BENYAMINA Elyas
elyas.benyamina@ens-paris-saclay.fr

Lab session # 3
ALTEGRAD 2023

10/31/23

1 Question 1

The article [1] explains the model.

Prediction of next tokens should only use previous tokens, and square mask allows us to do so by putting the value of future tokens to $-\infty$, which will lead the softmax function make our model ignore these tokens. It's accurante when we want to perform some tasks such as translation as the order of tokens in the input sequence matters.

According to "Attention is all you need", and contrary to the first lab where we used RNNs, Transformers do not inherently have a notion of the sequential order of tokens within an input sequence. Indeed, without positional encoding, the model would treat all tokens as if they were unordered. In some language tasks such as translation tasks, the order of words has a crucial importance to understand the meaning and the context of a word in a sentence, and to produce an accurate output.
Positional encoding injects information about the position of a word to its embedding (in this article, they compute a vector corresponding to position which and they add it at the end of the embedding, which is possible as they have same dimension).

2 Question 2

The classification head, need to be tailored to the specific requirements of the task we want to perform, including the number of classes and the type of output.
In language modeling, the goal is to predict the most likely next word or token given the context (and the output is a probability vector over all tokens of the dictionary), whereas classification tasks involve categorizing text into predefined classes (and the output is a probability vector over all tokens of the dictionary).
That's why we have to change the classification head, and one of the main advantages of this model is that we can use the same pre-trained model for these different tasks, by changing the last layer (classification layer) only.
Let's notice that language modeling require all token representations to capture context, while classification tasks focus on summarizing the information in the sequence to make a single classification decision.

3 Question 3

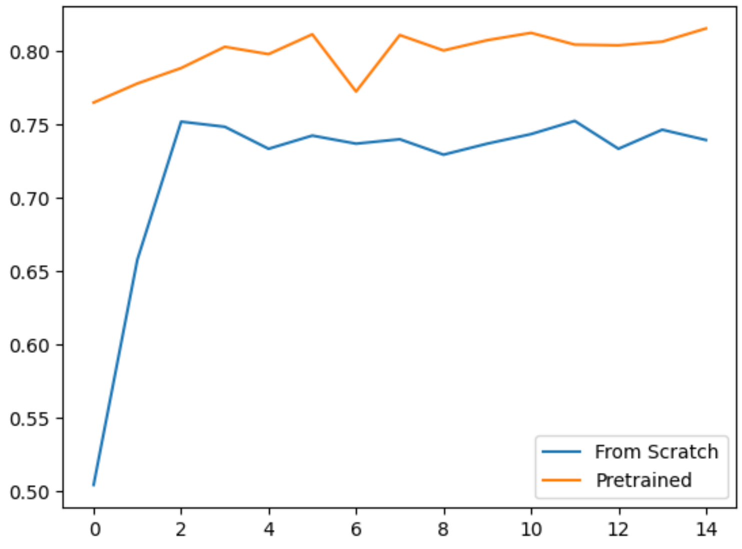
1. Trainable parameters in embedding step (n.Embedding(ntoken,nhid) layer): $n_{token}n_{hid}$
2. Trainable parameters for one attention layer : The encoder layer uses three matrices Q, K and V, and they take vectors of size n_{hid} in input and the output are vectors of the same size. So for each matrix, there are $n_{hid}^2 + n_{hid}$ trainable parameters, when adding the bias. There is a linear layer at the end, so we have to add $n_{hid}^2 + n_{hid}$, by counting the bias. So there are $4(n_{hid}^2 + n_{hid})$ trainable parameters
3. Two layers of normalization per attention layer: $2n_{hid}$ for each layer
4. Feedforward neural network per attention layer: $2(n_{hid}^2 + n_{hid})$
5. Output layer: $n_{hid}n_{output} + n_{output}$

To sum up, there are $n_{token}n_{hid} + n_{layers}(6n_{hid}^2 + 10n_{hid}) + n_{hid}n_{output} + n_{output}$ trainable parameters, where $n_{hid} = 200$ and $n_{layers} = 4$.

For language modeling task, $n_{output} = n_{token} = 100$, and there are 1008100 trainable parameters.
For classification task, $n_{output} = 2$, and there are 988402 trainable parameters.
We obtain the same values in the notebook.

4 Question 4

The pretrained model has a better accuracy and improves slowly with the number of epochs. The model trained from scracth has a low accuracy and is improving significantly over time. We can notice that the pretrained model starts with an accuracy of 75% (which is quite high because the parameters have already been trained) and improves to 83%, and the model trained from scratch starts with an accuracy of 50% (which can be explained that at the beginning the weights are randomly and the binary classification is random if the dataset is balanced) and improves to 83%.



(a)

Figure 1: Evolution of accuracy

5 Question 5

The main limitation of the language modeling objective used in this notebook is that it used only previous tokens to predict next words, and so it is an unidirectional model.
Using this model, we may miss important context from next tokens, and the results obtained with this model may not be optimal in all situations.
The BERT model used in [2] proposes to mask some input tokens randomly and predicts word using the context given by the non masked inputs. We are using both previous and next tokens, and so we have context from both directions and we overcome the limitation mentioned just before.

References

[1] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Łukasz Kaiser Ashish Vaswani, NoamShazeer and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[2] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, arXiv:1810.04805, 2018.