1 Question 1

Square mask :
When we look at the implementation of the square mask, it creates a matrix where the upper triangular part is filled with $-\infty$ values. Thus, since we are doing language modeling here, by generating the next word of a group of words, we want to make sure that the prediction i can only depend on the outputs that are before i and not future outputs which would make no sense in our case.

Positional encoding :
As we can see the way we do the model, conversely to rnn, there here are no information about the position of words in the sentence. So this information needs to be added to our model because it's really informative. The positional embeddings are only computed once, and reused for all the sentences. Here in this lab we are using the same as in [2].

2 Question 2

We have to change classification head because we have to adapt to the task we have to do, we where training it on language modeling, and then when we want to do classification.So it's how we adapt the model. But most of the training is already done since we pretrained our model on language modeling.

The main difference between language modeling and classification is the objective, in the first you want to predict words, in the second you want to do classification from a sentence or document in order to perform sentiment analysis or other. Thus, the outputs are not the same, one is probability of words, the other is the probability to be classified a certain way.

3 Question 3

for this question, we are going to compute it one by one using the architecture base and classifier seen in the lab.

Architecture base :

- **Embedding** : $dim_{vocab} * n_{hid}$ Here we are training the embedding matrix.
- **Positional Encoding** : 0. As said in question 1, it's not a trainable encoding we have here.
- **Transformer** :
  - We have 4 transformer so the result we see next is going to be **times 4**.
  - **Multi-Head Attention** : So we have 3 matrix that are Query, Key, Value : Each one is learn with weights and bias, so it's : $3 * (n_{hid}^2 + n_{hid})$ (each one have a $n_{hid} * n_{hid}$ matrix to learn and n_{hid} biases. (I don't take in account the head here, because it's the same parameters wise).
  - **Compute self attention** : So we are doing a linear transformation at the end when we concatenate the heads and we have $(n_{hid}^2 + n_{hid})$ parameters.
  - **Normalize** : so in order to normalize we need two n_{hid} vectors, so it's $2 * n_{hid}$
  - **Feed Forward** : So we have two feed forward so it's : $2 * (n_{hid}^2 + n_{hid})$ parameters.
  - And we **normalize** again : $2 * n_{hid}$
- **Classification** : We are doing a linear to do classification from n_{hid} to $n_{classes}$ so we have $n_{hid} * n_{classes} + n_{classes}$ parameters
- **Language modeling** : we are doing a linear transformation : n_{hid} to dim_{vocab} so we have $n_{hid} * dim_{vocab} + dim_{vocab}$ parameters.

Thus we have those two formulas :

- **Classification** : $N_{params} = dim_{vocab} * n_{hid} + 4 * (4 * (n_{hid}^2 + n_{hid}) + 2 * n_{hid} + 2 * (n_{hid}^2 + n_{hid}) + 2 * n_{hid}) + n_{hid} * n_{classes} + n_{classes}$
- **Language Modeling** :$N_{params} = dim_{vocab} * n_{hid} + 4 * (4 * (n_{hid}^2 + n_{hid}) + 2 * n_{hid} + 2 * (n_{hid}^2 + n_{hid}) + 2 * n_{hid}) + n_{hid} * dim_{vocab} + dim_{vocab}$

We then have the two result we want, (taking $n_{hid} = 200$ and $n_{vocab} = 100$ (Same as the one i print in my notebook) :

$N_{params,classi} = 1008100$
$N_{params,languagemodeling} = 988402$

4 Question 4:

We can see that after 15 epochs, we still have a performance gap between our two models. The one with pretrain weights is slightly better in terms of accuracy. We can see we are doing fine tuning because the model with pretrained weights as an increasing accuracy and then a ceiling. However, our model meets a lower ceilling and never achieve to be as preferment as the pretrained weights even if the gap closed a bit from what it was at the start and after at 15 epochs. We can also see that our models reach really quickly this ceiling, maybe our task is too 'easy' and cannot really make better results.

This might be explained by the fact that we learn our words representation from a little corpus , but the pretrained weights can have been learn through a bigger corpus hence the links between words is better understood and the representation is better. More over the long term dependency are better represented that way. All in all, having pretrained weights make the representation capture more information as the pretrained weight are learned on very large amounts of data and not only on one specific task.

5 Question 5

Our model is unidirectional because of the masked attention we use implied by the fact our language modeling task is predicting the next tokens in a sequence given the previous ones. Hence, when after you want to transfer the knowledge to a sentence-level task such as classification or also token-level task such as question answering where having all the information is mandatory, the weights are not optimal as they weren't trained to use all the information they had access to.
In order to correct that, [1] used another type of masked attention : they randomly mask a percentage of words in the sentence and the model has to predict the masked words, by doing so the model uses right and left context to make its prediction. Moreover (this is not concerning the mask but I think it's still really interesting), to have even more precise weights, they use a a "next sentence prediction" , where two sentences are fed to the model, and it has to predict whether they follow each other or not. It helps the model learn relationships between sentences. All in all, all of this is made to have a solid and precise representation of the words.

The downside of this new architecture is that BERT cannot do language generation or if one tries to make BERT generate text by having an all masked input it's not going to be of good quality.

References

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

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.