Virgile Rennard
virgile@rennard.org

Lab session # 8
ALTEGRAD 2019

01/22/20

1 Question 1

In the case of attention of y over x, as in the classical encoder-decoder framework, we have the queries that come from the previous layer, y is Q, and K, V come from x. In the case of Self attention over x, Q, K, V all come from x

2 Question 2

In the simple self-attention, for each query we try to assess its compatibility with each key using a dot product. This dot product is performed in the depth direction, meaning that the matrix multiplication QKT is the product of an n x d matrix multiplied by a d x n matrix, resulting in an n2 d complexity. 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, again resulting in n2 d, which is the total complexity of self-attention. The same idea can be derived for getting the Recurrent complexity. For the case of the convolutional layer, as we have a 1D convolution, the sum of the row-wise dot products of a filter $W \in \mathbb{R}^{k \times d}$ with a region matrix $A \in \mathbb{R}^{kd}$, where k is the length of the filter and d is the depth dimension Therefore, we have $O(d)$ for one dot product (d multiplications + d1 additions) we perform in total k dot products (there are k rows in W and A), which amounts to $O(kd)$ and finally, at the layer level, we apply the filter over the input $nk + 1$ times (where n is the length of the input), let' say n times since $n >> k$. This gives us a final complexity of $O(n = kd)$ The extra d comes from the fact that we are repeating this for every one of the d filters.

3 Question 3

"Multi-head attention allows the model to jointly attend to information from different representation at different positions"[3]. Having multiple head of small dimension is a cheap way to get more information than a large one, and is able to work better for parallelization

4 Question 4

Let $E \in \mathbb{R}^{n \times d_{model}}$ a matrix containing d_{model} dimensional column vectors E_t, encoding the position t in an input sequence of length n. The function $e : \{1, \dots, n\} \longrightarrow \mathbb{R}^{d_{model}}$ induces this matrix and is defined as

$$e(t) = E_{t,:} := \begin{bmatrix} sin(t/f_1) \\ cos(t/f_1) \\ \vdots \\ cos(t/f_{d_{model}/2}) \end{bmatrix}$$

with the frequencies $f_m = 1/\lambda_m$. We need to prove that we have a linear transformation $T^{(k)} E_{t,:} = E_{t+k,:}$ for any $k \in \{1, \dots, n\}$ $t \in \{1, \dots, n-k\}$.
This stems from the fact that we can find a definition for $T^{(k)}$ with no dependence on t :

$$T^{(k)} = \begin{bmatrix} \nu_1^{(k)} & 0 & \dots & 0 \\ 0 & \nu_2^{(k)} & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & nu_{\substack{(k) \\ d_{model}/2}} \end{bmatrix}$$

0 denots 2×2 all zero matrices, and the $d_{model}/2$ transposed rotation matrices $\nu^{(k)}$ are defined by

$$\nu_m^{(k)} = \begin{bmatrix} cos(r_m k) & -sin(r_m k) \\ sin(r_m k) & cos(r_m k) \end{bmatrix}^T$$

What we want to prove is therefore :

$$\nu_m^{(k)} * \begin{bmatrix} sin(\lambda_m t) \\ cos(\lambda_m t) \end{bmatrix} = \begin{bmatrix} sin(\lambda_m (t+k)) \\ cos(\lambda_m (t+k)) \end{bmatrix}$$

this means :

$$sin(\lambda k + \lambda t) = sin(rk)cos() + cos(rk)sin(\lambda t)$$

and

$$cos(\lambda k + \lambda t) = cos(rk)cos(\lambda t) - sin(rk)sin(\lambda t)$$

Using the formula of trigonometric addition, we get $\lambda k = rk$, $\lambda t = \lambda t$, and $r = \lambda$. With that, we simply get :

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

Giving us the result.
This technique is used because there is no notion of word order (1st word, 2nd word, ..) in the proposed architecture. All words of input sequence are fed to the network with no special order or position (unlike common RNN or ConvNet architectures), thus, model has no idea how the words are ordered. Consequently, a position-dependent signal is added to each word-embedding to help the model incorporate the order of words. Based on experiments, this addition not only avoids destroying the embedding information but also adds the vital position information. In the case of RNNs, we feed the words sequentially to RNN, i.e. n-th word is fed at step n, which helps the model incorporate the order of words.

5 Question 5

The masking, combined with the fact that the output embeddings are offset by one, makes it so that predictions in the transformer for a given i position can depend on the outputs known only from the outputs at position less than i It puts the padding values to 0 and the other to one, making it so that the illegal information is not used.

6 Question 6

the original paper [3] has a table with all the number of coefficients. It is explained that there are 65 millions coefficient. This is clearly untrainable on a single computer

7 Question 7

Training was inconclusive, so I'll take examples shown in [1]. :

- "Portugal, Spanien und Deutschland sind europäische Länder" $\longrightarrow$ "portugal , spain and germany are european countries"
- "Portugal, Spanien und Deutschland sind Länder in Europa" $\longrightarrow$ "portugal , spain , spain and germany are countries in europe"
- "Im Jahr 2014 wurde Deutschland Fußballweltmeister" $\longrightarrow$ "in , germany was ¡UNK¿""

8 Question 8

We will base our answer on [2]. It is explained that BERT's model architecture is a multi layer bidirectional transformer encoder, that is based on Vaswani et al. paper. It has L Transformer blocks of hidden size H, and A self attention heads. The input representation represents both a sentence and a pair of sentence in one token sequence (question, answer). One sentence could be a span of continuous text rather than a linguistic sentence. The same idea comes with the fact that a sequence refers to the input token sequence to BERT, one or two sentences packed together. They are representend with WprdPiece embeddings. The final hidden state corresponds to a classification toker. The pairs are packed together into a single sequence, and are differentiated in two ways, with a separator token, and a learned embedding token indicating whether it belongs to sentence A or B. One token's input representation is constructed by summing the corresponding token, segment, and position embeddings. As we've previously mentioned, as the self attention heads are of small dimensions, and the sentences are in one token, this is a poor choice of representation.

References

[1] http://vandergoten.ai/2018-09-18-attention-is-all-you-need/.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.