Khaoula Belahsen
khaoula.belahsen@yahoo.fr

1 Question 1

From the expression of Attention(Q,K,V) = softmax($\frac{QK^T}{\sqrt{d}}$)we have Q, K, V are defined as follows in the following cases :

- Attention of y over x in the classical encoder-decoder framework :
  - K = V = x
  - Q = y
- Self-attention over x :
  - K = Q = V = x

2 Question 2

- **For the Self-attention** :

  We have an input of size n x d Q=K=V, composed of n words represented as being of depth d.

  A dot product is performed in the depth direction. The matrix multiplication QK^T is the product of an n x d matrix by a d x n matrix. This gives a complexity of O(n^2 * d). We then apply a softmax, which is n*n operations and then multiply the resulting n x n matrix by the n x d matrix V.
  The total complexity is thus **O(n^2 * d)**.

- **For the CNN** :

  Let's represent the computation with the following elements :

  - A $\in \mathbb{R}^{nxd}$ the matrix representing our input.
  - W $\in \mathbb{R}^{kxd}$ one of the filters, with k the kernel size.
  - We suppose that we have d filters ? (in order to preserve the depth ?)

  We have that one layer of the CNN is equivalent to the following operations :

  - Each filter passing is equivalent to computing k dot products between W and $A_k \in \mathbb{R}^{kxn}$ a portion of the input. This dot product contains d multiplications and d-1 summations. In total it has a complexity of O(k * d).

  - Each filter slides (n-k+1) time over the input matrix and thus performs n-k+1 dot products. This takes the overall complexity to 0((n-k+1) * k * d). If we suppose that $k << n$, we have 0(n * k * d).

  - Supposing there are d filters, the previous computation is performed d times. And thus the overall complexity is **O(n * k * d^2)**.

- **For the RNN** :

  We scan through the data from left to right using weights representing the transition between the input-hidden, hidden-hidden and hidden-output.

  If we consider that the input sentence is composed of n elements of size d, then the recurrent network will swipe through the input n times, while performing O(d) operation. If we consider d different swipes, we have a complexity of **O(n * d^2)**.

  The RNN layer is equivalent to a 1D conv layer with a kernel size of 1. ?

3 Question 3

Using Multiple heads of small dimension allows the model to learn relevant information in different representative child spaces at different positions.

Using mutli-heads also allows to compute the attention mechanism for simpler spaces (in terms of dimensionality) in a parallel way, thus speeding the computation time.

4 Question 4

Let's write for $i \in \{1, .., \frac{dmodel}{2}\}$:

$$\begin{bmatrix} \sin \lambda_i(pos+k) \\ \cos \lambda_i(pos+k) \end{bmatrix} = \begin{bmatrix} \sin(\lambda_i pos)\cos(\lambda_i k) + \cos(\lambda_i pos)\sin(\lambda_i k) \\ \cos(\lambda_i pos)\cos(\lambda_i k) - \sin(\lambda_i pos)\sin(\lambda_i k) \end{bmatrix} = \begin{bmatrix} \cos(\lambda_i k) & \sin(\lambda_i k) \\ -\sin(\lambda_i k) & \cos(\lambda_i k) \end{bmatrix} \begin{bmatrix} \sin(\lambda_i pos) \\ \cos(\lambda_i pos) \end{bmatrix}$$

Where : $\lambda_i = \frac{1}{10000^{\frac{2i}{dmodel}}}$.

In the expression above, we uncover a rotation matrix, let's call it r_i^k.

We have that :

$$r_i^k = \begin{bmatrix} \cos(\lambda_i k) & \sin(\lambda_i k) \\ -\sin(\lambda_i k) & \cos(\lambda_i k) \end{bmatrix}$$

Therefore we can define the following matrix :

$$R^k = \begin{pmatrix} r_1^k & 0 & \cdots & 0 \\ 0 & r_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{\frac{dmodel}{2}}^k \end{pmatrix}$$

We conclude that :

$$PE_{pos+k} = R^k PE_{pos}$$

Which concludes the demonstration for the linear relation between PE_{pos+k} and PE_{pos}.

Advantage of Positional encoding :

In RNNs and LSTMs, the hidden state at position t+1 depends on the hidden state from position t. The network accumulates information by identifying the relative positions of each word . However, in the Transformer, there is no notion of sequence. Positional encoding seems to be a way to retrieve information about relative positions of words.

5 Question 5

The masking is used by applying it to the matrices of dot products. We multiply it by a low-triangular like matrix in order to disable all the elements above the diagonal, which would represent illegal information. This step is performed after the softmax because it includes an exponential and because we cant the resulting values to sum to 1.

6 Question 6

Number of parameters :

- In the encoder : a stack of N=6 layers. Each of them has two sublayers. The first sublayer is a multi-head self-attention, thus it requires N_{heads} projection spaces, each of them is represented by three matrices $W_i^Q \in \mathbb{R}^{dmodel,dk}$, $W_i^K \in \mathbb{R}^{dmodel,dk}$, and $W_i^V \in \mathbb{R}^{dmodel,dv}$ thus : $N_{heads} \times dmodel \times (2d_k + d_v)$ parameters to estimate in the encoder self-attetion layer. The second sublay is a feed forward layer that involves : $W_1 \in \mathbb{R}^{dmodel \times h}$, $b_1 \in \mathbb{R}^h$, $W_2 \in \mathbb{R}^{h \times dmodel}$, and $b_2 \in \mathbb{R}^{dmodel}$ thus the sublayer involves : $h \times (dmodel + 1) + dmodel \times (h + 1)$ parameters to estimate. The total number of parameters in the encoder : $N \times [h \times (dmodel + 1) + dmodel \times (h + 1) + N_{heads} \times dmodel \times (2d_k + d_v)]$.
- In the decoder : a stack of N=6 layers. Each of them has 3 sublayers. The first one is the multi-head self-attention of the decoder, and same as in the encoder, it requires $N_{heads} \times dmodel \times (2d_k + d_v)$ parameters to estimate. The second sublayer is the multi-head attention between the output of the encoder and the output of the multi-head self attention of the decoder, thus it requires the same number of parameters as above (we take the same parameters for all the attention layers) thus $N_{heads} \times dmodel \times (2d_k + d_v)$ parameters to estimate. And the third sublayer is a feed forward layer with the same dimensions as in the encoder stack so $h \times (dmodel + 1) + dmodel \times (h + 1)$ parameters. In the total, we need $N \times [h \times (dmodel + 1) + dmodel \times (h + 1) + 2 \times N_{heads} \times dmodel \times (2d_k + d_v)]$ parameters for the decoder.

Finally, the number of parameters for the encoder + the decoder is : $N \times [2(\times h \times (dmodel + 1) + dmodel \times (h + 1)) + 3 \times N_{heads} \times dmodel \times (2d_k + d_v)]$. In **[?]**, we have $dmodel = 512$, $h = 2048$, $N_{heads} = 8$ and $d_v = d_k = 64$ thus : 6558720 parameters.

7 Question 7

Section non-traitée pour le moment

8 Question 8

Since BERT is a multi-task model (it is designed to perform different tasks : questions answering, language inference,...), the input of the model may not be a single sentence, but two sentences for example. Thus, the input is represented as a "sequence" that combines the sentences inputs. Then, the WordPiece embedding is applied to these sequence. If the sequence contains two sentences, the model separates them by [SEP], and by adding a learned embedding indicating whether the sentence of the sequence is A or B.

References