DeepLearning Specialization Coursera

Sequence Models: Week 4

Transformer Network

Although the vanishing gradient problem of RNN model is overcome by GRU and LSTM model, all the models perform tasks in sequential order of word tokens. Transformer network on the other hand allows parallel computation of work tokens which makes it faster.

Transformer network combines the attention-based representation and CNN style of parallel processing. Attention model comprises two types of representation, Self Attention and Multi-Head Attention. Self Attention computes N representations of N words in a sentence in parallel. Multi-Head Attention computes the multiple version of these representations which is very rich representations for performing machine translation task.

Self Attention

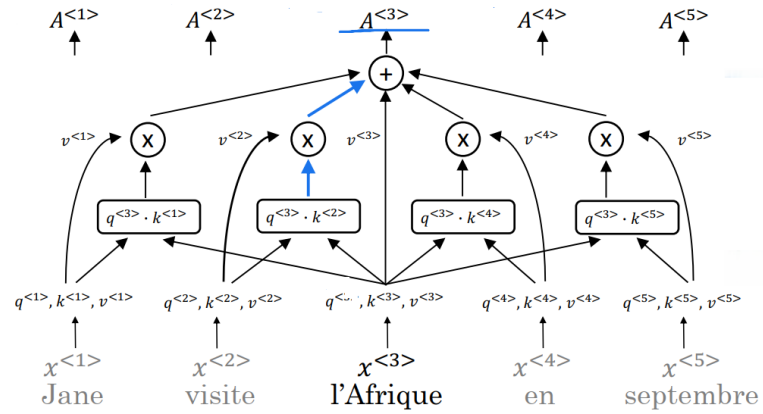
Compute attention based vector representation A(q, K, V) of each word in input sentence. Attention representation of current word is computed using the representation of surrounding words in parallel.

Transformers Attention

$$A(q,K,V)=\sum_i \frac{\exp(q\cdot k^{<i>})}{\sum_j \exp(q\cdot k^{<j>})} v^{<i>}$$

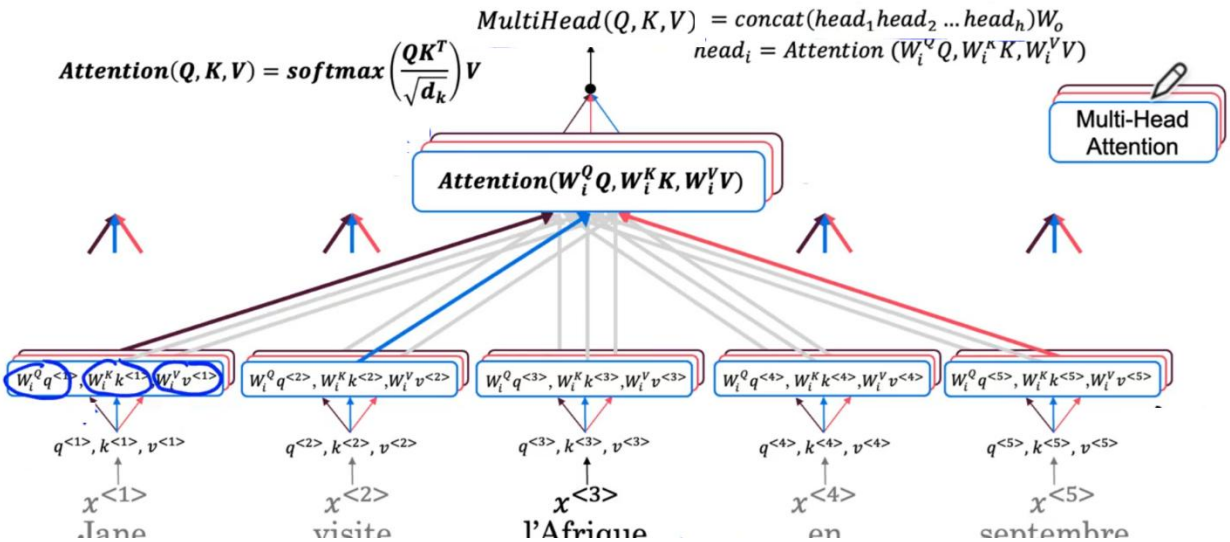
Each word has three vectors, q: Query, K: Key, V: Value where q, K and V is computed using weight matrix W^Q, W^K, W^V which is multiplied by input word X. Compute the inner product of q and $K^{<t>}$ for each word and then calculate Softmax for each word presentation $A^{<t>}$.

