Romain AMÉDÉE
romain.amedee@telecom-sudparis.eu

1 Question 1

The role of the square mask in our implementation is to ensure causality. Predictions on the next tokens are made only based on previously predicted tokens and future tokens are ”masked” in order for the model to be able to predict yet unseen tokens which is a feature needed for generative models. This is achieved by putting the value of future tokens to -inf which will give zero after going through the softmax function therefore ignoring their contributions when deriving self-attention.

Positional encoding is applied to the input embeddings before they are passed through the Transformer layers. This allows the model to capture both the intrinsic characteristics of tokens and their positions in the sequence. Positional encoding complements the self-attention mechanism, making the model more accurate and better at comprehending the notions of context and causality in the sentences.

2 Question 2

Replacing the classification head (fine-tuning) is done in order to use a pretrained model that has a specific task for another task. The model in itself is preserved but the desired output (given by the classification head) can be chosen by doing so.

The goal of language modeling is to build a model that is able to give preductions of words based on the context provided and previously translated tokens. This is particularlly useful for text generation or translations.

Classification involves assigning predefined labels to input data (text, documents, sentences) based on its content.

3 Question 3

Let us estimate the number of trainable parameters.

3.1 Embedding

The first step of our transformer is the embedding step. We use a vocabulary of size $n_{token} = 100$ and a hidden dimension for the transformer layers of $n_{hid} = 200$. Therefore we have :

$$W_{embdeding} = n_{token} \times n_{hid} = 20000$$

3.2 Attention layer

Next step is the attention layer. The model performs three linear operations (Query, Value, Key) through the MultiheadAttention() class. The dimensions of those operations are n_{hid} for both input and output. Then one last linear operation is performed with same dimensions. Therefore, addind n_{hid} biases we have the following number of parameters for the attention layer :

$$P_{attention} = 3 \times (n_{hid}^2 + n_{hid}) + (n_{hid}^2 + n_{hid}) = 4 \times (n_{hid}^2 + n_{hid}) = 160800$$

3.3 Fully-connected layer

Next step are the two linear layers for each attention head with dimension mappings $n_{hid} \longrightarrow n_{hid}$. Counting the bias, we have the following number of parameters :

$$P_{linear} = 2 \times (n_{hid}^2 + n_{hid}) = 80400$$

3.4 Norm Layer

We have two layer norms with each $2 \times n_{hid}$ parameters. Therefore :

$$P_{norm} = 2 \times 2 \times n_{hid} = 800$$

3.5 Total

We have 4 transformer blocks in the implementation so their will be a factor 4 for the parameters of the encoder.

3.5.1 Language modeling

For the modelling task, we ue a linear operation of size (n_{hid}, n_{token}). Counting the bias we have :

$$P_{modelling} = n_{hid} \times n_{token} + n_{token} = 200 * 100 + 100 = 20100$$

Finally the total number of parameters for the language modelling part is :

$$Total_{modelling} = W_{embedding} + 4 \times (P_{attention} + P_{linear} + P_{norm}) + P_{modelling} = 1008101$$

3.5.2 Classification

For the classification task, we use a linear operation of size $(n_{hid}, n_{classes})$. Counting the bias we have :

$$P_{classification} = n_{hid} \times n_{classes} + n_{classes} = 200 * 2 + 2 = 402$$

Finally the total number of parameters for the classification part is :

$$Total_{classification} = W_{embedding} + 4 \times (P_{attention} + P_{linear} + P_{norm}) + P_{classification} = 988402$$

4 Question 4

Let us observe the accuracies of the pretrained model and the model ”from scratch” :

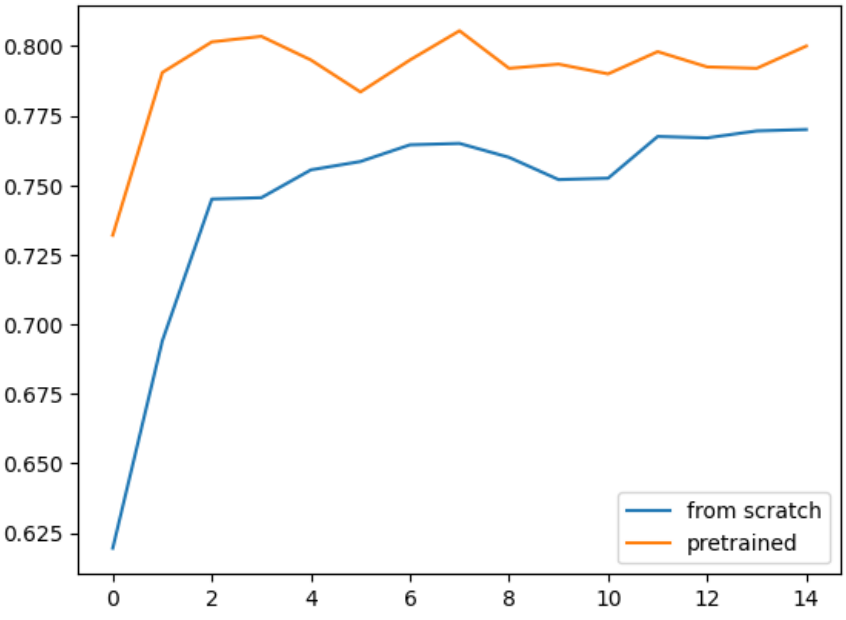

Figure 1: Accuracies of our two models ”pretrained” and ”from scratch”

Firstly, the starting point for the accuracies of the two models are very different. The accuracy of the model "from scratch" starts at approximately 63% whereas it starts at 80% for the pretrained model.

After a few epochs (approx. 8), the accuracy starts stabilizing for both models. The pre-trained model stabilizes at $\approx 80\%$ while the model from scratch ends up at $\approx 76\%$.

Overall, the pre-trained model has better results and performances than the model from scratch. The starting point for the accuracy is higher for the pre-trained model because its weights were already put through training while the weights of the model from scratch start at zero for which a low accuracy is expected. The higher ending point for the pre-trained model could be explained by the quantity and the quality of the data used for training.

5 Question 5

One of the limitations of the language modeling objective used in our notebook, is that it is unidirectional and lacks context from both directions.

Our model is trained to predict the next word or token in a sequence based only on previous tokens. Therefore, it only has access to the left context and lacks information about the right context. This limits the model's understanding of the entire context and can result in suboptimal performance.

On the other hand, the masked language model (MLM) objective introduced in [1] overcomes this limitation by training the model to predict masked tokens in a bidirectional manner. BERT masks a random subset of tokens in the input sequence and trains the model to predict the masked tokens by capturing context from both directions.

References

[1] 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.