Boubacar SOW
boubacar.sow@ens-paris-saclay.fr

Lab session # 4
ALTEGRAD 2023

11/14/23

1 Manual Computation of Model Parameters

Upon examination of the Python code for the model located in `fairseq/fairseq/models/roberta/model.py`, it is evident that the model's architecture is a standard implementation of the transformer. The model comprises:

- **Embedding Vectors**: The size of the vectors can be easily obtained by executing `model["model"]["encoder.sentence_encoder.embed_tokens.weight"].shape` in Python. Consequently, the dimensions of the embedding matrix are (32000, 512). Therefore, at this stage, the number of parameters is 32000×512.
- **Positional Embeddings**: These are distinct from positional encodings, which are not learned. The dimensions of the learned positional embedding are (258, 512), which can be obtained by executing `model["model"]["encoder.sentence_encoder.embed_positions.weight"].shape`.
- **Transformer Encoder Layers**: Each layer consists of
  - **Multiheaded Self Attention Mechanism**: This includes four matrices of size (512, 512) for the Q, K, V, and output projection weights. Biases are not considered in this computation.
  - **Linear Layers**: There are two linear layers, each of size (512, 512).

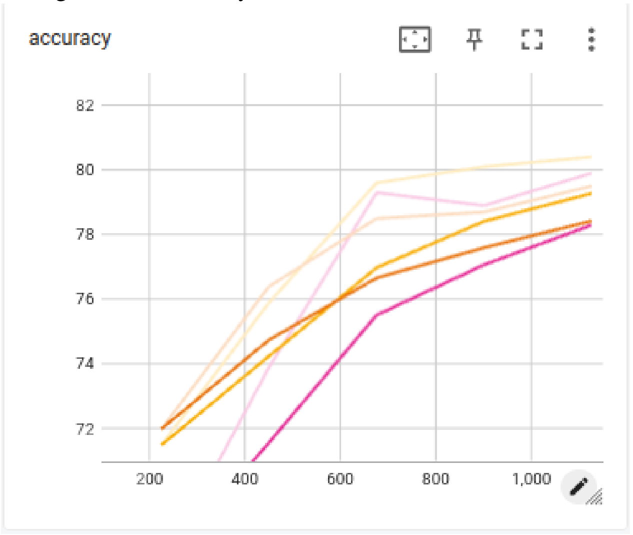
  Normalizing layers and the Roberta head are not considered in this computation.

Therefore, the total number of parameters is computed as follows:

$$32000 \times 512 + 258 \times 512 + 4(4 \times 512 \times 512 + 2 \times 512 \times 512) = 22807552$$

2 Task 3: Report of the average accuracy and its standard deviation on the test set

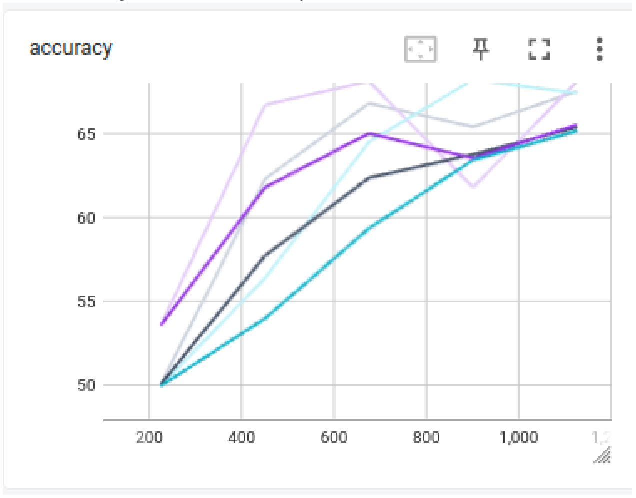
Figure 1: Accuracy of the Roberta small fr model



From the statistics provided, it appears that the model's performance improves with each epoch for all three seeds. This is a good sign as it indicates that the model is learning from the data and improving its predictions over time.
For **seed 1**, the model starts with an accuracy of **67.7%** and improves to **79.9%** by the fifth epoch.
For **seed 2**, the model starts with a higher initial accuracy of **72%**, and also ends with a slightly lower final accuracy of **79.5%** compared to seed 1.

For **seed 3**, the model starts with an accuracy of **71.5%** and achieves the highest final accuracy of **80.4%** among the three seeds. The mean accuracy of this model is **80%** and its mean standard deviation is **0.80%**. It's interesting to note that despite the different starting points (due to the different seeds), all three models converge to a similar level of accuracy by the fifth epoch.

Figure 2: Accuracy of a random model



We can observe that the accuracies for this random model are much lower than the RoBERTa model. The highest accuracy being only **68.1%** The variance of this model is also higher.

3 Detailed Explanation of Parameters in LoraConfig

LoRA, which stands for Low Rank Adaptation of Large Language Models, is a method that allows for the efficient adaptation of pre-trained language models. The `LoraConfig` class provides a way to control how LoRA is applied to the base model through several parameters:

- **"r"**: This parameter represents the rank of the update matrices. It is expressed as an integer. A lower rank results in smaller update matrices with fewer trainable parameters, thus increasing the efficiency of LoRA. However, choosing a higher rank may diminish the efficiency gains of LoRA while potentially improving the model's performance.
- **"target_modules"**: This parameter specifies the modules to which the LoRA update matrices should be applied. In the original LoRA paper, the update matrices are applied only to the "Q" and "V" attention matrices. However, some studies have shown that applying LoRA to additional layers or even all layers can lead to better performance.
- **"alpha"**: This is the LoRA scaling factor. It scales the learning weights and is typically set to 16.
- **"bias"**: This parameter determines whether the bias parameters should be included in the training process.
- **"lora_dropout"**: This is the dropout probability we choose
- **"module_to_save"**: This is a list of modules, apart from LoRA layers, that should be trained and saved in the final checkpoint.
- **"layers_pattern"**: This is a list of layers that LoRA transforms.
- **"layers_pattern"**: This parameter is used to match layer names in `target_modules` if `layers_to_transform` is specified.
- **"rank_pattern"**: This is a mapping from layer names or regular expression patterns to ranks that differ from the default rank specified by "r".
- **"alpha_pattern"**: This is a mapping from layer names or regular expression patterns to alphas that differ from the default alpha specified by `lora_alpha`.
- **"task_type"**: This parameter is used to specify the type of task for which the model is being trained.

In our implementation we only used the parameters, **"r"**, **"lora_alpha"**, **"target_modules"**, **"lora_dropout"**, **"bias"** and **"task_type"**.

References