1 Question 1

Using the following code snippet, we can understand the architecture of the small RoBERTa model:

```python
from fairseq.models.roberta import RobertaModel

RobertaModel.from_pretrained("models/RoBERTa_small_fr/", checkpoint_file='model.pt')
```

This model is made of:

- an token embedding which maps the 32 000 tokens into vectors of length 512. Thus, it has $32\,000 \times 512$ parameters.
- a learned poistional embedding which has 258×512 parameters.
- 4 transformer encoder layers. Each one has:
  - self attention mechanism of $3 \times 512 \times 512$ parameters for projecting Q, V, K and 512×512 parameters for projecting the result (weights without bias).
  - 2 linears layers of each 512×512 parameters (weights without bias).
  - 2 normalization layers which we don't take into account.
  - 2 dropout layers which don't have learnable parameters.
- a language model head which we don't take into account

Thus, RoBERTa has a total of $(32\,000 + 258 + 6 \times 4 \times 512) \times 512 = 22\,807\,552$ learnable parameters.

2 Result analysis on Fairseq

When finetuning the pretrained RoBERTa model, I obtained the following accuracy on a pretrained model:

Epoch	Worst Accuracy	Best Accuracy	Average	Standard Deviation
1	67.4	71.6	70.1	2.37
2	74.8	76.7	75.8	0.95
3	78.7	79.8	79.3	0.56
4	79	80	79.5	0.5
5	79.5	80.4	80	0.46

This is what I obtained on a random model.

Epoch	Worst Accuracy	Best Accuracy	Average	Standard Deviation
1	50	54.3	51.5	2.45
2	56.7	66.7	61.9	5.00
3	64.6	67.9	66.4	1.68
4	61.7	68.2	65.2	3.27
5	67.4	68.1	67.7	0.37

The result are rather consistent, no matter the seed, or the pretrained model.
We easily see that the pretrained model are doing much better and are closer to their convergence values. This is because those pretrained models have already seen and understood language, even if they were dedicated to another task.

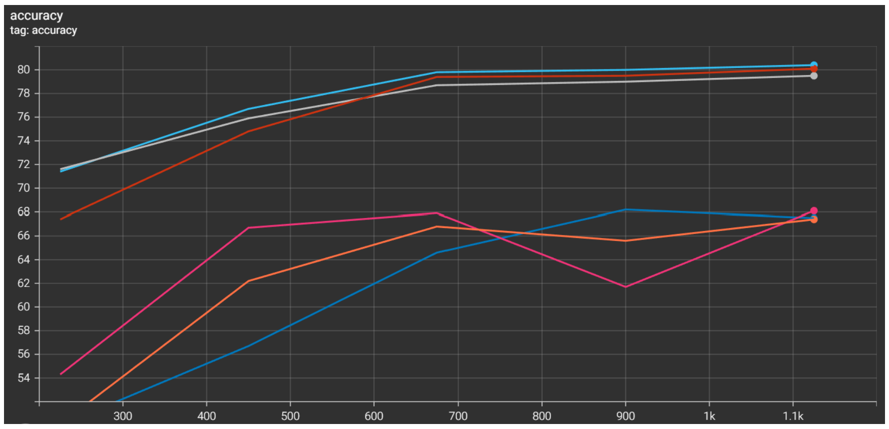

Figure 1: Accuracy scores for 3 different seeds on test set. Random models are below while pretrained models are on top.

3 Result analysis on HuggingFace

I didn't succeed in running the prediction on the test set at every epoch. Thus I can only access the evaluation accuracy at the end of each epoch. This is what I obtained:

Epoch	Worst Accuracy	Best Accuracy	Average	Standard Deviation
1	65.5	69	67	2.55
2	71.5	72.5	72.2	0.82
3	71.5	76.5	74.5	3.74
4	76.5	78	77	1.22
5	77	78.5	78	1.22

We can notice that the results are comparable to the ones observed on Fairseq.

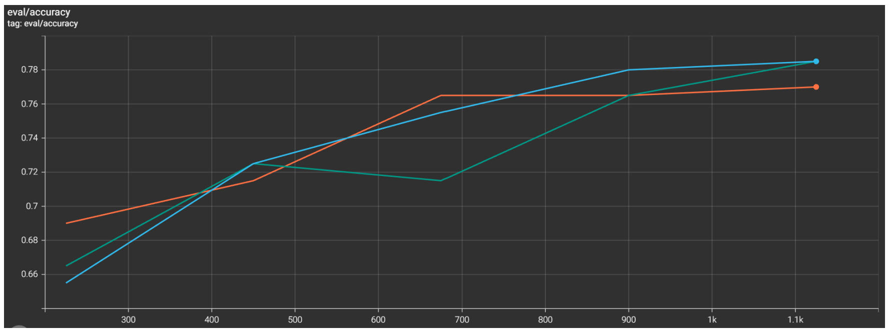

Figure 2: Accuracy scores for 3 different seeds on evaluation set.

4 Question 2

There are several points on which we can compare Fairseq and HugginFace:

- Fairseq only works on `Pytorch`, while HuggingFace works on both `Pytorch` and `Tensorflow`. I personnally prefer `Tensorflow`, but I think it is better when a tool has a fixed way of working. Thus Fairseq is better on that point.
- the community is bigger on HuggingFace. Fairseq is starred 20k times while HuggingFace is starred 74k times on GitHub.
- the use of each tool is slightly different. I think HuggingFace does everything Fairseq does, but it provides a web interface with models and datasets. HuggingFace is selling a cloud solution to run models while Fairseq is made to be run locally.

That is why I globally prefer HuggingFace.

References