Romy BEAUTÉ
romybeaute.univ@gmail.com

Lab session # 3
ALTEGRAD 2021

12/09/21

1 Question 1 : What is the role of the square mask in our implementation? What about the positional encoding?

As we use a generative pre-training [1] framework in a context of language modeling task, the pre-training steps of our model consist in learning to predict the tokens at the next positions (of the sentence), given the previous ones only. In order to prevent the attention mechanism of our model from using information of tokens at the next positions to predict them (overfitting) we implement a square attention mask (along with the input sentence) that will "mask" we tokens on future positions. We implement a positional encoding layer in order to give some informations about the (relative or absolute) position of a specific token in the input, in order to account for the position of a specific word in a sentence.

2 Question 2 : Why do we have to replace the classification head? What is the main difference between the language modeling and the classification tasks?

The classification head has to be replaced in order to produce an output distribution over new target tokens, and to prevent using a model that has already been pre-trained on a particular classification task. The key difference between Language Modeling (LM) and Classification Tasks (CT) is that LM will try to **generate** the most next suitable word by predicting it given a previous sequence (of words). By essence, LM will not rely on labelled data, but on unsupervised learning of complex data corpus. At the contrary, CT will use encoded information to classify words. For example, CT can be used after LM.

3 Question 3 : How many trainable parameters does the model have in the case of 1) language modeling task ; 2) classification task? Please detail your answer.

Recall parameters impremented :

```
ntokens = len(ind2token) #5maybe -4 #fill me # the size of vocabulary
nhid = 200  # the dimension of the feedforward network model in nn.TransformerEncoder
nlayers = 4  # the number of nn.TransformerEncoderLayer in nn.TransformerEncoder
nhead = 2  # the number of heads in the multiheadattention models
dropout = 0  # the dropout value
nclasses = 2 # for classification task only
model = Model(ntokens, nhead, nhid, nlayers, ntokens, dropout).to(device)
print(ntokens)
>> out : ntokens = 50001
```

In our implementation, we have 4 transformer layers (L=4), 200 hidden dimensions *(nhid=4)*. Starting with the base module of our model, a sequence of words (corresponding to the vocabulary input, composed of tokens) are first passed to the Embedding layer : each word will be embedded with a fully connected layer. During this operation, the number of parameters is therefore equal to the number of tokens *ntokens* multipled by the hidden dimension *nhid*.
In the Transformer layer, the multi-head self-attention will make a linear projection of the words in each attention head. The model also contains 2 fully connected layers that are (individually) normalized in a normalisation layer.
Moreover, we note that we have *nlayers = 4*, Transformer layers, which multiply the number of single layers by 4 in our case. We also note that the classification head will act as a linear projection from the hidden dimensions *nhid* to the classes *nclasses* (for classification tasks), or directly to the individual tokens (for modeling tasks). This will be the main difference in the resulting number of training parameters of the model. Note that in our case, we have *nclasses=2* and *ntokens = 50001*.

4 Question 4 : Interpret the results.

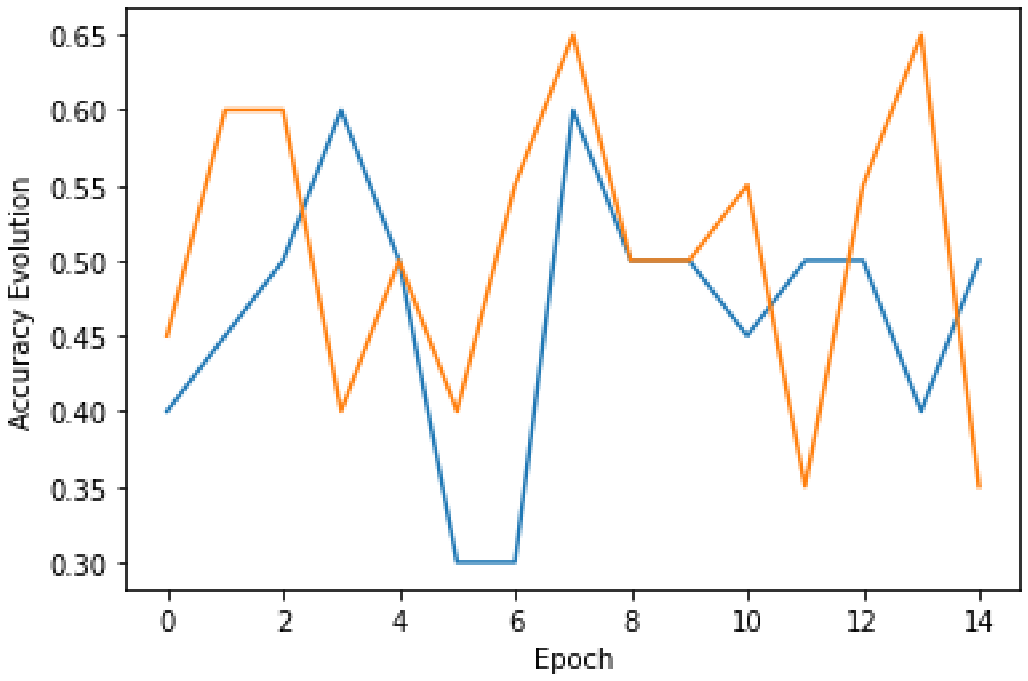

Figure 1: Caption

Here, we see that the pre-trained model (orange curve) starts with a better accuracy than the the model from scratch, and will stabilized in higher values. Indeed, in 1, the pre-trained model's accuracy is close to 0.45 at epoch 0, compared to the model from scratch that shows a 0.40 accuracy. The pre-trained model indeed has an advantage on the model from scratch, as it has been fed with prior information. The classification doesn't convey the same aim : it will not seek for an deep understanding of the word as in modelling (worse semantic representation). It is therefore with no surprise that we observe an overall better accuracy for the pre-trained model.

5 Question 5 : What is one of the limitations of the language modeling objective used in this notebook, compared to the masked language model objective introduced in [1]

In the BERT model described in [1], a bidirectional model is used to predict masked unseen words from both direction. This means that both previous words (before the word in the sentence, ie left) and future words (after in the sentence, ie right) with respect to the position of the word in the sentence we want to model. In bidirectional models, more words are used to pre-train the model, and therefore more information can be extracted and used to generate a word that has the most appropriate meaning in a specific context (for a given input). Therefore, to improve the model used in this notebook, one solution could be to take inspiration from the paper and implementing a deep unidirectional architectures for the pre-training layers of the model. This should allow to tackle a broader set of NLP tasks, and to tackle them in a more consistent way.

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.