Basile Terver
basile.terver@polytechnique.edu

1 Question 1 & Task 1

Let us follow the [3] architecture, which is based on the transformer architecture [4] studied in the previous lab.
A Transformer encoder layer is made up of a self-attention layer and a simple, position-wise fully connected feedforward network.
In Section 3.2.2 of [4], the number of parameters of each multi-attention head is given by the parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. The multi-head attention is given by

$$\begin{aligned} \text{Multihead(Q,K,V)} &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{1}$$

We employ h parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{model}/h$.
In each layer, after the attention sub-layer, the feedforward consists of two linear transformations with a ReLU activation, that is

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

The dimensionality of input and output is d_{model} so $W_1, W_2 \in \mathbb{R}^{d_{model} \times d_{model}}$.
After both the self-attention and the feedforward sublayers, the "Add and norm" sublayer does not add any trainable parameters.
In our code `nhid` and `nhead` correspond respectively to d_{model} and h from the paper.
The parameters of the base model should therefore be, for each of the transformer encoder layers $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{model}}$ for $i \in \{1, \dots, h\}$ and W_1, W_2. This yields

$$\begin{aligned} N \times \Big[h\Big(d_{model}d_k + d_{model}d_k + d_{model}d_v\Big) + hd_v d_{model} + 2\big(d_{model}^2 + d_{model}\big)\Big] \\ = N \times \Bigg[\underbrace{3d_{model}^2}_{attention} + \underbrace{d_{model}^2}_{concat} + \underbrace{2\big(d_{model}^2 + d_{model}\big)}_{feedforward} \Bigg] \end{aligned} \tag{3}$$

trainable parameters, where N is the number of transformer unit layers.
Printing all named trainable parameters of our base model, we realize that they are some more parameters, not present in the article [4], such as biases associated to W^O, W^Q, W^V, W^K, that is N times $4d_{model}$ more parameters. Moreover, each "Add and Norm" layer has $4d_{model}$ trainable parameters, so that is $4Nd_{model}$ more parameters.

1.1 Positional embeddings

In addition to this transformer model, one should not forget the embedding layer. In the original paper [4], embedding layer maps the input of size the number of tokens n_{tokens} to the feature vector of size d_{model}, which adds $n_{tokens}d_{model}$ parameters. BERT [1] and RoBERTa [3], the positional embedding values are trainable parameters, hence a $d_{model}(T+2)$, where T is the maximum length of a sentence and $T + 256$. The $+2$ comes from the eos and sos tokens.

1.2 Final trainable parameters

In the end, the base model has

$$N \times \Bigg[\underbrace{3(d_{model}^2 + d_{model})}_{attention} + \underbrace{d_{model}^2 + d_{model}}_{concat} + \underbrace{2\big(d_{model}^2 + d_{model}\big)}_{feedforward} + \underbrace{4d_{model}}_{norm} \Bigg] + \underbrace{n_{tokens}(d_{model} + (T+2)) + 2d_{model}}_{embedding} \tag{4}$$

trainable parameters.

1.3 Numerical results

We have $N = 4$ Transformer Encoder Layers, $d_{model} = 512$ with 8-headed attention heads ($h = 8$) and a vocabulary of size $n_{tokens} = 32000$.This yields

$$24d_{model}^2 + d_{model}(n_{tokens} + T + 2) + 42d_{model} = 24 \times (512)^2 + 512(32000 + (256 + 2)) + 42 \times 512 = 22829056. \quad (5)$$

trainable parameters, as printed in the notebook.

2 Task 3 & 4

Like in the previous lab, in Figure 1, we can see a performance gap between the pre-trained model, which has better accuracy on the validation set throughout the training epochs. The training we perform here is for a classification task, which, as seen in Question 1, is very different from the language modeling task on which the model has been pre-trained.
Yet, throughout 5 training epochs on the classification task, it seems that the accuracy of the "from sratch" model is not about to catch up with the accuracy of the pre-trained model. Both accuracy curves seem to plateau, the "from sratch" curve at 0.62-0.65 and the pre-trained one at 0.80-0.85 accuracy.
This can be explained by the fact that language models acquire a high-level language representation which allows, as a corollary to perform specific tasks such as sentence classification.

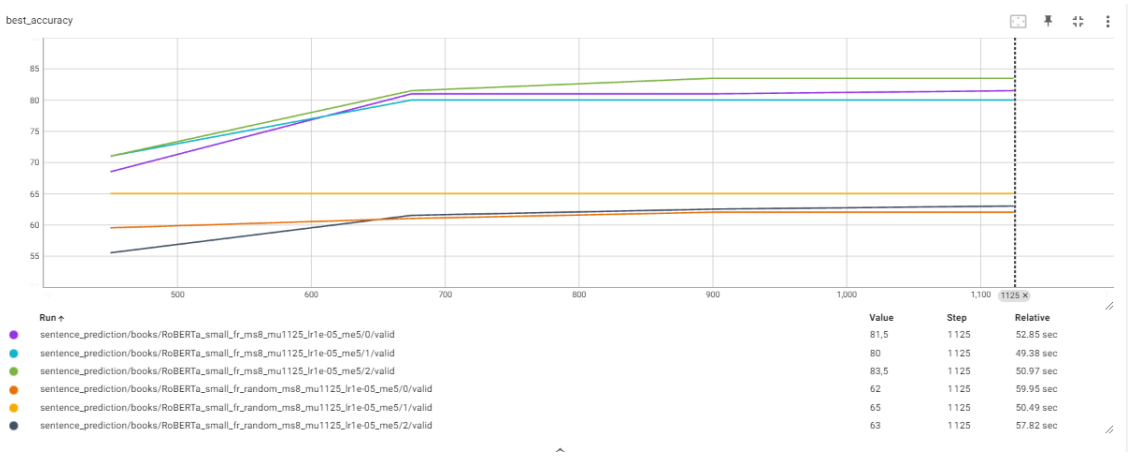

Figure 1: Plot of the evolution of the validation accuracy of the pre-trained and the "from scratch" model on the validation set during training on the classification task.

Remark: the number of samples for the training of the random might not be the same as for the pretrained model, so we will fix the number of backpropagations to 1125, which is the right number of backpropagations for the pretrained model. Indeed, $num\_samples * MAX\_EPOCH/MAX\_SENTENCES = 1125$ with $num\_samples = 1800$ sentences for training.

3 Question 2

The goal of LORA [2] is to train a lower rank matrix instead of the original weights of the pre-trained large language model. If the dimension of an embedding is d_{model}, the pre-trained wieght matrix W_0 has dimension d_{model}^2, we instead train ΔW with low rank $r << d_{model}$.
Let us define the parameters of the LORA function:

- $r = 16$ is the rank of the low rank matrix approximation
- $lora\_alpha = 32$: scaling factor of the low rank approximation adapter. The higher it is the less the original prediction plays a role
- $target\_modules$: modules on which we apply LORA. Here, we only apply it on the attention submodule of each attention layer, not the feedforward submodules.
- $bias = False$: no additive bias in low rank approxiamtion.
- $task\_type = CAUSAL\_LM$: Our finetuning on questions and answers is language modeling task, not a classification task.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.