Celine Hajjar
celinehajjar98@gmail.com

Lab session # 3
ALTEGRAD 2021

11/30/21

Please note: I followed the INF582 course last year, so I had already done this lab (hence the similarities you might notice between the two submissions). However, I now have a deeper understanding of the models being manipulated, so it was interesting for me to do the lab again.

Question 1

Square Mask In the implementation of a standard language model, the task is to infer the next token given a sequence. Therefore, the mask is used in order to put the attention weight of future tokens to 0, and only consider past tokens for the prediction.

Positional Encoding Since all the words in a sentence are processed simultaneously, and not sequentially like in recurrent neural networks, the model doesn't have any sense of the relative or absolute position of each word. Positional Encoding allows us to inject information about the ordering of the sequence into the model, which is very important for a thorough understanding of natural language.

Question 2

In the language modeling task, the classification head gives a classification over all the tokens in the vocabulary, where the most likely token to come after the input sequence has the maximal value. Thus, the parameter n_classes is nothing but n_tokens.
However, in the classification task, given an input sequence, we want to predict whether the sentiment is positive or negative. Hence, the number of classes is 2.
The main difference between these two approaches is the number of classes we're trying to predict from.

Question 3

Base model Let us first compute the number of parameters in the base model. The layers that account for trainable parameters are the embedding layer and each of the 4 transformer encoder layers.

1. **Embedding:** The embedding layer accounts for $n_{tokens} \times dim\_emb = 50001 \times 200 = 10000200$ parameters, 50001 being the size of the vocabulary and 200 being the embedding dimension.
2. **One transformer encoder layer:** The trainable parameters of each transformer encoder layer come from 1 multi-head attention layer, 2 linear layers, and 2 norm layers. A multi-head attention layer has [1]:
$$n_{head} \times (dim\_emb \times \frac{dim\_emb}{A}) \times 3 + 3 \times dim\_emb + (dim\_emb^2 + dim\_emb)$$
which is equal to 160800 in our case (the factor 3 in the formula is due to the number of matrices we manipulate, namely, Query, Key, Value). The linear layers account each of $n_{hid} \times n_{hid} + n_{hid} = 40200$ parameters (so 80400 in total) and the norm layers account each for 400 parameters (800 in total). This leaves us with 242 000 parameters for every transformer layer, so 968 000 in total for the transformer blocks.

Thus, we have 10 000 200+968 000 = 10 968 200 parameters in the base model.

Classification In the classification task, we add to that $n_{hid} * n_{classes} + n_{classes} = 402$ parameters for the classification head.

Language modeling In the language modeling task, $n_{classes} = n_{tokens}$ in the formula above, so we have 10 050 201 parameters.
All in all, we have:

- classification task: 10968602 parameters
- language modeling: 21018401 parameters

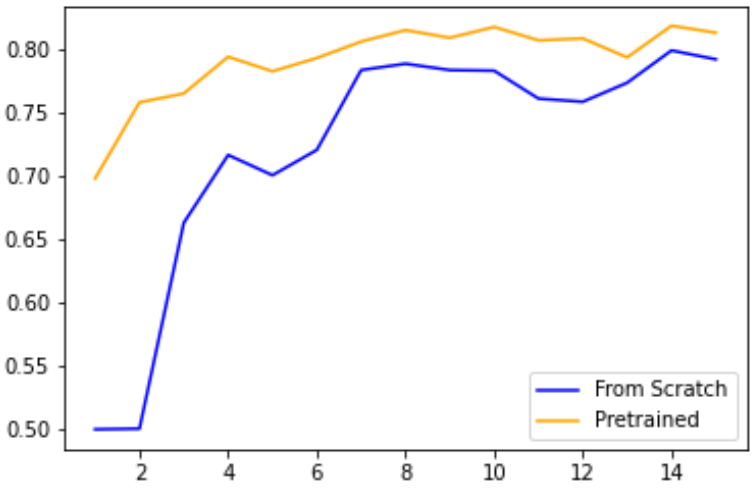

Figure 1: Accuracy comparison

Question 4

As seen in figure 1, the pretrained model has a better validation accuracy than the model trained from scratch, across all epochs. It starts out with an accuracy of 70% and increases slightly throughout training to reach 81%. The model trained from scratch starts with a lower accuracy of 50%(completely random) and increases significantly across the first 4 epochs. The rate of increase then matches that of the pretrained model, and the final accuracy is 79%. The pretrained model needs fewer epochs to reach an accuracy that is close to the final one, which is logical because the parameters are already pretrained, and not randomly generated like in the other model.

Question 5

The main limitation of the language modeling objective implemented in this lab is that it only masks the tokens that come after the region of interest. Thus, it only takes into account the left-to-right direction of the text. The Masked Language Model pre-training objective proposed in [2] alleviates the previously mentioned unidirectionality constraint by randomly masking some of the tokens from the input, and trying to predict the token from its context. It thus enables the representation to fuse the left and the right context.

References

[1] Dileep Kumar Patchigolla (https://stats.stackexchange.com/users/82404/dileep-kumar patchigolla). How to account for the no:of parameters in the multihead self-attention layer of bert. Cross Validated. URL:https://stats.stackexchange.com/q/435993 (version: 2021-07-05).

[2] Quoc V. Le and Tomás Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.