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



Multi-Head Attention

Multi-Head attention calculates multiple self attention for each word. Each self attention representation is called head. Each head representation is computed using each W^Q, W^K, W^V vectors and finally computed heads are concatenated and multiplied by weight vector W^O.

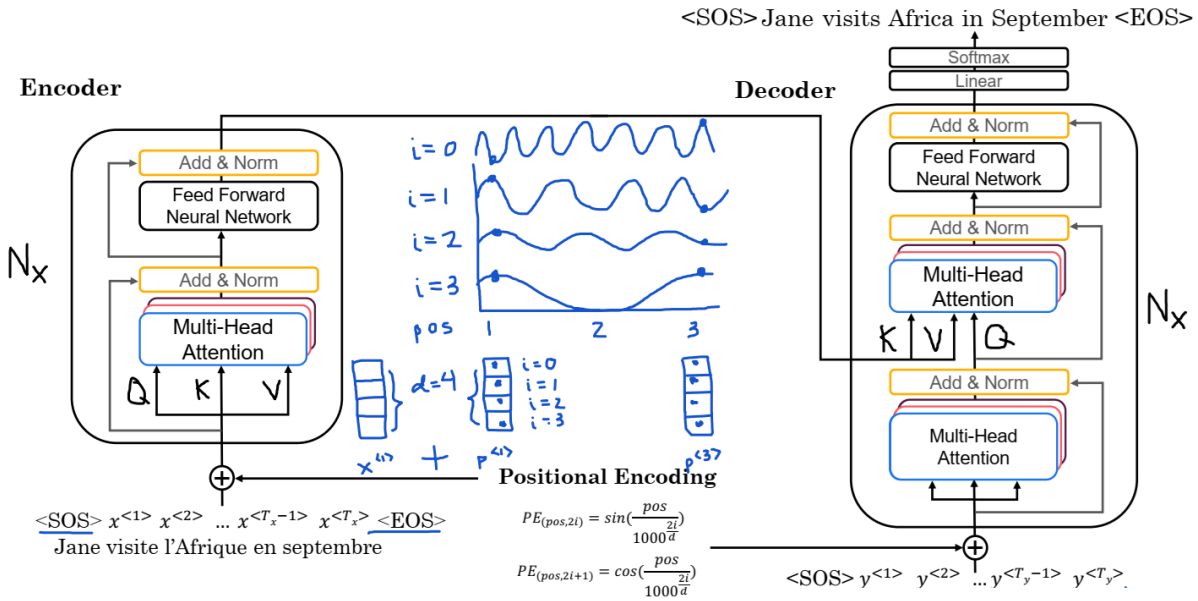


Transformers

Word embeddings are fed into Encoder block containing Multi-Head attention layer. Output of attention layer then fed into a feed forward NN which helps determine features in the sentence. Encoder block is repeated N_X times with typical value of six which is then fed into a decoder block. Decoder block generates the output sentence and in each step decoder block input is the previously generated output sequence which goes into a Multi-Head attention layer. First Multi-Head attention layer in Decoder block generates Q vector and for Encoder's output generates K and V vector second Multi-Head attention layer. Second Multi-Head attention layer output is fed into a feed forward NN. Similar to Encoder block, Decoder block is also repeated N_X times and predicts the next word in the sentence.

Positional encoding is done to determine the position of words in sentence before feeding into Multi-Head attention layer of Encoder block. Positional encoding vector p has same dimension d as word embedding vector dimension. Encoding for position pos and ith element of position vector uses sine and cosine function. The ith element number has relation with k=d dimension value as k//2. Position encoding creates unique position vector of words in sentence. Position encoding vector is added to the word embedding vector to influence the position of word in sentence. Encoder block output contains contextual and semantic meaning, along with positional encoding information. Positional encoding is also passed through the network as residual connection as RESNET. Similar to batch normalization Add & Norm layer is repeated

across entire Transformer architecture to speed up learning. Output of Decoder block follows Linear and Softmax layer to predict single word in each time step. Masked Multi-Head attention layer used in Decoder block while training a Transformer model.



[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng