AMÉDÉE Romain
romain.amedee@telecom-sudparis.eu

1 Question 1

Let us count the number of trainable parameters in the model.

First we proceed with the **word embeddings**. Our dictionnary is made of the **32000** most frquent tokens of the XLMR dictionnary. The size of the embedded vectors is **512**. Therefore we have the following number of parameters for the embedding :

$$W_{embed} = N_{tokens} \times Dim_{embed} = 32000 \times 512 = 16384000$$

Then there is the positionnal encoding :

$$W_{positional} = 258 * 512 = 132096$$

Then, we have 4 transormer blocks made of :

- An attention layer. The model performs three linear operations (Query, Value, Key). The dimensions of those operations are **512** for both input and output. Then one last linear operation is performed with same dimensions for projecting the result. Therefore, we have the following number of parameters for the attention layer (without biases) :

$$W_{attention} = 3 \times Dim^2_{embed} + Dim^2_{embed} = 4 \times Dim^2_{embed} = 4 \times 512^2 = 1048576$$

- Two fully connected layers with dimension mappings $512 \longrightarrow 512$. Therefore, we have the following number of parameters :

$$W_{linear} = 2 \times Dim^2_{embed} = 2 \times 512^2 = 524288$$

For the transformer block we have :

$$W_{transformer} = W_{attention} + W_{linear} = 1572864$$

Here, we do not count the bias, the parameters for the norm layers or the parameters of the language model head. Finally the total number of parameters is :

$$W_{tot} = W_{embed} + W_{positionnal} + 4 \times W_{transformer} = 22807552$$

2 Task 3 & Task 4 : Fairseq results

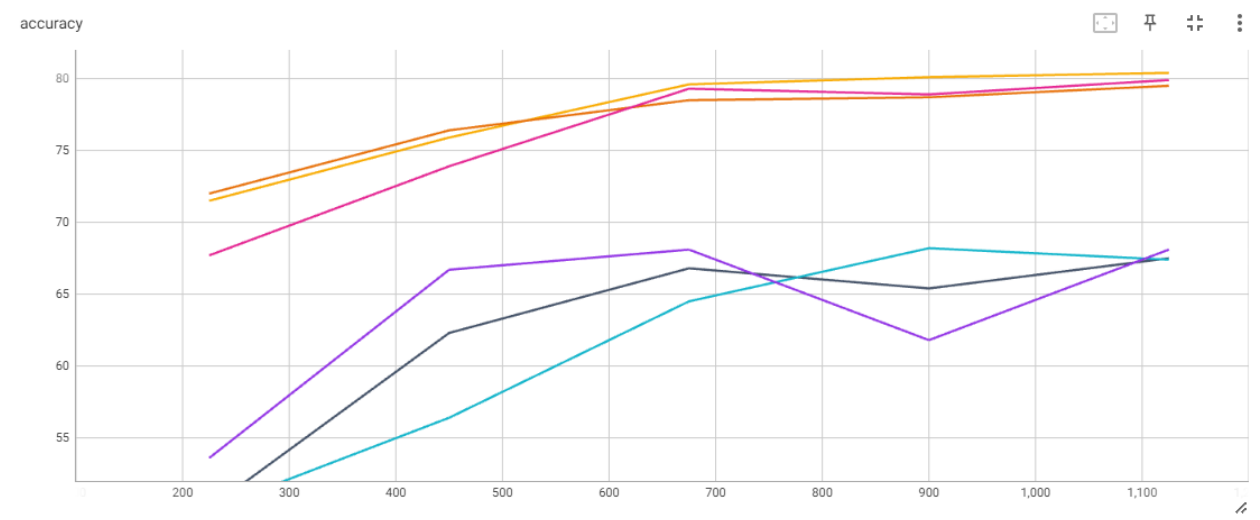

Figure 1: Test accuracy for each seed of the two models (3 upper are pretrained and 3 lower are random models

As expected, the accuracy for the pretrained model us consistently higher than the accuracy of the model where we introduced randomness. We also notice that the results for the pretrained model are more consistent whereas for the random models the accuracy can diminish through the epochs. For both models the accuracies converge.

For each SEED of each model we choose the checkpoint with the best validation accuracy :

	Valid. Accuracy	Test accuracy	Step
PT — SEED 1	81%	79.3%	675
PT — SEED 2	80%	78.5%	675
PT — SEED 3	83.5%	80.1%	900
RD — SEED 1	62%	65.4%	900
RD — SEED 2	65%	66.7%	450
RD — SEED 3	62.5%	68.2%	900

Table 1: Results for each seed and for each model (PT = pretrained, RD = random)

For each epoch we compute the corresponding average accuracy and standard deviation :

	Average accuracy	Standard deviation
PT — Epoch 1	70.4%	1.92%
PT — Epoch 2	75,4%	1.92%
PT — Epoch 3	79.13%	0.46%
PT — Epoch 4	79.23%	0.61%
PT — Epoch 5	79.93%	0.36%
RD — Epoch 1	51.23%	1.67%
RD — Epoch 2	61,8%	4.22%
RD — Epoch 3	66,46%	1.49%
RD — Epoch 4	65.13%	2.61%
RD — Epoch 5	67.6%	0.31%

Table 2: Average test accuracy and standard deviation for each epoch (PT = pretrained, RD = random)

For both the pretrained model and the random model, the average test accuracy increases in general exept between epoch 3 and 4 for the random-based model. The perfomance do become better throuch the epochs but with much better results for the pretrained model (between 20% and 10% better). However, we notice that the standard deviations are much more consistent and decrease through the epochs whereas it is rather random for the random-based model. This means the results are rather inconsistent between seeds for the randomness-based model which is expected.

3 Task 5 : HuggingFace Transformer

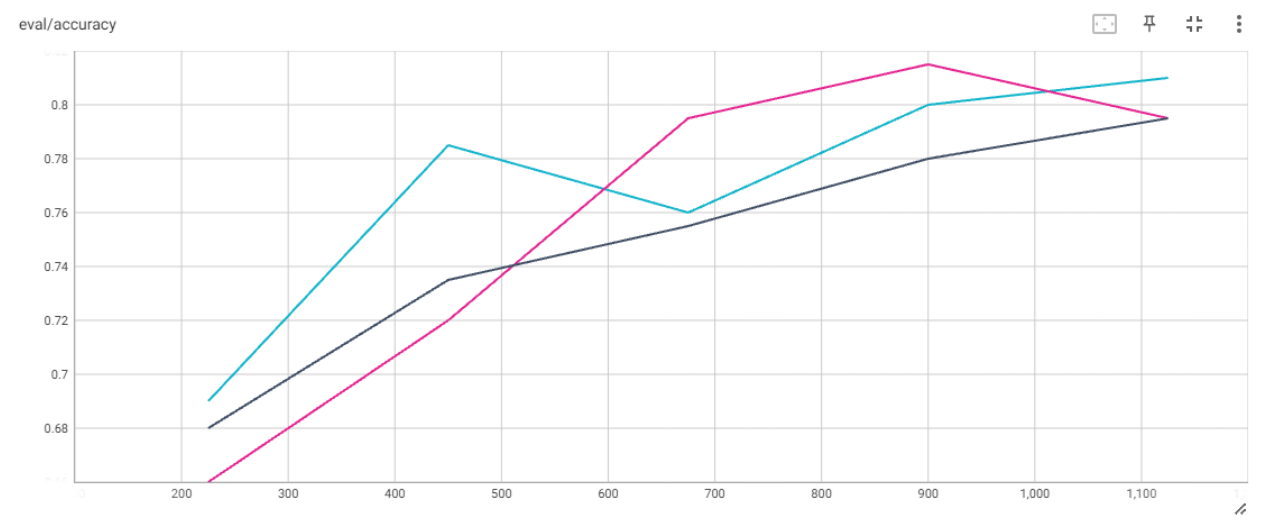

Figure 2: Eval accuracy for each seed of the HuggingFace model

Here I did not manage to display the test accuracies in the tensorboard so I can only show the results for the evaluation (validation). We notice that the accuracies still converge and we could expect the same behavior for the test accuracies.

4 Question 2

Here are the parameters used in the LoRA configuration :

- **Rank (r=16)** : the rank of the low-rank matrices learned during the fine-tuning process.
- **Scaling (lora_alpha=32)** : a scaling factor for the LoRA weight matrices.
- **Targe modules (target_module=["query_key_value"])** : a list of the names of the modules to apply LoRA to. In this case, we are applying LoRA to the query, key, and value attention weights.
- **Dropout (lora_dropout=0.05)** : the dropout probability for the LoRA layers.
- **Bias (bias="none")** : specifies whether to use a bias term in the LoRA layers.
- **Task type (task_type="CAUSAL_LM")** : type of task that the model is being fine-tuned for. Here it specifies that we are fine-tuning the model for a causal language modeling task.