



Estimation of transformer parameters

- The decoder uses the output but also the final "keys" and "values" vectors of the encoder to compute the conditional probabilities of the ouput's words.
- It will predict the conditional probability of the target words one after the other according to the encoder outputs and the previous target words.
- To do this, it hides the target words that follow the word to be predicted by assigning a zero probability to the coordinates corresponding to them in its "value" vector.



Generation of the sequence by the decoder

- The decoder starts with the encoder output as input. It then generates the most probable word (or punctuation mark) according to it (its weights).
- The decoder then uses the output of the encoder and the word it has just generated to generate the next word.
- The self attention encoder-decoder layer uses the final keys and values of the encoder in addition to the characters already generated by the decoder.
- The decoder stops generating words when it emits the special “end of sentence” character.
- Note that the size of the vocabulary is necessarily finite. In practice it is a few tens of thousands.
- The decoder can't invent new words, but it can sequence their characters.

Transfert learning for the NLP

- Since 2018 models have pre-trained weights on large database (books, wikipedia, etc...).
- We therefore use these pre-trained models and finely calibrate their weight for a specific task.
- The two best known (but rapidly changing) are BERT and the GPT models (OpenAI-GPT, GPT2 and GPT3).
- BERT takes into account the whole context (the past and future tense of the sentence), it is especially useful for :
  - Analysis of the feeling (positive or negative sentences).
  - More generally, sentence classification (spam, non-spam etc...)
  - Question/answer.
- GPT has been trained to predict the next word in a sentence. This model uses only the decoder of the transformer, it is especially useful for :
  - Chatbot.
  - Text generation in general.

Text generation

GPT2 (now GPT3) has been trained to predict the next word in a sentence. It is the basic model for generating text. However, it only computes the probability law of the next sub-word, a method is needed to draw it randomly among all sub-words. The main methods are :

- Greedy search : We simply choose the most likely sub-word. This tends to create repetitive sequences.
- Beam search : We compute the probabilities of the sequences of words of length NB, and we choose the most probable sequence of words. NB is a hyperparameter to set. This method always tends to create repetitions and natural language seems more random.
- Top K sampling : We draw randomly among the K most probable sub-words (by renormalizing the probability law on these K sub-words). However, the K hyperparameter may not be adapted to all probability laws.
- Top P sampling : We choose K such that the sum of the probabilities of K sub-words is equal to at least P.
- In practice, the two previous methods are mixed with a maximum K.